IBM® Tivoli® Netcool/OMNIbus Gateway for Message Bus
12.0

Reference Guide
February 11, 2021

SC14-7650-16

Notice

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 89.

Edition notice

This edition (SC14-7650-16) applies to version 12.0 of IBM Tivoli Netcool/OMNIbus Gateway for Message Bus and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces (SC14-7650-15).

Contents

About this guide

The following sections contain important information about using this guide.

Document control page

The IBM Tivoli Netcool/OMNIbus Gateway for Message Bus documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM Tivoli Netcool/OMNIbus 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
SC14-7650-00	February 25, 2011	First IBM publication.
SC14-7650-01	December 2, 2011	The following properties were updated in "Properties and command line options" on page 10 : • **Gate.Reader. DetailsTableName** • **Gate.Mapper. ForwardHistoricDetails** • **Gate.Mapper. ForwardHistoricJournals** • **Gate.Reader. JournalTableName** "Table replication definition file" on page 29 updated.
SC14-7650-02	April 6, 2012	"Installing the gateway" on page 6 updated. "Configuring HTTPS/SSL connections between the gateway and Message Bus" on page 50 updated.
SC14-7650-03	May 11, 2012	"Configuring HTTPS/SSL connections between the gateway and Message Bus" on page 50 updated.
SC14-7650-04	November 30, 2012	Guide updated for Netcool/OMNIbus V7.4 release. "Installing the gateway" on page 6 updated.

Table 1. Document modification history (continued)

Document version	Publication date	Comments
SC14-7650-05	March 13, 2014	Support for SCA-LA added. "Summary" on page 2 updated. Integrating with SCA-LA added. Explanations added to the descriptions of the following properties in "Properties and command line options" on page 10 whose default values depend on whether you are running the gateway in XML mode or SCA-LA mode: • **Gate.MapFile** • **Gate.StartupCmdFile** • **Gate.Reader. TblReplicateDefFile** • **Gate.XMLGateway. TransformerFile** • **Gate.XMLGateway. TransportFile** • **Gate.XMLGateway. TransportType** Description of the `scalaTransport.properties` file added to "Configuring transport module properties files" on page 61. "Running the gateway" on page 85 updated.
SC14-7650-06	April 1, 2014	"Summary" on page 2 updated. Note added to the description of the **Gate.Java.Arguments** property in "Properties and command line options" on page 10 "Preventing unbound reads from files or sockets being exploited in a denial of service attack" on page 31 added. Description for the new **readTimeout** property added to the SCA-LA transport properties table in "Configuring transport module properties files" on page 61.
SC14-7650-07	June 12, 2014	"Summary" on page 2 updated.
SC14-7650-08	March 12, 2015	The term SCA-LA has been replaced by IBM Operations Analytics - Log Analysis. Chapter 1, "Gateway for Message Bus," on page 1 updated. "Summary" on page 2 updated. An additional map definition file, `xml1300.map`, is now listed in the "Configuration files for gateways operating with IBM Operations Analytics - Log Analysis" table entry. 64 bit support added. "Gateway dependencies" on page 6 updated. "Installing probes and gateways on Tivoli Netcool/OMNIbus V8.1" on page 7 added. Added instructions for installing the gateway on Tivoli Netcool/OMNIbus Version 8.1.

Table 1. Document modification history (continued)		
Document version	**Publication date**	**Comments**
SC14-7650-08	March 12, 2015	"Properties and command line options" on page 10 updated. Updates were made to these properties: **Name**, **PropsFile**, **Gate.MapFile**, **Gate.StartupCmdFile**, **Gate.Reader.Password**, **Gate.Reader. TblReplicateDefFile**, **Gate.XMLGateway. DateFormat**, **Gate.XMLGateway. TransformerFile**, **Gate.XMLGateway. TransportFile**, and **Gate.XMLGateway. TransportType**, **Gate.Java.Arguments**, **Gate.Java.ClassPath**, **Gate.Java.Debug**, and **Gate.Java.Library Path**. "Preventing unbound reads from files or sockets being exploited in a denial of service attack" on page 31 updated to replace the term SCA-LA with IBM Operations Analytics - Log Analysis. Also, updates to clarify Java arguments versus gateway properties.
SC14-7650-08	March 12, 2015	"Integrating with IBM Operations Analytics - Log Analysis" on page 33 updated to discuss the `xml1300.mapdefinition` file. Added a section to "Troubleshooting the IBM Operations Analytics - Log Analysis integration" on page 38 that discusses when it might be necessary to edit the `NmosObjInst` field in the `xml1300.map` definition file.
SC14-7650-09	December 10, 2015	Chapter 1, "Gateway for Message Bus," on page 1 updated. "Summary" on page 2 updated. An additional map definition file, `xml1302.map`, is now listed in the summary table. "Properties and command line options" on page 10 updated with the following properties added to the common properties table: **ConfigCryptoAlg** **ConfigKeyFile** The availability of these properties has been extended so that in addition to being used with the gateway they can also be used in the transport module. "Integrating with IBM Operations Analytics - Log Analysis" on page 33 updated to discuss the `xml1302.map` definition file.
SC14-7650-09	December 10, 2015	"Configuring HTTPS/SSL connections between the gateway and Message Bus" on page 50 updated. Added "Using the nc_httpcertimport utility" on page 54 Added "Encrypting gateway and transport module properties" on page 73 Added property descriptions for the `socketTransport.properties` file. You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode. A number of sections in the guide have been updated to account for these two modes of operation.

Table 1. Document modification history (continued)		
Document version	**Publication date**	**Comments**
SC14-7650-10	May 31, 2017	Guide updated for Version 8 of the gateway. "Setting the path to the JRE libraries" on page 7 was updated . "Configuring transport module properties files" on page 61 was updated to reflect the support for encrypted passwords with the Transport Module. The following properties were added to the HTTP/HTTPS transport to support the new functionality added to send batched events from the gateway to the object server: • **batchHeader** • **batchFooter** • **bufferFlusher** • **bufferSize** • **httpTimeout** • **retryLimit** • **retryWait** Note added to the transport properties files to state that all property values must be unquoted.
SC14-7650-11	August 18, 2017	"Configuring SSL connections between the gateway and IBM Operations Analytics - Log Analysis" on page 52 updated to include information about exporting the server certificate from the SCALA server.
SC14-7650-12	April 12, 2018	"Summary" on page 2 updated. The following properties were added to the HTTP Transport: • **password** • **username** Support for integrating with Kafka added. "Integrating the gateway with Kafka" on page 42 added.
SC14-7650-13	October 11, 2018	Kafka details in the following topics updated: • Chapter 1, "Gateway for Message Bus," on page 1 • "Using the transport module" on page 60 • "Integrating the gateway with Kafka" on page 42

Table 1. Document modification history (continued)

Document version	Publication date	Comments
SC14-7650-14	April 19, 2019	Guide updated for Version 10 of the gateway. "Summary" on page 2 updated. The following properties were added to the properties file: • **Gate.XMLGateway.TransformerInputType** • **Gate.XMLGateway.PublishTraceOn** The following properties were added to the HTTP Transport: • **useNullHostnameVerifier** • **batchSeparator** • **concurrentRequestThreads** The following topic added: "Configuring the gateway to produce JSON data" on page 32. Version 10 addresses the following APAR and test fix mergeup: • **IJ09147**: Provides support for ON INSERT ONLY functionality in the gateway map file. • **PATCH gateway-libngjava-7**: Merge up test fixes in release 7 for GA release 8. Test fix includes improvements on string handling optimization and enhancement on exception logging.

Table 1. Document modification history (continued)

Document version	Publication date	Comments

SC14-7650-15	March 20, 2020	Guide updated for Version 11 of the gateway.
		
		A description for **`Gate.XMLGateway.MessageKey`** was added to "Gateway-specific properties" on page 20.
		
		The following topic was updated: "Integrating the gateway with Kafka" on page 42.
		
		The probe makes use of the following new features in the Kafka transport:
		
		• Kafka producer uses the following settings if they are not configured in the client configuration:
		
		`max.block.ms=20000`
		`request.timeout.ms=30000`
		
		• Kafka producer sends message with the message key assigned by the application.
		• New properties in the `kafkaTransport.properties` file for Kafka Liveness check.
		• Kafka producer periodically checks Kafka liveness.
		• Kafka producer halts outbound messaging as soon as the Kafka target is internally detected to be unreachable.
		• Kafka producer resends failed messages after reconnecting to the Kafka target.
		• Kafka producer reconnection is initiated by the gateway after hitting `TransportSendMsgException`.
		• Zookeeper watch service is disabled in the producer mode.
		
		"Known issues with the Gateway for Message Bus" on page 88 added.
		
		Version 11 also addresses the following APAR:
		
		• **IJ17528**: Fixed issue whereby UTC data is not converted to the date-time format configured in **`Gate.XMLGateway.DateFormat`**. This change introduces the following new behavior: When no value is set for **`Gate.XMLGateway.DateFormat`**, all UTC data will be in millisecond units in outbound messages for JSON and XML use cases.

Table 1. Document modification history (continued)		
Document version	**Publication date**	**Comments**
SC14-7650-16	February 11, 2021	Updated for version 12. Summary Table updated. The dependency bundle updated: `gateway-libngjava-9`, `common-transportmodule-27` "Integrating the gateway with Kafka" on page 42 updated. Descriptions for the following properties added to "Properties and command line options" on page 10: • **`Gate.XMLGateway.ForwardKeyValuePairs`** • **`Gate.XMLGateway.ForwardMappedFields`** • **`Gate.XMLGateway.ReplaceXmlInvalidChars`** • **`Gate.XMLGateway.XmlInvalidCharsFormatter`** • **`Gate.XMLGateway.MessageKey`** Version 12 addresses the following enhancement request: **RFE 136634:** Send dynamic values populating from the ObjectServer field in the JMS Header (multiple fields) for JMS Gateway. This version also addresses the following APAR: **APAR IJ27815**: The XSLT process hits exception upon alert data that contains characters invalid to the XML standard.

Note : The IBM Tivoli Netcool/OMNIbus Gateway for Message Bus was previously documented in the *IBM® Tivoli® Netcool/OMNIbus Message Bus Integration* Version 6.0 reference guide (SC23-8594-00).

Conventions used in this guide

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

Operating system-dependent variables and paths

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

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

```
$OMNIHOME/gates
```

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

```
%OMNIHOME%\gates
```

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

Note : The names of environment variables are not always the same in Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX and Linux environments.

Operating system-specific directory names

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

`$OMNIHOME/platform/`*`arch`*

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

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

Table 2. Directory names for the arch variable

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

OMNIHOME location

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

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

Chapter 1. Gateway for Message Bus

The IBM Tivoli Netcool/OMNIbus Gateway for Message Bus reads Tivoli Netcool/OMNIbus events from an ObjectServer and then converts these events to Extensible Markup Language (XML) format. The transformer module then converts these events from XML format to another format that you specify. The transport module then forwards these events to the target application.

You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode:

1. When you configure the gateway in standard mode, you specify which one of the following transport modules to use:
   - Java Message Service (JMS)
   - Data file
   - Message Queue Telemetry Transport (MQTT)
   - HTTP/HTTPS (Hypertext Transfer Protocol/HTTP Secure)
   - Socket
   - Kafka
2. When you configure the gateway in LA mode, you use the SCALA transport module.

In summary, the default or recommended way of determining which mode of operation you use is as follows:

- Standard mode operation
  - Create a server (using the Server Editor `nco_xigen`) named `G_XML`.
  - Locate the gateway **`Name`** property in the supplied default properties file, `G_XML.props`.
  - Specify the value `G_XML` for the **`Name`** property (which matches the name of the server name created with `nco_xigen`).
- LA mode operation
  - Create a server (using the Server Editor `nco_xigen`) named `G_SCALA`.
  - Locate the gateway **`Name`** property in the supplied default properties file, `G_SCALA.props`.
  - Specify the value `G_SCALA` for the **`Name`** property (which matches the name of the server name created with `nco_xigen`).

Configuration details and other information pertinent to installing, configuring, and running the gateway are described in the following sections:

Note : The gateway can also be used with the IBM Tivoli Netcool/OMNIbus Probe for Message Bus. If you want to use the gateway with the IBM Tivoli Netcool/OMNIbus Probe for Message Bus, see "Using the gateway with the Probe for Message Bus" on page 86 before installing or configuring the gateway.

Summary

Each gateway works in a different way to provide an interface to the ObjectServer. Use this summary information to learn about the Gateway for Message Bus.

You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode. Table 3 on page 2 organizes the summary information according to these two modes.

Table 3. Summary information

Summary information for gateway standard mode and LA mode operation	
Gateway target	XML event consumers
Gateway executable file name	`nco_g_xml`
Gateway jar file	`$OMNIHOME/gates/java/nco_g_xml.jar`
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=swg21665219

Table 3. Summary information (continued)

Requirements	A currently supported version of Tivoli Netcool/OMNIbus
	`common-transformer-11`
	`common-transportmodule-27`
	`gateway-libngjava-9`
	`common-sslutility-1` (This requirement applies only for UNIX operating system platforms that the gateway supports.)
	Java™ Runtime Environment (JRE) 1.7
	The MQTT transport requires the file: `wmqtt.jar`. To obtain and use this JAR file, perform the following steps:
	1. Download the JAR file from the following site:
	http://www-01.ibm.com/support/docview.wss?rs=203&uid=swg24006006
	2. Copy the JAR to the following directory:
	`$OMNIHOME/java/jars`
	3. Include the JAR file in probe classpath by editing either `nco_p_message_bus.env` for UNIX platforms, or `nco_p_message_bus.bat` for Windows platforms.
	Examples:
	On UNIX platforms, at the foot of the `nco_p_message_bus.env` file, include the following lines:
	`WMQTT_JAR=${OMNIHOME}/java/jars/wmqtt.jar`
	`CLASSPATH=${CLASSPATH}:${WMQTT_JAR};`
	On Windows platforms, in the `nco_p_message_bus.bat` replace the following line:
	`set MESSAGEBUS_CLASSPATH=%TRANSFORMER_CLASSPATH%; %TRANSPORT_CLASSPATH%; %JACKSON_CLASSPATH%`
	with the following lines:
	`set WMQTT_CLASSPATH=%JARS_DIR%/wmqtt.jar`
	`set MESSAGEBUS_CLASSPATH=%TRANSFORMER_CLASSPATH%; %TRANSPORT_CLASSPATH%; %JACKSON_CLASSPATH%; %JETTY_CLASSPATH%; %WMQTT_CLASSPATH%`

Multicultural support	Available
IP environment	IPv4 and IPv6
Current version of the gateway for standard mode operation	
	12.0 (patch name: `gateway-nco-g-xml-12`)
Current version of the gateway for LA mode operation	

Table 3. Summary information (continued)	
	9.0 (patch name: `gateway-nco-g-xml-9`) **Note :** Package version 7.0 is the minimum version for running the gateway with OMNIbus Insight Pack Version 1.3.0.2 and Network Manager Insight Pack Version 1.3.0.0 deployed against IBM Operations Analytics - Log Analysis.
Configuration files for the gateway standard mode operation	

	Map definition file:
	`$OMNIHOME/gates/xml/xml.map`
	Properties file:
	`$OMNIHOME/gates/xml/G_XML.props`
	Table replication definition file:
	`$OMNIHOME/gates/xml/xml.reader.tblrep.def`
	Environment (.env) file for Unix:
	On Netcool/OMNIbus 8.1
	`$OMNIHOME/platform/<arch>/bin64/nco_g_xml.env`
	Startup command file
	`$OMNIHOME/gates/xml/xml.startup.cmd`

Transport module configuration files for the gateway standard mode operation	
	The following list identifies the transport properties files related to the different transport modules. Thus, you would use the `jmsTransport.properties` file if your transport module is JMS, the `mqttTransport.properties` file if your transport module is MQTT, and so forth. JMS - `$OMNIHOME/java/conf/jmsTransport.properties` Data file - `$OMNIHOME/java/conf/fileTransport.properties` MQTT - `$OMNIHOME/java/conf/mqttTransport.properties` HTTP and HTTPS - `$OMNIHOME/java/conf/httpTransport.properties` Socket - `$OMNIHOME/java/conf/socketTransport.properties` Kafka - `$OMNIHOME/java/conf/kafkaTransport.properties`, `$OMNIHOME/java/conf/kafkaClient.properties`, and `$OMNIHOME/java/conf/kafkaConnectionProperties.json` **Note :** The gateway supports Java Message Service (JMS) API 1.1 and target applications that use later versions provided they maintain backward compatibility as currently expected by the JMS specification.

Table 3. Summary information (continued)	
Transformer module Extensible Markup Language (XML) configuration files for the gateway standard mode operation	
	$OMNIHOME/java/conf/transformers.xml
Transformer module Extensible Stylesheet Language Transformations (XSLT) configuration files for the gateway standard mode operation	
	• $OMNIHOME/java/conf/cbe2nvpairs.xsl • $OMNIHOME/java/conf/netcool2cbe.xsl • $OMNIHOME/java/conf/netcool2nvpairs.xsl • $OMNIHOME/java/conf/netcool2wef.xsl • $OMNIHOME/java/conf/netcool2json.xsl • $OMNIHOME/java/conf/wbe2nvpairs.xsl • wbepl2nvpairs.xsl • wbm2nvpairs.xsl • $OMNIHOME/java/conf/wef2nvpairs.xsl
Configuration files for the gateway LA mode operation	
	Properties file: $OMNIHOME/gates/xml/scala/G_SCALA.props Table replication definition file: $OMNIHOME/gates/xml/scala/xml.reader.tblrep.def Environment (.env) file: $OMNIHOME/platform/linux2x86/bin/nco_g_xml.env $OMNIHOME/platform/linux2x86/bin64/nco_g_xml.env Startup command file: $OMNIHOME/gates/xml/scala/xml.startup.cmd

Table 3. Summary information (continued)	

	Map definition files:
	`$OMNIHOME/gates/xml/xml.map`
	Use the `xml.map` file if your gateway is running with Network Manager Insight Pack Version 1.3.0 and higher deployed against IBM Operations Analytics - Log Analysis.
	`$OMNIHOME/gates/xml/scala/xml1300.map`
	Use the `xml1300.map` file if your gateway is running with OMNIbus Insight Pack Version 1.3.0.0 and Version 1.3.0.1 and higher and Network Manager Insight Pack Version 1.3.0.0 and higher deployed against IBM Operations Analytics - Log Analysis.
	`$OMNIHOME/gates/xml/scala/xml1302.map`
	Use the `xml1302.map` file if your gateway is running with OMNIbus Insight Pack Version 1.3.0.2 and higher and Network Manager Insight Pack Version 1.3.0.0 and higher deployed against IBM Operations Analytics - Log Analysis.

Transport module properties file for the gateway LA mode operation	
	`$OMNIHOME/gates/xml/scala/scalaTransport.properties`
Transformer module XML configuration file for the gateway LA mode operation	
	`$OMNIHOME/gates/xml/scala/scalaTransformers.xml`
Transformer module XSLT configuration file for the gateway LA mode operation	
	`$OMNIHOME/java/conf/netcool2scala.xsl`
SSL configuration utility for the gateway LA mode operation	
	`nc_httpcertimport` **Note :** The `nc_httpcertimport` utility is supported on all UNIX operating system platforms that the gateway supports. The `nc_httpcertimport` utility is not supported on Windows operating systems.

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.

Gateway dependencies

For Tivoli Netcool/OMNIbus V8.1, all gateways are installed using the Tivoli Netcool/OMNIbus installer. For Tivoli Netcool/OMNIbus V8.1, all gateways are 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.

Setting the path to the JRE libraries

If you are running the gateway on a Windows operating system, you must set the path to the JRE libraries.

If you are running the gateway on Tivoli Netcool/OMNIbus v8.1, then you must also add the following directories to the PATH environment variable:

```
C:\IBM\Tivoli\Netcool\omnibus\platform\win32\lib;
C:\IBM\Tivoli\Netcool\platform\win32\bin;
C:\IBM\Tivoli\Netcool\platform\win32\lib;
C:\IBM\Tivoli\Netcool\omnibus\platform\win32\bin;
C:\IBM\Tivoli\Netcool\omnibus\bin
```

Setting the classpath on Windows operating systems

Before you can run the gateway on Windows operating systems you must set the classpath to include the paths to the following Java libraries:

- `nco_g_xml.jar`
- `ngjava.jar`
- `TransportModule.jar`
- `Transformer.jar`
- `com.ibm.ws.ejb.thinclient_8.5.0.jar`
- `com.ibm.ws.sib.client.thin.jms_8.5.0.jar`
- `com.ibm.ws.orb_8.5.0.jar` (This is needed where a non-IBM JRE is used.)
- `JSON4J.jar`

You can do this by setting the **`Gate.Java.ClassPath`** property to include a semi-colon separated list of the paths to each of these libraries. For example:

```
'C:\\IBM\\Tivoli\\Netcool\\omnibus\\gates\\java\\nco_g_xml.jar;C:\\IBM\\Tivoli\
\Netcool\\omnibus\\gates\\java\\ngjava.jar;C:\\IBM\\Tivoli\\Netcool\\omnibus\
\java\\jars\\TransportModule.jar;C:\\IBM\\Tivoli\\Netcool\\omnibus\\java\\jars\
\Transformer.jar;C:\\IBM\\Tivoli\\Netcool\\omnibus\\java\\jars\
\com.ibm.ws.ejb.thinclient_8.5.0.jar;C:\\IBM\\Tivoli\\Netcool\\omnibus\\java\
\jars\\com.ibm.ws.sib.client.thin.jms_8.5.0.jar;C:\\IBM\\Tivoli\\Netcool\
\omnibus\\java\\jars\\JSON4J.jar'
```

If you are running the gateway on Linux or UNIX Operating Systems, this is taken care of by the `nco_g_xml.env` file.

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

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

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

Installing probes and gateways using the Command Line Tool

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

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

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

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

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

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

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

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

Uninstalling probes and gateways using the Command Line Tool

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

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

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

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

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

Installing probes and gateways using the GUI

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

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

   installer_path`/IBMIM`

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

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

13. When the **Install Packages** panel appears indicating that you have successfully installed the probe or gateway, click **Finish**.

Uninstalling probes and gateways using the GUI

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

1. Start Installation Manager using the following command:

   installer_path/IBMIM

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

Configuring the gateway server

You must create the Gateway for Message Bus server in the Tivoli Netcool/OMNIbus interfaces file and set the appropriate environment variables.

Note : You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode.

To create the Gateway for Message Bus server in the Tivoli Netcool/OMNIbus interfaces file:

- If you are running the gateway in standard mode, create a server named `G_XML`.
- If you are running the gateway in LA mode, create a server named `G_SCALA`.

For details about how to run the gateway, see "Running the gateway" on page 85.

Creating the gateway server

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

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

Setting environment variables

Ensure that the Java virtual machine (JVM) initialization library (`{JRE_DIR}/jre/bin/j9vm`) is in the library path appropriate for the operating system the gateway is installed on.

Configuring the gateway

The Gateway for Message Bus is configured using a set of configuration files.

The gateway uses centralized property management, which separates properties from data processing configuration. Use the properties file to configure properties and the map definition, table replication definition, and startup command files to configure data processing.

The following topics describe the configuration files and how to use them:

- "Properties file" on page 10
- "Map definition file" on page 27
- "Startup command file" on page 28

Properties file

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

Configuring the properties file

Before running the gateway for the first time, you must configure the installed properties file.

Note : You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode.

To configure the properties file on UNIX and Linux operating systems, follow these steps:

1. Copy the appropriate properties file as follows:
   a. If you are running the gateway in standard mode, copy the properties file `$OMNIHOME/gates/xml/G_XML.props` to `$OMNIHOME/etc`.
   b. If you are running the gateway in LA mode, copy the properties file `$OMNIHOME/gates/xml/scala/G_SCALA.props` to `$OMNIHOME/etc`.
2. Make a backup copy of the `G_XML.props` or `G_SCALA.props` properties file before making any edits.
3. Edit the properties in the `G_XML.props` or `G_SCALA.props` properties file to suit your operating environment.

To configure the properties file on Windows operating systems, follow these steps:

1. Copy the appropriate properties file as follows:
   a. If you are running the gateway in standard mode, copy the properties file `%OMNIHOME%\gates\xml\G_XML.props` to `%OMNIHOME%\etc`.
   b. If you are running the gateway in LA mode, copy the properties file `%OMNIHOME%\gates\xml\scala\G_SCALA.props` to `%OMNIHOME%\etc`.
2. Make a backup copy of the `G_XML.props` or `G_SCALA.props` properties file before making any edits.
3. Edit the properties in the `G_XML.props` or `G_SCALA.props` properties file to suit your operating environment.

Note : If you are running the gateway in standard mode, pay particular attention to the **`Gate.XMLGateway.MessageID`** property defined in the `$OMNIHOME/gates/xml/G_XML.props` file. The default value for this property is `netcoolEvents`, which causes all events from the ObjectServer to be sent using the transformer named `netcoolEvents` in the transformer definition file. You can set the **`Gate.XMLGateway.MessageID`** property to either the name of another transformer (and therefore to always use that transformer definition file to transform and send the events), or to the name of an ObjectServer field which will contain the identifier to be used in the transformer definition file. This is done using the following notation: `@FieldName`.

Properties and command line options

You use properties to define the operational environment of the gateway. You can override the default values by changing the values of these properties that are defined in the properties file.

The following tables describe the properties available with the Gateway for Message Bus. For more information about generic and Inter-Process Communication (IPC) properties and command line options, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

The following sections describe the standard properties used to configure the gateway:

Note : You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode. Each of these modes uses its own properties file.

Common Tivoli Netcool/OMNIbus properties

Table 4 on page 11 describes the Tivoli Netcool/OMNIbus common properties defined in the following properties files:

`G_XML.props` - For gateways operating in standard mode.
`G_SCALA.props` - For gateways operating in LA mode.

Table 4. Common Tivoli Netcool/OMNIbus properties

Property name	Command line option	Description
`ConfigCryptoAlg` *string*	N/A	Use this property to specify the encryption algorithm that the gateway uses. Use this property with the **`ConfigKeyFile`** property and the `nco_aes_crypt` utility that is supplied with Tivoli Netcool/OMNIbus. The default is AES. **Note :** You need to set this property to AES_FIPS if you are encrypting properties in one of the transport module properties files. **Note :** The use of the encryption functionality is optional. If you do not plan to use the encryption functionality, keep the default values associated with the **`ConfigCryptoAlg`** and **`ConfigKeyFile`** properties. If you do make use of the encryption functionality then you can encrypt the string valued properties in the gateway properties files and the transport module properties files.

Table 4. Common Tivoli Netcool/OMNIbus properties (continued)		
Property name	**Command line option**	**Description**
ConfigKeyFile *string*	N/A	Use this property to specify the encryption key that is used to encrypt data. Use this property with the **ConfigCryptoAlg** property and the `nco_aes_crypt` utility that is supplied with Tivoli Netcool/OMNIbus. The default is `""`.
Connections *integer*	`-connections` *integer*	Use this property to specify the maximum number of client connections that can be made to the gateway server. The default is `30`.
Help *boolean*	`-help` *boolean*	Use this property to instruct the gateway to display application help information on startup and exit. The default is `FALSE`.
MaxLogFileSize *integer*	`-maxlogfilesize` *integer*	Use this property to specify the size (in bytes) that the gateway allocates for the log file. When the log file reaches this size, the gateway renames the log file by appending the name with the characters `.old` and creates a new log file. The default is `1024`.
MessageLevel *string*	`-messagelevel` *string*	Use this property to specify the level of the log file messages. The default is `warn`.
MessageLog *string*	`-messagelog` *string*	Use this property to specify the path to the message log file. If you are running the gateway in standard mode, the default for this property is `$OMNIHOME/log/G_XML.log`. If you are running the gateway in LA mode, the default for this property is `$OMNIHOME/log/G_SCALA.log`.

Table 4. Common Tivoli Netcool/OMNIbus properties (continued)		
Property name	**Command line option**	**Description**
Name *string*	`-name` *string*	Use this property to specify the name of the current gateway instance. If you want to run multiple gateways on one machine, you must use a different name for each instance. If you are running the gateway in standard mode, set this property to `G_XML`. If you are running the gateway in LA mode, set this property to `G_SCALA`. You need to have created a server (using the Server Editor `nco_xigen`) in the Tivoli Netcool/OMNIbus interfaces file.
Props.CheckNames *boolean*	N/A	Use this property to instruct the gateway to shut down if any property in the properties file is set to an invalid value. The default is `TRUE`.
PropsFile *string*	`-propsfile` *string*	Use this property to specify the location and name of the gateway properties file. If you are running the gateway in standard mode, the default for this property is `$OMNIHOME/etc/G_XML.props`. If you are running the gateway in LA mode, the default for this property is `$OMNIHOME/etc/G_SCALA.props`.
UniqueLog *boolean*	`-uniquelog` *boolean*	Use this property to specify that log file names are made unique by adding the PID of the gateway to the file name. The default is `FALSE`.
Version *boolean*	`-version` *boolean*	Use this property to instruct the gateway to display information about the application version on startup and exit. The default is `FALSE`.

Common gateway properties

Table 5 on page 14 describes the Tivoli Netcool/OMNIbus common gateway properties defined in the following properties files:

`G_XML.props` - For gateways operating in standard mode.
`G_SCALA.props` - For gateways operating in LA mode.

Table 5. Common gateway properties

Property name	Command line option	Description
`Gate.CacheHashTblSize` *integer*	`-chashtblsize` *integer*	Use this property to specify the number of elements the gateway allocates for the hash table cache. The default is 5023.
Map definition file for gateways operating in standard mode		
`Gate.MapFile` *string*	`-mapfile` *string*	Use this property to specify the mapping file for the gateway to use. If you are running the gateway in standard mode, the default is `$OMNIHOME/gates/xml/xml.map`.
Map definition file for gateways operating in LA mode		
`Gate.MapFile` *string*	`-mapfile` *string*	Use this property to specify the mapping file for the gateway to use. See “Summary” on page 2 for the mapping file you should use based on which version of Network Manager Insight Pack and OMNIbus Insight Pack you are running the gateway with deployed against IBM Operations Analytics - Log Analysis.
`Gate.NGtkDebug` *boolean*	`-ngtkdebug` *boolean*	Use this property to enable the logging of NGTK library debug messages. The default is TRUE.
`Gate.PAAware` *integer*	`-paaware` *integer*	This property indicates whether the gateway is Process Agent (PA) aware. The default is `0` (not PA aware). **Note :** This property is maintained by the PA server and is included in the properties file for information only.

Table 5. Common gateway properties (continued)		
Property name	**Command line option**	**Description**
Gate.PAAwareName *string*	-paname *string*	This property indicates the name of the PA controlling the gateway. The default is "". **Note :** This property is maintained by the PA server and is included in the properties file for information only.
Gate.StartupCmdFile *string*	-startupcmdfile *string*	Use this property to specify the location of the startup command file. If you are running the gateway in standard mode, the default for this property is $OMNIHOME/gates/xml/xml.startup.cmd. If you are running the gateway in LA mode, the default is: $OMNIHOME/gates/xml/scala/xml.startup.cmd.
Gate.Transfer.FailoverSyncRate *integer*	-fsyncrate *integer*	Use this property to specify the rate (in seconds) of the failover synchronization. The default is 60.
Gate.UnixAdminGrp *string*	-unixadmingrp *string*	Use this property to specify the administration group to which the gateway must belong if standard UNIX authentication is used. The default is ncoadmin.
Gate.UsePamAuth *boolean*	-usepamauth *boolean*	Use this property to specify whether Pluggable Authentication Module (PAM) authentication is used. The default is FALSE. **Note :** To run the gateway in FIPS 140-2 mode, set this property to TRUE.

Java properties

Table 6 on page 16 describes the Tivoli Netcool/OMNIbus available Java properties defined in the following properties files:

- G_XML.props - For gateways operating in standard mode.
- G_SCALA.props - For gateways operating in LA mode.

Table 6. Java properties		
Property name	**Command line option**	**Description**
`Gate.Java.Arguments` *string*	`-javaargs` *string*	Use this property to specify the Java arguments. The default is `" "`. **Note :** You can specify more than one argument for the **`Gate.Java.Arguments`** property by specifying a space-separated list of arguments within single quote marks. For example: `'arg1 arg2 arg3'`.
`Gate.Java.ClassPath` *string*	`-javacpath` *string*	Use this property to specify the path to the third-party and user-defined Java classes that create the required Java virtual machine (JVM) environment. The default is `$CLASSPATH`.
`Gate.Java.Debug` *boolean*	`-javadebug` *boolean*	Use this property to specify whether the debugging option is enabled for the JVM environment. The default is TRUE.
`Gate.Java.Library Path` *string*	`-javalibpath` *string*	Use this property to specify the path to the library files for the required JVM environment. The default is `" "`.

Mapping properties

Table 7 on page 16 describes the Tivoli Netcool/OMNIbus available mapping properties defined in the following properties files:

`G_XML.props` - For gateways operating in standard mode.
`G_SCALA.props` - For gateways operating in LA mode.

Table 7. Mapping properties		
Property name	**Command line option**	**Description**
`Gate.Mapper.Debug` *boolean*	`-mapperdebug` *boolean*	Use this property to specify whether the gateway includes mapper debug messages in the debug log. The default is TRUE.

Table 7. Mapping properties (continued)		
Property name	**Command line option**	**Description**
Gate.Mapper.Forward HistoricDetails *boolean*	`-mapperforhistdtls` *boolean*	Use this property to specify whether the gateway forwards all historic details on converted update. The default is FALSE. **Note :** The Gateway for Message Bus cannot forward events from the alerts.journal or alerts.details tables.
Gate.Mapper.Forward HistoricJournals *boolean*	`-mapperforhistjrnl` *boolean*	Use this property to specify whether the gateway forwards all historic journals on converted update. The default is FALSE. **Note :** The Gateway for Message Bus cannot forward events from the alerts.journal or alerts.details tables.

Reader properties

Table 8 on page 17 describes the Tivoli Netcool/OMNIbus available gateway reader properties defined in the following properties files:

`G_XML.props` - For gateways operating in standard mode.
`G_SCALA.props` - For gateways operating in LA mode.

Table 8. Reader properties		
Property name	**Command line option**	**Description**
Gate.Reader.Debug *boolean*	`-readerdebug` *boolean*	Use this property to specify whether the gateway includes gateway reader debug messages in the debug log. The default is TRUE.
Gate.Reader.Description *string*	`-readerdescription` *string*	Use this property to specify the application description for the reader connection. This description is used in triggers and allows you to determine which component of the gateway attempted to perform an action. The default is `Gateway Reader`.

Table 8. Reader properties (continued)		
Property name	**Command line option**	**Description**
Gate.Reader.Details TableName *string*	`-readerdetailstblname` *string*	Use this property to specify the name of the details table that the gateway reads. The default is `alerts.details`. **Note :** The Gateway for Message Bus cannot forward events from the alerts.journal or alerts.details tables.
Gate.Reader.Failback Enabled *boolean*	`-readerfailbackenabled` *boolean*	Use this property to specify whether failback is enabled for the ObjectServer. The default is TRUE.
Gate.Reader.Failback Timeout *integer*	`-readerfailbacktimeout` *integer*	Use this property to specify the time (in seconds) that the gateway reader awaits before entering failback mode. The default is `30`.
Gate.Reader.IducFlushRate *integer*	`-readeriducflushrate` *integer*	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`. **Attention :** 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.

Table 8. Reader properties (continued)		
Property name	**Command line option**	**Description**
Gate.Reader.Journal TableName *string*	`-readerjournaltblname` *string*	Use this property to specify the name of the journal table that the gateway reads. The default is `alerts.journal`. **Note :** The Gateway for Message Bus cannot forward events from the alerts.journal or alerts.details tables.
Gate.Reader.LogOSSql *boolean*	`-readerlogossql` *boolean*	Use this property to specify whether the gateway logs all SQL commands sent to the ObjectServer in debug mode. The default is FALSE.
Gate.Reader.Password *string*	`-readerpassword` *string*	Use this property to specify the password associated with the user specified by the **Gate.Reader.Username** property. The default is `" "`. If the ObjectServer from which the gateway reads alerts is running on Tivoli Netcool/ OMNIbus V8.1 or above in FIPS 140-2 mode, this password must be either plain text or encrypted using the `nco_aes_crypt` utility. For details about the encryption utilities, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*.
Gate.Reader. ReconnectTimeout *string*	`-readerreconntimeout` *string*	Use this property to specify the time (in seconds) between each reconnection poll attempt that the gateway makes if the connection to the ObjectServer is lost. The default is 30.
Gate.Reader.Server *string*	`-readerserver` *string*	Use this property to specify the name of the ObjectServer from which the gateway reads alerts. The default is NCOMS.

Table 8. Reader properties (continued)		
Property name	**Command line option**	**Description**
Gate.Reader.StatusTableName *string*	`-readerstatustblname` *string*	Use this property to specify the name of the status table that the gateway reads. The default is `alerts.status`.
Gate.Reader.TblReplicateDefFile *string*	`-readertblrepdeffile` *string*	Use this property to specify the path to the table replication definition file. If you are running the gateway in standard mode, the default is `$OMNIHOME/gates/xml/xml.reader.tblrep.def`. If you are running the gateway in LA mode, the default is: `$OMNIHOME/xml/scala/xml.reader.tblrep.def`.
Gate.Reader.Username *string*	`-readerusername` *string*	Use this property to specify the user name that is used to authenticate the ObjectServer connection. The default is `root`.

Gateway-specific properties

Table 9 on page 21 describes the Tivoli Netcool/OMNIbus available gateway-specific properties defined in the following properties files:

`G_XML.props` - For gateways operating in standard mode.
`G_SCALA.props` - For gateways operating in standard mode.

Table 9. Gateway-specific properties

Property name	Command line option	Description
Gate.XMLGateway.DateFormat *string*	`-dateformat` *string*	Use this property to specify the format of UTC (date) columns in the constructed XML. If you are running the gateway in standard mode, the default is `yyyy-MM-dd\'T\'HH:mm:ss`. If you are running the gateway in LA mode, the default is `yyyy-MM-dd\'T\'HH:mm:ssZ`. The default is different because it needs to provide timezone details about the date and time to IBM Operations Analytics - Log Analysis. You may never need to change the value of this property. If you do need to, see the following JDK Documentation for details on how to define the date format: http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html **Note :** For gateways running in LA mode: Ensure that the format of this property matches the format of the date/time field in the data source properties file. The data source properties file is called `omnibus1100.properties`.
Gate.XMLGateway.Debug *boolean*	`-xmlgateway debug` *boolean*	Use this property to specify whether the debug log includes gateway debug messages. The default is `TRUE`.
Gate.XMLGateway.FailbackEnabled *boolean*	`xmlgatewayfailback enabled` *boolean*	Use this property to specify whether failback is enabled for the gateway. The default is `TRUE`.
Gate.XMLGateway.FailbackTimeout *string*	`xmlgatewayfail backtimeout` *string*	Use this property to specify the time (in seconds) that the gateway waits before entering failback mode. The default is `30`.

Table 9. Gateway-specific properties (continued)		
Property name	**Command line option**	**Description**
Gate.XMLGateway.ForwardKeyValuePairs *boolean*	-forwardkeyvaluepairs *boolean*	Use this property to enable the forwarding of Netcool key-value pairs to the Transport Module. For details about configuring the JMS Transport, see "The jmsTransport.properties file" on page 61.) For details about dynamic value assignment, see "Dynamic value assignment for jmsHeaderMap and jmsPropertyMap" on page 27 This property works in conjunction with **Gate.XMLGateway.ForwardMappedFields**. The default is FALSE.
Gate.XMLGateway.ForwardMappedFields *boolean*	-forwardmappedfields *boolean*	Use this property to specify the mapped fields defined in the gateway's map file to be added in the forwarded key-value pairs. This property takes effects only when **Gate.XMLGateway.ForwardKeyValuePairs** is set to TRUE. If forwarding key-value pairs is enabled, leaving the **Gate.XMLGateway.ForwardMappedFields** property blank causes all mapped field to be added to the key-value pairs. To reduce the memory footprint, specify only the mapped fields required by the relevant use case in the Transport Module. Default: "".
Gate.XMLGateway.MessageID *string*	-messageid *string*	Use this property to specify the message ID for the gateway. The message ID determines which transformation is required. The message ID will be either a specific transformation or @*col_name* (where col_name is a field within Tivoli Netcool/OMNIbus), which then uses the value of the column specified as the ID to decide which transformation is required. The default is netcoolEvents.

Table 9. Gateway-specific properties (continued)		
Property name	**Command line option**	**Description**
Gate.XMLGateway.MessageKey *string*	`-messagekey` *string*	Use this property to specify a static or a dynamic key to be used for every message that goes into Kafka transport (Kafka producer). A dynamic key is a real-time value sourced from one of the columns defined in the map file (`G_XML.map`). To use a dynamic key, the property value must start with `@`. For example, to enable a dynamic message key in a Kafka producer send operation, set **Gate.XMLGateway.MessageKey** to `@ServerSerial`. The message key effects the Kafka producer's send operation in relation to topic partitioning. For consistency of partitioned delivery, the dynamic key must be present in every IDUC fetch, and so the choice of message key must adhere to that requirement. When you specify the name of the column providing the key value using the **Gate.XMLGateway.MessageKey** property, events received through the Kafka transport will be forwarded to partitions as determined by the following algorithm: `Partition number=hash(key) % number of partitions` This method returns different partition numbers evenly when a larger number of partitions is used, and the events will be distributed more evenly among the partitions. The default is ''.
Gate.XMLGateway.MessageType *string*	`-messagetype` *string*	Use this property to specify whether the value defined by the **Gate.XMLGateway.MessageID** property is a transformer module name or transformer module endpoint. This property takes the following values: `NAME`: The value defined by the **Gate.XMLGateway.MessageID** property is a transformer module name. `ENDPOINT`: The value defined by the **Gate.XMLGateway.MessageID** property is a transformer module endpoint. The default is `NAME`.

Table 9. Gateway-specific properties (continued)

Property name	Command line option	Description

Gate.XMLGateway.PublishTraceOn *boolean*	`-pubtraceon` *boolean*	Use this property to specify whether the gateway prints the following event messages before and after transformation:
		`Log message:`
		`Transforming message -`
		`[<event_message>].`
		`Log message:`
		`Transformed message - [<event_message>].`
		The default is FALSE.

Gate.XMLGateway.ReconnectAttempts *integer*	`-reconnectattempts` *integer*	Use this property to specify the number of times that the gateway should attempt to reconnect to the target system. The default is 12.
Gate.XMLGateway.ReconnectDelay *integer*	`-reconnectdelay` *integer*	Use this property to specify the time (in seconds) between each reconnection attempt. The total delay is the specified value plus the connection timeout value. The default is 20.
Gate.XMLGateway.ReconnectTimeout *integer*	`xmlgateway reconntimeout` *integer*	Use this property to specify the time (in seconds) between each reconnection poll attempt if the gateway loses the connection to the ObjectServer. The default is 30.
Gate.XMLGateway.ReplaceXmlInvalidChars	`-replacexmlinvalidchars`	Use this property to enable the replacement of invalid characters found in Netcool data during the construction of XML event prior to the XSLT process. The invalid characters concerned refer to the control character sets in the XML standard, see https://www.w3.org/TR/xml/#charsets. Unconverted control characters in an XML document may trigger a Java exception in the XLST process. The replacement of each invalid character follows the format defined by the **Gate.XMLGateway.XmlInvalidCharsFormatter** property. The default is FALSE.

Table 9. Gateway-specific properties (continued)

Property name	Command line option	Description
Gate.XMLGateway.ThreadPoolSize *integer*	`-thrpoolsize` *integer*	Use this property to specify the number of threads in the XML transformation thread pool. Increasing the size of the thread pool allows more XML messages to be concurrently processed using the transformation system. The default is `10`.
Gate.XMLGateway.TransformerFile *string*	`-transformerfile` *string*	Use this property to specify the location of the transformer module file. If you are running the gateway in standard mode, the default is `$OMNIHOME/java/conf/transformers.xml`. If you are running the gateway in LA mode, the default is `$OMNIHOME/java/conf/scalaTransformers.xml`.
Gate.XMLGateway.TransformerInputType *string*	`-transformeritype` *string*	Use this property to specify whether the input format of the event into the transformer is `xml` or `json`. When `json` is used then the event will be passed into the transformer as JSON. This means that a transformer module that can handle JSON is used in place of the default XSLT transformer. The default is `xml`. **Note :** For details of using this property, see "Configuring the gateway to produce JSON data" on page 32.
Gate.XMLGateway.TransportFile *string*	`-transportfile` *string*	Use this property to specify the location of the transport module properties file. If you are running the gateway in standard mode, the default is `$OMNIHOME/java/conf/jmsTransport.properties`. If you are running the gateway in LA mode, the default is `$OMNIHOME/java/conf/scalaTransport.properties`.

Table 9. Gateway-specific properties (continued)

Property name	Command line option	Description
Gate.XMLGateway.TransportType *string*	`-transporttype` *string*	Use this property to specify either the transport module method to be used or the name of the transport module class (JMS, Data file, MQTT, HTTP and HTTPS, or Socket) to use. If you are running the gateway in standard mode, the default is `JMS`. You can also specify one of the following values if your gateway is running in standard mode: • **MQTT** • **HTTP** • **Socket** • **File** If you are running the gateway in LA mode, the default is `SCALA`.
Gate.XMLGateway.XmlInvalidCharsFormatter	`-xmlinvalidcharsformatter`	Specify the format for invalid character replacement. This property takes effects only when **Gate.XMLGateway.ReplaceXmlInvalidChars** is set to TRUE. The configured format must fulfill the following:criteria: • Adhere to Java print format syntax, otherwise a runtime exception will occur. • Typically, numeral format are recommended, such as `%d, %x, %X`. • White spaces are valid print format. To suppress invalid characters, apply empty configuration. The resultant Netcool data, embedded with tokens produced from the formatter, must be compliant to the input document standard (which is always XML) and the output document standard (which can be XML, JSON, or others) for the XSLT process. This requires extra care from the user. Format violation may escape detection within the gateway, for example when a string valid to the input document standard is invalid to the output document standard. For example, the `xml-to-json` transformation process does not validate the document produced (for performance sake). The outcome of this scenario is the receiver's rejection of the malformed event. The default is `[U+%04X]`

Dynamic value assignment for jmsHeaderMap and jmsPropertyMap

The requisite for using dynamic values is to set the **Gate.XMLGateway.ForwardKeyValuePairs** property to TRUE. When **Gate.XMLGateway.ForwardKeyValuePairs** is not empty, the specific mapped field name must be in the configuration.

The range of data source for the dynamic value is the mapped fields (on the left side of the equal sign) defined in the gateway's replication map. The set of Netcool data differs for INSERT, UPDATE and DELETE. For INSERT and UPDATE, data availability is controlled by the condition clause, for example ON INSERT ONLY. DELETE event contains only the data of deletion timestamp ServerName and ServerSerial.

In the following example gateway map file, MyNode is a mapped field with a customized name which is different from the origin table column.

```
CREATE MAPPING StatusMap
(
        'Identifier' =    '@Identifier'           ON INSERT ONLY,
        'MyNode'     =    '@Node'                 ON INSERT ONLY,
        'NodeAlias'  =    '@NodeAlias'            ON INSERT ONLY
);
```

Map definition file

The mapping definition file contains mappings that define how the gateway maps data received from the ObjectServer to the destination data collecting application.

Note : You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode.

You can configure the mapping functions of the gateway by using the mapper properties defined in one of the following properties files:

G_XML.props - For gateways operating in standard mode.
G_SCALA - For gateways running in LA mode.

For details of these properties, see "Mapping properties" on page 16.

Running the gateway in standard mode

If you are running the gateway in standard mode, the default map definition file, xml.map, is located in the following directory:

$OMNIHOME/gates/xml

Running the gateway in LA mode

If you are running the gateway with OMNIbus Insight Pack Version 1.1.0.x and higher deployed against IBM Operations Analytics - Log Analysis, use the xml.map definition file, which is located in the following directory:

$OMNIHOME/gates/xml/scala

If you are running the gateway with OMNIbus Insight Pack Version 1.3.0.0 and Version 1.3.0.1 and higher and Network Manager Insight Pack Version 1.3.0.0 and higher deployed against IBM Operations Analytics - Log Analysis, use the xml1300.map definition file, which is located in the following directory:

$OMNIHOME/gates/xml/scala

If you are running the gateway with OMNIbus Insight Pack Version 1.3.0.2 and higher and Network Manager Insight Pack Version 1.3.0.0 and higher deployed against IBM Operations Analytics - Log Analysis, use the xml1302.map definition file, which is located in the following directory:

$OMNIHOME/gates/xml/scala

The order of the columns in the map definition file is important. It must match the order of the columns in the `OMNIbus1100.properties` file in the OMNIbus Insight Pack for IBM Operations Analytics - Log Analysis. You may need to edit the gateway map file to ensure that the order of the columns does match.

When running the gateway in LA mode, the gateway uses several lookup tables to provide easy conversion from a numeric value to a string for the numeric columns used in the map definition file. As a result, the event data ingested by IBM Operations Analytics - Log Analysis contain string values only; this allows you to search against string values directly and makes the search results easier to understand. For example, you can search for `Severity=Critical` instead of `Severity=5`.

The contents of the lookup tables should be consistent with the conversions in the ObjectServer. If you make any change to the conversions in the ObjectServer, you must update the gateway map file to keep them identical. Changes or additions to the lookup table will take effect only after the gateway is stopped and restarted.

Information flow

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

1. The reader component reads from the source ObjectServer and passes the data to the mapper component.
2. The mapper component transforms the data it receives into an appropriate form for the target writer using the map definition file. It then passes the data to the writer component.
3. The writer component writes the data it receives to the destination data collecting application.

Startup command file

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

Note : You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode.

You can specify the location of the startup command file using the Tivoli Netcool/OMNIbus **`Gate.StartupCmdFile`** property, which is defined in the `G_XML.props` file for gateways operating in standard mode and `G_SCALA.props` file for gateways operating in LA mode. The default startup command file `xml.startup.cmd` is located in the following directory:

`$OMNIHOME/gates/xml` - For gateways operating in standard mode.
`$OMNIHOME/gates/xml/scala` - For gateways operating in LA mode.

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

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

- `GET PROPS | SHOW CONFIG` - Use these commands to display the current configuration of the gateway by listing all properties and their values.
- `GET PROPERTY '`*`property`*`'` - Use this command to display the gateway property value, where *property* is the property value you want to query.
- `SET PROPERTY '`*`prop`*`' TO ( '`*`string`*`' | `*`integer`*` | TRUE | YES | FALSE | NO );` - Use this command to specify the value of a gateway property in the properties file.
- `SET LOG LEVEL TO` - Use this command to set the level of message logging. This command can take the following values: `FATAL, ERROR, WARNING, INFORMATION` or `DEBUG`. The default is `DEBUG`.

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

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

Table replication definition file

The gateway replicates data between ObjectServer tables and the gateway target application. 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.

Note : You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode.

You can specify the location of the table replication definition file using the Tivoli Netcool/OMNIbus **Gate.Reader.TblReplicateDefFile** property, which is defined in the `G_XML.props` file for gateways operating in standard mode and `G_SCALA.props` file for gateways operating in LA mode. The default name for table replication definition file is `xml.reader.tblrep.def` and is located in the following directory:

`$OMNIHOME/gates/xml` - For gateways operating in standard mode.
`$OMNIHOME/gates/xml/scala` - For gateways operating in LA mode.

If you are running the gateway using a transport module, the default table replication definition file is located in the following directory:

`$OMNIHOME/gates/xml`

If you are running the gateway with IBM Operations Analytics - Log Analysis, the default table replication definition file is located in the following directory:

`$OMNIHOME/gates/xml/scala`

Running the gateway in standard mode

If you are running the gateway in standard mode, the default table replication definition file contains example commands. You should make a backup copy of the default table replication definition file for future reference.

Note : You should use the `REPLICATE` command to replicate data from the primary tables (`alert.status`, `alert.journal`, `alert.details`) and dynamic secondary tables.

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 10. Optional replication commands	
Command	**Description**
FILTER WITH '*filter*'	Filters the database rows selected for replication, where *filter* defines the filter that the gateway uses in the WHERE clause of the SQL SELECT. Filtering is positive by default, which means that only those events that match the filter definition are replicated. You can use a negative filter by putting an exclamation mark (!) before the equals sign (=) in the filter clause. For example, the following filter clause replicates all events whose severity is not 5: FILTER WITH 'Severity !=5'
INTO '*targettable*'	This specifies the table in the target database to which data will be replicated, where *targettable* is the name of the target table. **Note :** The INTO clause can only be used when the target is a table. **Note :** This clause is not required when replicating the main alerts.status, alerts.details, and alerts.journal ObjectServer tables. Even though these tables are dynamic, these primary tables have specific properties to specify the name of the target database table for storing their data. See the following properties in the properties file supplied with the gateway. • Gate.*gw_name*.DetailsTableName • Gate.*gw_name*.JournalTableName • Gate.*gw_name*.StatusTableName Where the value for *gw_name* is found in the gateway guide summary table.
ORDER BY '*order*'	Order results by the SQL SELECT ORDER BY clause used to get data. A potential use case might be to order by first occurrence, so that alerts are processed in chronological order, in which case the value specified for *order* would be 'FirstOccurrence'.
AFTER IDUC DO '*afteriduc*'	Updates replicated rows, where *afteriduc* specifies which field to update with what value. This uses the SQL UPDATE action to execute on rows retrieved by the SQL SELECT action used to get data, for example, 'SentToCRM=1'.

Running the gateway in LA mode

You can configure the gateway to replicate with LA using the UPDATE, INSERT, or FT_INSERT table replication definition settings. FT_INSERT uses the Accelerated Event Notification (AEN) fast track channel to send events from an ObjectServer to the Message Bus Gateway; this is the recommended replication setting when configuring the Message Bus Gateway to send events to LA.

If you set the gateway up for AEN, it requires triggers to be activated on the ObjectServer and the FT_INSERT replication settings to be specified. For details about how to set up this configuration, see "Configuring event forwarding using AEN" on page 34.

For details of the best practices for Log Analysis deployment scenarios, see Deployment scenarios. For further details about monitoring accelarated events, see Monitoring accelerated events.

Message log

The gateway creates a message log file to store all messages that it generates while running. You can use properties in the properties file and environment variables to specify the name of the message log file.

To control the file name of the message log file, you use the **MessageLog** property. For example, if the message log file is called *logfilename*.log, then *logfilename*.log is the value that is specified in the property.

Preventing unbound reads from files or sockets being exploited in a denial of service attack

To prevent unbound reads from files or sockets being exploited in a denial of service attack, use the **Gate.Java.Arguments** property with the following Java arguments:

`-Dcom.ibm.csi.netcool.integrations.max_line_length`: This Java argument specifies the maximum amount of data that the gateway attempts to read from the socket or file at any time.

`-Dcom.ibm.csi.netcool.integrations.transporter.read_timeout`: This Java argument specifies the timeout period for the gateway when reading on the socket.

The effect of these Java arguments is to make reading from files and sockets bounded in terms of the amount of data that can be read at one time (`max_line_length`) and the amount of time that can be spent waiting for data to be provided (`read_timeout`) where appropriate. You set these Java arguments using the **Gate.Java.Arguments** property in the gateway properties file.

To configure the `-Dcom.ibm.csi.netcool.integrations.max_line_length` Java argument, set the **Gate.Java.Arguments** property as follows:

```
Gate.Java.Arguments  :  '-Dcom.ibm.csi.netcool.integrations.max_line_length=xxxxx'
```

Where *xxxxx* is a number of bytes. The default is 1048576 (1MB).

To configure the `-Dcom.ibm.csi.netcool.integrations.max_line_length` Java argument, set the **Gate.Java.Arguments** property as follows:

```
Gate.Java.Arguments  :
       '-Dcom.ibm.csi.netcool.integrations.transporter.read_timeout=xxxxx'
```

Where *xxxxx* is the length of the timeout period in seconds. The default is 30 seconds.

Note : You can specify more than one Java argument for the **Gate.Java.Arguments** property by specifying a space-separated list of Java arguments within single quote marks. For example: '*arg1 arg2 arg3*'.

If you are using the transport module with IBM Operations Analytics - Log Analysis, you do not need to set `read_timeout` using a Java argument. You can instead use the **readTimeout** property in the `scalaTransport.properties` file.

Advanced properties

The gateway provides Error handling, Process Agent control, and Authentication features.

Error handling

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

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

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

Process Agent control

You can control how the gateway runs by using 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. Do not change these properties manually.

Authentication

Use the **Gate.UsePamAuth** property to specify how the gateway authenticates users.

Either standard UNIX authentication or PAM authentication can be used with the ObjectServer gateway. By default, the gateway uses standard UNIX authentication. To use PAM authentication, set the **Gate.UsePamAuth** property to TRUE.

Configuring the gateway to produce JSON data

To configure the gateway to produce JSON data, use the following steps:

1. Edit the gateway configuration in the following gateway properties file:

   `$OMNIHOME/gates/<arch>/G_XML.props`

   Where *<arch>* is the architecture directory, for example `linux2x86`.
2. Set **Gate.XMLGateway.TransformerInputType** to `json`:

   ```
   Gate.XMLGateway.TransformerFile      : '$OMNIHOME/java/conf/transformers.xml'
   Gate.XMLGateway.TransformerInputType : 'json'
   ```
3. Make the following configuration settings in the transformer file:

   Configure the host and port to be consistent with the host and port in the `httpTransport.properties` file.

   ```
   <tns:transformer name="netcoolEvents" type="northbound" endpoint="http://
   host:port/"
   className="com.ibm.tivoli.netcool.integrations.transformer.EmptyTransformer">
   </tns:transformer>
   ```

   If Message Bus Probe is run as the consumer for JSON formatted events sent from this gateway, the probe's transformer file requires the following configuration:

   ```
   <tns:transformer name="netcool2nvpairs" type="southbound" endpoint="http://
   host:port/"
   ```

```
className="com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer">
    <tns:property name="xsltFilename" type="java.lang.String" value="${OMNIHOME}/
java/conf/netcool2nvpairs.xsl" description="XSLT file for converting Netcool
events to name/value pairs"/>
</tns:transformer>
```

4. Configure the transport properties.
   a. For the HTTP transport, edit the following transport properties file:

      `$OMNIHOME/java/conf/httpTransport.properties`

      The following configuration is the minimum requirement when batch mode is ON with the JSON payload. By default, batch mode is ON with `bufferSize=10`. To disable batch mode, set `bufferSize=1`. All other property values in the default `httptransport.properties` file already include the minimum requirement for single JSON event functionality:

      ```
      bufferSize=15
      bufferFlushTime=30
      batchHeader=[
      batchFooter=]
      batchSeparator=,
      ```

   b. For the File transport, edit the following transport properties file:

      `$OMNIHOME/java/conf/filetransport.properties`

      In this file, the client should also be set to read from the JSON file:

      ```
      jsonFilename=${OMNIHOME}/var/jsonAlarm.txt
      ```

Note : The configuration in all other transports remains unchanged.

Integrating with IBM Operations Analytics - Log Analysis

The Gateway for Message Bus includes a separate set of configuration files for operating with IBM Operations Analytics - Log Analysis.

To run the gateway with IBM Operations Analytics - Log Analysis, use the following steps:

1. Create a gateway server named G_SCALA in the Tivoli Netcool/OMNIbus interfaces file. See “Configuring the gateway server” on page 9.
2. Copy the default properties file (`G_SCALA.props`) supplied with the gateway to the following location: `$OMNIHOME/etc`
3. Set the following properties as shown:
   - Set the **Name** property to G_SCALA.
   - Set the **PropsFile** property to `$OMNIHOME/etc/G_SCALA.props`.
4. Set the **Gate.MapFile** property to the map file that corresponds to the version of Network Manager Insight Pack you are running in IBM Operations Analytics - Log Analysis.

   Note : The **Gate.MapFile** property specifies a map file that configures the gateway to send data required by the `OMNIbus1100` datasource. If your gateway is running with Network Manager Insight Pack Version 1.1.0.x and higher that is deployed against IBM Operations Analytics - Log Analysis, use the `xml.map` file. Otherwise, if your gateway is running with OMNIbus Insight Pack Version 1.3.0.0 and Version 1.3.0.1 and higher and Network Manager Insight Pack Version 1.3.0.0 and higher that is deployed against IBM Operations Analytics - Log Analysis, use the `xml1300.map` file. By default, the `G_SCALA.props` file supplied with the gateway specifies the `xml.map` file.

   Otherwise, if your gateway is running with OMNIbus Insight Pack Version 1.3.0.2 and higher and Network Manager Insight Pack Version 1.3.0.0 and higher that is deployed against IBM Operations Analytics - Log Analysis, use the `xml1302.map` file.

   To specify the `xml1300.map` or `xml1302.map` file:

   a. Remove the comment (*#*) from the following line (depending on which map file you are using):

```
Gate.MapFile : '$OMNIHOME/gates/xml/scala/xml1300.map'

Gate.MapFile : '$OMNIHOME/gates/xml/scala/xml1302.map'
```

b. Add a comment (#) to the following line:

```
#Gate.MapFile : '$OMNIHOME/gates/xml/scala/xml.map'
```

5. Change the endpoint specified in the `scalaTransformers.xml` file to the URL of your data collecting application.
6. Configure the SSL connection between the gateway and the data collecting application associated with the IBM Operations Analytics - Log Analysis. See "Configuring SSL connections between the gateway and IBM Operations Analytics - Log Analysis" on page 52.
7. Edit the properties in the `scalaTransport.properties` file to suit your environment. See "Configuring transport module properties files" on page 61.
8. Run the gateway using the following command:

   ```
   $OMNIHOME/bin/nco_g_xml -propsfile $OMNIHOME/etc/G_SCALA.props
   ```

   .
9. Troubleshoot any problems that occur. See "Troubleshooting the IBM Operations Analytics - Log Analysis integration" on page 38.

   Note : By default, the `NmosObjInst` column is defined in the ObjectServer alerts.status table and is populated by Network Manager. The enriched value is required by the Network Manager Insight Pack Version 1.3.0.0 and higher to perform end to end searches. The OMNIbus Insight Pack Version 1.3.0.0 and higher and Network Manager Insight Pack Version 1.3.0.0 and higher share the same data source definitions. Therefore, the OMNIbus Insight Pack Version 1.3.0.0 and higher requires a value for the `NmosObjInst` column in the alerts.status table even though the value is not used.

   By default, the `NmosObjInst` column is installed in the ObjectServer schema for the alerts.status table. If the `NmosObjInst` column is not deleted from the alerts.status table, the default `xml1300.map` definition file can be used. However, if the `NmosObjInst` column has been deleted from the alerts.status table, you need to either add an integer column called `NmostObjInst` to the alerts.status table or modify the `xml1300.map` definition file. See "Failed to find column NmosObjInst" on page 39 for instructions on how to modify the `xml1300.map` definition file if you choose this option.

Configuring event forwarding using AEN

You can configure the Tivoli Netcool/OMNIbus ObjectServer to send events to a Message Bus gateway running in IBM Operations Analytics - Log Analysis (LA) mode by using the Accelerated Event Notification (AEN) system. The AEN system is generally used to accelerate events that could present a risk in the managed environment through ObjectServers in a tiered deployment or to the AEN pop-up client.

Overview

When running the gateway in LA mode, the AEN system is used to send each individual event that is received by the ObjectServer to IBM Operations Analytics - Log Analysis. In effect, this prevents the roll-up of multiple events into a single instance, which occurs as part of the normal IDUC system. For the purpose of analytics, IBM Operations Analytics - Log Analysis must see all event instances that occur in the environment. Although it is not required that you configure event forwarding using AEN when running in LA mode, it is recommended that you do so if you want to track changes in the log analysis tool.

For details about creating an AEN channel within Tivoli Netcool/OMNIbus, see the IBM Tivoli Netcool/OMNIbus Knowledge Center: http://www-01.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/common/kc_welcome-444.html?lang=en

When running the gateway in LA mode, the gateway is configured for the acceptance of event inserts from either IDUC or AEN.

The steps that you follow to configure event forwarding using AEN depend on the following:

Which version of OMNIbus Insight Pack you are running with the gateway.
Which version of the ObjectServer (and, by extension, the version of Tivoli Netcool/OMNIbus) you are using.
Which version of Network Manager Insight Pack you are using.

Note : These versions only apply if you are running the gateway in LA mode and you are configuring event forwarding using AEN.

Table 11 on page 35 provides a matrix to help you determine which steps to follow to configure event forwarding using AEN.

Table 11. Configuring event forwarding matrix

OMNIbus Insight Pack Version	ObjectServer Version	Network Manager Insight Pack Version	For configuration steps, see
Version 1.3.0, 1.3.0.0, 1.3.0.1	Version 8.1 (specifically, Tivoli Netcool/OMNIbus Version 8.1 Fix Pack 2)	Version 1.3.0, 1.3.0.0, 1.3.0.1	"Configuring event forwarding - OMNIbus Insight Pack Versions 1.3.0, 1.3.0.0, and 1.3.0.1" on page 35
Version 1.3.0.2	Version 8.1 (specifically, Tivoli Netcool/OMNIbus Version 8.1 Fix Pack 5 and higher)	Version 1.3.0.0	"Configuring event forwarding - OMNIbus Insight Pack Version 1.3.0.2 and higher" on page 36

Note : If the Network Manager Insight Pack is installed into your IBM Operations Analytics - Log Analysis and you are running Network Manager IP Edition, apply the `$NCHOME/omnibus/extensions/scala_itnm_configuration.sql` file to the ObjectServer. Note that the correct version of the `scala_itnm_configuration.sql` file is installed with the ObjectServer fix pack as listed in Table 11 on page 35. The purpose of installing these fix packs is to defer sending events to IBM Operations Analytics - Log Analysis until these events have been enriched by Network Manager IP Edition.

Specifically, the Tivoli Netcool/OMNIbus fix packs identified in Table 11 on page 35 provide updates to the AEN triggers, which delay sending an event to the ObjectServer until Network Manager can enrich these events. It is recommended that you install the ObjectServer fix pack as listed in Table 11 on page 35 if the Network Manager Insight Pack is installed into your IBM Operations Analytics - Log Analysis. Thus, you will be able to use the end to end search functionality that the Network Manager Insight Pack provides to ensure that events in IBM Operations Analytics - Log Analysis have the required information to search on. You install the updates to the AEN triggers by running the following SQL file:

```
/space/mahes/IBM/tivoli/netcool/omnibus/extensions/
scala/scala_itnm_configuration.sql
```

For more information, see *IBM Netcool Operations Insight V1.3 Integrations Guide*.

Configuring event forwarding - OMNIbus Insight Pack Versions 1.3.0, 1.3.0.0, and 1.3.0.1

To configure the gateway for event forwarding using AEN, use the following steps:

1. Edit the `G_SCALA.props` file and add the property definition shown below:

   ```
   Gate.Reader.Description : 'SCALA Gateway Reader'
   ```

   This ensures that the gateway reader will now identify itself to the ObjectServer specifically as a Gateway Reader interfacing with IBM Operations Analytics - Log Analysis.

2. Edit the gateway table replication definition file:

   $OMNIHOME/gates/xml/scala/xml.reader.tblref.def

   Change the alerts.status replication line to:

```
    REPLICATE FT_INSERT FROM TABLE 'alerts.status'
        USING MAP 'StatusMap';
```

3. Create an AEN channel called scala using the Tivoli Netcool/OMNIbus Administration console.

   a. On the **Columns** tab, add just the Serial column as the selected column list for the alerts.status table only.

   b. On the **Recipients** tab, add a channel recipient that sets the Application Description to SCALA Gateway Reader. Leave all other elements blank.

4. **Note :** This step is not required if you are running Tivoli Netcool/OMNIbus version 8.1. By default, the triggers are installed into the ObjectServer in Tivoli Netcool/OMNIbus version 8.1 and are no longer in the $OMNIHOME/extensions/ directory.

   Create and install the triggers in the ObjectServer. It is best to create a new trigger group called scala_triggers, so that triggers specific to the gateway running with IBM Operations Analytics - Log Analysis can be placed in this group and disabled and enabled as required.

   The following code shows the two basic triggers that are needed to send all new inserts using the event fast track notifications to the gateway. You can create these using either nco_sql or the Netcool/OMNIbus Administration GUI.

```
create or replace trigger group scala_triggers;

create or replace trigger scala_insert
group scala_triggers
priority 2
comment 'Fast Track an event insert for alerts.status to SCALA Gateway'
after insert on alerts.status
for each row
begin
    iduc evtft 'scala', insert, new
end;

create or replace trigger scala_reinsert
group scala_triggers
priority 2
comment 'Fast Track an event insert for alerts.status to SCALA Gateway'
after reinsert on alerts.status
for each row
begin
    iduc evtft 'scala', insert, new

end;
```

The event can now be forwarded using an AEN channel.

Note : As an alternative, Steps 3 and 4 can be achieved by running the following SQL file:

$OMNIHOME/extensions/scala/scala_configuration.sql

Configuring event forwarding - OMNIbus Insight Pack Version 1.3.0.2 and higher

The Version 1.3.0.2 OMNIbus Insight Pack contains a new field PubType, which allows you to differentiate between inserts and updates. For this feature to work correctly, the scala_reinsert AEN trigger needs to be configured to send an update event rather than an insert event. The SCALA triggers have been updated in Tivoli Netcool/OMNIbus Version 8.1 Fix Pack 5 to work with OMNIbus Insight Pack Version 1.3.0.2. If you have an existing ObjectServer instance and are installing OMNIbus Insight Pack Version 1.3.0.2, you will need to run upgrade scripts provided in Tivoli Netcool/OMNIbus Version 8.1 Fix Pack 5 to modify the existing triggers where indicated in the instructions. If you are creating a new ObjectServer instance and have installed Tivoli Netcool/OMNIbus, it is recommended that you install OMNIbus Insight Pack Version 1.3.0.2.

Specifically, Tivoli Netcool/OMNIbus 8.1 Fix Pack 2 and above provide updates to the AEN triggers, which delay sending an event to the ObjectServer until Network Manager IP Edition can enrich these events. The triggers in Tivoli Netcool/OMNIbus 8.1 Fix Pack 2 and above are compatible with OMNIbus Insight Pack Version 1.3.0.0 and Version 1.3.0.2. The triggers in Tivoli Netcool/OMNIbus 8.1 Fix Pack 5 are compatible with OMNIbus Insight Pack Version 1.3.0.2.

It is recommended that you install the relevant Tivoli Netcool/OMNIbus Version 8.1 Fix Pack if the Network Manager Insight Pack is installed into your IBM Operations Analytics - Log Analysis. Thus, you will be able to use the end to end search functionality that the Network Manager Insight Pack provides to ensure that events in IBM Operations Analytics - Log Analysis have the required information to search on.

You install the updates to the AEN triggers by running the following SQL file:

```
$OMNIHOME/extensions/scala/scala_itnm_configuration.sql
```

To configure the gateway for event forwarding using AEN, use the following steps:

1. Edit the gateway table replication definition file:

   $OMNIHOME/gates/xml/scala/xml.reader.tblref.def

   Change the alerts.status replication line to:

   ```
   REPLICATE FT_INSERT,FT_UPDATE FROM TABLE 'alerts.status'
           USING MAP 'StatusMap';
   ```

2. If you are not using the Network Manager Insight Pack, enable the scala_insert and scala_reinsert triggers in the scala_triggers group, which are provided by default.

   Note : If your ObjectServer instance was created before the Tivoli Netcool/OMNIbus Version 8.1 Fix Pack 5 was applied, you will need to run the update scripts provided in $OMNIHOME/etc/ to update the default triggers to the latest level.

3. If you are using the Network Manager Insight Pack, install the updates to the AEN triggers by running the following SQL file:

   ```
   $OMNIHOME/extensions/scala/scala_itnm_configuration.sql
   ```

4. Specify the following triggers using either nco_sql or the Netcool/OMNIbus Administration GUI:

   ```
   create or replace trigger scala_reinsert group scala_triggers
   priority 2 comment 'Fast Track an event insert for alerts.status to
   SCALA Gateway' after reinsert on alerts.status for each row begin
   iduc evtft 'scala', update, new end;
   ```

Changing the StatusMap in the xml1300.map file

When using AEN to send de-duplicated events you should modify the first field in the StatusMap defined in the xml.map, xml1300.map, or xml1302.map definition file. These map definition files reside in the following directory:

$OMNIHOME/gates/xml/scala

Note : You use the xml.map file if your gateway is running with Network Manager Insight Pack Version 1.3.0 and higher deployed against IBM Operations Analytics - Log Analysis. You use the xml1300.map file if your gateway is running with OMNIbus Insight Pack Version 1.3.0.0 and Version 1.3.0.1 and higher and Network Manager Insight Pack Version 1.3.0.0 and higher deployed against IBM Operations Analytics - Log Analysis. You use the xml1302.map file if your gateway is running with OMNIbus Insight Pack Version 1.3.0.2 and higher and Network Manager Insight Pack Version 1.3.0.0 and higher deployed against IBM Operations Analytics - Log Analysis.

Replace **FirstOccurrence** to **LastOccurrence**, as in the following example:

```
CREATE MAPPING StatusMap('LastOccurrence'  =  '@LastOccurrence',
                         'Summary'         =    '@Summary',
.
```

```
.
.
```

```
CREATE MAPPING StatusMap('LastOccurrence'  =  '@LastOccurrence',
                         'Summary'          =    '@Summary',
.
.
.
```

This change in field name means that IBM Operations Analytics - Log Analysis records the timestamp of each event that it receives as the time that event occurred, rather than when the first instance of that event occurred. This change in field name also ensures that deduplicated events have unique timestamps.

Troubleshooting the IBM Operations Analytics - Log Analysis integration

This topic describes common error messages that you may encounter when you start the IBM Operations Analytics - Log Analysis integration for the first time. It also includes a description of the error messages that the IBM Operations Analytics - Log Analysis transport module generates.

Error 1 in gateway log file

```
2013-11-06T09:28:17: Error: E-GJA-000-000: [ngjava]: XMLGateway:
[THREAD=29]: Unexpected communication failure to data collector
'https://localhost:9987/Unity/DataCollector' detected.
Exception Msg = [java.security.NoSuchAlgorithmException:
SSLContext Default implementation not found: ],
Exception Type = [java.net.SocketException]
```

Cause

The IBM Operations Analytics - Log Analysis keystore or truststore file does not exist or its path is not specified correctly.

Solution

Check that the value of the **keyStore** property and the **trustStore** property in the `$OMNIHOME/java/conf/scalaTransport.properties` file are set to valid keystore/truststore files. On Windows platforms, the delimiter in the path should be escaped, for example: `keyStore=C:\\IBM\\Tivoli\\Netcool\\omnibus\\java\\security\\client.jks`.

Error 2 in gateway log file

```
2013-09-06T12:22:42: Error: E-GJA-000-000: [ngjava]: XMLGateway: [THREAD=23]:
Failed to send batch to data collector. HTTP response code != 200. [{
responseCode:404
responseMessage:Not Found responseException:null authenticationRequired:false
cookieJar:null responseContent:
{&amp;quot;BATCH_STATUS&amp;quot;:&amp;quot;NONE&amp;quot;,&amp;quot;RESPONSE_MESSAGE&amp;quot;:
&amp;quot;CTGLA0401E :
Missing data source &amp;quot;,&amp;quot;RESPONSE_CODE&amp;quot;:404}
```

Cause

The data source does not match the one configured in IBM Operations Analytics - Log Analysis.

Solution

Check the values of the **jsonMsgHostname** and **jsonMsgPath** properties in the `$OMNIHOME/java/conf/scalaTransport.properties` file. These values should match the values set for the host name and path of the data source configured in IBM Operations Analytics - Log Analysis administrative settings.

Error 3 in gateway log file

```
2013-11-05T14:59:09: Error: E-GJA-000-000: [ngjava]: XMLGateway: [THREAD=24]:
Unexpected communication failure to data collector
'https://localhost:9987/Unity/DataCollector' detected.
```

```
Exception Msg = [com.ibm.jsse2.util.j: PKIX path building failed:
java.security.cert.CertPathBuilderException:
unable to find valid certification path to requested target],
Exception Type = [javax.net.ssl.SSLHandshakeException]
```

Cause

The IBM Operations Analytics - Log Analysis host certificate is not available in the `trustStore`.

Solution

Check that the value of **`trustStore`** property in the `$OMNIHOME/java/conf/scalaTransport.properties` file is set correctly. If it is, import the IBM Operations Analytics - Log Analysis host certificate to the file specified by the **`trustStore`** property.

Error 4 in gateway log file

```
2013-11-05T14:59:09: Error: E-GJA-000-000: [ngjava]: XMLGateway: [THREAD=24]:
Unexpected communication failure to data collector
'https://localhost:9987/Unity/DataCollector' detected.
Exception Msg = [com.ibm.jsse2.util.j: PKIX path building failed:
java.security.cert.CertPathValidatorException:
The certificate expire at Tue Dec 04 18:59:23 GMT 2012],
Exception Type = [javax.net.ssl.SSLHandshakeException]
```

Cause

The IBM Operations Analytics - Log Analysis host certificate in the trust store has expired.

Solution

Check that the value of the **`trustStore`** property in the `$OMNIHOME/java/conf/scalaTransport.properties` file is set correctly. If it is, import the new IBM Operations Analytics - Log Analysis host certificate to the file specified by the **`trustStore`** property.

The SSL connection could fail if the server side certificate expires or if the IBM Operations Analytics - Log Analysis data collector has been upgraded.

If you need to renew the server side certificate, you must recreate the certificate and import it into the keystore file. Similarly, if you upgrade the IBM Operations Analytics - Log Analysis data collector, you must create a new certificate and import it into the keystore file. For details see “Configuring SSL connections between the gateway and IBM Operations Analytics - Log Analysis” on page 52.

Failed to find column NmosObjInst

The gateway fails to start up with the following error:

```
2015-02-20T08:20:23: Error: E-GOB-102-088: [ngobjserv]: Reader: Failed to find
column 'NmosObjInst' that is specified in map 'StatusMap' for the source table
'alerts.status'. (0:No error)
```

Cause

This error occurs if you are using one of the following map definition files:

- `xml1300.map` definition file to connect to an OMNIbus Insight Pack 1.3.0.0 and higher
- `xml1302.map` definition file to connect to an OMNIbus Insight Pack Version 1.3.0.2 and higher

The error occurs when the `NmostObjInst` column has been removed from the alerts.status table in the ObjectServer.

Solution

To resolve this issue, either add an integer column called `NmostObjInst` to the alerts.status table or modify the `xml1300.map` or `xml1302.map` definition file. Follow these steps if you choose to modify the `xml1300.map` or `xml1302.map` definition file:

1. Locate the `xml1300.map` or `xml1302.map` definition file in the following directory:

```
$OMNIHOME/gates/xml/scala
```

2. Make a backup copy of the `xml1300.map` or `xml1302.map` definition file.
3. Open the `xml1300.map` or `xml1302.map` definition file for editing.
4. Locate the `StatusMap` section and, specifically, the `NmostObjInst` field.
5. Change the value of the `NmostObjInst` field from `'NmosObjInst' = '@NmosObjInst',` to `'NmosObjInst' = '',`.

   Note : Do not remove the `NmostObjInst` field entirely or the gateway will fail to send data to the ObjectServer.
6. Save the edits and exit the `xml1300.map` or `xml1302.map` definition file.
7. Restart the gateway to pick up the changes.

IBM Operations Analytics - Log Analysis transport error messages

The following table describes the error messages that are generated by the IBM Operations Analytics - Log Analysis transport module.

Table 12. IBM Operations Analytics - Log Analysis transport error messages

Error	Description	Action
`[JSON Exception]: Failed to send batch to Unity data collector '%s': Msg=[%s], Stack=[%s]`	The gateway failed to send a data batch to the data collector for the reason described in the JSON message error.	Check the JSON message format construction using the `netcool2scala.xsl` file.
`[General Exception]: Failed to send batch to Unity data collector '%s': Msg=[%s], Stack=[%s]`	The gateway failed to send a data batch to the data collector due to a general problem.	Check the message provided.
`Failed to send batch message to data collector. No results handle returned.`	The gateway failed to send a data batch to the data collector for an unknown reason.	Check the IBM Operations Analytics - Log Analysis data collector.
`Failed to send batch to data collector. HTTP response code != 200. [%s].`	The gateway failed to receive the HTTP 200 (OK) response from the data collector.	Check the detailed data collector response in the message provided and check the data collector.
`Successfully sent batch to data collector, but failed with response code '%u'. [%s]`	The data collector returned the HTTP 200 (OK) response, but the response message indicates a failure.	Examine the response code and message in relation to the data collector.
`Failed to send event log record '%u': %s`	The gateway failed to send an individual event log record.	Check that the log record matches that of the data collector as defined by the gateway map file.

Table 12. IBM Operations Analytics - Log Analysis transport error messages (continued)		
Error	**Description**	**Action**
`Batch would exceed maximum size of '%u' for log data. Setting batch size to '%u'.`	The batch file size parameter in the `scalaTransport.properties` file (expressed in bytes) has been set to a value higher than allowed for log data.	The gateway will use a lower batch size. To stop this warning message being generated, reduce the value set for the **eventBufferSize** property in `scalaTransport.properties`.
`Number of log record successes and failures do not equal batch size.`	There is a discrepancy in recorded number of successes and failures and the total batch size.	Check the IBM Operations Analytics - Log Analysis data collector and contact IBM support.

Known Issues when running the gateway in LA mode

Use this information to understand the known issues associated with running the gateway in IBM Operations Analytics - Log Analysis (LA) mode.

No error messages appearing in the gateway log file and no events being sent to IBM Operations Analytics - Log Analysis

If you configure the hostname on the IBM Operations Analytics - Log Analysis server incorrectly, no error messages are written to the gateway's log file and no events are sent to IBM Operations Analytics - Log Analysis.

Symptom

To confirm that you have configured the hostname incorrectly, start the gateway with the command line option `-messagelevel debug` to start debug logging. You then see multiple repeating messages in the log file with the following format:

```
Debug: D-GJA-000-000: [ngjava]: XMLGateway: [THREAD=16]:
    [POST]: Authenticating against REST with username:unityadmin Password: ********
Debug: D-GJA-000-000: [ngjava]: XMLGateway: [THREAD=16]:
    Form Based Auth URL: https://omnihost:9987/Unity/j_security_check
Debug: D-GJA-000-000: [ngjava]: XMLGateway: [THREAD=16]:
    [GET]: Using REST address: https://omnihost:9987/Unity/CSRFToken
```

Cause

The hostname was not properly configured when IBM Operations Analytics - Log Analysis was installed. For instance, you may have specified the hostname in short format (for example: `omnihost`) instead of the expected fully qualified domain name (FQDN) (for example: `omnihost.domain.com`). As a result, IBM Operations Analytics - Log Analysis cannot retrieve tokens and return them to the gateway.

Solution

To resolve this issue, use the following steps:

1. Compare your configuration of the hostname with that described in the known issue referenced at the end of these steps.
2. Correct the configuration of the hostname as appropriate.
3. Reinstall IBM Operations Analytics - Log Analysis.

The gateway starts sending events to IBM Operations Analytics - Log Analysis.

Configuring the hostname correctly in IBM Operations Analytics - Log Analysis is described in the following known issue documented in the IBM SmartCloud Analytics - Log Analysis Knowledge Center:

http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.iwa.doc_1.0/tshoot/iwa_ts_dchp_issue_c.html?resultof=%22%68%6f%73%74%6e%61%6d%65%22%20%22%68%6f%73%74%6e%61%6d%22%20

Help text for the nc_httpcertimport utility missing options

If you invoke the `nc_httpcertimport` utility with the `-?` or `--help` parameters, or invoke the utility with an invalid parameter, the utility displays usage information. The usage information is missing the version or help parameters as follows:

```
-v   , --version  - version of tool
-?   , --help     -display usage information
```

These are valid parameters and can be used.

Integrating the gateway with Kafka

The Message Bus Gateway can be configured to integrate with a Kafka server as an event producer.

The Message Bus Gateway connects to the Kafka server using the Kafka transport. The Kafka transport runs on the following:

- `zookeeper-3.4.8.jar`
- `kafka-clients-0.10.0.1.jar`

Check the Apache Kafka compatibility matrix for the support of the target system with respect to the dependency.

Configuring the Message Bus Gateway to publish Kafka events

The following configuration files are supplied with the gateway for the integration with Kafka:

- `G_XML.props`
- `kafkaTransport.properties`
- `kafkaConnectionProperties.json`
- `kafkaClient.properties`

To configure the Message Bus Gateway to publish Kafka events, use the following steps:

1. Install/update the Message Bus Gateway. See “Installing the gateway” on page 6.
2. Edit the gateway configuration in the following gateway properties file:

   `$OMNIHOME/gates/<arch>/G_XML.props`

   Where *<arch>* is the architecture directory, for example `linux2x86`
3. Update the following property values with the appropriate path:

   ```
   Gate.XMLGateway.TransportFile        : '$OMNIHOME/java/conf/
   kafkaTransport.properties'
   Gate.XMLGateway.TransportType        : 'KAFKA'
   ```

4. Configure the Kafka transport properties.
   a. Edit the Kafka transport configuration in the following transport properties file:

      `$OMNIHOME/java/conf/kafkaTransport.properties`

   b. Update the following property value with the appropriate path:

```
kafkaClientMode=PRODUCER
connectionPropertiesFile=$OMNIHOME/java/conf/kafkaConnectionProperties.json
```

5. For descriptions of the Kafka transport properties, see the “Kafka transport properties table” on page 44.
6. Configure the Kafka connection properties.

   Kafka connection properties are defined in the `kafkaConnectionProperties.json` file. This file contains the following properties:

```
{
    "zookeeper_client" :
        {
            "target" : "",
            "properties" : "",
            "java_sys_props" : "",
            "topic_watch": true,
            "broker_watch": true
        },
    "brokers" : "",
    "topics": "",
    "kafka_client" :
        {
            "properties" : "",
            "java_sys_props" : ""
        }
 }
```

   a. Within the sample configuration file supplied with the gateway, update the path to the ZooKeeper client properties file.

```
"zookeeper_client" :
  {
      "target" : "localhost:2181",
      "properties" : "<Path & File of zookeeper client properties file>",
      "java_sys_props" : "",
      "topic_watch": true,
      "broker_watch": true
  },
```

   b. Update the path to the Kafka client properties file:

```
  "brokers" : "PLAINTEXT://localhost:9092",
  "topics": "topicABC,topicXYZ",
  "kafka_client" :
    {
      "properties" : "<full_omnihome_path>/java/conf/kafkaClient.properties",
      "java_sys_props" : ""
    }
```

7. For descriptions of the Kafka connection properties, see the “Kafka connection properties table” on page 46.
8. Configure the Kafka client properties.

   a. Edit the Kafka client configuration in the following properties file:

      `$OMNIHOME/java/conf/kafkaClient.properties`

   b. Uncomment the common Kafka client properties and the producer specific properties in the `kafkaClient.properties` file.

   c. For details of the configuration required by the connection protocols supported by the Kafka producer, see “Kafka configuration for different connection protocols” on page 47.

Note : The default setting for the Kafka producer’s **`key.serializer`** and **`value.serializer`** is `org.apache.kafka.common.serialization.StringSerializer`. The current Kafka Transport does not support any Kafka producer using a serializer class other than

`org.apache.kafka.common.serialization.StringSerializer`. Specifying any other serializer class causes issues in the producer's send operation.

9. Run the gateway using the following command:

   ```
   $OMNIHOME/bin/nco_g_xml -propsfile $OMNIHOME/etc/G_XML.props
   ```

   .

   Note : The gateway property file must be copied over from `$OMNIHOME/gates/xml` to `$OMNIHOME/etc` if it is going to be run from that directory.

Kafka transport properties table

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

Table 13. Kafka transport properties

Property name	Description
kafkaClientMode	Use this property to set the transport as a Kafka client to run as a consumer or a producer. This property takes the following values: `CONSUMER`: A Kafka consumer reads data from topics. `PRODUCER`: A Kafka producer writes data to topics.
connectionPropertiesFile	Use this property to specify the JSON file holding the Kafka connection properties.
LivenessCriterion	Use this property to control the periodic Kafka liveness check feature. This property takes the following values: `TOPIC`: Enables liveness check. `NONE`: Disables liveness check. The default value is `TOPIC`
LivenessCheckInterval	Use this property to configure the interval (in seconds) to run the Kafka liveness check. The range of valid values is 10 to 1000 seconds. The default value is `20`
LivenessCheckTimeout	Use this property to configure the time (in seconds) the transport allows for the liveness check operation to complete. If the operation does not complete within this period, the transport regards it as a negative Kafka liveness status. The range of valid values is 5 to 60 seconds. The default value is 5

Table 13. Kafka transport properties (continued)	
Property name	**Description**
LockSendUntilReconnection	Use this property to configure the transport to lock negative liveness status and stop message processing. This property takes the following values: TRUE: Locks the negative liveness status after a failed liveness check report. FALSE: Does not lock the negative liveness status. This used for debugging purposes only. For details about setting this property, see "Kafka liveness test" on page 45. The default value is TRUE

Kafka liveness test

In PRODUCER mode, the Kafka transport can be enabled to run the Kafka target liveness test periodically. This test queries the Kafka target for metadata of the topics configured in the `kafkaConnectionProperties.json` file. The initial test is performed after the worker producer's initialization as a proof of an established connection to the Kafka target. During start up, the gateway will exit if the initial test fails.

Setting **`LivenessCriterion`** to `TOPIC` enables the periodic liveness check to be performed at the interval specified by the **`LivenessCheckInterval`** property. When the Kafka target is unreachable, or the topic does not exist, the query stalls. If the query returns no result within the period specified by the **`LivenessCheckTimeout`** property, the transport interrupts the unresponsive query and regards the test as failed.

The Kafka transport creates an adhoc Kafka producer for every liveness check cycle to run the topic metadata query. This is because the `KafkaProducer` class's `partitionsFor()` method is effective only for testing connectivity in the first call of the method which queries the remote system, and then stores the result in the producer instance. A subsequent `partitionsFor()` call for the same topic will access the local cache instead.

Setting **`LockSendUntilReconnection`** to TRUE locks a negative liveness status and stops message processing within the Kafka transport after a liveness check reports a negative status. You would only set the **`LockSendUntilReconnection`** property to FALSE for debugging purposes. Do not use this configuration in a production environment.

Note : Locking negative a liveness status prevents the operational problems that could be induced by the gateway not initiating a reconnection in time while the restarted Kafka target may have been using a new IP. If the transport is not renewed, subsequent liveness checks by adhoc Kafka producers (connecting to the target with a new IP) could mistakenly report a positive status contrary to the failed state of the worker producer.

Kafka Reconnection

The Kafka transport does not self-initiate a reconnection. During a phase of negative liveness, the gateway's send attempts to the transport will hit a `TransportSendMsgException`. Responding to this exception, the gateway initiates the reconnection in the Kafka transport.

After a successful reconnection, before processing new messages, the Kafka producer will resend the messages that previously failed to reach the target.

Kafka connection properties table

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

Table 14. Kafka connection properties

Property name		Description
zookeeper_client	**target**	Use this property to specify the ZooKeeper endpoint. When this property is empty, the transport will not initiate connection to ZooKeeper. The default is `empty`.
	properties	Use this property to specify the path to a file holding ZooKeeper client properties in key-value format, for example: `key=value` The default is `empty`.
	java_sys_props	Use this property to specify the path to a file holding ZooKeeper client Java system properties required in a secure connection. The default is `empty`.
	topic_watch	Use this property to enable the ZooKeeper topic watch service.Valid values are: `true`: Enable the ZooKeeper topic watch service. `false`: Disable the ZooKeeper topic watch service. The default is `true`.
	broker_watch	Use this property to enable the ZooKeeper broker watch service.Valid values are: `true`: Enable the ZooKeeper broker watch service. `false`: Disable the ZooKeeper broker watch service. The default is `true`.
brokers		Use this property to specify broker endpoints in a comma-separated list. For example: "`localhost:9092, localhost:9093, localhost:9094`" The brokers must belong to the same cluster managed by a zookeeper. The default is `empty`.
topics		Use this property to specify topics in a comma-separated list. For example: "`topic1, topic2, topic3`" The default is `empty`.

Table 14. Kafka connection properties (continued)		
Property name		**Description**
Kafka_client	**properties**	Use this property to specify the path to a file holding Kafka client properties. The default is `empty`.
	java_sys_props	Use this property to specify the path to a file holding the Kafka client's Java system properties required in a secure connection. The default is `empty`.

Configuring the Kafka compression

Kafka client jar supports three compression codecs: `gzip`, `lz4`, and `snappy`, which can be enabled by setting the value at the Kafka client property **`compression.type`** in in the `kafkaClient.properties` file.

`lz4` or `snappy` compression also requires the following additional setup:

1. Obtain the `lz4` and `snappy` library jar file from Kafka installation directory: `<kafka_installation_location>/libs`

   The `lz4` jar file is `lz4-<version>.jar`.

   The `snappy` jar file is `snappy-java-<version>.jar`.
2. Copy the jar files to: `$OMNIHOME/java/jars`
3. Add the full path of the jars file to the `$CLASSPATH` environment variable.

   For example:

   ```
   export CLASSPATH=$CLASSPATH:$OMNIHOME/java/jars/lz4-<version>.jar:$OMNIHOME/java/jars/snappy-java-<version>.jar
   ```

   Note : `Gate.Java.ClassPath` is assumed to be containing the `$CLASSPATH` token.

Enable Kafka logging

For troubleshooting Kafka issues, enable the Kafka client's Log4j debug logging with the following configuration in the `$OMNIHOME/java/conf/log4j.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Log4j2 Configuration for Kafka Transport -->
<Configuration status="debug">
  <Appenders>
    <File name="LOGFILE" fileName="<log_filename.log>">
      <PatternLayout pattern="%d %-5p [%t] %C{2} (%F:%L) - %m%n"/>
    </File>
  </Appenders>
  <Loggers>
    <Root level="debug" >
      <!-- Increase the logging level if necessary  -->
      <AppenderRef ref="LOGFILE"/>
    </Root>
  </Loggers>
</Configuration>
```

Kafka configuration for different connection protocols

Kafka supports three types of connection protocol:

- `SASL_PLAINTEXT`
- `SASL_SSL`

The following table describes the configuration required by each connection protocol.

Table 15. Connection protocol configuration

Connection protocol	Configuration required
SASL_PLAINTEXT	**Kafka producer properties** `security.protocol=SASL_PLAINTEXT` **Note :** Must combine with SASL-specific configurations.

SASL_SSL	**Kafka producer properties**
	`acks=all`
	`security.protocol=SASL_SSL`
	`ssl.protocol=TLSv1.2`
	`ssl.enabled.protocols=TLSv1.2`
	`ssl.truststore.location=<path>\<trust_store_file>`
	`ssl.truststore.password=<trust_store_password>`
	`ssl.truststore.type=JKS`
	`# Specify the SASL type (see table-16)`
	`sasl.mechanism=PLAIN (SASL: Kafka user access control)`
	`OR`
	`sasl.mechanism=GSSAPI (SASL: Kerberos)`
	`OR`
	`sasl.mechanism=SCRAM (SASL: SCRAM)`
	Java system properties
	`java.security.auth.login.config=<path>/user_jaas.conf`
	`https.protocols=TLSv1.2`
	Note : Must combine with SASL-specific configurations.

The following table describes SASL-specific configurations.

Table 16. SASL-specific configuration

SASL: Kafka user access control

Java system properties

```
java.security.auth.login.config=<path>/
user_jaas.conf
```

Example user_jass.conf

```
KafkaClient {

org.apache.kafka.common.security.plain.Plai
nLoginModule required
  serviceName="kafka"
  username="myUserName"
  password="myPasword";
};
```

SASL: Kerberos

Java system properties

```
java.security.auth.login.config=<path>/
user_jass.conf
java.security.krb5.conf=<path>/krb5.conf
```

Example user_jass.conf

When using IBM JDK

```
KafkaClient {

com.ibm.security.auth.module.Krb5LoginModule
required
   debug=true
   credsType=both
   useKeytab="<path>/kafka.keytab"
   principal="username/instance@realm";
};
```

When using Oracle JDK

```
KafkaClient {

com.sun.security.auth.module.Krb5LoginModule
required
   debug=true
   renewTicket=true
   serviceName="kafka"
   useKeyTab=true
   keyTab="<path>/kafka.keytab"
   principal="username/instance@realm";
};
```

Note : This is the generic format of principal: *username/instance@realm*. Some organizations might use *servicename* instead of *username* or without *username*

```
principal="servicename/instance@realm"
principal="instance@realm
```

Consult your organization administrator for principal information.

SASL: SCRAM

Java system properties

```
java.security.auth.login.config=<path>/user_jaas.conf
```

Example `user_jaas.conf`

```
KafkaClient {
    org.apache.kafka.common.security.scram.ScramLoginModule required
    tokenauth="true"
    username="<username>"
    password="<password>"
};
```

Note :

Kafka producer properties are configured in the file specified in the **kafka_client.properties** field.

Java system properties are configured in the file specified in the **kafka_client.java_sys_props** field.

Example configuration

kafkaConnectionProperties.json:

```
"kafka_client": {
    "properties": "/opt/IBM/tivoli/netcool/omnibus/java/conf/kafkaClient.properties",
    "java_sys_props": "/opt/IBM/tivoli/netcool/omnibus/java/conf/kafkaClient_javaSys.properties"
  }
```

kafkaClient_javaSys.properties:

```
java.security.auth.login.config=C:\IBM\Tivoli\Netcool\omnibus\java\conf
\kafka_client_jaas.conf
```

kafka_client_jaas.conf:

```
KafkaClient {
    org.apache.kafka.common.security.plain.PlainLoginModule required
    serviceName="kafka"
    username="alice"
    password="alice";
};
```

In broker list configuration, a broker endpoint without a protocol prefix is assumed to be using the protocol configured in the **security.protocol** property. An unconfigured **security.protocol** denotes PLAINTEXT.

Configuring SSL connections

You configure SSL connections by creating a truststore to store the Message Bus digital certificate and point the Gateway for Message Bus to the location of the truststore.

You configure SSL connections if you are using one of the following SSL connection scenarios:

- HTTPS/SSL connection between the gateway and Message Bus
- SSL connection between the gateway and IBM Operations Analytics - Log Analysis

Note : The second SSL connection scenario provides a utility called nc_httpcertimport that makes it easier to load a certificate from an HTTP server. Note that the nc_httpcertimport utility is supported on all UNIX operating system platforms that the gateway supports. The nc_httpcertimport utility is not supported on Windows operating systems. For a description of the nc_httpcertimport utility and its associated command line options see "Using the nc_httpcertimport utility" on page 54.

The following topics describe how to configure SSL connections for the previously describe scenarios:

- "Configuring HTTPS/SSL connections between the gateway and Message Bus" on page 50
- "Configuring SSL connections between the gateway and IBM Operations Analytics - Log Analysis" on page 52

Configuring HTTPS/SSL connections between the gateway and Message Bus

If you are using an HTTPS/SSL connection between the gateway and Message Bus, you must create a truststore to store the Message Bus digital certificate and point the gateway to the location of the truststore.

Configuring SSL connections between the gateway and Message Bus consists of two tasks:

1. Creating a client keystore file.
2. Creating a truststore file for the target application to which the gateway is connecting.

You create a client keystore file and a truststore file by using the Java keytool utility. The keytool utility is located in the following directory:

$NCHOME/platform/*arch*/*jre_directory*/jre/bin/

Where:

arch is the operating system you are running.
jre_directory is the installation directory of your Java Runtime Environment (JRE).

Note : When running the gateway on a 64-bit operating system, use the Java `keytool` utility that delivers with the 64-bit Java Runtime Environment (JRE).

When creating a truststore file for the target application to which the gateway is connecting, you also need to edit property values in the transport file specified in the **Gate.XMLGateway.TransportFile** property. The **Gate.XMLGateway.TransportFile** property is defined in the `G_XML.props` file.

The default transport file specified by the **Gate.XMLGateway.TransportFile** property in the `G_XML.props` file is `$OMNIHOME/java/conf/jmsTransport.properties`.

Creating a client keystore file

To create a client keystore file named `client.jks` and store in it a certificate for your client, run the following `keytool` command from the $NCHOME/platform/*arch*/*jre_directory*/jre/bin/ directory:

```
keytool -genkey -alias youralias -keystore $OMNIHOME/java/security/client.jks
```

Note : You will be prompted to create a password for the client keystore file. Then `keytool` will prompt you for the details of the certificate to be entered; for each prompt enter something appropriate for your organization.

Creating a truststore file for the target application server to which the gateway is connecting

To create a truststore file for the target application to which the gateway is connecting, perform the following steps:

1. Export the server certificate from the host running the target application using Mozilla Firefox:
   a. Click the lock icon in the address bar.
   b. View and export the certificate to a file (for example: *xml-host*`.crt`).

      Where:

      xml-host: Specifies the name of a host server that is running a gateway with the HTTPS transport module.
2. Import the server certificate to the host where the gateway is running:
   a. Copy the server certificate file to the host where the gateway is running.
   b. Run the following `keytool` command from the $NCHOME/platform/*arch*/*jre_directory*/jre/bin/ directory:

      ```
      keytool -import -keystore $OMNIHOME/java/security/cacerts.jks -file
      xml-host.crt -alias xml-host
      ```

   Where:

   arch is the operating system you are running.
   jre_directory is the installation directory of your Java Runtime Environment (JRE).
   xml-host: Specifies the name of a host server that is running a gateway with the HTTPS transport module.
3. Open the transport file specified by the **Gate.XMLGateway.TransportFile** property defined in the `G_XML.props` file and modify the following transport file properties:

a. Set the value of the **keyStore** property to the full path of the keystore file. For example, if you created a keystore file in the location $OMNIHOME/java/security, then specify that path and the name of the keystore file (for example, client.jks) in the **keyStore** property in the httpTransport.properties file.

b. Set the value of the **trustStore** property to the full path of the truststore file. For example, if you created a truststore file in the location $OMNIHOME/java/security, then specify that path and the name of the truststore file (for example, cacerts.jks) in the **trustStore** property in the httpTransport.properties file.

c. Set the value of the **keyStorePassword** property to the password that you set for the client keystore file.

d. Set the value of the **trustStorePassword** property to the password that you set for the truststore file.

Configuring SSL connections between the gateway and IBM Operations Analytics - Log Analysis

If you are using an SSL connection between the gateway and IBM Operations Analytics - Log Analysis, you must create a truststore file to store the IBM Operations Analytics - Log Analysis digital certificate and point the gateway to the location of the truststore.

Configuring SSL connections between the gateway and IBM Operations Analytics - Log Analysis consists of two tasks:

1. Creating a client keystore file.
2. Creating a truststore file for the target application to which the gateway is connecting.

You create a client keystore file and a truststore file by using the Java keytool utility. The keytool utility is located in the following directory:

$NCHOME/platform/*arch*/*jre_directory*/jre/bin/

Where:

arch is the operating system you are running.
jre_directory is the installation directory of your Java Runtime Environment (JRE).

Note : When running the gateway on a 64-bit operating system, use the Java keytool utility that delivers with the 64-bit Java Runtime Environment (JRE).

You create a truststore file for the target application to which the gateway is connecting by using the nc_httpcertimport utility, which is packaged with the gateway. The nc_httpcertimport utility is located in the following directory:

$OMNIHOME/bin

When creating a truststore file for the target application to which the gateway is connecting, you also need to edit property values in the transport file specified in the **Gate.XMLGateway.TransportFile** property. The **Gate.XMLGateway.TransportFile** property is defined in the G_SCALA.props file.

The default transport file specified by the **Gate.XMLGateway.TransportFile** property in the G_SCALA.props file is $OMNIHOME/java/conf/scalaTransport.properties.

Creating a client keystore file

To create a client keystore file named client.jks and store in it a certificate for your client, run the following keytool command from the $NCHOME/platform/*arch*/*jre_directory*/jre/bin/ directory:

keytool -genkey -alias *youralias* -keystore $OMNIHOME/java/security/client.jks

Note : You will be prompted to create a password for the client keystore file. The keytool will prompt you for the details of the certificate to be entered; for each prompt enter something appropriate for your organization. You must copy the client certificate from the SCALA server location:

$UNITYHOME/wlp/usr/servers/Unity/resources/security/client.crt

To review the keytool reference information to review tools and concepts related to SSL security, see the following link:

https://www.ibm.com/support/knowledgecenter/SSYKE2_7.0.0/com.ibm.java.security.component.70.doc/security-component/keytoolDocs/keytool_overview.html

Exporting the server certificate from the SCALA server

To export the server certificate from the SCALA server, perform the following steps:

1. Open the Firefox web browser.
2. Enter the following SCALA URL

   https://analysis-server1:9987/Unity

   .
3. Click the padlock icon and select **More Information**.
4. Click **Security** and select **View Certificate**.
5. Select **Details**.
6. Scroll to the bottom of the page and select **Export**.
7. In the **Save As** bar, enter host1.cert and click **Save**.
8. Copy host1.cert to /opt on Machine 1.

Creating a truststore file for the target application server to which the gateway is connecting

To create a truststore file for the target application server to which the gateway is connecting, perform the following steps:

1. Import the server certificate to the host where the gateway is running by invoking the nc_httpcertimport utility as follows:

   ```
   $OMNIHOME/bin/nc_httpcertimport -k $OMNIHOME/java/security/cacerts.jks
    -h host --alias scala-host
   ```

   Where:

   - *host*: Specifies the Uniform Resource Identifier (URI) of the LA target server from which to import the certificate. You must specify a port number in the URI specification. For example:

     ```
     $OMNIHOME/bin/nc_httpcertimport -k $OMNIHOME/java/security/cacerts.jks
     -p password
     -h http://testserver1.xyzcompany.com:9988/unity
     --alias scala-host
     ```

     In the example, the URI specifies that the LA target server is running on host testserver1.xyzcompany.com.
   - *scala-host*: Specifies the name of the host server that is running a gateway with IBM Operations Analytics - Log Analysis.
2. When prompted for the password, specify the password for the truststore file.

   Note : Take note of this password as you will need to specify this password for the **trustStorePassword** property in the scalaTransport.properties file in the next step.
3. Open the transport file specified by the **Gate.XMLGateway.TransportFile** property and modify the following transport file properties:

   a. Set the value of the **keyStore** property to the full path of the keystore file. For example, if you created a keystore file in the location $OMNIHOME/java/security/client.jks as in the

previous example, you do not need to edit the **keyStore** property in the `scalaTransport.properties` file because that is the default value. Otherwise, if you created a keystore file in a different location specify that location and the keystore file in the **keyStore** property.

b. Set the value of the **trustStore** property to the full path of the truststore file. For example, if you created a truststore file in the location `$OMNIHOME/java/security/cacerts.jks` as in the previous example, you do not need to edit the **trustStore** property in the `scalaTransport.properties` file because that is the default value. Otherwise, if you created a truststore file in a different location specify that location and the truststore in the **trustStore** property.

c. Set the value of the **keyStorePassword** property to the password that you set for your client keystore file.

d. Set the value of the **trustStorePassword** property to the password that you set for your truststore file.

Using the nc_httpcertimport utility

You use the `nc_httpcertimport` utility to make it easier to load a certificate from an HTTP server when configuring the Gateway for Message Bus to connect to IBM Operations Analytics - Log Analysis through Secure Socket Layer (SSL). This topic provides reference information related to the `nc_httpcertimport` utility.

NAME

`nc_httpcertimport` - Loads a certificate from an HTTP server

COMMAND LINE SYNTAX

```
nc_httpcertimport [-k keystore_file] -h host_URL
                  [-p password] [-a certificate_alias]
                  [-l] [-x] [-v] [-?]

nc_httpcertimport [--keystore keystore_file] --host host_URL
                  [--password password] [--alias certificate_alias]
                  [--list] [--accept] [--version] [--help]
```

OPTIONS

Option	Description
-k *keystore_file* --keystore *keystore_file*	The -k and --keystore options specify the name of the keystore file to create (if no keystore file exists) or the name of an existing keystore file in which to add certificates. The keystore file you specify for *keystore_file* should include the path. For example: `--keystore $OMNIHOME/java/security/ cacerts.jks`
-h *host_URL* --host *host_URL*	The -h and --host options specify the Uniform Resource Identifier (URI) of the server running the application from which to import the certificate. You must specify a port number in the URI specification. For example: `-h https://testserver1.xyzcompany.com: 9987/Unity` **Note :** The URI you specify should be an HTTPS URI. By default, IBM Operations Analytics - Log Analysis uses the port 9987. Note also that the URI includes the

Option	Description
	name of the application from which to import the certificate, which in the example is `Unity`.
`-p` *password* `--password` *password*	The `-p` and `--password` options specify the password of the truststore file that you create using the Java `keytool` utility. The password of the truststore file is optional on the command line. If you do not specify a password on the command line, the `nc_httpcertimport` utility displays a prompt for the password. One reason you might not want to specify the password of the truststore file on the command line is because the `nc_httpcertimport` utility echoes it on the display terminal screen. However, when prompted for the password, the `nc_httpcertimport` utility does not echo it on the display terminal screen. **Note :** If you are creating a new trustore file, the `nc_httpcertimport` utility prompts you twice for the password. The password will not echo onto the display terminal screen.
`-a` *certificate_alias* `--alias` *certificate_alias*	The `-a` and `--alias` options specify the alias to associate with the certificate to be added to the keystore file specified in *keystore_file*. For example: `--alias scala-host`
`-l` `--list`	The `-l` and `--list` options display a list of all the certificates returned by the server application specified in *host_URL*.
`-x` `--accept`	The `-x` and `--accept` options accept the first server certificate from the server application specified in *host_URL* without prompting and adds the certificate to the keystore file specified in *keystore_file*. If the server application returns more than one certificate, the `nc_httpcertimport` utility accepts only the first certificate.
`-v` `--version`	The `-v` and `--version` options display version information for the `nc_httpcertimport` utility.
`-?` `--help`	The `-?` and `--help` options display usage information for the `nc_httpcertimport` utility. This usage information includes descriptions of the valid syntax for the command line options.

DESCRIPTION

The `nc_httpcertimport` utility loads an SSL certificate from an HTTP server and either adds it to an existing keystore file, or creates a new keystore file to which it adds the certificate. Using this utility allows you to avoid having to manually extract the SSL certificate from a web browser and to use the Java tooling to add it to a truststore file. The `nc_httpcertimport` utility is interactive by default and requires users to accept SSL certificates. To run the `nc_httpcertimport` utility with no user interaction, specify the `-p` (or `--password`) and `-x` (or `--accept`) options.

Note : The `nc_httpcertimport` utility generates a file to store SSL certificates. These SSL certificates can be used as a keystore file or a truststore file.

NOTES

The `nc_httpcertimport` utility is supported only on UNIX platforms supported by the gateway. The `nc_httpcertimport` utility is not supported on Windows operating systems.

EXAMPLE

```
$OMNIHOME/bin/nc_httpcertimport -k $OMNIHOME/java/security/cacerts.jks
-p password
-h http://testserver1.xyzcompany.com:7789/unity
--alias scala-host
```

ERROR MESSAGES

The following table describes the possible error messages that the nc_httpcertimport utility can generate.

Error Message	Description and Action
Command line option *option* is not recognized.	You specified an invalid command line option. To resolve the issue, specify a valid command line option. The valid command options are described in "OPTIONS" on page 54. You can also specify the -? or --help option to display usage information for the nc_httpcertimport utility.
The certificate number that you specified is invalid.	You were running the nc_httpcertimport utility in interactive mode and the utility requested that you enter the number of a SSL certificate you trust and want to add to the keystore file. However, the SSL certificate number you specified is invalid. To resolve the issue, enter a valid number of a SSL certificate you trust and want to add to the keystore file. Or, you can specify q to quit the nc_httpcertimport utility.
Failed to access the system console. Not able to run in interactive mode. Specify the --help option for details on how to run the utility in non-interactive mode.	The nc_httpcertimport utility was unable to access the system console. The most likely reason is that you attempted to run nc_httpcertimport without user interaction (non-interactive mode) and specified invalid options. To run the nc_httpcertimport utility without user interaction (non-interactive mode), specify the -p (or --password) and -x (or --accept) options. You can also specify the -? or --help option to display usage information for the nc_httpcertimport utility.
You specified an http URI for the host. An https URI is required.	You specified either the -h or --host option and followed it with the URI of the server application from which to import the SSL certificate. However, the URI you specified started with http://. To resolve the issue, the URI you specify should be an HTTPS URI. Thus, the URI should start with https://.
The required keystore file *keystore_file* was not found. Verify that the install is not corrupted.	You specified either the -k or --keystore option and followed it with the path and name of the required keystore file. However, the nc_httpcertimport utility was unable to find the specified keystore file. To resolve the issue ensure that the keystore file that you specify with the -k or --keystore option exists.
Failed to connect to the host due to IO exception. *exception*	The nc_httpcertimport utility called an internal method to create an SSL socket connection. This error indicates that the attempt to create the SSL socket connection to the host specified in *host_URL* failed because the server is not running, a firewall is in the way, or you incorrectly specified the URI in the *host_URL*. The I/O exception specified in *exception* generally provides information about the specific reason for the failure to connect to the server.

Error Message	Description and Action
	To resolve the issue, use the information in *exception* to identify the specific cause of the error.
`Failed to open keystore file due to IO exception` *exception*	The `nc_httpcertimport` utility called an internal method to open the keystore file that you specified with the `-k` or `--keystore` option. However, the internal method failed to open the specified keystore file due to the I/O exception specified in *exception*. One reason this error can occur is that you are running the `nc_httpcertimport` utility against an existing keystore file and the user ID that the utility is running under does not have access to the keystore file. Another possible reason for this error to occur is that you are creating a new keystore file and the `nc_httpcertimport` utility does not have access to the directory where the keystore file resides. To resolve the issue, use the information in *exception* to identify the specific cause of the error.
`The --list and --accept parameters cannot be specified at the same time.`	You invoked the `nc_httpcertimport` utility with the `--list` and `--accept` (or `-l` and `-x`) options. This is an invalid combination of options because the `--list` and `-l` options display a list of all the SSL certificates returned by the server application specified in *host_URL*. The `--accept` and `-x` options accept the first server certificate from the server application specified in *host_URL* without prompting and instruct `nc_httpcertimport` to add the SSL certificate to the keystore file specified in *keystore_file*. To resolve the issue, specify either the option to display a list of all the certificates or the option to accept the first server certificate.
`Argument` *arg_name* `requires a value.` `Please use the --help option to check the valid syntax of the command line arguments. There is a missing value.`	You invoked the `nc_httpcertimport` utility with the command line option *arg_name*. The *arg_name* command line option requires that you specify a valid value. Or, you invoked the `nc_httpcertimport` utility with the command line option *arg_name* and specified a value that starts with `-`. Command line option values cannot start with a `-`. To resolve the issue, use the `-?` or `--help` option to check the valid syntax of the command line options. You can also review the command line option descriptions in “OPTIONS” on page 54. Also, ensure that you do not specify values with `-`.
`This utility requires that you set the the NCHOME environment variable.`	The `nc_httpcertimport` utility was performing operations on the keystore file that you specified with the `-k` or `--keystore` option. However, `nc_httpcertimport` determined that the NCHOME environment variable was not set. To correct the issue, set the NCHOME environment variable.
`Internal error retrieving the server certificate chain.`	The `nc_httpcertimport` utility called an internal method to obtain the trust manager chain from the Java Secure Socket Extension (JSSE) trust manager. This error indicates that the internal method discovered that the trust manager chain is null and therefore the currently running Java virtual machine (JVM) was terminated. If this error occurs contact IBM technical support.
`Unable to find a default X509 trust`	The `nc_httpcertimport` utility called an internal method to obtain an instance of a default X509 trust manager from the current keystore file. (X509 trust

Error Message	Description and Action
`manager for the JVM.`	managers manage X509 certificates.) This error indicates that the internal method was unable to find a default X509 trust manager for the currently running Java virtual machine (JVM). As a result, the internal method creates an X509 trust manager. Typically, this error would only occur if your environment has an issue with the JVM. The gateway should be picking up the X509 trust manager that delivers with it. If this error occurs contact IBM technical support.
`This utility requires you specify the URI of the server running the application from which you want to add certificates.`	You invoked the `nc_httpcertimport` utility with the `-h` or `--host` option, but did not supply a URI to the *host_URL* argument. To resolve the issue, you must specify the URI of the server running the application from which to import the SSL certificate. You can also specify the `-?` or `--help` option to check the valid syntax of the `-h` or `--host` options or see "OPTIONS" on page 54 for additional information.
`Specify either a keystore filename or a list of all certificates returned by the server.`	You invoked the `nc_httpcertimport` utility without specifying one of the following command line options: `-l` or `--list` - To display a list of all the certificates returned by the server application specified in *host_URL*. `-k` or `--keystore` *keystore_file* - To specify the specify the name of the keystore file To resolve the issue, you need to specify one of the previously described options. You can also use the `-?` or `--help` option to check the valid syntax of the command line options. You can also review the command line option description for the previously described options in "OPTIONS" on page 54.
`The passwords that you specified do not match.`	You invoked the `nc_httpcertimport` utility without specifying the `-p` or `--password` options. Thus, the `nc_httpcertimport` utility requests that you enter and then reenter the password for the truststore file. In this case, the passwords that you specified do not match. To resolve the issue, invoke the `nc_httpcertimport` utility without specifying the `-p` or `--password` options. Then make sure you specify identical passwords to the password prompts that the `nc_httpcertimport` utility displays.
`The host server URI does not contain a port number.`	You invoked the `nc_httpcertimport` utility with the `-h` or `--host` option and specified a URI for the *host_URL* argument. However, the URI you specified did not contain a port number for the server application from which to import the SSL certificate. To resolve the issue, invoke the `nc_httpcertimport` utility with the `-h` or `--host` option and this time specify a URI that contains a port number. You can also use the `-?` or `--help` option to check the valid syntax of the command line options. You can also review the command line option description for the previously described options in "OPTIONS" on page 54.
`The SSL connection to the host failed with an SSLException` *`exception`*`.`	The `nc_httpcertimport` utility called an internal method to create an SSL socket connection. This error indicates that the attempt to create the SSL socket connection to the host specified in *host_URL* failed because of the reason specified in *exception*.

Error Message	Description and Action
	This error indicates that the attempt to create the SSL socket connection to the host specified in *host_URL* failed because the server is not running, a firewall is in the way, or you incorrectly specified the URI in the *host_URL*. The SSL exception specified in *exception* generally provides information about the specific reason for the failure to connect to the server. To resolve the issue, use the information in *exception* to identify the specific cause of the error.
`The SSL Handshake failed with the following error error.`	The `nc_httpcertimport` utility called an internal method to create an SSL socket. This error indicates that the SSL handshake operation failed because of the reason specified in *error*. This error indicates that the attempt to create the SSL socket connection to the host specified in *host_URL* failed because the server is not running, a firewall is in the way, or you incorrectly specified the URI in the *host_URL*. The error specified in *error* generally provides information about the specific reason for the failure to connect to the server. To resolve the issue, use the information in *error* to identify the specific cause of the error.
`The SSL Handshake failed with the following error error. Verify that the keystore file contains certificates.`	The `nc_httpcertimport` utility called an internal method to create an SSL socket. This error indicates that the SSL handshake operation failed because of the reason specified in *error*. To resolve the issue, ensure that the keystore file that you specified with the `-k` or `--keystore` option contains one or more valid SSL certificates. Also, use the information in *error* to identify the specific cause of the error.
`Error unable to parse URI: uri.`	You invoked the `nc_httpcertimport` utility with the `-h` or `--host` option and specified a *host_URL*. When the `nc_httpcertimport` utility attempts to create an SSL certificate import instance it checks the specified *host_URL*. In this case, `nc_httpcertimport` could not parse the *host_URL* you specified. To resolve the issue, enter a valid *host_URL*. Use the `-?` or `--help` option to check the valid syntax of the command line options. You can also review the command line option description for the `-h` and `--host` options in “OPTIONS” on page 54.
`Error error reading user response from the standard input.`	You were running the `nc_httpcertimport` utility in interactive mode and the utility requested that you enter the number of an SSL certificate you trust and want to add to the keystore file. However, the `nc_httpcertimport` utility detected an error in reading the SSL certificate you specified. To resolve the issue, enter a valid number of an SSL certificate you trust and want to add to the keystore file. Or, you can specify q to quit the `nc_httpcertimport` utility.

The transport and transformer modules

The Gateway for Message Bus reads Tivoli Netcool/OMNIbus events from an ObjectServer and then converts these events to Extensible Markup Language (XML) format. The transformer module then

converts these events from XML format to another format that you specify. The transport module then forwards these events to the target application.

Using the transport module

You need to choose one of several available transport protocols and then configure properties in the associated transport properties files.

You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode. The transport protocol and transport properties file you choose depend on the gateway's mode of operation.

Before using the transport module, you must specify the transport protocol and transport properties file using the **Gate.XMLGateway.TransportType** and **Gate.XMLGateway.TransportFile** properties defined in the `G_XML.props` and `G_SCALA.props` properties files. After installing the gateway, these properties have the following default values:

- For gateways operating in standard mode, the following properties in the `G_XML.props` file have these default values:

  Gate.XMLGateway.TransportType - `JMS`
  Gate.XMLGateway.TransportFile - `$OMNIHOME/java/conf/jmsTransport.properties`
- For gateways operating in LA mode, the following properties in the `G_SCALA.props` file have these default values:

  Gate.XMLGateway.TransportType - `SCALA`
  Gate.XMLGateway.TransportFile - `$OMNIHOME/java/conf/scalaTransport.properties`

The transport module properties files are located in the following directory:

`$OMNIHOME/java/conf/`

Note : Refer to the Configuring the properties file topic for information about configuring the `G_XML.props` and the `G_SCALA.props` files.

Table 17 on page 60 identifies the available transport module protocols, the associated transport module protocol values to use for the**Gate.XMLGateway.TransportType** property and the associated transport module properties file names to use for the **Gate.XMLGateway.TransportFile** property for gateway standard and LA operation.

Table 17. Transport module protocols and properties files

Transport module protocol	Value for Gate.XMLGateway.TransportType	Value for Gate.XMLGateway.TransportFile
Transport module protocols and properties files for the gateway standard mode operation		
Java Message Service (JMS)	`JMS`	`jmsTransport.properties`
Data file	`File`	`fileTransport.properties`
Message Queue Telemetry Transport (MQTT)	`MQTT`	`mqttTransport.properties`
HTTP and HTTPS (Hypertext Transfer Protocol/HTTP Secure)	`HTTP`	`httpTransport.properties`
Socket	`Socket`	`socketTransport.properties`

Table 17. Transport module protocols and properties files (continued)		
Transport module protocol	**Value for Gate. XMLGateway. TransportType**	**Value for Gate. XMLGateway. TransportFile**
Kafka	Kafka	kafkaTransport.properties
Transport module protocol and properties file for the gateway LA mode operation		
	SCALA	scalaTransport.properties

Configuring transport module properties files

Transport properties files define how the gateway sends events using the transport module.

The following sections describe the properties files that the gateway can use to send events using the transport module:

- "The jmsTransport.properties file" on page 61
- "The fileTransport.properties file" on page 63
- "The mqttTransport.properties file" on page 64
- "The httpTransport.properties file" on page 65
- "The socketTransport.properties file" on page 69
- "The scalaTransport.properties file" on page 70

Note : You can configure the gateway to operate in one of two modes of operation: standard mode or IBM Operations Analytics - Log Analysis (LA) mode. The transport module properties files associated with gateways running in standard mode are `jmsTransport.properties`, `fileTransport.properties`, `mqttTransport.properties`, `httpTransport.properties` and `socketTransport.properties`. The transport module properties file associated with gateways running in LA mode is `scalaTransport.properties`.

Note : On Windows platforms, the delimiter in all the paths defined in the transport module properties files should be escaped, for example, if you are setting the path for the key store, the path should be defined as follows:

```
keyStore=C:\\IBM\\Tivoli\\Netcool\\omnibus\\java\\security\\client.jks
```

The jmsTransport.properties file

Table 18 on page 62 describes the properties defined in the `jmsTransport.properties` file. which is the transport module properties file associated with the Java Message Service (JMS) transport module protocol.

Note : You configure the JMS-related properties if you set the **`Gate.XMLGateway.TransportType`** and **`Gate.XMLGateway.TransportFile`** properties to JMS and `jmsTransport.properties` in the `G_XML.props` properties file.

Table 18. JMS transport module properties	
Property name	**Description**
initialContextFactory	Use this property to specify the context factory class name of the JMS provider. **Note :** The **initialContextFactory** is the initial context factory for the Java Naming Directory Interface (JNDI) provider being used. The default is the Websphere JNDI provider.
jmsFilter	Use this property to specify the expression for JMS servers for sending the intended messages or events to the probe.

jmsHeaderMap	Use this property to assign a value to the JMS header. Multiple instances of this property is supported.
	Configuration format:
	`jmsHeaderMap=<jms_header_name>:<value>`
	Supported JMS headers for sender update:
	• `JMSCorrelationID`
	• `JMSType`
	The assigned value can be static or dynamic.
	A static value is a string enclosed with single quotes, for example:
	`jmsHeaderMap=<jms_header_name>:'This is static value'`
	A dynamic value can be obtained from raw Netcool event data by configuring a data source in the property. The data source can be any mapped field name specified in the gateway's map field, for example:
	`jmsHeaderMap=<jms_header_name>:MyNode`
	For details about using dynamic values, see "Dynamic value assignment for jmsHeaderMap and jmsPropertyMap" on page 27.

Table 18. JMS transport module properties (continued)	
Property name	**Description**

jmsPropertyMap	Use this property to assign a value to the JMS property. Multiple instances of this property is supported.
	Configuration format:
	`jmsHeaderMap=<jms_property_name>:<value>`
	JMS property names are customizable. The assigned value can be static or dynamic.
	A static value is a string enclosed with single quotes, for example:
	`jmsHeaderMap=MyJmsProperty:'This is static value'`
	A dynamic value can be obtained from raw Netcool event data by configuring a data source in the property. The data source can be any mapped field name specified in the gateway's map field, for example:
	`jmsHeaderMap=jmsPropertyMap:MyNode`
	For details about using dynamic values, see "Dynamic value assignment for jmsHeaderMap and jmsPropertyMap" on page 27.

providerURL	Use this property to specify the URL of the JMS provider. You can specify the URL of the data store for the JNDI provider, or you can specify the path to a binding file. **Note :** If you specify a binding file, always create that file using the IBM WebSphere® MQ JMS Administration tool. In addition, copy the WebSphere jar files to the `$OMNIHOME/java` directory. For information about creating binding files and JMS Administration tool jar files, see the IBM WebSphere MQ information center.
queueConnectionFactory	Use this property to specify the identifier of the queue connection factory.
topicConnectionFactory	Use this property to specify the identifier of the topic connection factory.
username	Use this property to specify the user name for the JMS connection.
password	Use this property to specify the password for the JMS connection.

Note : You must ensure that all property values in the transport properties file are unquoted; that is, contain no quotation marks.

The fileTransport.properties file

Table 19 on page 64 describes the properties defined in the `fileTransport.properties` file. which is the transport module properties file associated with the Data File transport module protocol.

Note : You configure the data file-related properties if you set the **Gate.XMLGateway.TransportType** and **Gate.XMLGateway.TransportFile** properties to `File` and `fileTransport.properties` in the `G_XML.props` properties file.

Table 19. Data file transport module properties

Property name	Description
jsonFilename	Use this property to specify the full path to a JSON file to run through the gateway. The default is `${OMNIHOME}/var/jsonAlarm.txt`.
streamFilename	Use this property to specify the full path to a stream capture file to run through the gateway. The default is `${OMNIHOME}/var/xmlAlarm.txt`. **Note :** You can specify either the **streamfilename** property or the **xmlfilename** property but not both.
xmlFilename	Use this property to specify the full path to a standard stream of XML to run through the gateway. **Note :** You can specify either the **streamfilename** property or the **xmlfilename** property but not both.
sleepInterval	Use this property to specify the interval (in milliseconds) that the gateway waits between iterations when tailing the source file. The default is `1000` milliseconds.

Note : You must ensure that all property values in the transport properties file are unquoted; that is, contain no quotation marks.

The mqttTransport.properties file

Table 20 on page 64 describes the properties defined in the `mqttTransport.properties` file, which is the transport module properties file associated with the MQTT transport module protocol.

Note : You configure the MQTT-related properties if you set the **Gate.XMLGateway.TransportType** and **Gate.XMLGateway.TransportFile** properties to `MQTT` and `mqttTransport.properties` in the `G_XML.props` properties file.

Table 20. MQTT transport module properties

Property name	Description
connectionURL	Use this property to specify the URL of the MQTT bus provider. The default is `tcp://localhost:1883`.
clientId	Use this property to specify an ID for each MQTT client connection to the MQTT provider. The **clientId** of each MQTT client connection must be unique. The default is `MQTTClient`.

Table 20. MQTT transport module properties (continued)	
Property name	**Description**
topicName	Use this property to specify the name of a topic to subscribe to for messages. To subscribe to multiple topics, you can define multiple topic names. The default is `""`. **Note :** The MQTT wildcard character *#* can be used to define a wildcard set of topics to subscribe to. For example, a value of *#* would register interest in all topics known to the MQTT provider.
cleanStart	Use this property to specify whether the connection should perform a clean start. The default is `false`. If you specify a value of `false`, any message stored by the MQTT provider for a connection client ID will be returned on startup.
keepAlive	Use this property to specify the frequency (in seconds) with which the MQTT provider connection is polled during periods of inactivity. Polling keeps the connection alive and active. The default is `30`.

Note : You must ensure that all property values in the transport properties file are unquoted; that is, contain no quotation marks.

The httpTransport.properties file

Table 21 on page 66 describes the properties defined in the `httpTransport.properties` file, which is the transport module properties file associated with the HTTP and HTTPS transport module protocol.

Note : You configure the HTTP- and HTTPS-related properties if you set the **`Gate.XMLGateway.TransportType`** and **`Gate.XMLGateway.TransportFile`** properties to HTTP and `httpTransport.properties` in the `G_XML.props` properties file.

Table 21. HTTP/HTTPS transport module properties	
Property name	**Description**
batchHeader	Use this property to specify the header for batch messages. A batch message consists of a group of Netcool events of which every event is originally an XML file, but each has been stripped of its XML declaration to ensure XML parsing at the endpoint. To complete the batched events as an XML file, **batchHeader** must contains the following: • The XML declaration: `<?xml version="1.0" encoding="UTF-8">` • An additional XML opening tag for enclosing the batched events. **Note : batchFooter** must have an XML closing tag corresponding to the opening tag specified within this property. The default is `""`.
batchFooter	Use this property to specify the footer for batch message. **Note : batchFooter** must have an XML closing tag corresponding to the opening tag specified within the **batchHeader** property. The default is `""`.
batchSeparator	Use this property to set the separator used between events in a batch. This property is usually set to a comma (`,`) for the batched JSON events use case, that is: when **Gate.XMLGateway.TransformerInputType** is `json` and **batchSize** is greater than 1. The default is `""`.
bufferFlushTime	Use this property to specify the interval (in seconds) at which the gateway flushes the buffer. When the difference between the current time and the time at which the last event was put into the buffer is equal to or greater than the **bufferFlushTime** interval, all buffered events are batched up for HTTP posting. The default is 20.
bufferSize	Use this property to specify the maximum size that the gateway allows for the buffer. When the buffer size is reached, all events in the buffer are batched up for HTTP posting. The default is 10.

Table 21. HTTP/HTTPS transport module properties (continued)	
Property name	**Description**
clientURL	Use this property to specify a URL definition for a HTTP/HTTPS destination. The default is "".
concurrentRequestThreads	Use this property to set the thread pool size for HTTP requests. The default is 1.
httpTimeout	Use this property to specify the time (in seconds) that the gateway allows for the HTTP connection. The default is 10.
retryLimit	Use this property to specify the maximum number of HTTP operation retries that the gateway can make. After a successful HTTP operation, the gateway makes no more retries. The default is 2.
retryWait	Use this property to specify the time (in seconds) that the gateway waits before retrying an HTTP operation . The default is 5.
threadPoolSize	Use this property to specify the number of threads that should be used to handle client connections to the internal HTTP/HTTPS server used by the transport module. The default is 10.
useNullHostnameVerifier	Use this property to disable hostname verification in the HTTP client against the web server's SSL certificate. This unblocks the authentication exception caused by the hostname in a valid SSL certificate being different from the web server's current hostname. The default is false.
username	Use this property to specify the username to use for basic authentication. Leave this property blank for no authentication. The username and password in the HTTP client has the following constraints: • **username** and **password** cannot override the credentials embedded in the endpoint URL passed from the Transformer module. • For any endpoint without prior configured credentials, the **username** and **password** will be used in POST operations, if these properties are configured. The default is "".

Table 21. HTTP/HTTPS transport module properties (continued)	
Property name	**Description**
password	Use this property to specify the password to use in basic authentication. This property is ignored if **username** is blank. The **password** property can be set to the value generated from the encryption by **ConfigCryptoAlg**. For more information see "Encrypting gateway and transport module properties" on page 73 The default is "".

Note : You must ensure that all property values in the transport properties file are unquoted; that is, contain no quotation marks.

HTTP operation retry

If the gateway fails to send out a message using the HTTP operation, it retries for the number of times specified by the **retryLimit** property. When the target returns a positive response, the gateway stops retrying.

The gateway waits for the number of seconds specified by the **retryWait** property between each HTTP operation retry.

When the gateway is in retry mode, if the condition for the next message sending is ripe (see the description for **bufferSize** and **bufferFlushTime**) the new message is put on hold until the retry of the previous message is completed. This prevents concurrent attempts of message sending which might disturb event ordering at the receiving end.

Event transformation and batching

The netcool event produced from the empty Transformer is in the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
    <tns:netcoolEvent type="update"
      xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool">
        <tns:netcoolField name="Serial" type="integer">1</tns:netcoolField>
        <tns:netcoolField name="Node" type="string">1.1.1.1</tns:netcoolField>
        <tns:netcoolField name="Acknowledged" type="integer">0</tns:netcoolField>
        <tns:netcoolField name="ServerName" type="string">NCOMS_P</tns:netcoolField>
        <tns:netcoolField name="ServerSerial" type="integer">1</tns:netcoolField>
    </tns:netcoolEvent>
```

To batch multiple Netcool events into XML compliant format, you must make some configuration changes in the Transformer Module and Transport Module.

Transformer Module

`transformer.xsl`

```
<tns:transformer name="netcoolEvents" type="northbound"
     endpoint="http://<ip>:<port>"
     className="com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer">
<tns:property name="xsltFilename" type="java.lang.String"
     value="${OMNIHOME}/java/conf/netcoolstripxmlheader.xsl"
     description="XSLT file for converting Netcool events to NC events"/>
</tns:transformer>
```

This XLST definition strips the XML declaration from every Netcool event produced from the Transformer Module. The header-stripped event is then placed in the HTTP Transport's buffer.

When the condition is ripe (refer to the **bufferSize** and **bufferFlushTime** properties), events in the buffer are batched up and enclosed by the string values specified by **batchHeader** and **batchFooter**.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:netcoolEventList  xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool">
    <tns:netcoolEvent type="update"
       xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool">
        <tns:netcoolField name="Serial" type="integer">1</tns:netcoolField>
        <tns:netcoolField name="Node" type="string">1.1.1.1</tns:netcoolField>
        <tns:netcoolField name="Acknowledged" type="integer">0</tns:netcoolField>
        <tns:netcoolField name="ServerName" type="string">NCOMS_P</tns:netcoolField>
        <tns:netcoolField name="ServerSerial" type="integer">1</tns:netcoolField>
    </tns:netcoolEvent>
    <tns:netcoolEvent type="update"
       xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool">
        <tns:netcoolField name="Serial" type="integer">2</tns:netcoolField>
        <tns:netcoolField name="Node" type="string">2.2.2.2</tns:netcoolField>
        <tns:netcoolField name="Acknowledged" type="integer">0</tns:netcoolField>
        <tns:netcoolField name="ServerName" type="string">NCOMS_P</tns:netcoolField>
        <tns:netcoolField name="ServerSerial" type="integer">2</tns:netcoolField>
    </tns:netcoolEvent>
</tns:netcoolEventList>
```

HTTP Transport configurations for event-format-oriented use cases

Table 22. Use Case 1: Gateway send batched events

Property name	Description
bufferSize	2 or above.
batchHeader	`<?xml version="1.0" encoding="UTF-8"? ><tns:netcoolEventList xmlns:tns=http:// item.tivoli.ibm.com/omnibus/netcool">`
batchFooter	`</tns:netcoolEventList>`

Use Case 2: Gateway send single event

Table 23. Use Case 2: Gateway send single event

Property name	Description
bufferSize	1
batchHeader	Leave empty.
batchFooter	Leave empty.

The socketTransport.properties file

Table 24 on page 70 describes the properties defined in the `socketTransport.properties` file, which is the transport module properties file associated with the socket transport module protocol.

Note : You configure the socket-related properties if you set the **Gate.XMLGateway.TransportType** and **Gate.XMLGateway.TransportFile** properties to `Socket` and `socketTransport.properties` in the `G_XML.props` properties file.

Table 24. Socket transport module properties	
Property name	**Description**
serverPort	Use this property to specify the port to which the events will be sent.
clientAddress	Use this property to specify the address to which the gateway connects. The format of the address is *host*:*port*.
subscribeMessages	This property is not used by the gateway.
subscribeResponses	Use this property to specify a list of response processors for responses to the subscribe messages.
disconnectMessages	Use this property to specify messages sent when disconnecting an asynchronous client.
disconnectResponses	Use this property to specify response processors for the disconnect messages.
activeAlarmsMessages	Use this property to specify messages sent when requesting active alarms as an asynchronous client.
messageTerminator	Use this property to specify a regex that should match the end of each alert from the target system.
readResponseAttempts	Use this property to specify how many times the probe should attempt to read a response to a message.
readerResponseDelay	Use this property to specify how long, in seconds, the probe should wait between attempts to read a response.
threadPoolSize	Use this property to specify the maximum number of threads to use to process connections made to the probe when it is running as a server.

Note : You must ensure that all property values in the transport properties file are unquoted; that is, contain no quotation marks.

The scalaTransport.properties file

Table 25 on page 71 describes the properties defined in the `scalaTransport.properties` file, which is the transport module properties file associated with the SCALA transport module protocol.

Note : You configure the SCALA-related properties if you set the **Gate.XMLGateway.TransportType** and **Gate.XMLGateway.TransportFile** properties to SCALA and `scalaTransport.properties` in the `G_SCALA.props` properties file.

The gateway uses the Java Secure Socket Extension (JSSE) framework to implement secure transport over Secure Socket Layer (SSL) and Transport Layer Security (TLS). The JSSE framework hides the complexities of the underlying protocols associated with SSL and TLS. Thus, the following transport properties defined in the `scalaTransport.properties` file map directly to the JSSE properties: **keyStore**, **keyStorePassword**, **trustStore**, and **trustStorePassword**.

Table 25. Properties in the scalaTransport.properties file	
Property name	**Description**
scalaURL	Use this property to specify the URL of the target data collector application to which the gateway connects. The default is `http(s)://`*`some.host.com`*`:`*`port`*`/Unity/`*`DataCollector`*.
scalaRetryMax	Use this property to specify the maximum number of attempts that the gateway makes to connect to the target data collector application before dropping the message. If you set this property to 0 (zero), the gateway tries to connect to the target data collector application indefinitely. The default is `0`. **Note :** While the IBM Operations Analytics - Log Analysis data collector is not reachable, events are buffered in the gateway. This allows the gateway to handle temporary or intermittent connectivity problems to the IBM Operations Analytics - Log Analysis data collector. A prolonged period of connectivity loss potentially results in event loss due to the gateway's inability to buffer large numbers of events if the event rate is extremely high. In situations where the event rate is high, it is unlikely that the gateway can send all of the data and catch up once the IBM Operations Analytics - Log Analysis data collector is available.
scalaRetryPeriod	Use this property to specify the amount of time (in seconds) between each reconnection attempt to the target data collector application. The default is `30` seconds.
keyStore	Use this property to specify the location of the Java keystore file that contains the private keys for any HTTPS ports. See the JSSE for Java security property **`javax.net.ssl.keyStore`**. The default is `$OMNIHOME/java/security/client.jks`.
keyStorePassword	Use this property to specify the password to access the private keys for any HTTPS ports from the keystore file specified in the **keyStore** property. See the JSSE for Java security property **`javax.net.ssl.keyStorePassword`**. The default is `password`.
trustStore	Use this property to specify the location of the Java trust store file that contains the server's public key for any HTTPS clients. See the JSSE for Java security property **`javax.net.ssl.trustStore`**). The default is `$OMNIHOME/java/security/cacerts.jks`.

Table 25. Properties in the scalaTransport.properties file (continued)

Property name	Description
trustStorePassword	Use this property to specify the password to access the server's public key for any HTTPS clients from the trust store file specified in the **trustStore** property. See the JSSE for Java security property **javax.net.ssl.trustStorePassword**. The default is `password`.
threadPoolSize	Use this property to specify the number of threads that the HTTP servers share to process incoming requests. The default is 16 threads.
username	Use this property to specify the username to use for authentication with the target data collector application. The default is no value (that is, the property is blank). Leave this property blank for no authentication.
password	Use this property to specify the password to use for authentication with the target data collector application. The **password** property is ignored if the **username** property is left blank. The default is no value (that is, the property is blank).
eventBufferSize	Use this property to specify the maximum number of events to contain in each batch of log record data that the transport module sends to the target data collector application. The default is `200` events.
eventBufferFlushTime	Use this property to specify the amount of time (in seconds) to wait for new events before flushing the buffer. The flush timer is reset on each event added to the batch. The default is 30 seconds.
enableTrace	Use this property to enable diagnostic tracing of communications between the transport module and the target data collector application. The default is `false` (that is, disable diagnostic tracing).
jsonMsgHostname	Use this property to specify the hostname that corresponds to the data source for received data. This will be the name of the host where the gateway is running. The default is `<local hostname>`.
jsonMsgPath	Use this property to specify the path that corresponds to the data source for received data. The default is `NCOMS`.

Table 25. Properties in the scalaTransport.properties file (continued)	
Property name	**Description**
readTimeout	Use this property to specify the length of time (in seconds) that the gateway allows for reading on a socket. The default is 30 seconds.

Note : You must ensure that all property values in the transport properties file are unquoted; that is, contain no quotation marks.

Encrypting gateway and transport module properties

The gateway supports the use of 128 bit, 192 bit, and 256 bit encryption keys when encrypting gateway and transport module string value properties. You perform a number of steps to enable the encryption of gateway and transport module string value properties.

The following list summarizes the tasks you need to perform to enable the encryption of gateway and transport module string value properties. Some of these tasks involve editing gateway properties files and transport module properties files. See "Summary" on page 2 for the gateway and transport module properties files you would use depending on whether you are running the gateway in standard mode or LA mode.

Note : Encrypting transport module string value properties requires transport module Version 10.0.23 or later. Also, for encrypted string values for passwords in the transport module properties files the gateway ensures that their associated unencrypted values do not appear in log files or Java error messages. However, the gateway does not guarantee this will happen for other properties.

- Set the following properties in the appropriate gateway properties file:
  - **ConfigCryptoAlg** - You can set this property to either the value `'AES_FIPS'` or the value `'AES'` (the default value).
  - **ConfigKeyFile** - You set this property to the file path and file name of the key file that was generated by the `nco_keygen` utility.

  The transport module inherits the configuration of **ConfigCrytoAlg** and **ConfigKeyFile** from the gateway properties.

  Note : You need to set the **ConfigCryptoAlg** property to the value `'AES_FIPS'` in the gateway properties file.
- Run the `nco_keygen` utility to generate a key and store it in a key file.
- Use the `nco_aes_crypt` utility to encrypt a gateway or transport property string value with the key that was generated by the `nco_keygen` utility.

  For details on the tasks associated with property value encryption, see the following topic on the Tivoli Netcool/OMNIbus Knowledge Center:

  https://www-304.ibm.com/support/knowledgecenter/SSSHTQ_8.1.0/com.ibm.netcool_OMNIbus.doc_8.1.0/omnibus/wip/admin/reference/omn_adm_propsfileencryption.html?lang=en
- Configure the Java Runtime Environment (JRE) for FIPS 140-2, which includes:
  - Editing the `java.security` file
  - Installing the "Unrestricted Java Cryptography Extension (JCE) policy files for SDK"

  For details on how to configure the JRE for FIPS 140-2, see the following topic on the Tivoli Netcool/OMNIbus Knowledge Center: https://www-304.ibm.com/support/knowledgecenter/SSSHTQ_8.1.0/com.ibm.netcool_OMNIbus.doc_8.1.0/omnibus/wip/install/task/omn_con_configuringjreforfips.html.

Using the transformer

The transformer reads the XML event stream and converts the event data into a format suitable for the destination application.

For the probe, the endpoint from which the events are generated determines which transformation is required. For the gateway, the message ID determines which transformation is required; the message ID will either be a hard coded value or `@col_name`, which then uses the value of the column specified as the ID to decide which transformation is required.

If the destination system is Netcool/OMNIbus, the probe uses the transformer to transform the XML messages into a set of typed name-value pairs. For this conversion, the transformer uses the `netcool2nvpairs.xsl` file to convert Netcool® XML events. For other XML events, the probe uses the XSLT file created for that type of XML event to generate the name-value pairs.

If the destination system is an XML event source, the gateway uses the transformer to transform a Netcool event into an XML representation of the event that can then be sent to another application. For this conversion, the transformer uses the XSLT file related to that particular XML event source.

This section contains the following topics:

XSLT files

Each XML event source generates events in a format that is specified by its own XML schema. You create an Extensible Stylesheet Language Translation (XSLT) file to transform events from that event source to another XML format, making it possible for applications to share XML events.

The following types of XSLT files are used by the transformer:

- Input XSLT files: These are used by the Probe for Message Bus to send events to the ObjectServer.
- Output XSLT files: These are used by the Gateway for Message Bus to send events from the ObjectServer.

The following table shows the XSLT files that are supplied with the transformer.

Table 26. XSLT files supplied with the transformer

XSLT file	Description
`$OMNIHOME/java/conf/cbe2nvpairs.xsl`	This XSLT file converts Common Base Event (CBE) events into name-value pairs for the probe to read.
`$OMNIHOME/java/conf/netcool2cbe.xsl`	This XSLT file converts a Netcool event into an event in CBE format.
`$OMNIHOME/java/conf/netcool2nvpairs.xsl`	This XSLT file converts Netcool events into name-value pairs that the probe can read.
`$OMNIHOME/java/conf/netcool2scala.xsl`	This XSLT file converts Netcool events into a format suitable for the target data collection application.
`$OMNIHOME/java/conf/netcool2wef.xsl`	This XSLT file converts Netcool events from the gateway into Web Services Distributed Management (WSDM) Event Format (WEF) events.

Table 26. XSLT files supplied with the transformer (continued)	
XSLT file	**Description**
`$OMNIHOME/java/conf/netcool2json.xsl`	This XSLT file converts Netcool events from the gateway into JavaScript Object Notation (JSON) events.
`$OMNIHOME/java/conf/wbe2nvpairs.xsl`	This XSLT file converts WebSphere Business Event (WBE) events into name-value pairs for the probe to read.
`wbepl2nvpairs.xsl`	This XSLT file includes `wbm2nvpairs.xsl` for handling WebSphere Business Monitoring (WBM) events that are contained within WBE events.
`wbm2nvpairs.xsl`	This example XSLT file contains support XSLT match functions for handling specific WBM trade array event elements.
`$OMNIHOME/java/conf/wef2nvpairs.xsl`	This XSLT file converts WEF events into name-value pairs for the probe to read.

If you require other types of XSLT files, you must create them. The following topics provide information that will help you to create XSLT files.

For details of the syntax required for XSLT files, see the XSL Transformations page on the W3C Web site:

http://www.w3.org/TR/xslt

Probe XSLT files

The input can be any XML message that is to be inserted into the ObjectServer. This type of XSLT file must generate a set of name-type-value elements in the following format:

name:*type*:"*value*"

where:

- *name* consists of alphanumeric characters and underscores.
- *type* can be `string`, `utc`, or `integer`.
- *value* is any arbitrary string that does not include a new-line character.

The module is supplied with an XSLT template, `addnvpairs.xsl`, that you must include in your XSLT file. You can call the template as a function, providing the function with the name, type, and value, to format the output correctly. To use the support template, include the XSLT file using the following XSLT include directive:

```
<xsl:include href="addnvpair.xsl"/>
```

Note : The `href` parameter for the file is relative to the location of the XSLT file that is including it. All XSLT files supplied with the module are installed in the same directory. If the XSLT file that you create is not in the same directory, you must specify the relative path to the support template within the `href` parameter. The module is supplied with a basic name-value pair XSLT file (`netcool2nvpairs.xsl`) which converts a Netcool XML event into a name-value pair for consumption by the probe.

The following example shows the content of the `netcool2nvpairs.xsl` with the include directive highlighted:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="1.0"
        xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool/nvpairs"
```

```
        xmlns:ens="http://item.tivoli.ibm.com/omnibus/netcool"
        exclude-result-prefixes="tns ens">
<xsl:output method="text"/>
<xsl:strip-space elements="*"/>

<xsl:include href="addnvpair.xsl"/>

<xsl:template match="/">
    <xsl:for-each select="ens:netcoolEvent">
        <xsl:call-template name="AddNVPair">
            <xsl:with-param name="name">
                <xsl:text>NetcoolEventAction</xsl:text>
            </xsl:with-param>
            <xsl:with-param name="type">
                <xsl:text>string</xsl:text>
            </xsl:with-param>
            <xsl:with-param name="value">
                <xsl:value-of select="@type"/>
            </xsl:with-param>
        </xsl:call-template>
        <xsl:apply-templates/>
    </xsl:for-each>
</xsl:template>

<xsl:template match="ens:netcoolEvent/ens:netcoolField">
    <xsl:call-template name="AddNVPair">
        <xsl:with-param name="name">
            <xsl:value-of select="@name"/>
        </xsl:with-param>
        <xsl:with-param name="type">
            <xsl:value-of select="@type"/>
        </xsl:with-param>
        <xsl:with-param name="value">
            <xsl:value-of select="."/>
        </xsl:with-param>
    </xsl:call-template>
</xsl:template>

</xsl:stylesheet>
```

Within this example there are two other directives that you should include in all probe XSLT files:

- `<xsl:output method="text"/>`

  This sets the XSLT to plain text, rather than the default of XML. As the output is required in name-type-value elements, it must be in plain text.
- `<xsl:strip-space elements="*"/>`

  This directive forces all unessential whitespace to be stripped from the output.

Gateway XSLT files

The input into a gateway XSLT file is an XML representation of an ObjectServer event. The output from this XSLT conversion must be in the format required by the target application. It can be an alternative XML format or a plain text string. You must determine which on an application-by-application basis.

The columns available within the XML are defined by the gateway mapping. The output is published by the gateway to the defined topic.

Using the transformer testing tool

The transformer testing tool helps you verify the XSLT file created for an XML event source.

The transformer testing tool allows you to check that the XSLT file that you have created for an XML event source generates XML events in the expected format. This tool runs the source XML through the XSLT file and prints the result of the XSLT transformation.

To start the transformer testing tool, run the following command:

```
java -cp $OMNIHOME/java/jars/Transformer.jar
com.ibm.tivoli.netcool.integrations.transformer.XSLTransformer XSLT_file
Source_file
```

where:

- *XSLT_file* is the name of the XSLT file that you are testing.
- *Source_file* is the name of the XML file conforming to the schema file of the event source.

Example output from the transformer testing tool

The following is sample output from the transformer testing tool:

```
java -cp $OMNIHOME/java/jars/Transformer.jar
com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer
netcool2nvpairs.xsl netcool.xml
Output from applying transformer 'netcool2nvpairs.xsl' to source file
'netcool.xml':-
NetcoolEventAction:string:update
Identifier:string:"GATEWAY:Gateway Reader@hostname.Mon Nov 10 14:37:55 2008"
NodeAlias:string:"hostname"
Manager:string:"ConnectionWatch"
Agent:string:""
AlertGroup:string:"Gateway"
AlertKey:string:"GATEWAY:Gateway Reader"
Severity:integer:"0"
Summary:string:"A GATEWAY process Gateway Reader running on
hostname has disconnected as username gateway"
StateChange:utc:"2008-11-10T14:38:33"
FirstOccurrence:utc:"2008-11-10T14:37:55"
LastOccurrence:utc:"2008-11-10T14:37:55"
InternalLast:utc:"2008-11-10T14:37:55"
Poll:integer:"0"
Type:integer:"1"
Tally:integer:"1"
Class:integer:"0"
Grade:integer:"0"
Location:string:""
OwnerUID:integer:"65534"
OwnerGID:integer:"0"
Acknowledged:integer:"0"
Flash:integer:"0"
EventId:string:""
ExpireTime:integer:"0"
ProcessReq:integer:"0"
SuppressEscl:integer:"0"
Customer:string:""
Service:string:""
PhysicalSlot:integer:"0"
PhysicalPort:integer:"0"
PhysicalCard:string:""
TaskList:integer:"0"
NmosSerial:string:""
NmosObjInst:integer:"0"
NmosCauseType:integer:"0"
LocalNodeAlias:string:""
LocalPriObj:string:""
LocalSecObj:string:""
LocalRootObj:string:""
RemoteNodeAlias:string:""
RemotePriObj:string:""
RemoteSecObj:string:""
RemoteRootObj:string:""
X733EventType:integer:"0"
X733ProbableCause:integer:"0"
X733SpecificProb:string:""
X733CorrNotif:string:""
URL:string:""
ExtendedAttr:string:""
ServerName:string:"NCOMS"
ServerSerial:integer:"1841"
```

Configuring the transformer definition file

The transformer definition file maps the XML event sources to their related XSLT files, and directs the probe and the gateway to use the XSLT file associated to the XML event source.

The `transformers.xml` transformer definition file, located in the `$OMNIHOME/java/conf` directory, defines how the messages that the probe reads, or that the gateway sends, are transformed. This file is divided into two logical sections, one for the probe (southbound) and one for the gateway (northbound).

After creating an XSLT file for each event source, you create a transformer entry in the transformer definition file. This enables the transformer to use the specified XSLT file when transforming events for that event source.

By default, the probe section enables the following transformations:

- messages received on a topic name of `cbe` are transformed by the `cbe2nvpairs` XSLT file
- messages received on a topic name of `wef` are transformed by the `wef2nvpairs` XSLT file
- messages received on a topic name of `wbe` are transformed by the `wbe2nvpairs` XSLT file
- messages received on a topic name of `netcool` are transformed by the `netcool2nvpairs` XSLT file

By default, the gateway section enables the following transformations:

- Netcool events that have an identifier of `cbeEvents` are transformed into CBE events (using the netcool2cbe XSLT file) and published to the JMS using the topic `cbe`
- Netcool events that have an identifier of `wefEvents` are transformed into WEF events (using the netcool2wef XSLT file) and published to the JMS using the topic `wef`
- Netcool events that have an identifier of `netcoolEvents` are published to the JMS using the topic `netcool` (without using an XSLT transformer)

Each entry takes the following format:

```
<tns:transformer name="transformer_name" type="southbound | northbound"
endpoint="event_endpoint" className="class_name">

<tns:property name="property_name" type="property_type" value="property_value"
description="description"/>

</tns:transformer>
```

where:

- `transformer_name` specifies the name of the transformer definition for an event source.
- `type` specifies the type of transformer being defined.

  This is either `southbound` for transformers used by the probe, or `northbound` for transformers used by the gateway.
- `event_endpoint` maps the endpoint from which the event arrived to an XSLT transformer and determines the endpoint to which the event source sends events.
- `class_name` is the name of the class to be used.
- `property_name` is the name of a property to be set in the transformation.

  For XSLT transformers, this is the name of the XSLT file to use.
- `property_type` identifies the type of entry that is set by the `property_name` field.

  For an XSLT file name, this would be `string`.
- `property_value` is the field that specifies the path to the XSLT file created for the event source.
- `description` specifies the description for the XSLT file created for the event source.

Sample transformer file

The following is a sample transformer file with transformers for various event sources:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <tns:transformers
    xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool/transformer"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

< !-- Southbound (probe) transformer definitions -->

< tns:transformer name="cbe2nvpairs" type="southbound" endpoint="cbe"
   className="com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer">

< tns:property name="xsltFilename" type="java.lang.String" value="${OMNIHOME}/java
  /conf/cbe2nvpairs.xsl" description="XSLT file for converting CBE events to
  name/value pairs"/>

< /tns:transformer>

< tns:transformer name="wef2nvpairs" type="southbound" endpoint="wef"
    className="com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer">

< tns:property name="xsltFilename" type="java.lang.String" value="${OMNIHOME}/
    java/conf/wef2nvpairs.xsl" description="XSLT file for converting WEF events
    to name/value pairs"/>

< /tns:transformer>

< tns:transformer name="netcool2nvpairs" type="southbound" endpoint="netcool"
   className="com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer">

< tns:property name="xsltFilename" type="java.lang.String" value="${OMNIHOME}/
    java/conf/netcool2nvpairs.xsl" description="XSLT file for converting Netcool
    events to name/value pairs"/>

< /tns:transformer>

< !-- Northbound (gateway) transformer definitions -->

< tns:transformer name="netcool2wef" type="northbound" endpoint="wef"
    className="com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer">

< tns:property name="xsltFilename" type="java.lang.String" value="${OMNIHOME}/
    java/conf/netcool2wef.xsl" description="XSLT file for converting Netcool
    events to WEF events"/>

< /tns:transformer>

< tns:transformer name="netcool2cbe" type="northbound"  endpoint="cbe"
    className="com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer">

< tns:property name="xsltFilename" type="java.lang.String" value="${OMNIHOME}/
    java/conf/netcool2cbe.xsl" description="XSLT file for converting Netcool
    events to CBE events"/>

< /tns:transformer>

< tns:transformer name="netcoolEvents" type="northbound" endpoint="netcool"
    className="com.ibm.tivoli.netcool.integrations.transformer.
    EmptyTransformer">

< /tns:transformer>

< /tns:transformers>
```

You can prevent events from an event source being transformed by specifying an empty transformer for that event source. The following example shows the entry for an event source with empty transformer details:

```
<tns:transformer name="empty" id="empty" className="com.micromuse.common.
     transformer.EmptyTransformer"/>
  </tns:transformers>
```

Using the XML validation tool

You can use the XML validation tool to verify transformed XML events.

The XML validation tool allows you to validate transformed XML events against the XML schema of an event source.

To validate the XML output, run the following command:

```
java -cp $OMNIHOME/java/jars/Transformer.jar
com.ibm.tivoli.netcool.integrations.transformer.XMLValidator XML_schema
XML_file
```

where:

- *XML_schema* is the XML schema against which you are validating the XML output.
- *XML_file* is the name of the XML output file whose format you are validating.

Sample implementation using JMS

This section describes a gateway implementation with an Enterprise Service Bus (ESB) using JMS.

The following diagram shows how the gateway can work with an Enterprise Service Bus using JMS:

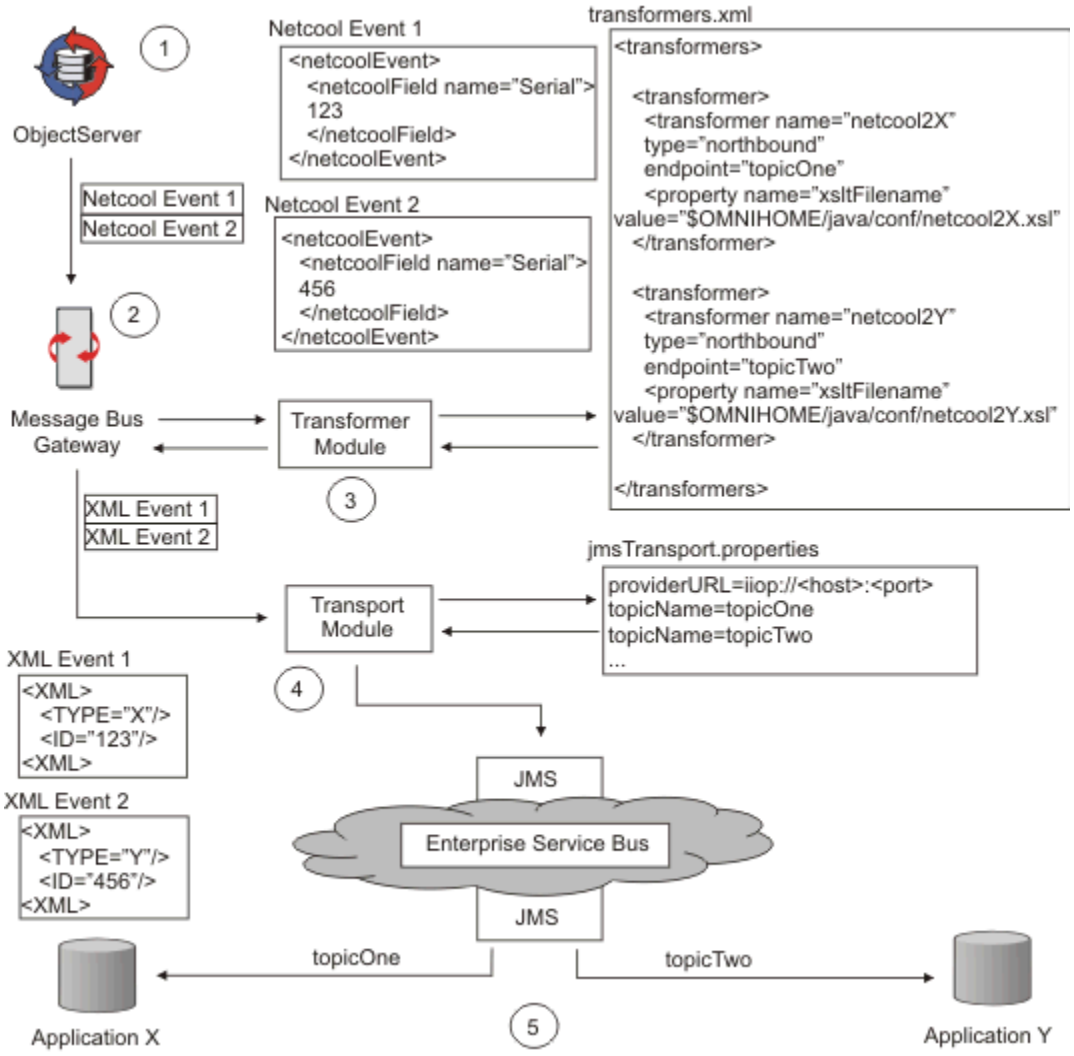

Figure 1. Gateway acquiring data from the ObjectServer and sending it to target applications using JMS

In this example, the ObjectServer generates events whose data needs to be written in XML event format to two different applications, Application X and Application Y.

The flow of data between the ObjectServer and the applications is as follows:

1. The ObjectServer generates two events, Netcool Event 1 and Netcool Event 2, that must be written in XML format to the two different applications:

Netcool Event 1 must be converted to XML format and written to Application X.

Netcool Event 2 must be converted to XML format and written to Application Y.

2. The gateway reads Netcool Event 1 and Netcool Event 2.
3. The gateway uses the transformer module to convert Netcool Event 1 and Netcool Event 2 into XML events using the transformation defined by the `transformers.xml` file.

   For each event source, the `transformers.xml` file contains a transformer entry. Within this entry, the transformation name (which equates to the message ID of the Netcool event) determines which .xsl file to use to convert the event to XML format and to which endpoint to publish the XML event.

   The transformer module converts Netcool Event 1 into XML Event 1 (an event of type X).

   The transformer module converts Netcool Event 2 into XML Event 2 (an event of type Y).
4. The gateway uses the transport module to publish XML Event 1 to Topic 1 and to publish XML Event 2 to Topic 2.
5. Application X subscribes to Topic 1 and receives XML Event 1. Application Y subscribes to Topic 2 and receives XML Event 2.

Error messages

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

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

Table 27. Error messages

Error	Description	Action
`Failed to get a handle to the properties manager`	The gateway could not install a property handle.	Contact IBM software support.
`Failed to send message:`	The gateway could not send the message to the destination application using the transport module.	Check that the transport module is running correctly.
`Failed to create XML netcool event:` `Failed to transform netcool event XML:`	The transformer could not convert the Netcool events to the appropriate XML format.	Check the transformer definition for the event source in the transformer definition file.
`Failed to look-up entry:`	The message ID could not be found in the alert.	Make sure the message ID field is set in the alert.
`Connection to target system has been lost. Returning closed connection state.`	The gateway's connection to the target system has been lost.	Check that the target system is running as expected.
`Lost connection to target system`	The gateway's connection to the target system has been lost.	Check that the target system is running as expected.

Transporter and transformer error messages

The transporter and transformer modules generate error messages.

The following table describes the error messages that are generated by the transporter and transformer modules.

Table 28. Common error messages

Error	Description	Action
`Failed to get message text`	The JMS transport module failed to get the text from the JMS message that it received.	There was a problem with the format of the message received from the JMS.
`Unsupported message type`	The JMS has received a message that is not a text message.	Only messages in text format are supported.
`No transformer defined for name` *`name`*	The gateway is trying to find the correct transformer to use to transform an event, but the message ID for that transformer is not present in the `transformers.xml` file.	Add a transformer to the `transformers.xml` file that corresponds to the message ID.
`No endpoint defined for name` *`name`*	The gateway is trying to find the correct endpoint to send a message to, but the transformer in the `transformers.xml` file for this message does not specify an endpoint.	Specify an **`endpoint`** for the transformer that corresponds to this message.
`No transformer defined for endpoint` *`endpoint`*	The probe has received an event from an endpoint for which there is no transformer in the `transformers.xml` file.	Add a transformer to the `transformers.xml` file for this endpoint.
`Invalid entry in transformer definition file - duplicate endpoint` *`endpoint`* `in southbound transformer entry`	Two southbound transformers in the `transformers.xml` file have been specified with the same endpoint.	You can only create one southbound transformer for each endpoint. Remove one of the transformers from the `transformers.xml` file.
`Invalid entry in transformer definition file - duplicate name` *`name`* `in northbound transformer entry`	Two northbound transformers in the `transformers.xml` file have been specified with the same name.	You cannot create two northbound transformers with the same name. Rename one of the transformers.
`Invalid transformer type` *`type`* `specified in transformers file`	A transformer in the `transformers.xml` file specifies a transformer type that is not supported.	The message type should either be northbound or southbound. Update the transformer in the `transformers.xml` file.

Table 28. Common error messages (continued)

Error	Description	Action
`Property type type does not exist`	The type specified for a property definition with a transformer is not a Java object.	Correct the transformer in the `transformers.xml` file by specifying a valid Java object name, for example, java.lang.String.
`Invalid property name propName in transformer name`	The name specified for a property definition with a transformer is not valid.	Correct the transformer in the `transformers.xml` file by specifying a valid name. This should be XSLT file name.
`Method method_name in class transformerClass has thrown an exception`	The method specified in the transformer definition file has failed.	Use the information in the message to diagnose the problem.
`Failed to read transformer definition file`	The transformer module failed to read the transformer definition file (`transformers.xml`).	Check that the file is specified correctly and that the permissions on the `transformers.xml` file are set correctly.
`Unknown property`	A transformer in the `transformers.xml` file contained a property that is not recognized.	Update the transformer in the `transformers.xml` file.
`Failed to transform message`	The transformer module could not transform an XML event.	There may be a problem with the XLST file. Try testing it using the transformer validation tool.
`Failed to read XSLT file`	The transformer module failed to read an XSLT file specified within the transformer definition file.	Check that the file is specified correctly and that the permissions on the .xsl file are set correctly.
`Invalid transformer class name found`	A transformer in the `transformers.xml` file contains an invalid transformer class name.	Check that you have specified the transformer class correctly in the `transformers.xml` file.
`Failed to close file` `Failed to find file`	The named file was not found.	Check that the name of the file has been specified correctly in the properties file.
`Stream file filename is not a stream capture file`	The file identified is not a stream capture file and cannot be used by the transport module.	Specify an alternative file name in the transport properties file.
`I/O exception whilst opening file`	The transport module could not open the file specified.	Check the permissions on the file indicated.
`Failed to read object from file`	The probe failed to read the stream capture file.	Try recreating the stream capture file and running it through the probe again.

Table 28. Common error messages (continued)

Error	Description	Action
`Error in stream capture file`	The stream capture file identified contains an error.	Try recreating the stream capture file and running it through the probe again.
`Failed to find initial context`	The class name in the transport properties file is incorrect.	Check the setting of the **`initialContextFactory`** property.
`Failed to create subscriber for topic: topicName`	The transport module failed to subscribe to a topic with the JMS.	Use the information in the message to diagnose the problem.
`JNDI lookup for topic: topicName` `Failed to find topicName in JNDI`	The transport module could not find one of the topics specified in the JNDI.	Check that the topic has been specified correctly in the transport file.
`Failed to start JMS subscriber`	The transport module failed to start the JMS subscriber.	Use the information in the message to diagnose the problem.
`Failed to send message`	The transport module failed to send a message to the endpoint specified in the transformer.	Use the information in the message to diagnose the problem.
`Failed to find topicName in JNDI`	The transport module cannot find one of the topics in the JNDI.	Check that the topic has been specified correctly in the transport file.
`Failed to close topic connection`	The transport module failed to close a topic connection within JMS.	Use the information in the message to diagnose the problem.
`Invalid property propName`	The transport properties file contains a property that is not supported.	Update the transport properties file.
`Invalid transport class name found`	The **`transportModule`** property of the transport file contains an invalid class name.	Change the value of the **`transportModule`** property to a class that exists in the jar file.
`Failed to load properties file`	The transport module could not read the transport properties file.	Check that the properties file exists and that the permissions are set correctly.
`Invalid property propName in file transportFile does not exist in transport class transportClass`	The transport file contained a property that is not supported by the transport module.	Update the transport properties file. For a description of the all the properties that the transport file supports, see "Configuring transport module properties files" on page 61.

Table 28. Common error messages (continued)		
Error	**Description**	**Action**
`Method methodName in class transportClass has thrown an exception`	The method specified in the transport file has failed. It depends on the method but the method would probably tell you what's wrong.	Use the information in the message to diagnose the problem.
`key + " contains non-existent environment variable value`	A value in the transport properties file or in the transformer file contains an environment variable which does not exist.	Set the environment variable specified to an appropriate value.
`Failed to create JMS connection`	The transport module failed to create a connection to the JMS.	Check that you have specified the parameters correctly in the transport properties file.

Running the gateway

This topic describes how to run the gateway.

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

`$OMNIHOME/bin/nco_g_xml -propsfile $OMNIHOME/etc/gateway.props`

Where *gateway* is `G_XML` or `G_SCALA`.

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

`%OMNIHOME%\bin\win32\nco_g_xml.exe -propsfile %OMNIHOME%\etc\gateway.props`

Where *gateway* is `G_XML` or `G_SCALA`.

Running the gateway as a Windows service

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

1. Run a console as an administrator.
2. In the console, install the gateway as a Windows service using the following command:

   ```
   > nco_g_xml.exe /INSTALL /CMDLINE "-propsfile %OMNIHOME%\etc\G_XML.props"
   ```
3. Since auto-sourcing of `CLASSPATH` is not possible in service mode, the `CLASSPATH` must be configured manually: To configure the **`Gate.Java.ClassPath`** property, use the following steps:
   a. Run `%OMNIHOME%\bin\win32\nco_g_xml.bat <output_file>` to generate `CLASSPATH`.
   b. Copy the `CLASSPATH` string from the `<output_file>`.
   c. Edit the `G_XML.props` file (see the argument passed in the service installation),
   d. Paste the `CLASSPATH` string to **`Gate.Java.ClassPath`**,

   Important : Convert all single slash in the paths to double backslashes (i.e. \\).
4. Go to **Windows Services**. Right click on **Netcool/Omnibus XML Gateway** and go to **Properties**. In the **`Start`** parameter specify the parameter you will use to start the gateway. For example: `-messagelevel debug -propsfile %OMNIHOME%/etc/G_XML.props`.
5. Start the **XML Gateway** service.
6. For Omnibus 810, copy the `nco_g_xml jar` file from `%OMNIHOME%/bin/win32` to `%OMNIHOME%/bin` and follow the steps above to start the Windows Service.

To remove the gateway as a Windows service, run the following command:

```
nco_gxml /REMOVE
```

Note : If the service is not removed from Windows Services, restart Windows.

Using the gateway with the Probe for Message Bus

You can use the Gateway for Message Bus with the Probe for Message Bus to process XML messages stored in a file or transmitted using the JMS, HTTP/HTTPS, or MQTT transport protocols.

You can use the probe and gateway together as a single implementation. In this scenario, the probe uses the transport module to acquire XML events, uses the transformer module to convert them into name-value pairs, tokenizes them, and then sends them as Netcool events to the ObjectServer. The gateway reads Netcool XML events from the ObjectServer, uses the transformer module to convert them into a format appropriate for the destination application, and uses the transport module to send the transformed events to their destination application.

The following sections describe how to use the probe and gateway together as a single implementation:

- "Requirements" on page 86
- "Sample implementation using JMS" on page 86
- "Installing and running the components" on page 88

Requirements

Several software packages are required to operate the Probe for Message Bus and the Gateway for Message Bus together.

You can download the probe, the gateway, and all required installation packages from the IBM Passport Advantage® Online website:

http://www-306.ibm.com/software/howtobuy/passportadvantage/

To use the probe and the gateway together, you will require the following packages:

- omnibus-*arch*-common-transformer-*version*
- omnibus-*arch*-common-transportmodule-*version*
- omnibus-*arch*-gateway-libngjava-*version*
- omnibus-*arch*-probe-nonnative-base-*version*
- omnibus-*arch*-gateway-nco-g-xml-*version*
- omnibus-*arch*-probe-nco-p-xml-*version*

where *arch* is the operating system you are installing the components on and *version* is the package version.

Sample implementation using JMS

This section describes an implementation of the probe and gateway working together.

The following diagram shows an example of the probe and gateway working together, with an Enterprise Service Bus (ESB) using JMS:

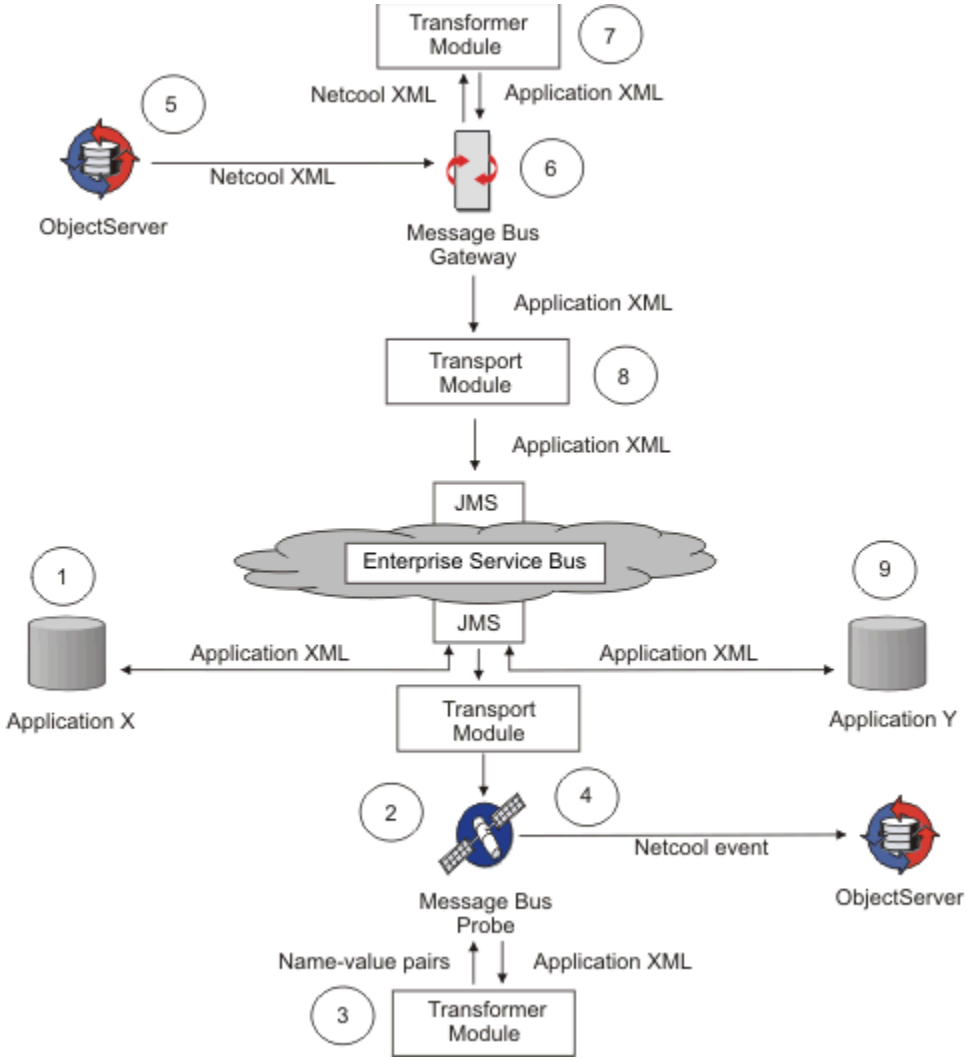

Figure 2. Probe and gateway working with ESB using JMS

The flow of data between XML event sources and consumers, and Netcool/OMNIbus is as follows:

1. Applications generate XML events and publish them to topics in the ESB.
2. The probe uses the transport module to subscribe to the topics in the ESB and receives the XML events published to those topics.
3. The probe uses the transformer module to convert the XML events in name-value pairs using the transformation defined by the `transformers.xml` file.

   For each event source, the `transformers.xml` file contains a transformer entry. This entry identifies the source of the XML event and determines which .xsl file to use to convert the XML event.
4. The probe parses the name-value pairs in Netcool events and sends them to the ObjectServer.
5. The ObjectServer generates events that need to be written in XML format to various applications.
6. The gateway reads the Netcool events generated by the ObjectServer.
7. The gateway uses the transformer module to convert the Netcool events into XML events using the transformation defined by the `transformers.xml` file.

   For each event source, the `transformers.xml` file contains a transformer entry. Within this entry, the transformation name (which equates to the message ID of the Netcool event) determines which .xsl file to use to convert the event to XML format and to which endpoint to publish the XML event.
8. The gateway uses the transport module to publish the transformed XML events to topics in the ESB.
9. The applications subscribe to the topics and receive the XML events published by the gateway.

Note : XML events are not always transformed. For example, if the event source is a Gateway for Message Bus and the event consumer is a Probe for Message Bus, the XML events are not transformed by the transformer module.

Installing and running the components

To use the probe and gateway together, you must install the required components in the correct order.

Install the components in the following order:

1. `common-transformer`
2. `common-transportmodule`
3. `gateway-libngjava`
4. `probe-nonnative-base`
5. `gateway-nco-g-xml`
6. `probe-nco-p-xml`

Running the probe and gateway together

Start the probe first, using the following command:

```
$OMNIHOME/probes/nco_p_xml
```

When the probe is running, start the gateway using the following command:

```
$OMNIHOME/bin/nco_g_xml
```

Known issues with the Gateway for Message Bus

This section explains some known issues with the Gateway for Message Bus.

- The gateway repeats reconnection attempt even when the retry countdown reaches zero until the transport connection is established.
- In Windows the gateway cannot shut down gracefully after **Ctrl-C**.

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

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

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

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

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

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

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

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

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

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

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

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

IBM Corporation
Software Interoperability Coordinator, Department 49XA

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

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

Trademarks

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

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

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

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

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

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

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

Part Number:

(1P) P/N:

SC14-7650-16