

Netcool/OMNIbus ObjectServer Gateway
Version 7 Release 3

Reference Guide



Netcool/OMNIbus ObjectServer Gateway
Version 7 Release 3

Reference Guide



Note

Before using this information and the product it supports, read the information in “Notices” on page 55.

Edition notice

This edition applies to version 7 release 3 modification 1 of IBM Tivoli Netcool/OMNIBus ObjectServer Gateway (product number 5724-S42) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1996, 2011.

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

Contents

Chapter 1. IBM Tivoli Netcool/OMNIBus

ObjectServer Gateway 1

Features of the ObjectServer Gateway 1

Unidirectional ObjectServer Gateways 2

Bidirectional ObjectServer Gateways 3

Chapter 2. Installing the gateway on Tivoli Netcool/OMNIBus V7.3.0 or later . 5

Installing the gateway on UNIX and Linux operating
systems 5

Installing the gateway on Windows operating
systems 6

Chapter 3. Configuration of the unidirectional ObjectServer Gateway . . 7

Unidirectional gateway properties 7

Hash table cache 16

Error handling 17

Process Agent control 17

Authentication 17

Buffer size for unidirectional ObjectServer

Gateways 17

Secure connections for unidirectional

ObjectServer Gateways 18

Failback for unidirectional ObjectServer

Gateways 18

Store and forward for unidirectional ObjectServer

Gateways 19

Resynchronization properties of unidirectional

ObjectServer Gateways 19

Chapter 4. Configuration of the bidirectional ObjectServer Gateway . . 21

Bidirectional gateway properties 21

Hash table cache 35

Error handling 36

Process Agent control 36

Authentication 36

Buffer size for bidirectional ObjectServer

Gateways 36

Secure ObjectServer connections for bidirectional

ObjectServer Gateways 37

Failback for bidirectional ObjectServer Gateways . 37

Store and forward for bidirectional ObjectServer

Gateways 38

Alternative deletion strategy. 38

Resynchronization properties of bidirectional

ObjectServer Gateways 39

Chapter 5. ObjectServer Gateway mapping 41

Mapping attributes 42

Example mapping 43

Chapter 6. Startup command file . . . 49

SHOW PROPS 49

GET CONFIG 49

FAILOVER SYNCH. 49

Chapter 7. Table replication definition file 51

Example table replication definition file 53

Notices 55

Trademarks 57

Index 59

Chapter 1. IBM Tivoli Netcool/OMNIBus ObjectServer Gateway

The ObjectServer Gateway is used to replicate table data (for example, alert-related data) between different IBM Tivoli Netcool/OMNIBus ObjectServers.

ObjectServer Gateways consist of readers and writers. Readers extract alerts from a source ObjectServer. Writers send the alert data to a target ObjectServer. An ObjectServer Gateway can be unidirectional or bidirectional. ObjectServer Gateways can be used to:

- Maintain a backup ObjectServer.
- Replicate alerts between different Network Operations Centers (NOCs).
- Create a tiered architecture.

The following table provides a summary of the gateway:

Table 1. Summary of the ObjectServer Gateway

Gateway target	IBM Tivoli Netcool/OMNIBus ObjectServer V7.1 or later
Gateway executable filename	nco_g_objserv_uni for a unidirectional gateway nco_g_objserv_bi for a bidirectional gateway Note: Both binaries use the Netcool® Gateways Toolkit (NGTK) library, which provides the basic framework for the gateway process and are configured independently of each other. The NGTK library is installed as a part of Tivoli Netcool/OMNIBus.
Patch number	1.0
Gateway supported on	Solaris, AIX®, HP-UX, Linux, Linux for System z, Windows
Configuration files	\$OMNIHOME/etc/server_name.props
Requirements	A currently supported version of TivoliTivoli Netcool/OMNIBus
Licensing	Electronic licensing is no longer implemented in IBM® Tivoli® Netcool products. All IBM Tivoli Netcool products now use the IBM software licensing process.
Remote connectivity	Yes
Failover or failback functionality	Available

Features of the ObjectServer Gateway

Unidirectional and bidirectional ObjectServer Gateways share functions that they use to pass data between ObjectServers.

Passing table data

The gateway can replicate the data in any table between ObjectServers. Details of the tables to be replicated are stored in the table definition file.

Centralized property management

The gateway uses centralized property management and separates properties from data processing configuration. Configuration of the unidirectional and bidirectional gateways is performed using configuration files.

Failback

The failback function comes into operation when a gateway loses its connection to the primary ObjectServer; this enables the gateway to connect to a backup ObjectServer. The failback function also enables the gateway to reconnect to the primary ObjectServer when it becomes active again.

Unidirectional ObjectServer Gateways

The unidirectional ObjectServer Gateway enables alerts to flow in one direction, from a source ObjectServer to a destination ObjectServer.

Changes made in the source ObjectServer are reflected in the destination ObjectServer, but changes in the destination ObjectServer are not reflected in the source ObjectServer.

The following figure shows the configuration of a unidirectional ObjectServer Gateway:

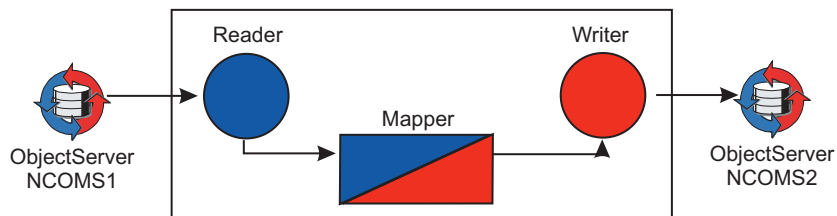


Figure 1. Unidirectional ObjectServer Gateway

Information flow

The unidirectional 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 receives data from the reader component, transforms the data into an appropriate form for the target writer using the map definition file, and passes the data to the writer component.
3. The writer component receives the source data from the mapper component and writes it to the destination ObjectServer.

Bidirectional ObjectServer Gateways

The bidirectional ObjectServer Gateway enables alerts to flow in both directions between a source and a destination ObjectServer.

Any changes made in the source ObjectServer are replicated in the destination ObjectServer, and changes in the destination ObjectServer are replicated in the source ObjectServer. This ensures that both ObjectServers contain the same alerts and allows you to maintain a backup ObjectServer.

The following figure shows the configuration of a bidirectional ObjectServer Gateway:

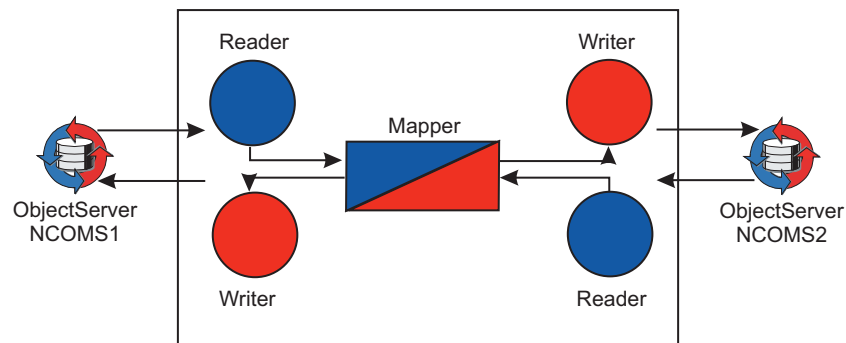


Figure 2. Bidirectional ObjectServer Gateway

Information flow

The bidirectional gateway is comprised of a mapper and two reader/writer components (one for each ObjectServer). The flow of information between the components is as follows:

1. A source reader/writer component reads from the source ObjectServer and passes the data to the mapper component.
2. The mapper component receives data from the source reader/writer component, transforms the data into an appropriate form using the map definition file, and passes the data to the target reader/writer component.
3. A second reader/writer component receives the source data from the mapper component and writes it to the destination ObjectServer.

Note: Bidirectional gateways can be used to create a failover pair of ObjectServers or to communicate between virtual pairs of ObjectServers.

Chapter 2. Installing the gateway on Tivoli Netcool/OMNIBus V7.3.0 or later

With the introduction of Tivoli Netcool/OMNIBus V7.3.0, all gateways are installed using the Tivoli Netcool/OMNIBus installer. You can install the gateway using the installation wizard, using a text-based installer (console mode), or using settings predefined in a text file (silent mode).

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

Installing the gateway on UNIX and Linux operating systems

To install the gateway on UNIX and Linux operating systems, use the following steps:

1. Download the installation package for the gateway from the Passport Advantage Online Web site:
`http://www-306.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm`
2. Make a backup of any existing configuration files that you want to retain.
3. Extract the contents of the installation package to a temporary directory.
4. To install the gateway using the installation wizard, use the following steps:
 - a. Run the following command:
`$NCHOME/omnibus/install/nco_install_integration`
 - b. When the installation wizard starts, specify the extracted directory that contains the README.txt file as the location of the gateway installation files.
 - c. Accept the license conditions.
5. To install the gateway using console mode, use the following steps:
 - a. Run the following command:
`$NCHOME/omnibus/install/nco_install_integration -i console`
 - b. When the text-based installer starts, specify the extracted directory that contains the README.txt file as the location of the gateway installation files.
 - c. Accept the license conditions.
6. To install the gateway using silent mode, use the following steps:
 - a. Create a text file named `reponse.txt` and add the following entries:
`PROBE_OR_GATE_LOCATION=README_directorypath`
`LICENSE_ACCEPTED=true`
where `README_directorypath` is the path to the directory containing the README.txt file in the extracted package.
 - b. Run the following command:
`$NCHOME/omnibus/install/nco_install_integration -i silent -f response_path/response.txt`
where `response_path` is the full path to the response.txt file.

In each case, the gateway is installed in the `$NCHOME/omnibus/gates` directory.

Installing the gateway on Windows operating systems

To install a gateway on Windows operating systems, use the following steps:

1. Download the installation package for the gateway from the Passport Advantage Online Web site:
`http://www-306.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm`
2. Make a backup of any existing configuration files that you want to retain.
3. Extract the contents of the package to a temporary directory.
4. To install the gateway using the installation wizard, use the following steps:
 - a. Run the following command:
`%NCHOME%\omnibus\install\nco_install_integration`
 - b. When the installation wizard starts, specify the extracted directory that contains the README.txt file as the location of the gateway installation files.
 - c. Accept the license conditions.
5. To install the gateway using console mode, use the following steps:
 - a. Run the following command:
`%NCHOME%\omnibus\install\nco_install_integration -i console`
 - b. When the text-based installer starts, specify the extracted directory that contains the README.txt file as the location of the gateway installation files.
 - c. Accept the license conditions.
6. To install the gateway using silent mode, use the following steps:
 - a. Create a text file named `reponse.txt` and add the following entries:
`PROBE_OR_GATE_LOCATION=README_directorypath`
`LICENSE_ACCEPTED=true`
where `README_directorypath` is the path to the directory containing the README.txt file in the extracted package.
 - b. Run the following command:
`%NCHOME%\omnibus\install\nco_install_integration -i silent -f response_path\response.txt`
where `response_path` is the full path to the response.txt file.

In each case, the gateway is installed in the `%NCHOME%\omnibus\gates` directory.

Chapter 3. Configuration of the unidirectional ObjectServer Gateway

The unidirectional gateway is configured using a properties file. This 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.

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

```
nco_g_objserv_uni -name
```

Note: To reduce latency, run the gateway on the same box to which it writes alerts. You can also reduce latency by adjusting the buffer size.

The default location for the properties file for the unidirectional gateways is:

```
$OMNIHOME/etc/server_name.props
```

The default properties file must be copied to the \$OMNIHOME/etc folder.

Note: The properties files for the unidirectional and bidirectional gateways contain similar properties; however, they are described separately for clarity.

Unidirectional gateway properties

You can configure ObjectServer Gateways by using properties defined in a properties file. The unidirectional ObjectServer Gateway and the bidirectional ObjectServer Gateway have some of these properties in common, but each set of properties is described separately for ease of reference.

For information about the common properties and Interprocess Communication (IPC) properties, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* SC14-7608. The following table describes the common gateway properties.

Table 2. Properties and command line options used by unidirectional gateways

Property name	Command line option	Description
Gate.CacheHash TblSize <i>integer</i>	<code>-chashtblsize integer</code>	Use this property to specify the size (in elements) that the gateway allocates for the hash table cache. The default is 5023.
Gate.MapFile <i>string</i>	<code>-mapfile string</code>	Use this property to specify the location of the map definition file. The default is \$OMNIHOME/gates/objserv_uni/objserv_uni.map.
Gate.StartupCmdFile <i>string</i>	<code>-startupcmdfile string</code>	Use this property to specify the location of the startup command file. The default is \$OMNIHOME/objserv_uni/objserv_uni.startup.cmd.

Table 2. Properties and command line options used by unidirectional gateways (continued)

Property name	Command line option	Description
Gate.Transfer.FailoverSyncRate <i>integer</i>	<code>-fsynccrate</code> <i>integer</i>	Use this property to specify the rate (in minutes) of the failover synchronization. The default is 60.
Gate.NGtkDebug <i>boolean</i>	<code>-ngtkdebug</code> <i>boolean</i>	Use this property to specify whether the NGTK library should log debug messages. The default is TRUE. Note: You can specify which debug messages are included in the debug log file using the Gate.Mapper.Debug , Gate.Reader.Debug and Gate.Writer.Debug properties.
Gate.PAAware <i>integer</i>	<code>-paaware</code> <i>integer</i>	Use this property to specify 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.
Gate.PAAwareName <i>string</i>	<code>-paname</code> <i>string</i>	Use this property to specify the name of the Process Agent 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.UsePamAuth <i>boolean</i>	<code>-usepamauth</code> <i>boolean</i>	Use this property to specify whether PAM authentication is used. The default is FALSE. Note: To run the gateway in FIPS 140-2 mode, you must set this property to TRUE.
Gate.UnixAdminGroup <i>string</i>	<code>-unixadmingroup</code> <i>string</i>	Use this property to specify the administration group to which the gateway must belong if standard UNIX authentication is used. The default is " ".
MaxLogFileSize <i>integer</i>	<code>-maxlogfilesize</code> <i>integer</i>	Use this property to specify the maximum size (in kilobytes) of 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.

Table 2. Properties and command line options used by unidirectional gateways (continued)

Property name	Command line option	Description
OldTimeStamp <i>boolean</i>	<code>-oldtimestamp</code> <i>boolean</i>	<p>Use this property to specify old-style timestamp format the gateway uses in the log file.</p> <p>Set the value to TRUE to specify the timestamp format used in Tivoli Netcool/OMNIBus V7.2.1, or earlier.</p> <p>For example: dd/MM/YYYY hh:mm:ss AM or dd/MM/YYYY hh:mm:ss PM when the locale is set to en_GB on a Solaris 9 computer.</p> <p>Set this value to FALSE to display the timestamp in ISO 8601 format.</p> <p>For example: YYYY-MM-DDThh:mm:ss, where T separates the date and time, and hh is in 24-hour clock.</p> <p>The default is FALSE.</p> <p>Note: Do not set the OldTimeStamp property to TRUE when running in UTF-8 mode.</p>
N/A	<code>-utf8enabled</code> <i>boolean</i>	<p>Use this command line option to control the encoding of data that is passed into, or generated by, this gateway when running on Windows.</p> <p>Set the value of <code>-utf8enabled</code> to TRUE to run the application in UTF-8 mode.</p> <p>The default is FALSE, which causes the default system code page to be used.</p> <p>Note: The OldTimeStamp property must not be set to TRUE when running in UTF-8 mode.</p>
Mapping properties:		
Gate.Mapper.Debug <i>boolean</i>	<code>-mapperdebug</code> <i>boolean</i>	<p>Use this property to specify whether the gateway includes mapper debug messages in the debug log.</p> <p>The default is TRUE.</p>
Gate.Mapper.Forward HistoricDetails <i>boolean</i>	<code>-mapperforhistdtls</code> <i>boolean</i>	<p>Use this property to specify whether the gateway forwards all historic details on converted update.</p> <p>The default is FALSE.</p>
Gate.Mapper.Forward HistoricJournals <i>boolean</i>	<code>-mapperforhistjrn1</code> <i>boolean</i>	<p>Use this property to specify whether the gateway forwards all historic journals on converted update.</p> <p>The default is FALSE.</p>
Gateway reader properties:		

Table 2. Properties and command line options used by unidirectional gateways (continued)

Property name	Command line option	Description
Gate.Reader.CommonNames <i>string</i>	<code>-readercommonnames</code> <i>string</i>	<p>If the gateway is connecting to an ObjectServer using SSL, and the Common Name field of the received certificate does not match the name specified by the Gate.Reader.Server property (for example, in a failover pair or a virtual server setting), use this property to specify a comma-separated list of acceptable SSL Common Names.</p> <p>The default setting is to use the Gate.Reader.Server property.</p>
Gate.Reader.Debug <i>boolean</i>	<code>-readerdebug</code> <i>boolean</i>	<p>Use this property to specify whether the gateway includes gateway reader debug messages in the debug log.</p> <p>The default is TRUE.</p>
Gate.Reader.Description <i>string</i>	<code>-readerdescription</code> <i>string</i>	<p>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.</p> <p>The default is " ".</p>
Gate.Reader.DetailsTableName <i>string</i>	<code>-readerdetailstblname</code> <i>string</i>	<p>Use this property to specify the name of the details table that the gateway reads.</p> <p>The default is alerts.details.</p>
Gate.Reader.FailbackEnabled <i>boolean</i>	<code>-readerfailbackenabled</code> <i>boolean</i>	<p>Use this property to specify failback for this ObjectServer.</p> <p>The default is TRUE.</p>
Gate.Reader.FailbackTimeout <i>integer</i>	<code>-readerfailbacktimeout</code> <i>integer</i>	<p>Use this property to specify the time (in seconds) that the gateway allows before entering failback mode.</p> <p>The default is 30.</p>

Table 2. Properties and command line options used by unidirectional gateways (continued)

Property name	Command line option	Description
Gate.Reader.IDUCFlushRate <i>integer</i>	<code>-readeriducflushrate</code> <i>integer</i>	<p>Use this property to specify the rate (in seconds) of the granularity of the reader.</p> <p>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.</p> <p>The default is 0.</p> <p>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.</p>
Gate.Reader.IgnoreStatusFilter <i>boolean</i>	<code>-readerignorestatusfilter</code> <i>boolean</i>	<p>Use this property to permit rows from the alerts.details table and alerts.details table to be passed from the collection layer to the aggregation layer in a multi-tier setup.</p> <p>The default is FALSE, that is, rows are not passed between the layers.</p>
Gate.Reader.JournalTableName <i>string</i>	<code>-readerjournaltblname</code> <i>string</i>	<p>Use this property to specify the name of the journal table that the gateway reads.</p> <p>The default is alerts.journal.</p>
Gate.Reader.LogOSSql <i>boolean</i>	<code>-readerlogossq</code> <i>boolean</i>	<p>Use this property to specify whether the gateway logs all SQL commands sent to the ObjectServer in debug mode.</p> <p>The default is FALSE.</p>

Table 2. Properties and command line options used by unidirectional gateways (continued)

Property name	Command line option	Description
Gate.Reader.Password <i>string</i>	<code>-readerpassword</code> <i>string</i>	<p>Use this property to specify the password associated with the user specified by the Gate.Reader.Username property.</p> <p>The default is " ".</p> <p>If the ObjectServer from which the gateway reads alerts is running on Tivoli Netcool/OMNIBus V7, 7.1, 7.2, or 7.2.1 this password must be encrypted by the <code>nco_g_crypt</code> utility.</p> <p>If the ObjectServer from which the gateway reads alerts is running on Tivoli Netcool/OMNIBus V7.2.1 in FIPS 140-2 mode, this password must be either plain text or encrypted using the <code>nco_aes_crypt</code> utility.</p> <p>For details about the encryption utilities, see the <i>IBM Tivoli Netcool/OMNIBus Administration Guide</i> (SC14-7605).</p>
Gate.Reader.Reconnect Timeout <i>integer</i>	<code>-readerreconntimeout</code> <i>integer</i>	<p>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.</p> <p>The default is 30.</p>
Gate.Reader.Server <i>string</i>	<code>-readerserver</code> <i>string</i>	<p>Use this property to specify the name of the ObjectServer from which the gateway reads alerts.</p> <p>The default is NCOMS.</p>
Gate.Reader.Status TableName <i>string</i>	<code>-readerstatustblname</code> <i>string</i>	<p>Use this property to specify the name of the status table that the gateway reads.</p> <p>The default is <code>alerts.status</code>.</p>
Gate.Reader.Tbl ReplicateDefFile <i>string</i>	<code>-readertblrepdeffile</code> <i>string</i>	<p>Use this property to specify the path to the table replication definition file.</p> <p>The default is <code>\$OMNIHOME/gates/objserv_uni/objserv_uni.reader.tblrep.def</code>.</p>
Gate.Reader.Username <i>string</i>	<code>-readerusername</code> <i>string</i>	<p>Use this property to specify the username that is used to authenticate the ObjectServer connection.</p> <p>The default is <code>root</code>.</p>
Gateway writer properties:		

Table 2. Properties and command line options used by unidirectional gateways (continued)

Property name	Command line option	Description
Gate.Writer.BufferSize <i>integer</i>	<code>-writerbufsize integer</code>	Use this property to specify the number of entries that the gateway stores in the buffer before flushing, if buffering is enabled. This property can be used to fine-tune the efficiency of the gateway. The default is 25. Note: The gateway flushes the buffer when the end of a batch of SQL statements has been reached regardless of the buffer size.
Gate.Writer.CommonNames <i>string</i>	<code>-writercommonnames string</code>	If the gateway is connecting to an ObjectServer using SSL, and the Common Name field of the received certificate does not match the name specified by the Gate.Writer.Server property (for example, in a failover pair or a virtual server setting), use this property to specify a comma-separated list of acceptable SSL Common Names. The default setting is to use the Gate.Writer.Server property.
Gate.Writer.Debug <i>boolean</i>	<code>-writerdebug boolean</code>	Use this property to specify whether the gateway includes gateway writer debug messages in the debug log. The default is TRUE.
Gate.Writer.Description <i>string</i>	<code>-writerdescription string</code>	Use this property to specify the application description for the writer 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 " ".
Gate.Writer.Failback Enabled <i>boolean</i>	<code>-writerfailback enabled boolean</code>	Use this property to specify failback for this ObjectServer. The default is TRUE.
Gate.Writer.Failback Timeout <i>integer</i>	<code>-writerfailback timeout integer</code>	Use this property to specify the time (in seconds) that the gateway allows before checking for the return of the master ObjectServer and failing back. The default is 30.
Gate.Writer.LogOSSql <i>boolean</i>	<code>-writerlogossql boolean</code>	Use this property to specify whether the gateway logs all SQL commands sent to the ObjectServer in debug mode. The default is FALSE.

Table 2. Properties and command line options used by unidirectional gateways (continued)

Property name	Command line option	Description
Gate.Writer.Password <i>string</i>	<code>-writerpassword <i>string</i></code>	<p>Use this property to specify the password associated with the user that is specified by the Gate.Writer.Username property.</p> <p>The default is " ".</p> <p>If the ObjectServer to which the gateway writes alerts is running on Tivoli Netcool/OMNIBus v7, 7.1, 7.2, or 7.2.1, this password must be encrypted by the <code>nco_g_crypt</code> utility.</p> <p>If the ObjectServer to which the gateway writes alerts is running on Tivoli Netcool/OMNIBus V7.2.1 in FIPS 140-2 mode, this password must be either plain text or encrypted using the <code>nco_aes_crypt</code> utility.</p> <p>For details about the encryption utilities, see the <i>IBM Tivoli Netcool/OMNIBus Administration Guide</i> (SC14-7605).</p>
Gate.Writer.Reconnect Timeout <i>integer</i>	<code>-writerreconntimeout <i>integer</i></code>	<p>Use this property to specify the time (in seconds) between each reconnection poll attempt if the gateway loses the connection to the ObjectServer.</p> <p>The default is 30.</p>
Gate.Writer.Refresh CacheOnUpdate <i>boolean</i>	<code>-writerrefcacheonupd <i>boolean</i></code>	<p>Use this property to specify whether the hash table cache for this ObjectServer is refreshed.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • TRUE: The cache is resynchronized with the target ObjectServer prior to processing each collection of row updates for a table in the current IDUC window. • FALSE: The gateway assumes that its cache is accurate and does not resynchronize it. <p>The default is TRUE.</p>
Gate.Writer.SAF <i>boolean</i>	<code>-writersaf <i>boolean</i></code>	<p>Use this property to specify that the gateway stores all table entries if the destination ObjectServer is unavailable and to forward them when the ObjectServer becomes available again.</p> <p>The default is FALSE.</p>

Table 2. Properties and command line options used by unidirectional gateways (continued)

Property name	Command line option	Description
Gate.Writer.SAFFile <i>string</i>	<code>-writersaffile</code> <i>string</i>	Use this property to specify the name of the file that the gateway uses to store table entries while the destination ObjectServer is unavailable. The default is \$OMNIHOME/var/objserv_uni_NCO_GATE_Writer.store. Note: This file is only used if the Gate.Writer.SAF property is set to TRUE.
Gate.Writer.Server <i>string</i>	<code>-writerserver</code> <i>string</i>	Use this property to specify the name of the ObjectServer to which the gateway writes alerts. The default is REMOTE.
Gate.Writer.Username <i>string</i>	<code>-writerusername</code> <i>string</i>	Use this property to specify the username that is used to authenticate the ObjectServer connection. This username is used to establish both the writer's IDUC connection and the subsidiary SQL command connection. The default is root.
Resynchronization properties:		
Gate.Resync.Enable <i>boolean</i>	<code>-resyncenable</code> <i>boolean</i>	Use this property to specify that the gateway uses resynchronization. The default is TRUE.
Gate.Writer.SAF ReplayOnResync <i>boolean</i>	<code>-writersafreplayonresync</code> <i>boolean</i>	Use this property to specify how store-and-forward (SAF) replays on resynchronization. You have the following options: <ul style="list-style-type: none"> • TRUE: SAF replays regardless of whether Gate.Resync.Enable has been set to TRUE. • FALSE: SAF replays only when Gate.Resync.Enable has been set to FALSE. The default is FALSE.
Gate.Resync.LockType <i>string</i>	<code>-resynclocktype</code> <i>string</i>	Use this property to specify the locking option on the source and destination ObjectServers while resynchronizing events. You have the following options: <ul style="list-style-type: none"> • FULL: The gateway locks both the source and target ObjectServers. • PARTIAL: The gateway only locks the destination ObjectServer. • NONE: The gateway locks neither the source nor the target ObjectServer. The default is FULL.

Table 2. Properties and command line options used by unidirectional gateways (continued)

Property name	Command line option	Description
Gate.Resync.Type <i>string</i>	-resync <i>type string</i>	<p>Use this property to specify the type of resynchronization that the gateway performs.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • NORMAL: The gateway checks the contents of the two ObjectServers specified by the Gate.Reader.Server and Gate.Writer.Server properties, and, if necessary, resynchronizes them. • UPDATE: The gateway performs the resynchronization with updates. • MINIMAL: The gateway only performs updates for the events that have changed since a failure occurred. <p>The default is NORMAL.</p>
Gate.Writer.UseBulkInsCmd <i>boolean</i>	-usebulkinscmd <i>boolean</i>	<p>Use this property to specify bulk inserts for faster resynchronization.</p> <p>Note: You can only set this property to TRUE if you are using the latest version of both ObjectServer and the gateway.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • TRUE: The gateway changes the format of the insert statement to enable the ObjectServer to process bulk inserts more efficiently. • FALSE: The gateway makes no changes to the insert statement before sending events to the ObjectServer. <p>The default is FALSE.</p>

Hash table cache

The gateway uses a hash table cache to store details of tables that require transferring from one ObjectServer to another.

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

The cache aids performance optimization by providing the gateway with an in-memory summarized view of the contents of the ObjectServers to which it is linked. This means that the gateway does not have to query an ObjectServer to check for the existence of an event, or the **Serial** value or **Tally** value of an event; it can check the cache of the target ObjectServer instead.

You can control the size of the hash table cache by using the **Gate.CacheHashTblSize** property. By default, the size of the hash table cache is

5023 elements (or rows). This can be increased if the status table has a large number of rows (for example, in excess of 20,000).

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

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 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.Writer.Debug** properties; these can be set 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.

Important: To run the gateway in FIPS 140-2 mode, you must set the **Gate.UsePamAuth** property to TRUE.

Buffer size for unidirectional ObjectServer Gateways

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

To set the buffer size, use the **Gate.Writer.BufferSize** property. This property can be adjusted to fine-tune the efficiency of the gateway.

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

Secure connections for unidirectional ObjectServer Gateways

When an ObjectServer is running in secure mode, the gateway must make its connection either as a known ObjectServer user or as the root user.

If you want to resynchronize security data when your ObjectServers are running in secure mode, you should run the gateway as the root user. If you fail to do this, when you attempt the resynchronization the gateway quits and the destination ObjectServer will have no security data. This is because the gateway deletes the destination permissions and so cannot insert rows copied from the source table. Running the gateway as the root user overcomes this problem because it does not require permissions to be set explicitly.

Failback for unidirectional ObjectServer Gateways

Two ObjectServers can be set up as a pair, with one acting as the primary and the other as the backup. You can specify how the backup ObjectServer fails back to the primary ObjectServer using the failback properties defined in the properties file.

Example unidirectional gateway failback configuration

The following figure shows an example unidirectional gateway failback configuration:

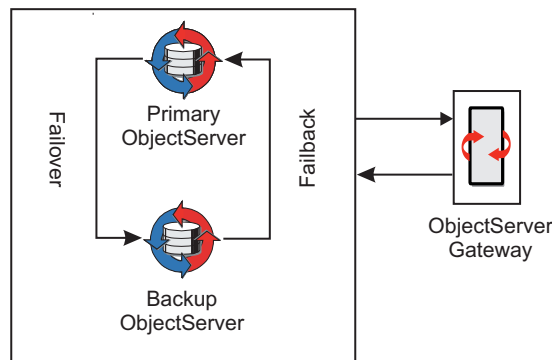


Figure 3. Unidirectional ObjectServer Gateway

Setting up failback

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

Enabling failback

To enable failback, in the gateway properties file, set the **Gate.Reader.Failback** and **Gate.Writer.Failback** properties to TRUE. When the primary ObjectServer fails, the reader and writer fail over to the backup ObjectServer without shutting down. When the reader or writer have detected that they are now connected to a backup ObjectServer, they periodically poll for the return of the primary ObjectServer. When the primary ObjectServer has been detected again, the reader or writer automatically fail back to the primary ObjectServer.

To specify the frequency with which the reader and writer parts of the gateway poll the failed (primary) ObjectServer, set the **Gate.Reader.FailbackTimeout** and **Gate.Writer.FailbackTimeout** properties.

Store and forward for unidirectional ObjectServer Gateways

The gateway supports store and forward on any table when the destination ObjectServer goes offline. This feature can be configured to store and forward either all or none of the tables being replicated.

To activate the store and forward function, set the **Gate.Writer.SAF** property to TRUE. Then, specify the file to which the alerts are written while the destination ObjectServer is offline using the **Gate.Writer.SAFFile** property and restart the gateway. To deactivate the store and forward function, set the **Gate.Writer.SAF** property to FALSE and restart the gateway.

Resynchronization properties of unidirectional ObjectServer Gateways

You specify how unidirectional ObjectServer Gateways perform resynchronization using the resynchronization properties defined in the properties file.

The following table describes the resynchronization properties of unidirectional ObjectServer Gateways.

Table 3. Resynchronization properties and command line options of unidirectional ObjectServer Gateways

Property name	Command line option	Description
Gate.Resync.Enable <i>boolean</i>	<code>-resyncenable</code> <i>boolean</i>	Use this property to specify that the gateway uses resynchronization. The default is TRUE.
Gate.Resync.Type <i>string</i>	<code>-resynctype</code> <i>string</i>	Use this property to specify the type of resynchronization that is required. You have the following options: <ul style="list-style-type: none">• NORMAL: The gateway checks the contents of the two ObjectServers specified by the Gate.Reader.Server property and Gate.Writer.Server property, and, if necessary, resynchronizes them.• UPDATE: The gateway performs the resynchronization with updates. This option ensures that updates made in the upper tier are not lost during the resynchronization The default is NORMAL.

Chapter 4. Configuration of the bidirectional ObjectServer Gateway

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

To start the bidirectional gateway, enter the following command on the command line:

```
nco_g_objserv_bi -name
```

To reduce latency, run the gateway on the same box to which it writes alerts. You can also reduce latency by adjusting the buffer size.

For information about the common properties and Interprocess Control (IPC), see the *IBM Tivoli Netcool/OMNIBus Administration Guide* (SC14-7605).

The default location for the properties file for the unidirectional and bidirectional gateways is:

```
$OMNIHOME/etc/server_name.props
```

The default properties file must be copied to the \$OMNIHOME/etc folder.

Note: The properties files for the unidirectional and bidirectional gateways contain similar properties; however, they are described separately for clarity.

Bidirectional gateway properties

You can configure ObjectServer Gateways by using properties defined in a properties file. The unidirectional ObjectServer Gateway and the bidirectional ObjectServer Gateway have some of these properties in common, but each set of properties is described separately for ease of reference.

For information about the common properties and Interprocess Communication (IPC) properties, see the *IBM Netcool/OMNIBus Probe and Gateway Guide* (SC23-6387). The following table describes the common gateway properties.

Table 4. Common gateway properties and command line options

Property name	Command line option	Description
Gate.CacheHashTblSize <i>integer</i>	<code>-chashtblsize <i>integer</i></code>	Use this property to specify the size (in elements) that the gateway allocates for the hash table cache. The default is 5023.

Table 4. Common gateway properties and command line options (continued)

Property name	Command line option	Description
Gate.MapFile <i>string</i>	-mapfile <i>string</i>	Use this property to specify the location of the map definition file. The default is \$OMNIHOME/gates/objserv_uni/objserv_uni.map.
Gate.StartupCmdFile <i>string</i>	-startupcmdfile <i>string</i>	Use this property to specify the location of the startup command file. The default is \$OMNIHOME/objserv_uni/objserv_uni.startup.cmd.
Gate.Transfer.FailoverSyncRate <i>integer</i>	-fsyncrate <i>integer</i>	Use this property to specify the rate (in seconds) of the failover synchronization. The default is 60.
Gate.NGtkDebug <i>boolean</i>	-ngtkdebug <i>boolean</i>	Use this property to specify whether the NGTK library should log debug messages. The default is TRUE. Note: You can specify which debug messages are included in the debug log file using the Gate.Mapper.Debug , Gate.Reader.Debug and Gate.Writer.Debug properties.
Gate.PAAware <i>integer</i>	-paaware <i>integer</i>	Use this property to specify 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.
Gate.PAAwareName <i>string</i>	-paname <i>string</i>	Use this property to specify the name of the Process Agent 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.

Table 4. Common gateway properties and command line options (continued)

Property name	Command line option	Description
Gate.Resync.BackoffRetryTime <i>integer</i>	-backoffretrytime <i>integer</i>	Use this property to specify the time in seconds to backoff before attempting to resynchronize. The default is 60.
Gate.UsePamAuth <i>boolean</i>	-usepamauth <i>boolean</i>	Use this property to specify whether PAM authentication is used. The default is FALSE.
Gate.UnixAdminGroup <i>string</i>	-unixadmingroup <i>string</i>	Use this property to specify the administration group to which the gateway must belong if standard UNIX authentication is used. The default is " ".
MaxLogFileSize <i>integer</i>	-maxlogfilesize <i>integer</i>	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 100.

Table 5. Properties and command line options used by bidirectional gateways

Property name	Command line option	Description
Gate.CacheHashTblSize <i>integer</i>	-chashtblsize <i>integer</i>	Use this property to specify the size (in elements) that the gateway allocates for the hash table cache. The default is 5023.
Gate.MapFile <i>string</i>	-mapfile <i>string</i>	Use this property to specify the location of the map definition file. The default is \$OMNIHOME/gates/objserv_uni/ objserv_uni.map.
Gate.StartupCmdFile <i>string</i>	-startupcmdfile <i>string</i>	Use this property to specify the location of the startup command file. The default is \$OMNIHOME/objserv_uni/ objserv_uni.startup.cmd.

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.Transfer.FailoverSyncRate <i>integer</i>	<code>-fsyncrate integer</code>	Use this property to specify the rate (in minutes) of the failover synchronization. The default is 60.
Gate.NGtkDebug <i>boolean</i>	<code>-ngtkdebug boolean</code>	Use this property to specify whether the NGTK library should log debug messages. The default is TRUE. Note: You can specify which debug messages are included in the debug log file using the Gate.Mapper.Debug , Gate.Reader.Debug and Gate.Writer.Debug properties.
Gate.PAAware <i>integer</i>	<code>-paaware integer</code>	Use this property to specify 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.
Gate.PAAwareName <i>string</i>	<code>-paname string</code>	Use this property to specify the name of the Process Agent 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.UsePamAuth <i>boolean</i>	<code>-usepamauth boolean</code>	Use this property to specify whether PAM authentication is used. The default is FALSE. Note: To run the gateway in FIPS 140-2 mode, you must set this property to TRUE.
Gate.UnixAdminGroup <i>string</i>	<code>-unixadmingroup string</code>	Use this property to specify the administration group to which the gateway must belong if standard UNIX authentication is used. The default is " ".
MaxLogFileSize <i>integer</i>	<code>-maxlogfilesize integer</code>	Use this property to specify the maximum size (in kilobytes) of 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.

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
OldTimeStamp <i>boolean</i>	<i>-oldtimestamp boolean</i>	<p>Use this property to specify old-style timestamp format the gateway uses in the log file.</p> <p>Set the value to TRUE to specify the timestamp format used in Tivoli Netcool/OMNIBus V7.2.1, or earlier.</p> <p>For example: dd/MM/YYYY hh:mm:ss AM or dd/MM/YYYY hh:mm:ss PM when the locale is set to en_GB on a Solaris 9 computer.</p> <p>Set this value to FALSE to display the timestamp in ISO 8601 format.</p> <p>For example: YYYY-MM-DDThh:mm:ss, where T separates the date and time, and hh is in 24-hour clock.</p> <p>The default is FALSE.</p> <p>Note: Do not set the OldTimeStamp property to TRUE when running in UTF-8 mode.</p>
N/A	<i>-utf8enabled boolean</i>	<p>Use this command line option to control the encoding of data that is passed into, or generated by, this gateway when running on Windows.</p> <p>Set the value of <i>-utf8enabled</i> to TRUE to run the application in UTF-8 mode.</p> <p>The default is FALSE, which causes the default system code page to be used.</p> <p>Note: The OldTimeStamp property must not be set to TRUE when running in UTF-8 mode.</p>
Mapping properties:		
Gate.Mapper.Debug <i>boolean</i>	<i>-mapperdebug boolean</i>	<p>Use this property to specify whether the gateway includes mapper debug messages in the debug log.</p> <p>The default is TRUE.</p>
Gate.Mapper.Forward HistoricDetails <i>boolean</i>	<i>-mapperforhistdtls boolean</i>	<p>Use this property to specify whether the gateway forwards all historic details on converted update.</p> <p>The default is FALSE.</p>
Gate.Mapper.Forward HistoricJournals <i>boolean</i>	<i>-mapperforhistjrn1 boolean</i>	<p>Use this property to specify whether the gateway forwards all historic journals on converted update.</p> <p>The default is FALSE.</p>
ObjectServer properties:		

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.ObjectServerA.BufferSize <i>integer</i>	<code>-objectserverabufsize</code> <i>integer</i>	<p>Use this property to specify the number of entries that the gateway stores in the buffer for this ObjectServer before flushing, if buffering is enabled. This property can be used to fine-tune the efficiency of the gateway.</p> <p>The default is 25.</p> <p>Note: The gateway flushes the buffer when the end of a batch of SQL statements has been reached regardless of the buffer size.</p>
Gate.ObjectServerA.CommonNames <i>string</i>	<code>-objectserveracommonnames</code> <i>string</i>	<p>If the gateway is connecting to an ObjectServer using SSL, and the Common Name field of the received certificate does not match the name specified by the Gate.ObjectServerA.Server property (for example, in a failover pair or a virtual server setting), use this property to specify a comma-separated list of acceptable SSL Common Names.</p> <p>The default setting is to use the Gate.ObjectServerA.Server property.</p>
Gate.ObjectServerA.Debug <i>boolean</i>	<code>-objectserveradebug</code> <i>boolean</i>	<p>Use this property to specify whether the gateway includes debug messages for this ObjectServer in the gateway debug log.</p> <p>The default is TRUE.</p>
Gate.ObjectServerA.DeleteIfNoDedup <i>boolean</i>	<code>-objectserveradelifnodelete</code> <i>boolean</i>	<p>Use this property to specify whether a deletion is applied if the gateway forwards deletes.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • FALSE: If you make this setting, the delete is always applied • TRUE: If you make this setting, the delete is not applied if the event in the target server indicates that the event has occurred again since the delete was issued. <p>The default is FALSE.</p> <p>Note: In most cases, this property is set to FALSE.</p>

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.ObjectServerA.Description <i>string</i>	<code>-objectserveradescription</code> <i>string</i>	Use this property to specify an application description for the connection to ObjectServer A. This description is used in triggers and allows you to determine which component of the gateway attempted to perform an action. The default is " ".
Gate.ObjectServerA.FailbackEnabled <i>boolean</i>	<code>-objectserverafailbackenabled</code> <i>boolean</i>	Use this property to enable failback for this ObjectServer. The default is FALSE.
Gate.ObjectServerA.FailbackTimeout <i>integer</i>	<code>-objectserverafailbacktimeout</code> <i>integer</i>	Use this property to specify the time (in seconds) that the gateway allows before checking for the return of the master ObjectServer and failing back. The default is 30.
Gate.ObjectServerA.LogOSSql <i>boolean</i>	<code>-objectserveralogossql</code> <i>boolean</i>	Use this property to specify whether the gateway logs all SQL commands sent to this ObjectServer in debug mode. The default is FALSE.
Gate.ObjectServerA.Password <i>string</i>	<code>-objectserverapassword</code> <i>string</i>	Use this property to specify the password associated with the user specified by the Gate.ObjectServerA.Username property. The default is " ".
		If the ObjectServer to which the gateway reads/writes alerts is running on IBM Tivoli Netcool/OMNIbus v7, 7.1, 7.2 or 7.2.1, this password must be encrypted by the <code>nco_g_crypt</code> utility. For details about the encryption utilities, see the <i>IBM Tivoli Netcool/OMNIbus Administration Guide</i> (SC14-7605).
Gate.ObjectServerA.ReconnectTimeout <i>integer</i>	<code>-objectserverareconnecttimeout</code> <i>integer</i>	Use this property to specify the time (in seconds) between each reconnection poll attempt if the connection to this ObjectServer is lost. The default is 30.

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.ObjectServerA.RefreshCacheOn Update <i>boolean</i>	<code>-objectserverarefcacheon</code> <i>boolean</i>	<p>Use this property to specify how the hash table cache is refreshed for this ObjectServer.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • TRUE: The cache is resynchronized with the target ObjectServer before each collection of row updates for a table in the current IDUC window is processed • FALSE: The gateway assumes that its cache is accurate and does not resynchronize it <p>The default is FALSE.</p>
Gate.ObjectServerA.SAF <i>boolean</i>	<code>-objectserverasaf</code> <i>boolean</i>	<p>Use this property to specify whether the gateway stores all table entries if the destination ObjectServer is unavailable and to forward them when the ObjectServer becomes available again.</p> <p>The default is FALSE.</p>
Gate.ObjectServerA.SAFFile <i>string</i>	<code>-objectserverasaffile</code> <i>string</i>	<p>Use this property to specify the name of the file that the gateway uses to store table entries while the destination ObjectServer is unavailable.</p> <p>The default is \$OMNIHOME/var/objserv_bi/NCO_GATE_ObjectServerA.store.</p> <p>Note: This file is only used if the Gate.ObjectServerA.SAF property is set to TRUE.</p>
Gate.ObjectServerA.Server <i>string</i>	<code>-objectserveraserver</code> <i>string</i>	<p>Use this property to specify the name of this ObjectServer.</p> <p>The default is NCOMS.</p>
Gate.ObjectServerA.Username <i>string</i>	<code>-objectserverausername</code> <i>string</i>	<p>Use this property to specify the username that is used to authenticate connection to this ObjectServer. This username is used to establish both the writer's IDUC connection and the subsidiary SQL command connection.</p> <p>The default is root.</p>
Gate.ObjectServerA.TblReplicateDefFile <i>string</i>	<code>-objectserveratblrepdef</code> <i>string</i>	<p>Use this property to specify the path to the table replication definition file.</p> <p>The default is \$OMNIHOME/gates/objserv_bi/objserv_bi.objectservera.tblrep.def.</p>

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.ObjectServerA. StatusTableName <i>string</i>	<code>-objectserverastatustbl</code> <i>string</i>	Use this property to specify the name of the status table that the gateway reads. The default is alerts.status.
Gate.ObjectServerA. JournalTableName <i>string</i>	<code>-objectserverajournaltbl</code> <i>string</i>	Use this property to specify the name of the journal table that the gateway reads. The default is alerts.journal.
Gate.ObjectServerA. DetailsTableName <i>string</i>	<code>-objectserveradetailstbl</code> <i>string</i>	Use this property to specify the name of the details table that the gateway reads. The default is alerts.details.
Gate.ObjectServerA. IDUCFlushRate <i>integer</i>	<code>-objectserveraiducflushrate</code> <i>integer</i>	Use this property to specify the rate (in seconds) of the granularity of the reader. If you set this property to 0, the reader gets its updates at the same granular rate as that of the ObjectServer to which it is connected. The default is 0. 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.
Gate.ObjectServerB. BufferSize <i>integer</i>	<code>-objectserverbbufsize</code> <i>integer</i>	Use this property to specify the number of entries that the gateway stores in the buffer for this ObjectServer before flushing, if buffering is enabled. This property can be used to fine-tune the efficiency of the gateway. The default is 25. Note: The gateway flushes the buffer when the end of a batch of SQL statements has been reached regardless of the buffer size.

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.ObjectServerB.CommonNames <i>string</i>	<code>-objectserverbcommonnames</code> <i>string</i>	<p>If the gateway is connecting to an ObjectServer using SSL, and the Common Name field of the received certificate does not match the name specified by the Gate.ObjectServerB.Server property (for example, in a failover pair or a virtual server setting), use this property to specify a comma-separated list of acceptable SSL Common Names.</p> <p>The default setting is to use the Gate.ObjectServerB.Server property.</p>
Gate.ObjectServerB.Debug <i>boolean</i>	<code>-objectserverbdebug</code> <i>boolean</i>	<p>Use this property to specify whether the gateway includes debug messages for this ObjectServer in the debug log.</p> <p>The default is TRUE.</p>
Gate.ObjectServerB.DeleteIfNoDedup <i>boolean</i>	<code>-objectserverbdeleteifnodelete</code> <i>boolean</i>	<p>Use this property to specify whether a deletion is applied if the gateway forwards deletes.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> FALSE: If you make this setting, the delete is always applied TRUE: If you make this setting, the delete is not applied if the event in the target server indicates that the event has occurred again since the delete was issued <p>The default is FALSE. Note: In most cases, this property is set to FALSE.</p>
Gate.ObjectServerB.Description <i>string</i>	<code>-objectserverbdescription</code> <i>string</i>	<p>Use this property to specify an application description for the connection to ObjectServer B. This description is used in triggers and allows you to determine which component of the gateway attempted to perform an action.</p> <p>The default is " ".</p>
Gate.ObjectServerB.FailbackEnabled <i>boolean</i>	<code>-objectserverbfailbackenable</code> <i>boolean</i>	<p>Use this property to specify failback for this ObjectServer.</p> <p>The default is FALSE.</p>
Gate.ObjectServerB.FailbackTimeout <i>integer</i>	<code>-objectserverbfailbacktimeout</code> <i>integer</i>	<p>Use this property to specify the time (in seconds) that the gateway allows before checking for the return of the master ObjectServer and failing back.</p> <p>The default is 30.</p>

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.ObjectServerB.LogOSSql <i>boolean</i>	<code>-objectserverblogossql</code> <i>boolean</i>	Use this property to specify whether the gateway logs all SQL commands sent to this ObjectServer in debug mode. The default is FALSE.
Gate.ObjectServerB.Password <i>string</i>	<code>-objectserverbpassword</code> <i>string</i>	Use this property to specify the password associated with the user specified by the Gate.ObjectServerB.Username property. The default is " ". If the ObjectServer to which the gateway reads/writes alerts is running on IBM Tivoli Netcool/OMNIbus v7, 7.1, 7.2, or 7.2.1, this password must be encrypted by the <code>nco_g_crypt</code> utility. For details about the encryption utilities, see the <i>IBM Tivoli Netcool/OMNIbus Administration Guide</i> (SC14-7605).
Gate.ObjectServerB.ReconnectTimeout <i>integer</i>	<code>-objectserverbreconnecttime</code> <i>integer</i>	Use this property to specify the time (in seconds) between each reconnection poll attempt if the gateway loses the connection to this ObjectServer, this property defines . The default is 30.
Gate.ObjectServerB.RefreshCacheOn Update <i>boolean</i>	<code>-objectserverbrefcacheonupdate</code> <i>boolean</i>	Use this property to specify whether the hash table cache is refreshed for this ObjectServer. You have the following options: <ul style="list-style-type: none"> • TRUE: The cache is resynchronized with the target ObjectServer prior to processing each collection of row updates for a table in the current IDUC window • FALSE - The gateway assumes that its cache is accurate and does not resynchronize it The default is FALSE.
Gate.ObjectServerB.SAF <i>boolean</i>	<code>-objectserverbsaf</code> <i>boolean</i>	Use this property to specify that the gateway stores all table entries if the destination ObjectServer is unavailable and to forward them when the ObjectServer becomes available again. The default is FALSE.

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.ObjectServerB.SAFile <i>string</i>	<code>-objectserverbsaffile</code> <i>string</i>	Use this property to specify the name of the file that the gateway uses to store table entries while the destination ObjectServer is unavailable. The default is \$OMNIHOME/var/objserv_bi/NCO_GATE_ObjectServerB.store. Note: This file is only used if the Gate.ObjectServerB.SAF property is set to TRUE
Gate.ObjectServerB.Server <i>string</i>	<code>-objectserverbserver</code> <i>string</i>	Use this property to specify the name of this ObjectServer. The default is NCOMS.
Gate.ObjectServerB.Username <i>string</i>	<code>-objectserverbusername</code> <i>string</i>	Use this property to specify the username used to authenticate the connection to this ObjectServer. This username is used to establish both the IDUC connection of the writer and the subsidiary SQL command connection. The default is root.
Gate.ObjectServerB.TblReplicateDefFile <i>string</i>	<code>-objectserverbtblrepdef</code> <i>string</i>	Use this property to specify the path to the table replication definition file. The default is \$OMNIHOME/gates/objserv_bi/objserv_bi.objectserverb.tblrep.def.
Gate.ObjectServerB.StatusTableName <i>string</i>	<code>-objectserverbstatustbl</code> <i>string</i>	Use this property to specify the name of the status table that the gateway reads. The default is alerts.status.
Gate.ObjectServerB.JournalTableName <i>string</i>	<code>-objectserverbjournaltbl</code> <i>string</i>	Use this property to specify the name of the journal table that the gateway reads. The default is alerts.journal.
Gate.ObjectServerB.DetailsTableName <i>string</i>	<code>-objectserverbdetailstbl</code> <i>string</i>	Use this property to specify the name of the details table that the gateway reads. The default is alerts.details.

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.ObjectServerB.IDUCFlushRate <i>integer</i>	<code>-objectserverbiducflushrate</code> <i>integer</i>	<p>Use this property to specify the rate, in seconds, of the granularity of the reader.</p> <p>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.</p> <p>The default is 0.</p> <p>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.</p>
Resynchronization properties:		
Gate.Resync.Enable <i>boolean</i>	<code>-resyncenable</code> <i>boolean</i>	<p>Use this property to specify that the gateway uses resynchronization.</p> <p>The default is TRUE.</p>
Gate.ObjectServerA.SAFReplayOnResync <i>boolean</i>	<code>-objectserverasafreplay</code> <i>boolean</i>	<p>Use this property to specify how store-and-forward (SAF) file for ObjectServerA replays on resynchronization.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • TRUE: SAF replays regardless of whether Gate.Resync.Enable has been set to TRUE. • FALSE: SAF replays only when Gate.Resync.Enable has been set to FALSE <p>The default is FALSE.</p>
Gate.ObjectServerB.SAFReplayOnResync <i>boolean</i>	<code>-objectserverbsafreplay</code> <i>boolean</i>	<p>Use this property to specify how store-and-forward (SAF) file for ObjectServerB replays on resynchronization.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • TRUE: SAF replays regardless of whether Gate.Resync.Enable has been set to TRUE. • FALSE: SAF replays only when Gate.Resync.Enable has been set to FALSE <p>The default is FALSE.</p>

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.Resync.Master <i>string</i>	-resyncmaster <i>string</i>	<p>Use this property to specify which ObjectServer the gateway should always treat as the master during resynchronization.</p> <p>Valid values are ObjectServerA and ObjectServerB.</p> <p>The default is "".</p> <p>Note: If you omit this property, the gateway always treats the ObjectServer that has been running the longest as the master.</p>
Gate.Resync.Preferred <i>string</i>	-resyncpreferred <i>string</i>	<p>Use this property to specify which ObjectServer the gateway should treat as the master during resynchronization if the Gate.Resynch.Master has been omitted and both ObjectServers have been running for the same length of time.</p> <p>Valid values are ObjectServerA and ObjectServerB.</p> <p>The default is "".</p>
Gate.Resync.Type <i>string</i>	-resynctype <i>string</i>	<p>Use this property to specify the type of resynchronization that the gateway performs.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • NORMAL: The gateway checks the contents of the two ObjectServers, and, if necessary, resynchronizes them. • UPDATE: The gateway performs the resynchronization with updates. • MINIMAL: The gateway performs the resynchronization with updates only for the events that have changed since a failure has occurred. <p>The default is NORMAL.</p>
Gate.Resync.LockType <i>string</i>	-resynclocktype <i>string</i>	<p>Use this property to specify the locking option on the source and destination ObjectServers while resynchronizing events.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • FULL: The gateway locks both the source and target ObjectServers. • PARTIAL: The gateway only locks the destination ObjectServer. • NONE: The gateway locks neither the source nor the target ObjectServer. <p>The default is FULL.</p>

Table 5. Properties and command line options used by bidirectional gateways (continued)

Property name	Command line option	Description
Gate.ObjectServerA.UseBulkInsCmd <i>boolean</i>	<code>-usebulkinscmd</code> <i>boolean</i>	<p>Use this property to specify bulk inserts for faster resynchronization.</p> <p>Note: You can only set this property to TRUE if you are using the latest version of both ObjectServer and the gateway.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • TRUE: The gateway changes the format of the insert statement it sends to ObjectServerA, which allows ObjectServerA to process the inserts more efficiently. • FALSE: The gateway makes no changes to the insert statement before sending events to ObjectServerA. <p>The default is FALSE.</p>
Gate.ObjectServerB.UseBulkInsCmd <i>boolean</i>	<code>-usebulkinscmd</code> <i>boolean</i>	<p>Use this property to specify bulk inserts for faster resynchronization.</p> <p>Note: You can only set this property to TRUE if you are using the latest version of both ObjectServer and the gateway.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • TRUE: The gateway changes the format of the insert statement it sends to ObjectServerB, which allows ObjectServerB to process the inserts more efficiently. • FALSE: The gateway makes no changes to the insert statement before sending events to ObjectServerB. <p>The default is FALSE.</p>

Hash table cache

The gateway uses a hash table cache to store details of tables that require transferring from one ObjectServer to another.

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

The cache aids performance optimization by providing the gateway with an in-memory summarized view of the contents of the ObjectServers to which it is linked. This means that the gateway does not have to query an ObjectServer to check for the existence of an event, or the **Serial** value or **Tally** value of an event; it can check the cache of the target ObjectServer instead.

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

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

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

You can use the following properties to specify which debug messages are included in the debug files. Set the properties to TRUE or FALSE as required.

- **Gate.Mapper.Debug**
- **Gate.ObjectServerB.Debug**
- **Gate.ObjectServerA.Debug**

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.

Important: To run the gateway in FIPS 140-2 mode, you must set the **Gate.UsePamAuth** property to TRUE.

Buffer size for bidirectional ObjectServer Gateways

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

The buffer size controls the number of entries that the gateway stores in its buffer before flushing them to the ObjectServer. The gateway uses separate buffers for the source and destination ObjectServers and their sizes can be set using the **Gate.ObjectServerA.BufferSize** and **Gate.ObjectServerB.BufferSize** properties. These properties can be adjusted to fine-tune the efficiency of the gateway.

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

Secure ObjectServer connections for bidirectional ObjectServer Gateways

When an ObjectServer is running in secure mode, the gateway must make its connection either as a known ObjectServer user or as the root user.

If you want to resynchronize security data when your ObjectServers are running in secure mode, you should run the gateway as the root user. If you fail to do this, when you attempt the resynchronization the gateway quits and the destination ObjectServer will have no security data. This is because the gateway deletes the destination permissions and so cannot insert rows copied from the source table. Running the gateway as the root user overcomes this problem because it does not require permissions to be set explicitly.

Failback for bidirectional ObjectServer Gateways

Two ObjectServers can be set up as a pair, with one acting as the primary and the other as the backup. You can specify how the backup ObjectServer fails back to the primary ObjectServer using the failback properties defined in the properties file.

Setting up failback

To set up failback, set the **BackupObjectServer** property for the backup ObjectServer to TRUE.

Example bidirectional gateway failback configuration

The following figure shows an example bidirectional gateway failback configuration:

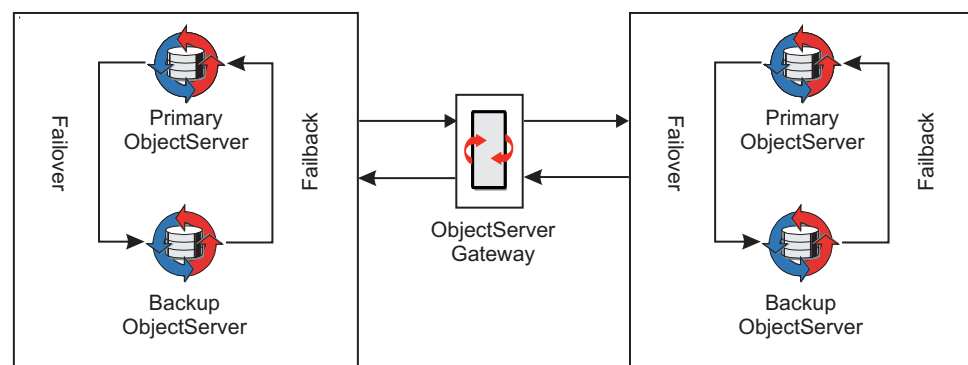


Figure 4. Bidirectional ObjectServer Gateway

Enable failback

To enable failback, in the gateway properties file, you must set the **Gate.ObjectServerA.Failback** property to TRUE (if ObjectServer A has a backup ObjectServer) and **Gate.ObjectServerB.Failback** property to TRUE (if ObjectServer B has a backup ObjectServer). When the primary ObjectServer fails, the gateway

fails over to the backup ObjectServer without shutting down. When the gateway is connected to a backup ObjectServer, it periodically polls for the return of the primary ObjectServer. When the primary ObjectServer has been detected again, the gateway automatically fails back to the primary ObjectServer.

To specify the frequency with which the gateway polls the failed ObjectServer, set the **Gate.ObjectServerA.FailbackTimeout** and **Gate.ObjectServerB.FailbackTimeout** properties.

Note: Failback does not apply when the bi-directional gateway connects to an ObjectServer that is not in a virtual pair. For example, if you are using a bi-directional gateway to maintain a backup ObjectServer, failback does not apply.

Store and forward for bidirectional ObjectServer Gateways

The gateway supports store and forward on any table when the destination ObjectServer goes offline. This feature can be configured to store and forward either all or none of the tables being replicated.

To activate the store and forward function for ObjectServer A, set the **Gate.ObjectServerA.SAF** property to TRUE, specify the file to which the alerts are written while the destination ObjectServer is offline using the **Gate.ObjectServerA.SAFFile** property and restart the gateway. To deactivate the store and forward function, set the **GateObjectServerA.SAF** property to FALSE and restart the gateway.

To activate the store and forward function for ObjectServer B, set the **Gate.ObjectServerB.SAF** property to TRUE, specify the file to which the alerts are written while the destination ObjectServer is offline using the **Gate.ObjectServerB.SAFFile** property, and restart the gateway. To deactivate the store and forward function, set the **Gate.ObjectServerB.SAF** property to FALSE and restart the gateway.

Alternative deletion strategy

You can set ObjectServer Gateways to collect insert, update, and delete events from a set of ObjectServers and pass them on to an ObjectServer that aggregates them and only passes back deletes.

For example, the aggregation ObjectServer may make available events pooled from a set of ObjectServer. Deletion events would be passed back by the aggregation ObjectServer as events are addressed. In this situation, you may want to prevent a deletion from being applied if the event in the target server indicates that the event has occurred again since the delete was issued; for example, if an attempted resolution to a problem has failed.

To set up this deletion strategy, set the **Gate.ObjectServerA.DeleteIfNoDedup** and **Gate.ObjectServerB.DeleteIfNoDedup** properties to TRUE.

Resynchronization properties of bidirectional ObjectServer Gateways

You specify how bidirectional ObjectServer Gateways perform resynchronization using the resynchronization properties defined in the properties file.

Bidirectional ObjectServer Gateways have more resynchronization properties than unidirectional ObjectServer Gateways.

The following table describes the resynchronization properties of bidirectional ObjectServer Gateways.

Table 6. Resynchronization properties and command line options of bidirectional ObjectServer Gateways

Property name	Command line option	Description
Gate.Resync.Enable <i>boolean</i>	<code>-resyncenable</code> <i>boolean</i>	Use this property to specify that the gateway uses resynchronization. The default is TRUE.
Gate.Resync.Master <i>string</i>	<code>-resyncmaster</code> <i>string</i>	Use this property to specify which ObjectServer the gateway should always treat as the master during resynchronization. Valid values are ObjectServerA and ObjectServerB. The default is "". Note: If you omit this property, the gateway always treats the ObjectServer that has been running the longest as the master.
Gate.Resync.Preferred <i>string</i>	<code>-resyncpreferred</code> <i>string</i>	Use this property to specify which ObjectServer the gateway should treat as the master during resynchronization if the Gate.Resync.Master has been omitted and both ObjectServers have been running for the same length of time. Valid values are ObjectServerA and ObjectServerB. The default is "".

Table 6. Resynchronization properties and command line options of bidirectional ObjectServer Gateways (continued)

Property name	Command line option	Description
Gate.Resync.Type <i>string</i>	-resync <i>type</i> <i>string</i>	<p>Use this property to specify the type of resynchronization that is required.</p> <p>You have the following options:</p> <ul style="list-style-type: none"> • NORMAL: The gateway checks the contents of the two ObjectServers specified by the Gate.Reader.Server and Gate.Writer.Server properties, and, if necessary, resynchronizes them. • UPDATE: The gateway performs the resynchronization with updates. • MINIMAL: The gateway only performs updates for the events that have changed since a failure occurred. <p>The default is NORMAL.</p>

Chapter 5. ObjectServer Gateway mapping

The gateway can replicate any table in the ObjectServer. To do this, the gateway maps data to the appropriate fields in the ObjectServer using a map definition file.

If you want to replicate user related system tables (for example, SecurityUsers, SecurityGroups, SecurityRoles, SecurityRoleGrants, and SecurityGroupMembers), you must also include details of these mappings in this file. The path of the map definition file is determined by the **Gate.Mapfile** property in the properties file.

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

- TO_STRING (<column_name>)
- TO_INTEGER (<column_name>)
- TO_TIME (<column_name>)

Syntax

Mappings for use with the ObjectServer writer must adhere to the following syntax:

```
CREATE MAPPING mappingname ( ' dest_fieldname' = ( ' @src_fieldname' |  
simple_expression | attribute ) [ ON INSERT ONLY ]      [ CONVERT TO type ]  
[ NOT NULL (' @src_fieldname') ] [ , 'dest_fieldname' = ( ' @src_fieldname' |  
simple_expression | attribute ) [ ON INSERT ONLY ]      [ CONVERT TO type ]  
[ NOT NULL (' @src_fieldname') ] ] ... ) ;
```

Where:

- *mappingname* is the name of the mapping to be created.
- *dest_fieldname* is the name of the field to be written in the destination ObjectServer.
- *src_fieldname* is the name of a field in the ObjectServer alerts.status table.
- *simple_expression* is an integer or a set of integers and operators.
- *attribute* is an attribute name.

The optional **ON INSERT ONLY** controls the updating of the field during the life of the alert; when omitted, the field is updated when any change in the state of the alert occurs. When included, the field is created once for the alert, but is never updated.

Tip: The **ON INSERT ONLY** option only applies when setting the value of variables.

The optional **CONVERT TO *type*** enables the mapping to define a forced conversion for situations where a source field may not match the type of the destination field. The *type* can be INTEGER, STRING, or DATE.

The optional **NOT NULL** indicates that the mapping is only performed if the source field is not null; that is, has a value set for it.

Mapping attributes

You use attribute names to include additional data in mapping definitions. You can specify two types of attribute: cache value access attributes or dynamic attributes.

Cache value access attributes

The gateway uses cache value attributes to access values that are stored in the cross-reference cache. The following table describes the cache value attributes that can be used in mapping definitions.

Table 7. Cache value access attributes

Attribute name	Description
STATUS.SERIAL	Cached serial number for the status table row that is associated with the current journal or details table row.
STATUS.SERVER_SERIAL	Cached server serial number for the status table row that is associated with the current journal or details table row.
STATUS.SERVER_NAME	Cached server name for the status table row that is associated with the current journal or details table row.
STATUS.IDENTIFIER	Cached identifier for the status table row that is associated with the current journal or details table row.
JOURNAL.SERIAL	Cached serial number of the journal table row.
DETAILS.IDENTIFIER	Cached identifier of the details table row.

Dynamic attributes

Dynamic attributes enable the gateway to access dynamic values that are automatically generated by the gateway. The following table describes the dynamic attributes that can be used in mapping definitions.

Table 8. Dynamic attributes

Attribute name	Description
ACTION_CODE	This attribute displays a single character string that specifies the type of operation performed. Valid values are: <ul style="list-style-type: none">• I: Insert• U: Update• D : Delete
ACTION_TIME	This attribute displays the time in UTC that the action occurred.
DELETEDAT	This attribute displays the date on which the row was deleted, if applicable.

Example mapping

Mappings define how the gateways replicate tables by assigning data to appropriate fields in the ObjectServer.

The following example shows the mappings for the ObjectServer tables into which the gateway writes:

```
CREATE MAPPING StatusMap
(
  'Identifier' = '@Identifier' ON INSERT ONLY,
  'Node' = '@Node' ON INSERT ONLY,
  'NodeAlias' = '@NodeAlias' ON INSERT ONLY
                  NOTNULL '@Node',
  'Manager' = '@Manager' ON INSERT ONLY,
  'Agent' = '@Agent' ON INSERT ONLY,
  'AlertGroup' = '@AlertGroup' ON INSERT ONLY,
  'AlertKey' = '@AlertKey' ON INSERT ONLY,
  'Severity' = '@Severity',
  'Summary' = '@Summary',
  'StateChange' = '@StateChange',
  'FirstOccurrence' = '@FirstOccurrence' ON INSERT ONLY,
  'LastOccurrence' = '@LastOccurrence',
  'InternalLast' = '@InternalLast',
  'Poll' = '@Poll' ON INSERT ONLY,
  'Type' = '@Type' ON INSERT ONLY,
  'Tally' = '@Tally',
  'ProbeSubSecondId' = '@ProbeSubSecondId',
  'Class' = '@Class' ON INSERT ONLY,
  'Grade' = '@Grade' ON INSERT ONLY,
  'Location' = '@Location' ON INSERT ONLY,
  'OwnerUID' = '@OwnerUID',
  'OwnerGID' = '@OwnerGID',
  'Acknowledged' = '@Acknowledged',
  'BSM_Identity' = '@BSM_Identity',
  'Flash' = '@Flash',
  'EventId' = '@EventId' ON INSERT ONLY,
  'ExpireTime' = '@ExpireTime' ON INSERT ONLY,
  'ProcessReq' = '@ProcessReq',
  'SuppressEscl' = '@SuppressEscl',
  'Customer' = '@Customer' ON INSERT ONLY,
  'Service' = '@Service' ON INSERT ONLY,
  'PhysicalSlot' = '@PhysicalSlot' ON INSERT ONLY,
  'PhysicalPort' = '@PhysicalPort' ON INSERT ONLY,
  'PhysicalCard' = '@PhysicalCard' ON INSERT ONLY,
  'TaskList' = '@TaskList',
  'NmosSerial' = '@NmosSerial',
  'NmosObjInst' = '@NmosObjInst',
  'NmosCauseType' = '@NmosCauseType',
  'NmosDomainName' = '@NmosDomainName',
  'NmosEntityId' = '@NmosEntityId',
  'NmosManagedStatus' = '@NmosManagedStatus',
  'NmosEventMap' = '@NmosEventMap',
  'LocalNodeAlias' = '@LocalNodeAlias' ON INSERT ONLY,
  'LocalPriObj' = '@LocalPriObj' ON INSERT ONLY,
  'LocalSecObj' = '@LocalSecObj' ON INSERT ONLY,
  'LocalRootObj' = '@LocalRootObj' ON INSERT ONLY,
  'RemoteNodeAlias' = '@RemoteNodeAlias' ON INSERT ONLY,
  'RemotePriObj' = '@RemotePriObj' ON INSERT ONLY,
  'RemoteSecObj' = '@RemoteSecObj' ON INSERT ONLY,
  'RemoteRootObj' = '@RemoteRootObj' ON INSERT ONLY,
  'X733EventType' = '@X733EventType' ON INSERT ONLY,
  'X733ProbableCause' = '@X733ProbableCause' ON INSERT ONLY,
  'X733SpecificProb' = '@X733SpecificProb' ON INSERT ONLY,
  'X733CorrNotif' = '@X733CorrNotif' ON INSERT ONLY,
  'URL' = '@URL' ON INSERT ONLY,
```

```

'ExtendedAttr' = '@ExtendedAttr' ON INSERT ONLY,
'ServerName' = '@ServerName' ON INSERT ONLY,
'ServerSerial' = '@ServerSerial' ON INSERT ONLY
);

```

```

CREATE MAPPING JournalMap
(
'KeyField' = TO_STRING(STATUS.SERIAL) + ":" +
             TO_STRING('@UID') + ":" +
             TO_STRING('@Chrono') ON INSERT ONLY,
'Serial' = STATUS.SERIAL,
'Chrono' = '@Chrono',
'UID' = TO_INTEGER('@UID'),
'Text1' = '@Text1',
'Text2' = '@Text2',
'Text3' = '@Text3',
'Text4' = '@Text4',
'Text5' = '@Text5',
'Text6' = '@Text6',
'Text7' = '@Text7',
'Text8' = '@Text8',
'Text9' = '@Text9',
'Text10' = '@Text10',
'Text11' = '@Text11',
'Text12' = '@Text12',
'Text13' = '@Text13',
'Text14' = '@Text14',
'Text15' = '@Text15',
'Text16' = '@Text16'
);

```

```

CREATE MAPPING DetailsMap
(
'KeyField' = '@Identifier' + '####' +
             TO_STRING('@Sequence') ON INSERT ONLY,
'Identifier' = '@Identifier',
'AttrVal' = '@AttrVal',
'Sequence' = '@Sequence',
'Name' = '@Name',
'Detail' = '@Detail'
);

```

```

CREATE MAPPING IducMap
(
'ServerName' = '@ServerName' ON INSERT ONLY,
'AppName' = '@AppName',
'AppDesc' = '@AppDesc' ON INSERT ONLY,
'ConnectionId' = '@ConnectionId' ON INSERT ONLY,
'LastIducTime' = '@LastIducTime'
);

```

```

# CREATE MAPPING SecurityUsersMap
# (
# 'UserID' = '@UserID' ON INSERT ONLY,
# 'UserName' = '@UserName',
# 'SystemUser' = '@SystemUser',
# 'FullName' = '@FullName',
# 'Passwd' = '@Passwd',
# 'UsePAM' = '@UsePAM',
# 'Enabled' = '@Enabled'
# );
#
#

```

```

# CREATE MAPPING SecurityGroupsMap
# (
#   'GroupID' = '@GroupID'      ON INSERT ONLY,
#   'GroupName' = '@GroupName',
#   'SystemGroup' = '@SystemGroup',
#   'Description' = '@Description'
# );
#
#
# CREATE MAPPING SecurityRolesMap
# (
#   'RoleID' = '@RoleID'      ON INSERT ONLY,
#   'RoleName' = '@RoleName',
#   'SystemRole' = '@SystemRole',
#   'Description' = '@Description',
#   'RoleScope' = '@RoleScope'
# );
#
#
# CREATE MAPPING SecurityRoleGrantsMap
# (
#   'GranteeType' = '@GranteeType'      ON INSERT ONLY,
#   'GranteeID' = '@GranteeID'      ON INSERT ONLY,
#   'RoleID' = '@RoleID'      ON INSERT ONLY
# );
#
#
# CREATE MAPPING SecurityGroupMembersMap
# (
#   'UserID' = '@UserID'      ON INSERT ONLY,
#   'GroupID' = '@GroupID'      ON INSERT ONLY,
#   'Compat' = '@Compat'
# );
#
#
# CREATE MAPPING CatalogRestrictionFiltersMap
# (
#   'RestrictionName' = '@RestrictionName'      ON INSERT ONLY,
#   'TableName' = '@TableName',
#   'DatabaseName' = '@DatabaseName',
#   'ConditionText' = '@ConditionText',
#   'CreationText' = '@CreationText'
# );
#
#
# CREATE MAPPING SecurityRestrictionFiltersMap
# (
#   'GranteeType' = '@GranteeType'      ON INSERT ONLY,
#   'GranteeID' = '@GranteeID'      ON INSERT ONLY,
#   'RestrictionName' = '@RestrictionName',
#   'DatabaseName' = '@DatabaseName',
#   'TableName' = '@TableName'
# );
#
#
# CREATE MAPPING SecurityPermissionsMap
# (
#   'ApplicationID' = '@ApplicationID'      ON INSERT ONLY,
#   'ObjectType' = '@ObjectType'      ON INSERT ONLY,
#   'Object' = '@Object'      ON INSERT ONLY,
#   'GranteeType' = '@GranteeType'      ON INSERT ONLY,
#   'GranteeID' = '@GranteeID'      ON INSERT ONLY,
#   'Allows' = '@Allows',
#   'Denies' = '@Denies',
#   'GrantOptions' = '@GrantOptions'
# );
#

```

```

#
# CREATE MAPPING ToolsMenusMap
# (
# 'MenuID' = '@MenuID' ON INSERT ONLY,
# 'Name' = '@Name',
# 'Owner' = '@Owner',
# 'Enabled' = '@Enabled'
# );
#

# # CREATE MAPPING ToolsMenuItemsMap
# (
# 'KeyField' = TO_STRING('@MenuID') + ":" +
# TO_STRING('@MenuItemID')
# ON INSERT ONLY,
# 'MenuID' = '@MenuID' ON INSERT ONLY,
# 'MenuItemID' = '@MenuItemID' ON INSERT ONLY,
# 'Title' = '@Title',
# 'Description' = '@Description',
# 'Enabled' = '@Enabled',
# 'InvokeType' = '@InvokeType',
# 'InvokeID' = '@InvokeID',
# 'Position' = '@Position',
# 'Accelerator' = '@Accelerator'
# );
#
#
# CREATE MAPPING ToolsActionsMap
# (
# 'ActionID' = '@ActionID' ON INSERT ONLY,
# 'Name' = '@Name',
# 'Owner' = '@Owner',
# 'Enabled' = '@Enabled',
# 'Description1' = '@Description1',
# 'Description2' = '@Description2',
# 'Description3' = '@Description3',
# 'Description4' = '@Description4',
# 'HasInternal' = '@HasInternal',
# 'InternalEffect1' = '@InternalEffect1',
# 'InternalEffect2' = '@InternalEffect2',
# 'InternalEffect3' = '@InternalEffect3',
# 'InternalEffect4' = '@InternalEffect4',
# 'InternalForEach' = '@InternalForEach',
# 'HasExternal' = '@HasExternal',
# 'ExternalEffect1' = '@ExternalEffect1',
# 'ExternalEffect2' = '@ExternalEffect2',
# 'ExternalEffect3' = '@ExternalEffect3',
# 'ExternalEffect4' = '@ExternalEffect4',
# 'ExternalForEach' = '@ExternalForEach',
# 'RedirectOut' = '@RedirectOut',
# 'RedirectErr' = '@RedirectErr',
# 'Platform' = '@Platform',
# 'JournalText1' = '@JournalText1',
# 'JournalText2' = '@JournalText2',
# 'JournalText3' = '@JournalText3',
# 'JournalText4' = '@JournalText4',
# 'JournalForEach' = '@JournalForEach',
# 'HasForcedJournal' = '@HasForcedJournal'
# );
#
#
# CREATE MAPPING ToolsActionAccessMap
# (
# 'ActionAccessID' = '@ActionAccessID' ON INSERT ONLY,
# 'ActionID' = '@ActionID',
# 'GID' = '@GID',
# 'ClassID' = '@ClassID'

```

```

# );
#
#
# CREATE MAPPING ToolsMenuDefsMap
# (
# 'Name' = '@Name' ON INSERT ONLY,
# 'DatabaseName' = '@DatabaseName',
# 'TableName' = '@TableName',
# 'ShowField' = '@ShowField',
# 'AssignField' = '@AssignField',
# 'OrderByField' = '@OrderByField',
# 'WhereClause' = '@WhereClause'
# );
#
# CREATE MAPPING ToolsPromptDefsMap
# (
# 'Name' = '@Name' ON INSERT ONLY,
# 'Prompt' = '@Prompt',
# 'Default' = '@Default',
# 'Value' = '@Value',
# 'Type' = '@Type'
# );
#
#
# CREATE MAPPING AlertsConversionsMap
# (
# 'KeyField' = '@KeyField' ON INSERT ONLY,
# 'Colname' = '@Colname' ON INSERT ONLY,
# 'Value' = '@Value' ON INSERT ONLY,
# 'Conversion' = '@Conversion'
# );
#
# CREATE MAPPING AlertsColVisualsMap
# (
# 'Colname' = '@Colname' ON INSERT ONLY,
# 'Title' = '@Title',
# 'DefWidth' = '@DefWidth',
# 'MaxWidth' = '@MaxWidth',
# 'TitleJustify' = '@TitleJustify',
# 'DataJustify' = '@DataJustify'
# );
#
#
# CREATE MAPPING AlertsColorsMap
# (
# 'Severity' = '@Severity' ON INSERT ONLY,
# 'AackedRed' = '@AackedRed',
# 'AackedGreen' = '@AackedGreen',
# 'AackedBlue' = '@AackedBlue',
# 'UnackedRed' = '@UnackedRed',
# 'UnackedGreen' = '@UnackedGreen',
# 'UnackedBlue' = '@UnackedBlue'
# );
#
#
# CREATE MAPPING MasterServergroupsMap
# (
# 'ServerName' = '@ServerName' ON INSERT ONLY,
# 'GroupID' = '@GroupID',
# 'Weight' = '@Weight'
# );

```

Chapter 6. Startup command file

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

After the gateway has started, you can use the **nco_sql** command to issue the commands manually. For details about **nco_sql**, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*.

SHOW PROPS

Use the **SHOW PROPS** command to display the current configuration of the gateway by listing all properties and their values.

Syntax

SHOW PROPS ;

Example

```
SHOW PROPS ;  
go
```

GET CONFIG

Use the **GET CONFIG** command to display the current configuration of the gateway by listing all properties and their values.

GET CONFIG is identical to the **SHOW PROPS** command; it may be removed from later versions of the ObjectServer gateway.

Syntax

GET CONFIG ;

Example

```
GET CONFIG ;  
go
```

FAILOVER SYNCH

Use the **FAILOVER SYNCH** command to synchronize data between primary and backup ObjectServers. The command specifies which master tables are transferred during the data transfer operation.

For information about ObjectServer failover, see the *IBM Tivoli Netcool/OMNIbus Administration Guide* (SC14-7605).

Syntax

**FAILOVER_SYNC [ADD 'TABLENAME' TO | REMOVE 'TABLENAME' FROM
] WRITERNAME ;**

Example

```
FAILOVER_SYNC ADD 'master.names' TO ObjectServerA;
FAILOVER_SYNC ADD 'master.groups' TO ObjectServerA;
FAILOVER_SYNC ADD 'master.members' TO ObjectServerA;
FAILOVER_SYNC ADD 'master.permissions' TO ObjectServerA;
FAILOVER_SYNC ADD 'master.profiles' TO ObjectServerA;
FAILOVER_SYNC ADD 'tools.actions' TO ObjectServerA;
FAILOVER_SYNC ADD 'tools.action_access' TO ObjectServerA;
FAILOVER_SYNC ADD 'tools.menus' TO ObjectServerA;
FAILOVER_SYNC ADD 'tools.menu_defs' TO ObjectServerA;
FAILOVER_SYNC ADD 'tools.menu_items' TO ObjectServerA;
FAILOVER_SYNC ADD 'tools.prompt_defs' TO ObjectServerA;
FAILOVER_SYNC ADD 'alerts.conversions' TO ObjectServerA;
FAILOVER_SYNC ADD 'alerts.col_visuals' TO ObjectServerA;
FAILOVER_SYNC ADD 'alerts.colors' TO ObjectServerA;
FAILOVER_SYNC ADD 'alerts.objclass' TO ObjectServerA;
FAILOVER_SYNC ADD 'alerts.objmenus' TO ObjectServerA;
FAILOVER_SYNC ADD 'alerts.objmenuitems' TO ObjectServerA;
```

Chapter 7. Table replication definition file

The table replication definition file defines the tables in the source that the ObjectServer replicates in the target system.

For unidirectional gateways, the table definition file is specified using the **Gate.Reader.TblReplicateDefFile** property. For bidirectional gateways, a table definition file can be specified for both ObjectServers using the **Gate.ObjectServerA.TblReplicateDefFile** and **Gate.ObjectServerB.TblReplicateDefFile** properties.

Syntax

```
REPLICATE {ALL | INSERTS, UPDATES, DELETES | FT_INSERTS,
FT_UPDATES, FT_DELETES} FROM TABLE sourcetable USING MAP
mapname [FILTER WITH filter_clause] [INTO destinationtable] [ ORDER BY
column_name ] [WITH NORESYNC][RESYNC DELETES FILTER
condition] [SET UPDTOINS CHECK TO {ENABLED|DISABLED|FORCED}]
[ AFTER IDUC DO command] [ CACHE FILTER condition]
```

Where:

- *sourcetable* is the table to be replicated in the target ObjectServer.
- *mapname* is the map definition that defines the table.
- *filter_clause* defines the filter the gateway uses to select rows for replicating. By default, filtering is inclusive. That is, the filter sends only those events that match the filter definition. However, you can specify that the filter sends events that do not match the filter definition by preceding the equals sign (=) with an exclamation mark (!). For example, the following filter clause sends all events whose severity is not set to 5: FILTER with 'Severity !=5'.
- *destinationtable* is the table to receive the replicated table. If this clause is omitted the name of the destination table is the same as the value of *sourcetable*.
- *column_name* is the column by which the rows returned to the gateway from the ObjectServer are ordered.
- *condition* is the SQL condition that the gateway adds to the SELECT statement when limiting the cache entries that the gateway retrieves during a cache refresh.
- *propertyname* is the property the gateway uses to filter the table data (only rows that satisfy the filter are replicated).
- *propertyvalue* is argument to be used in the filter.
- *targettable* is the name of the table in which to replicate the data.
- *delete_filter_clause* defines the resynchronization delete filter that the gateway issues to the target ObjectServer.

ALL is the equivalent of INSERTS, UPDATES, DELETES.

The optional WITH NORESYNC option allows you to specify tables that should not be resynchronized.

The optional ORDER BY option allows you to specify the order in which the rows are returned to the gateway from the ObjectServer. ASC specifies that the rows are

sorted in ascending order; DESC specifies descending order. If you specify neither ASC nor DESC, the rows are sorted in ascending order.

You can define multiple columns for sorting by specifying a comma-separated list; for example:

```
ORDER BY 'Serial DESC, StateChange ASC'
```

The optional RESYNC DELETE FILTER option is used when the rows in the target and source tables are not an exact match. This option allows you to define a resynchronization deletion filter that specifies which rows to remove before insertion into the target table.

The optional SET UPDTOINS CHECK TO option allows you to configure the update-to-insert functionality. ENABLED (the default setting) instructs the gateway to perform normal update-to-insert conversions - when an update is received from the source ObjectServer for a given table, and the row that has been updated does not exist within the destination ObjectServer, it converts the update to an insert, so that the row is repopulated within the destination ObjectServer. If the row does exist within the destination, the gateway will update the existing row with the update from the source ObjectServer; DISABLED instructs the gateway to always send an update to the destination ObjectServer for each update received from the source ObjectServer - any update sent to the destination for rows that do not exist are dropped as they will affect no rows within the destination ObjectServer; FORCED instructs the gateway to convert all updates from the source ObjectServer to an insert on the destination ObjectServer - if the row already exists within the destination ObjectServer, a de-duplication action is performed by the destination ObjectServer - this effectively makes the gateway behave like a probe.

The optional AFTER IDUC DO option allows you to specify a column and associated value that the gateway applies to all rows that have been inserted, updated, or deleted.

The optional CACHE FILTER option allows you to reduce the amount of data that the gateway retrieves during a unidirectional gateway cache refresh. The gateway adds the condition to the end of the select statement that it uses to retrieve cache entries.

Note: If you want to use the CACHE FILTER option, it must be the last entry in the table replication definition.

Forwarding deletes and memory size

When the forwarding of deletes is disabled (for example, if only inserts and updates are specified by the REPLICATE command), the deletion details are dropped by the ObjectServer gateway mapper and not passed on to the writer itself. This means that the cache entries for the deleted rows are not removed from the cache. To overcome this, the ObjectServer gateway mapper deletes cache entries for deleted rows in the destination ObjectServer cache on behalf of the ObjectServer gateway writer when the forwarding of deletes has been disabled. This allows you to maintain a stable memory size.

If the forwarding of deletes is enabled and the destination ObjectServer is not fast enough to keep up with the data being sent to it, the memory used for the gateway will grow in size.

Example table replication definition file

The table replication definition file defines how the ObjectServer gateway replicates tables between the source and target ObjectServers. Use this example to familiarize yourself with how the file works.

The following example shows a table replication definition file:

```
REPLICATE INSERT, DELETE FROM TABLE 'alerts.status'
  USING MAP 'StatusMap'
ORDER BY 'Serial ASC'
FILTER WITH 'Severity!=5'
SET UPDTOINS CHECK TO FORCED
AFTER IDUC DO 'Location=\'PASSED BY GW\''
CACHE FILTER 'ServerName IN (\'NCOMBS_P\',\'NCOMBS_B\')';

REPLICATE ALL FROM TABLE 'alerts.journal'
  USING MAP 'JournalMap';

REPLICATE ALL FROM TABLE 'alerts.details'
  USING MAP 'DetailsMap';

#####
# NOTE: If replication of the user related system tables is required, uncomment
# the replication definitions below. The associated maps will also need to be
# uncommented.
#####

# REPLICATE ALL FROM TABLE 'security.users'
# USING MAP 'SecurityUsersMap'
# INTO 'transfer.users';
#
# REPLICATE ALL FROM TABLE 'security.groups'
# USING MAP 'SecurityGroupsMap'
# INTO 'transfer.groups';
#
# REPLICATE ALL FROM TABLE 'security.roles'
# USING MAP 'SecurityRolesMap'
# INTO 'transfer.roles';
#
# REPLICATE ALL FROM TABLE 'security.role_grants'
# USING MAP 'SecurityRoleGrantsMap'
# INTO 'transfer.role_grants';
#
# REPLICATE ALL FROM TABLE 'security.group_members'
# USING MAP 'SecurityGroupMembersMap'
# INTO 'transfer.group_members';
```

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 wish 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
958/NH04
IBM Centre, St Leonards
601 Pacific Hwy
St Leonards, NSW, 2069
Australia

IBM Corporation
896471/H128B
76 Upper Ground
London SE1 9PZ
United Kingdom

IBM Corporation
JBF1/SOM1
294 Route 100
Somers, NY, 10589-0100
United States of America

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 document 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.

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.

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

Trademarks

AIX, IBM, the IBM logo, `ibm.com`[®], Netcool, Netcool/OMNIBus, and Tivoli are trademarks or registered 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.



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

Linux is a registered trademark of Linus Torvalds 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.

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

Other company, product, or service names may be trademarks or service marks of others.

Index

Special characters

\$OMNIHOME/etc folder 7, 21

A

activate store and forward
 bidirectional gateway 38
 unidirectional gateway 19
aggregation ObjectServer 38
alternative deletion strategy 38
attributes, cache value access 42
attributes, dynamic 42
attributes, mapping 42
authentication
 bidirectional gateway 36
 unidirectional gateway 17

B

backup ObjectServer
 bidirectional gateway 37
 unidirectional gateway 18
BackupObjectServer property
 bidirectional gateway 37
 unidirectional gateway 18
backward compatibility 1
bidirectional gateway
 properties file 21
bidirectional ObjectServer gateway
 alternative deletion strategy 38
 buffer size 36
 configuration 21, 39
 common gateway properties 21
 debugging 36
 description 3
 error handling 36
 example 3
 hash table cache 35
 information flow 3
 properties 37
 resynchronization properties 39
 secure mode connection 37
 start 21
 store and forward 38
buffer size
 bidirectional gateway 36
 unidirectional gateway 17

C

cache value access attributes 42
cache, hash table
 bidirectional gateway 35
 unidirectional gateway 16
command
 FAILOVER SYNCH 49
 GET CONFIG 49
 nco_g_objserv_bi 21
 nco_g_objserv_uni 7

command (*continued*)
 nco_sql 49
 SHOW PROPS 49
command file, startup 49
command line options
 bidirectional gateway
 common gateway properties 21
 resynchronization properties 39
 unidirectional gateway
 common gateway properties 7
 resynchronization properties 19
commands 49
compatibility
 with IBM Tivoli Netcool/OMNIbus
 version 3.x 1
configuration
 common gateway properties
 bidirectional gateway 21
 unidirectional gateway 7
 resynchronization properties
 bidirectional gateway 39
 unidirectional gateway 19
conversion functions 41

D

deactivate store and forward
 bidirectional gateway 38
 unidirectional gateway 19
debugging
 bidirectional gateway 36
 unidirectional gateway 17
deletion events 38
deletion filter, resynchronization 51
deletion strategy, alternative 38
dynamic attributes 42

E

enable failback
 bidirectional gateway 37
 unidirectional gateway 18
enable store and forward
 bidirectional gateway 38
 unidirectional gateway 19
error handling
 bidirectional gateway 36
 unidirectional gateway 17
example table replication definition
 file 53

F

failback
 description 1
 example 1
 setup
 bidirectional gateway 37
 unidirectional gateway 18
failover 49

FAILOVER SYNCH command 49
file, map definition 41, 43
file, table replication definition 51
fine-tuning
 bidirectional gateway 36
 unidirectional gateway 17

G

Gate.Mapper.Debug property
 bidirectional gateway 36
 unidirectional gateway 17
Gate.ObjectServerA.DeleteIfNoDedup 38
Gate.ObjectServerA.Failback property 37
Gate.ObjectServerA.FailbackTimeout
 property 37
Gate.ObjectServerB.DeleteIfNoDedup 38
Gate.ObjectServerB.Failback property 37
Gate.ObjectServerB.FailbackTimeout
 property 37
Gate.Reader.Debug property 17
Gate.Reader.Failback property
 bidirectional gateway 37
 unidirectional gateway 18
Gate.UsePamAuth property
 bidirectional gateway 36
 unidirectional gateway 17
Gate.Writer.BufferSize property
 bidirectional gateway 36
 unidirectional gateway 17
Gate.Writer.Debug property 17
Gate.Writer.Failback property
 bidirectional gateway 37
 unidirectional gateway 18
Gate.Writer.SAF property
 bidirectional gateway 38
 unidirectional gateway 19
Gate.Writer.SAFFile property
 bidirectional gateway 38
 unidirectional gateway 19
gateway mapping, ObjectServer 41
GET CONFIG command 49

H

handle errors
 bidirectional gateway 36
 unidirectional gateway 17
hash table cache
 bidirectional gateway 35
 unidirectional gateway 16

M

map definition file 41, 43
map definition file conversion
 functions 41
mapper 2
mapping attributes 42
mapping, ObjectServer gateway 41

N

- nco_g_objserv_bi command 21
- nco_g_objserv_uni command 7
- nco_sql command 49
- NGTK library 17, 36

O

- ObjectServer
 - backup 3
 - failover pair 3
 - secure mode connection
 - bidirectional gateway 37
 - unidirectional gateway 18
- ObjectServer failover 49
- ObjectServer gateway
 - bidirectional 3
 - common gateway properties 21
 - configuration 21
 - resynchronization properties 39
 - description 1
 - example mapping 43
 - features 1
 - licensing 1
 - mapping 41
 - summary 1
 - unidirectional 2
 - common gateway properties 7
 - resynchronization properties 19
 - user and password 18, 37
 - uses 1
- ObjectServers, primary and backup 49
- operation, table insert
 - bidirectional gateway 35
 - unidirectional gateway 16

P

- PAM authentication
 - bidirectional gateway 36
 - unidirectional gateway 17
- pass data between ObjectServers 1
- platforms, supported 1
- primary and backup ObjectServers 49
- properties file
 - bidirectional ObjectServer gateway 21
 - unidirectional gateway 7

R

- reader 2
- reader/writer 3
- readers 1
- reduce latency
 - bidirectional gateway 36
 - unidirectional gateway 17
- replicate data between ObjectServers 1
- replication, table 51
- resynchronization deletion filter 51
- resynchronization properties
 - bidirectional gateway 39
 - unidirectional gateway 19

S

- set up failback
 - bidirectional gateway 37
 - unidirectional gateway 18
- SHOW PROPS command 49
- start bidirectional gateway 21
- start unidirectional gateway 7
- startup command file 49
- store and forward
 - bidirectional gateway 38
 - unidirectional gateway 19
- supported platforms 1

T

- table data
 - centralized property management 1
 - property management, centralized 1
 - replication 1
- table data
 - passing 1
- table insert operation
 - bidirectional gateway 35
- table insert operations
 - unidirectional gateway 16
- table replication 51
- table replication definition file 51
- table replication definition file, example 53
- transfer data between ObjectServers 1

U

- unidirectional ObjectServer gateway
 - authentication 17
 - buffer size 17
 - configuration 7
 - common gateway properties 7
 - resynchronization properties 19
 - description 2
 - error handling 17
 - example 2
 - hash table cache 16
 - information flow 2
 - properties 19
 - properties file 7
 - start 7
 - store and forward 19
- unidirectional ObjectServer gateway
 - properties
 - ObjectServer
 - secure mode 18
 - secure mode connection 18
 - update-to-insert 51

W

- writer 2
- writers 1



Printed in USA

SC14-6090-00

