IBM® Tivoli® Netcool/OMNIbus Java Gateway for BMC Remedy ARS 3.0

Reference Guide
June 28, 2016

SC27-6553-02

Notice

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 59.

Edition notice

This edition (SC27-6553-02) applies to version 3.0 of IBM Tivoli Netcool/OMNIbus Java Gateway for BMC Remedy ARS and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-6553-01.

Contents

About this guide

The following sections contain important information about using this guide.

Document Control Page

Use this information to track changes between versions of this guide.

The IBM Tivoli Netcool/OMNIbus Java Gateway for BMC Remedy ARS documentation is provided in softcopy format only. To obtain the most recent version, please visit the IBM® Tivoli® Netcool® knowledge center:

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

Table 1. Document modification history

Document version	**Publication date**	**Comments**
SC27-6553-00	November 7, 2014	First IBM publication.
SC27-6553-01	August 6, 2015	Information about new gateway framework properties were added to the Properties and command line options and a new subtopic Suppressing Deletes on and after Resynchronization was created.
SC27-6553-02	June 28, 2016	Updates were made to the following section: "Summary" on page 1

Conventions used in this guide

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

Operating system-dependent variables and paths

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

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

```
$OMNIHOME/gates
```

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

```
%OMNIHOME%\gates
```

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

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

Operating system-specific directory names

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

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

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

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

Table 2. Directory names for the arch variable

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

OMNIHOME location

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

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

Chapter 1. Java Gateway for BMC Remedy ARS

The IBM Tivoli Netcool/OMNIbus Java Gateway for BMC Remedy ARS works with the BMC Remedy Action Request System (ARS), which is a help desk system operating on a variety of operating system platforms. The BMC Remedy ARS uses a system of trouble requests.

The Java Gateway for BMC Remedy ARS is a bidirectional gateway that creates requests in BMC Remedy ARS from alerts sent by the ObjectServer.

This guide contains the following sections:

Summary

Use this summary information to learn about the Java Gateway for BMC Remedy ARS.

The following table provides a summary of the gateway:

Table 3. Summary

Gateway target	The gateway supports BMC Remedy ARS (BMC Remedy Action Request System) version 7.6, version 8.1, and version 9.0. **Note :** Although the gateway supports interacting with a BMC Remedy ARS version 9.0 server, it does not support the BMC Remedy ARS 9.0 API.
Gateway jar file	`$OMNIHOME/gates/java/nco_g_bmc_remedy.jar`
Additional gateway jar files	Additional gateway jar files are installed in the following directory: `$OMNIHOME/gates/java`
Package Version	3.0
Gateway supported on	For details of supported operating systems, see the following Release Notice on the IBM Software Support website: https://www-304.ibm.com/support/docview.wss?uid=swg21687618

Table 3. Summary (continued)

Configuration files	Map definition file:
	`$OMNIHOME/gates/bmc_remedy/bmc_remedy.map`
	Properties files:
	`$OMNIHOME/gates/bmc_remedy/G_BMC_REMEDY.props`
	`$OMNIHOME/gates/bmc_remedy/log4j.properties`
	Table replication definition file:
	`$OMNIHOME/gates/bmc_remedy/ bmc_remedy.rdrwtr.tblrep.def`
	JavaScript files:
	`$OMNIHOME/gates/bmc_remedy/bmc_remedy.js`
	`$OMNIHOME/gates/bmc_remedy/ bmc_remedy.notification.js`
	Environment (.env) file:
	`$OMNIHOME/gates/bmc_remedy/nco_g_bmc_remedy.env`
	Conversion table:
	`$OMNIHOME/gates/bmc_remedy/ createConversionsTable.sql`
	Module-definition (.def) file:
	`$OMNIHOME/gates/bmc_remedy/omnibus.def`

Table 3. Summary (continued)	

Requirements	A currently supported version of IBM Tivoli Netcool/OMNIbus.
	Java runtime environment (JRE) 1.6.0 or later (32-bit or 64-bit)
	The following packages that are available from the BMC Remedy ARS server installation:
	• `arapi`*`X`*`_build`*`Y`*`.jar`, where *X* specifies the BMC Remedy ARS version number and *Y* specifies the build number.
	For example:
	`arapi7604_build002.jar`
	`arapi81_build001.jar`
	• `log4j-`*`xxx`*`.jar`
	where *xxx* specifies the version number of the Apache Log4j product. For example: `log4j-1.2.14.jar`.
	The following jar files from Tivoli Netcool/OMNIbus:
	• `$OMNIHOME/java/jars/jconn3.jar` (Sybase JConnect JDBC driver)
	• `$OMNIHOME/java/jars/niduc.jar`
	Note : The previously listed jar files and driver require the installation of the 'NCODesktop' feature in Tivoli Netcool/OMNIbus versions prior to version 8.1. The previously listed jar files require the installation of the 'GatewaySupport' feature in Tivoli Netcool/OMNIbus version 8.1.
	The following packages are provided with the gateway:
	• `common-jnetcool-2` package or higher
	• `gateway-nco-g-java-3` or higher

Remote Connectivity	Available
SSL support	Available
Multicultural support	Available For information about configuring multicultural support, including language options, see the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*.
IP environment	IPv4 and IPv6
Federal Information Processing Standards (FIPS)	IBM Tivoli Netcool/OMNIbus uses the FIPS 140-2 approved cryptographic provider: IBM Crypto for C (ICC) certificate 384 for cryptography. This certificate is listed on the NIST website at http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2004.htm. For details about configuring Netcool/OMNIbus for FIPS 140-2 mode, see the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*.

Overview

The Java Gateway for BMC Remedy ARS is a bidirectional gateway that creates requests in BMC Remedy ARS from alerts sent by the ObjectServer.

In a typical environment, users define workflows within BMC Remedy ARS to create trouble requests from the requests that the gateway creates. These trouble requests are updated, according to a predefined mapping, throughout the life-span of the alert. For each request that the gateway creates, BMC Remedy ARS sends back an associated request ID.

BMC Remedy ARS also passes back to the gateway any changes to the requests so that the gateway can update the original ObjectServer alert.

The following list describes some of the functions that the gateway supports:

- Flexible port mapping to BMC Remedy ARS.
- Flexible event filtering that allows you to specify which alerts the gateway uses to create the corresponding requests in BMC Remedy ARS.
- Reporting within Tivoli Netcool/OMNIbus of BMC Remedy ARS operation failures.
- Forwarding updates to BMC Remedy ARS requests when the associated events in Tivoli Netcool/OMNIbus are updated or deleted.
- The ability for BMC Remedy ARS to pass back any changes to the requests, so that the gateway can update the original ObjectServer alert.
- The gateway forwards journal entries from alerts in Tivoli Netcool/OMNIbus to BMC Remedy ARS.
- Conversion of ObjectServer values and BMC Remedy ARS values. These conversions are bidirectional.
- The gateway can perform two types of resynchronization: unidirectional and bidirectional.
- The gateway provides JavaScript files for processing bidirectional updates (that is, updates from BMC Remedy ARS to the ObjectServer).
- The gateway provides a store and forward capability to minimize any data loss should the gateway lose communication with BMC Remedy ARS.

Features of the IBM Tivoli Netcool/OMNIbus Java Gateway for BMC Remedy ARS

The IBM Tivoli Netcool/OMNIbus Java Gateway for BMC Remedy ARS has various features that enable you to create an interface between BMC Remedy ARS and the ObjectServer.

Event forwarding

The gateway uses the Insert, Delete, Update, or Control (IDUC) communication protocol to retrieve events from ObjectServer tables. The gateway can replicate the data in any table between the ObjectServer and the destination server. Details of the tables to be replicated are stored in the table replication definition file. The events retrieved from these tables are based on the table replication definition file configuration, including their filtering. Retrieved events are then passed through a mapper to assign values to target fields that the gateway is required to populate and update on the BMC Remedy ARS system. The mappers are specified in the table replication definition file and defined in the map definition file.

"Table Replication" on page 19 contains more information on replicating data between the ObjectServer and BMC Remedy ARS

See the *IBM Tivoli Netcool/OMNIbus Administration Guide* and the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* for more information on IDUC.

Mapping table data

The gateway writes the alerts received from the various tables in the ObjectServer onto BMC Remedy ARS in a format defined by the mappers defined in the map definition file. Each of the maps defines how to map an alert into a request in BMC Remedy ARS (as defined by the **`Gate.Remedy.Form`** property).

"Mapping" on page 21 contains more information on the map definition file.

Unidirectional and bidirectional resynchronization

The gateway can perform two types of resynchronization: unidirectional and bidirectional. There is also an automatic mode that causes the gateway to perform either unidirectional or bidirectional synchronization, depending on whether its cache is empty on startup.

"Resynchronization" on page 33 provides additional information about unidirectional and bidirectional resynchronization.

Installing the gateway

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

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

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

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

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

Installing probes and gateways using the Command Line Tool

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

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

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

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

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

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

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

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

Uninstalling probes and gateways using the Command Line Tool

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

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

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

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

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

Installing probes and gateways using the GUI

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

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

   `installer_path/IBMIM`

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

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

Uninstalling probes and gateways using the GUI

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

1. Start Installation Manager using the following command:

   `installer_path/IBMIM`

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

Installing the gateway on Tivoli Netcool/OMNIbus V7.4.0

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

You can install the gateway using any of the following:

- "The installation wizard" on page 7
- A text-based installer ("Console mode" on page 7)
- Settings predefined in a text file ("Silent mode" on page 7)

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

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

Obtaining the installation package

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

1. Download the installation package for the gateway from the Passport Advantage Online Web site:

   http://www-306.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm
2. Make a backup of any existing configuration files that you want to retain.
3. Extract the contents of the installation package to a temporary directory.

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

```
$NCHOME/omnibus/gates
```

The installation wizard

To install the gateway using the installation wizard:

1. Run the installer for your operating system:

   ```
   $NCHOME/omnibus/install/nco_install_integration
   ```
2. When the installation wizard starts, specify the extracted directory that contains the `README.txt` file as the location of the gateway installation files.
3. Accept the license conditions.

Console mode

To install the gateway in console mode:

1. Run the installer for your operating system:

   ```
   $NCHOME/omnibus/install/nco_install_integration -i console
   ```
2. When the text-based installer starts, specify the extracted directory that contains the `README.txt` file as the location of the gateway installation files.
3. Accept the license conditions.

Silent mode

To install the gateway in silent mode:

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

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

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

2. Run the installer for your operating system:

   ```
   $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.

Installing the BMC Remedy ARS jar files

Use this information to install the BMC Remedy ARS jar files.

Overview

For the gateway to run, you need to obtain and install jar files from the BMC Remedy ARS installation.

IBM does not distribute the BMC Remedy ARS jar files as part of the gateway installation. You need to obtain the BMC Remedy ARS jar files from the BMC Remedy ARS installation and install them separately from the gateway installation.

You need to install the following required BMC Remedy ARS jar files:

- `arapixxxx_buildyyy.jar`

  where *xxxx* specifies the version number and *yyy* specifies the build number of the Action Request (AR) System components. For example: `arapi7604_build002.jar`.

  This jar file includes the AR System Java API, Java utilities, and the AR System server.

- `log4j-xxx.jar`

  where *xxx* specifies the version number of the Apache Log4j product. For example: `log4j-1.2.14.jar`.

  This jar file contains the Apache Log4j libraries that the BMC Remedy ARS API uses.

Performing the installation of the BMC Remedy ARS jar files

Note : Before performing these steps, you should have installed BMC Remedy ARS (typically, on a different server than the Java Gateway for BMC Remedy ARS). It is assumed that you have knowledge of the BMC Remedy ARS solution and that you are also familiar with the BMC Remedy ARS APIs environment.

To install the BMC Remedy ARS jar files use the following step:

1. Copy the `arapixxxx_buildyyy.jar` and `log4j-zzz.jar` files from the installation directory on the BMC Remedy ARS server to the following directory on the Java Gateway for BMC Remedy ARS server:

   ```
   $NCHOME/omnibus/gates/java
   ```

Replace *xxxx* and *yyy* with the version number and build number of the BMC Remedy ARS components.

Replace *zzz* with the version number of the Apache Log4j product.

Note : There is no need to extract the files.

Configuring the gateway

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

The following table lists gateway configuration tasks. For each configuration task, the table lists the properties you use with that task, and the section in this guide that shows you how to complete the configuration task.

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

Table 4. Configuring the gateway

Configuration tasks	**Properties**	**See**
Required configuration tasks:		
Authentication configuration tasks		
Authenticate the gateway with BMC Remedy ARS	**`Gate.Remedy.Password`** **`Gate.Remedy.Username`**	"Authenticating the gateway With BMC Remedy ARS" on page 10
Authenticate the gateway with the ObjectServer **Note :** The gateway needs to authenticate with the ObjectServer only when the ObjectServer runs in secure mode.	**`Gate.RdrWtr.Password`** **`Gate.RdrWtr.Username`**	"Authenticating the gateway With the ObjectServer" on page 10
Connection Configuration Tasks		
Connect the gateway to the ObjectServer	**`Gate.RdrWtr.Server`**	"Connecting the gateway to the ObjectServer" on page 11
Connect the gateway to the BMC Remedy ARS server by: Establishing the connection to BMC Remedy ARS	**`Gate.Remedy.Port`** **`Gate.Remedy.Server`**	"Establishing the connection to BMC Remedy ARS" on page 11
Connect the gateway to the BMC Remedy ARS server by: Validating the gateway properties file	**`Gate.Remedy.Form`**	"Validating the gateway properties file" on page 12
Optional configuration tasks:		
Table replication configuration task		
Define the tables and event types that are replicated between the ObjectServer and BMC Remedy ARS.	**`Gate.RdrWtr.TblReplicateDefFile`**	"Table Replication" on page 19
Data mapping configuration task		

Table 4. Configuring the gateway (continued)

Configuration tasks	Properties	See
Define how fields in ObjectServer tables map to fields in BMC Remedy ARS.	**`Gate.Mapfile`**	"Mapping" on page 21
Improve gateway performance configuration task		
Specify the number of simultaneous connections the gateway makes to BMC Remedy ARS	**`Gate.Remedy.Connections`**	"Improving the performance of the gateway" on page 12
Journal updates configuration task		
Specify whether the gateway forwards historic journal information.	**`Gate.HistoricResync`**	"Historic journal forwarding" on page 25
Store and forward configuration task		
Controls the operation of the store and forward functionality.	**None**	" Store and forward capability" on page 25
FIPS mode and encryption configuration tasks		
Operate the gateway using FIPS mode and encryption.	**None**	"FIPS mode and encryption" on page 26
Operate the gateway using AES encryption.	**None**	"AES encryption" on page 26

Authentication

The gateway needs to authenticate itself with BMC Remedy ARS at all times. The gateway needs to authenticate itself with the ObjectServer only when the ObjectServer runs in secure mode.

Authenticating the gateway With BMC Remedy ARS

Create a dedicated gateway user on the BMC Remedy ARS system by specifying a user name and password on BMC Remedy ARS that only the gateway uses. This enables the gateway to track changes made to trouble tickets that BMC Remedy ARS creates. It also helps to ensure that the operation of the gateway does not interfere with any other user of the BMC Remedy ARS system.

Set the gateway's **`Gate.Remedy.Username`** and **`Gate.Remedy.Password`** to the user name and password of the gateway's BMC Remedy ARS account.

Authenticating the gateway With the ObjectServer

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

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

Connecting the gateway to the ObjectServer

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

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

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

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

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

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

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

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

Connecting to BMC Remedy ARS

The connection sequence that the gateway uses has two phases: establishing the connection to BMC Remedy ARS and data validation.

Establishing the connection to BMC Remedy ARS

Use the following properties to establish connection to BMC Remedy ARS:

- Set the **`Gate.Remedy.Server`** property to the name of the system where the gateway can interact with BMC Remedy ARS. Always specify this property.
- The **`Gate.Remedy.Port`** defines the port on the BMC Remedy ARS to use for the connection. The value of that property depends on whether the BMC Remedy ARS is using a port mapper. The following table shows how to specify the property.

Table 5. Setting the Gate.Remedy.Port property

Port mapper usage	**Value of Gate.Remedy.Port**
Port mapper in use	Set the value of the property to `0` (the default value).
Port mapper not in use	Set the value of the property to the TCP port to which the gateway connects.

- Set the **`Gate.Remedy.Username`** property to the username for the user of the gateway.
- Set the **`Gate.Remedy.Password`** property to the password for the user specified in the **`Gate.Remedy.Username`** property.

When the gateway starts, it connects to the specified system and port, if there is no port mapper, and logs in using the supplied user name and password. "Authentication" on page 10 contains more information on specifying the user name and password for BMC Remedy ARS.

On gateway startup, if the gateway cannot connect to BMC Remedy ARS for any reason, it shuts down. If the gateway starts up successfully and subsequently loses its connection with BMC Remedy ARS, the gateway goes into store and forward mode.

See " Store and forward capability" on page 25 for information on the store and forward feature.

Validating the gateway properties file

When the gateway starts, but before it connects to the BMC Remedy ARS server, it validates the gateway properties file, `G_BMC_REMEDY.props`, to ensure that each property has the correct syntax. After the gateway successfully connects to the BMC Remedy ARS server, it performs the following operations:

- Validates the BMC Remedy ARS form used to create tickets as defined in the **`Gate.Remedy.Form`** property.
- Validates the BMC Remedy ARS form used to read ticket updates as defined in the **`Gate.Remedy.UpdateForm`** property.
- Validates the mapping file as defined in the **`Gate.MapFile`** property. Specifically, the gateway:
  - Ensures that the mapping file contains at least the status map (`StatusMap`). If the status map is not present, the gateway shuts down.
  - Checks that every BMC Remedy ARS field residing in all maps is a valid BMC Remedy ARS field name.

Improving the performance of the gateway

To improve the performance of the gateway use the **`Gate.Remedy.Connections`** property (defined in the `G_BMC_REMEDY.props` file). The **`Gate.Remedy.Connections`** property takes an integer value that specifies the number of simultaneous connections the gateway makes with BMC Remedy ARS. In general, the more connections specified, the faster the gateway can create requests in BMC Remedy ARS from alerts sent by the ObjectServer.

In the following gateway properties file example, the **`Gate.Remedy.Connections`** property is set to the value 6, which means the gateway will make six simultaneous connections with BMC Remedy ARS:

```
# Property Name:Default
.
.
.
# Remedy gateway specific properties
.
.
.
Gate.Remedy.Connections:6
.
.
.
```

Note : If specified, the value of the **`Gate.Remedy.Connections`** property should be greater than 0 (zero). Also, the underlying hardware architecture must align with the number of connections.

Using an environment file to define Java options

Use the `nco_g_bmc_remedy.env` environment file to define Java options that you want the Java Gateway for BMC Remedy ARS to pick up at run time.

Overview

The *JRE_OPTS* environment variable defined in the `nco_g_bmc_remedy.env` file defines Java options specific to the Java Gateway for BMC Remedy ARS. The *JRE_OPTS* environment variable is supplied with Java options that define one system property: *log4j.configuration*. At start-up, the gateway picks up the contents of the *JRE_OPTS* environment variable.

The following sections discuss in more detail the `nco_g_bmc_remedy.env` file and the `log4j.properties` file (which is used to configure the log4j logging package).

The nco_g_bmc_remedy.env file

The `nco_g_bmc_remedy.env` file supplied with the gateway resides in the `$OMNIHOME/gates/bmc_remedy` directory and contains an environment variable called *JRE_OPTS*. The *JRE_OPTS* environment variable is supplied with the following Java options that make use of the `-D` switch to define the following system property that gets executed at run time:

- *log4j.configuration*

  This system property specifies the location and name (`log4j.properties`) of the properties file used to configure the log4j logging package that is distributed under the Apache Software License.

To define additional Java options specific to the gateway, add them to the *JRE_OPTS* environment variable in the `nco_g_bmc_remedy.env` file.

The log4j.properties file

The purpose of the `log4j.properties` file is to set the level of the BMC Remedy ARS debug messages.

The `log4j.properties` file supplied with the Java Gateway for BMC Remedy ARS resides in the `$OMNIHOME/gates/bmc_remedy` directory and has the following properties (key-value pairs):

```
log4j.rootLogger=INFO  1
```

The following list provides an explanation for the numbered item in the supplied `log4j.properties` file.

Note : The following description assumes an understanding of the format of a `log4j.properties` file.

1. Specifies the debug level of the root logger as `INFO`.

   You can change the debug level to the standard log4j debug levels (for example, TRACE, WARN, and so forth). This change impacts the information that BMC Remedy ARS (not the gateway) prints.

   Note : The root logger sends BMC Remedy ARS (not gateway) output to the gateway log.

Using the alert functions defined in bmc_remedy.notification.js

Use the alert functions defined in the `bmc_remedy.notification.js` file to perform a variety of operations on Tivoli Netcool/OMNIbus alerts. The alert functions are called by the Java Gateway for BMC Remedy ARS.

Overview

The `bmc_remedy.notification.js` file defines a variety of alert functions that operate on Tivoli Netcool/OMNIbus alerts. The file also imports functionality from several modules. The alert functions supplied in the `bmc_remedy.notification.js` file define a basic default behaviour for each of those functions. You can modify these alert functions to suit your environment.

The following sections discuss each of the imported modules and alert functions defined in the `bmc_remedy.notification.js` file.

Imported modules

The `bmc_remedy.notification.js` file uses the `require` function to load the following modules:

- Gateway log manager service -- Made available through the *logger* variable
- Simple OMNIbus Gateway (SOG) interface module -- Made available through the *sog* variable
- dateformat module -- Made available through the *dateformat* variable
- bmc_remedy module -- Made available through the *remedy* variable

The SOG interface module exposes the following method:

- `newrow` -- Returns an empty row that is ready to be populated.

The bmc_remedy module exposes the following method (through the `bmc_remedy.js` file):

- `getEntry` -- Returns the BMC Remedy ARS entry identified by the *requestid* from the configured form. For example:

```
var remedy = require("bmc_remedy");
.
.
.
var entry = remedy.getEntry("00000001232");
var field = entry.getField("fieldid");
```

The bmc_remedy module allows for calling back into BMC Remedy ARS to retrieve details of BMC Remedy ARS requests.

The addJournal function

The gateway calls the `addJournal` function to add a journal entry in the journal table for the alert specified in *alert*.

The `addJournal` function is defined as follows:

```
function addJournal(alert,message)
```

- *alert* -- Specifies the alert within Tivoli Netcool/OMNIbus for which you want to add a journal entry in the journal table.
- *message* -- Specifies a message to be concatenated after the following string:

  `Generated by Remedy Gateway:`

The update function

The gateway calls the `update` function whenever it receives a notification from the target system. In the case of BMC Remedy ARS this notification is the result of a ticket modification. The `update` function is defined as follows:

```
function update(alert,inputs)
```

- *alert* -- Specifies the alert within Tivoli Netcool/OMNIbus that is being updated.
- *inputs* -- Specifies the actual update from the target system (in this case, BMC Remedy ARS). This update is a name/value map of updated values. These names are in the target system namespace, which might not correspond to the names of columns in the Tivoli Netcool/OMNIbus alert, so values need to be converted from one namespace to the other. This requires creating an empty update row (by calling the `sog.newrow` method).

The following example shows a call to the `update` function:

```
function update(alert, inputs)
{
        values = sog.newrow();
        values.put("Severity", inputs.get("Severity"));
        alert.update(values);
}
```

The clear function

The gateway calls the `clear` function whenever the target ticket is deleted or cleared, in which case the gateway clears the Tivoli Netcool/OMNIbus alert. In the case of BMC Remedy ARS, the gateway performs the clear operation when the end user deletes the target BMC Remedy ARS ticket, as there is no open or closed status. The `clear` function is defined as follows:

```
function clear(alert,inputs)
```

- *alert* -- Specifies the alert within Tivoli Netcool/OMNIbus.
- *inputs* -- Specifies the actual values from the BMC Remedy ARS ticket.

The error function

The gateway calls the `error` function whenever an error occurs when the gateway tries to manipulate the corresponding ticket in the target system (in this case, BMC Remedy ARS). The `error` function is defined as follows:

```
function error(alert,inputs, message)
```

- *alert* -- Specifies the alert within Tivoli Netcool/OMNIbus.
- *inputs* -- Specifies the actual values from the BMC Remedy ARS ticket.
- *message* -- Specifies the error message text. This error message text can be added to a journal entry for the alert (by calling the `addJournal` method), for example, to identify the problem to the end user.

The setTargetId function

The gateway calls the `setTargetId` function whenever the alert is associated with a destination target system ticket. Typically, the gateway calls `setTargetId` after the user creates a BMC Remedy ARS ticket and a corresponding creation ticket alert is generated. The `setTargetId` function sets the target ID in the underlying alert and adds the target ID details to a journal entry for the alert (by calling the `addJournal` method).

The `setTargetId` function is defined as follows:

```
function setTargetId(alert,targetid)
```

- *alert* -- Specifies the alert within Tivoli Netcool/OMNIbus.
- *targetid* -- Specifies a string representation of the id. In the case of BMC Remedy ARS, this string is the request id of the form entry.

The logError function

The gateway calls the `logError` function to log errors associated with the alert specified in *alert*. The `logError` function actually calls the `error` method, which is implemented as part of the Tivoli Netcool/OMNIbus `Logger` interface. The gateway writes errors to the gateway log file.

The `logError` function is defined as follows:

```
function logError(alert, msg)
```

- *alert* -- Specifies the alert within Tivoli Netcool/OMNIbus to which the error message specified in *msg* applies.
- *msg* -- Specifies a message to be concatenated after the following string:

  `Alert [`*`ServerName`*`,`*`ServerSerial`*`]:`

Using the gateway log manager service to manage log messages

The gateway log manager service is a static class which enables log messages to be sent to the lower level gateway architecture. Any log messages of a level higher than that set by the **`MessageLevel`** property in the gateway's properties file will appear in the gateway's log file.

The following topics describe how to use the gateway log manager service.

Preparing to use the gateway log manager service

Typically, the gateway makes the gateway log manager service available through a JavaScript file. The name of the JavaScript file has the following format:

```
gatewayname.notification.js
```

Where: *gatewayname* specifies a string that maps to the name of the gateway.

The JavaScript file uses the `require` function to load the gateway log manager service and make it available through the *logger* variable, as follows:

```
.
.
.
var logger = require("logger");
.
.
.
```

The gateway log manager service exposes the following methods:

- `error` -- Logs an error message.
- `warning` -- Logs a warning message.
- `info` -- Logs an informational message.
- `debug` -- Logs a debug message.

Logging messages of type ERROR

Use the `error` method to log messages of type ERROR into the specified gateway log file.

Using the `error` method requires you to understand:

- The definition of the `error` method
- How to call the `error` method

Note : Use the **`MessageLevel`** property to specify the level of messages to receive for this gateway. Use the **`MessageLog`** property to specify the path, including the name of the log file, to which the `error` method writes messages of type ERROR for this gateway.

Understanding the definition of the error method

The `error` method is defined in the gateway log manager service.

The `error` method is defined as follows:

```
public void error( Object message );
```

Where:

- *message* specifies the message string to be logged.

The `error` method inserts the message specified in the *message* parameter into the log file specified in the **`MessageLog`** property for this gateway.

Upon completion, the `error` method returns no value.

Calling the error method

You can call the `error` method in any of the gateway's source files whenever you want to insert a log message of type ERROR into the log file specified in the **`MessageLog`** property for this gateway.

The following example shows a call to the `error` method:

```
.
.
.
logger.error("Target application unable to find the ticket. Please
             double-check if the ticket exists in the target
             application.");  1
.
.
.
```

1. Calls the `error` method, which writes the specified message to the log file specified in the **`MessageLog`** property for this gateway.

Logging messages of type WARNING

Use the `warning` method to log messages of type `WARNING` into the specified gateway log file.

Using the `warning` method requires you to understand:

- The definition of the `warning` method
- How to call the `warning` method

Note : Use the **`MessageLevel`** property to specify the level of messages to receive for this gateway. Use the **`MessageLog`** property to specify the path, including the name of the log file, to which the `warning` method writes messages of type `WARNING` for this gateway.

Understanding the definition of the warning method

The `warning` method is defined in the gateway log manager service.

The `warning` method is defined as follows:

```
public void warning( Object message );
```

Where:

- *message* specifies the message string to be logged.

The `warning` method inserts the message specified in the *message* parameter into the log file specified in the **`MessageLog`** property for this gateway.

Upon completion, the `warning` method returns no value.

Calling the warning method

You can call the `warning` method in any of the gateway's source files whenever you want to insert a log message of type `WARNING` into the log file specified in the **`MessageLog`** property for this gateway.

The following example shows a call to the `warning` method:

```
.
.
.
logger.warning("Unable to determine hostname");  1
.
.
.
```

1. Calls the `warning` method, which writes the specified message to the log file specified in the **`MessageLog`** property for this gateway.

Logging messages of type INFO

Use the `info` method to log messages of type `INFO` into the specified gateway log file.

Using the `info` method requires you to understand:

- The definition of the `info` method
- How to call the `info` method

Note : Use the **`MessageLevel`** property to specify the level of messages to receive for this gateway. Use the **`MessageLog`** property to specify the path, including the name of the log file, to which the `info` method writes messages of type `INFO` for this gateway.

Understanding the definition of the info method

The `info` method is defined in the gateway log manager service.

The `info` method is defined as follows:

```
public void info( Object message );
```

Where:

- *message* specifies the message string to be logged.

The `info` method inserts the message specified in the *message* parameter into the log file specified in the **`MessageLog`** property for this gateway.

Upon completion, the `info` method returns no value.

Calling the info method

You can call the `info` method in any of the gateway's source files whenever you want to insert a log message of type `INFO` into the log file specified in the **`MessageLog`** property for this gateway.

The following example shows a call to the `info` method:

```
.
.
.
logger.info("Ticket on the target system found:"); 1
.
.
.
```

1. Calls the `info` method, which writes the specified message to the log file specified in the **`MessageLog`** property for this gateway.

Logging messages of type DEBUG

Use the debug method to log messages of type `DEBUG` into the specified gateway log file.

Using the debug method requires you to understand:

- The definition of the debug method
- How to call the debug method

Note : Use the **`MessageLevel`** property to specify the level of messages to receive for this gateway. Use the **`MessageLog`** property to specify the path, including the name of the log file, to which the debug method writes messages of type `DEBUG` for this gateway.

Understanding the definition of the debug method

The debug method is defined in the gateway log manager service.

The debug method is defined as follows:

```
public void debug( Object message );
```

Where:

- *message* specifies the message string to be logged.

The debug method inserts the message specified in the *message* parameter into the log file specified in the **`MessageLog`** property for this gateway.

Upon completion, the debug method returns no value.

Calling the debug method

You can call the debug method in any of the gateway's source files whenever you want to insert a log message of type DEBUG into the log file specified in the **MessageLog** property for this gateway.

The following example shows a call to the debug method:

```
.
.
.
logger.debug("Form validated by target application"); 1
.
.
.
```

1. Calls the debug method, which writes the specified message to the log file specified in the **MessageLog** property for this gateway.

Table Replication

Use the table replication definition file to define which tables the gateway replicates and the types of events that it monitors in the ObjectServer.

Overview

The gateway creates and updates requests in BMC Remedy ARS when alerts are created and updated in the ObjectServer. The gateway populates and updates the fields in the request by replicating data from the ObjectServer. The gateway uses a table replication definition file called `bmc_remedy.rdrwtr.tblrep.def` to determine which ObjectServer tables to replicate. Specifically, the `bmc_remedy.rdrwtr.tblrep.def` file specifies REPLICATE command clauses that define how the gateway replicates data between an ObjectServer and a target database or application, in this case BMC Remedy ARS.

See the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* for descriptions of the syntax associated with the REPLICATE command clauses. See the *IBM Tivoli Netcool/OMNIbus Administration Guide* for a description of the ObjectServer SQL interface for defining and manipulating relational database objects such as tables and views. The `bmc_remedy.rdrwtr.tblrep.def` file makes use of the ObjectServer SQL interface. The *IBM Tivoli Netcool/OMNIbus Administration Guide* also describes the column names of the `alerts.status` and `alerts.journal` tables.

This topic describes the specific REPLICATE command and `FILTER WITH` clauses specified in the delivered `bmc_remedy.rdrwtr.tblrep.def` file. The topic also provides examples of other possible `FILTER WITH` clauses.

The information that follows describes the `bmc_remedy.rdrwtr.tblrep.def` file. See the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* for more information, including information on the structure of the table replication definition file, the REPLICATE command, and the available clauses.

The table replication definition file

The table replication definition file defines which tables the gateway replicates and the types of events that it monitors in the ObjectServer. The table replication definition file also identifies the mapping between data fields in the ObjectServer and those in a BMC Remedy ARS request.

The table replication definition file supplied with the gateway is named `bmc_remedy.rdrwtr.tblrep.def`, and resides in the `$OMNIHOME/gates/bmc_remedy` directory. Use the **`Gate.RdrWtr.TblReplicateDefFile`** property to specify the path to (including the name of) the table replication definition file.

The table replication definition file contains one or more REPLICATE command clauses and as supplied contains the following:

```
REPLICATE ALL FROM TABLE 'alerts.status' 1
    USING MAP 'StatusMap'; 2
```

```
REPLICATE ALL FROM TABLE 'alerts.journal' 3
    USING MAP 'JournalMap'; 4
  FILTER WITH "Text1 NOT LIKE 'Generated by Remedy Gateway*'"; 5

# REPLICATE ALL FROM TABLE 'alerts.details' 6
    #  USING MAP 'DetailsMap'  7
```

The following descriptions explain each of the numbered items in the supplied `bmc_remedy.rdrwtr.tblrep.def` file:

1. Specifies that the gateway replicates all insert, update, and delete operations from the source table (`alerts.status`) to the target table.
2. Specifies that the gateway uses the `StatusMap` defined in the `bmc_remedy.map` file to map Tivoli Netcool/OMNIbus alert fields to their corresponding BMC Remedy ARS request fields.

   Note : A separate data mapping file called `bmc_remedy.map` contains the definitions of the data mapping.

   See "Mapping" on page 21 for details of the data mapping definitions.
3. Specifies that the gateway replicates all insert, update, and delete operations from the source table (`alerts.journal`) to the target table.
4. Specifies that the gateway uses the `JournalMap` defined in the `bmc_remedy.map` file to map Tivoli Netcool/OMNIbus `alerts.journal` fields to their corresponding BMC Remedy ARS fields.
5. Specifies how the gateway should filter the rows that are selected for replication. The NOT LIKE operator instructs the gateway to perform a string comparison between the string `'Generated by Remedy Gateway*'` and the table column represented by the string `Text1`. The result is the gateway replicates all rows in the target table in which `Text1` does not contain the string `'Generated by Remedy Gateway*'`.

   This ensures that rows containing journal entries created by the gateway are not replicated.
6. Specifies that the gateway replicates all insert, update, and delete operations from the source table (`alerts.details`) to the target table.
7. Specifies that the gateway uses the `DetailsMap` defined in the `bmc_remedy.map` file to map Tivoli Netcool/OMNIbus `alerts.details` entries to their corresponding BMC Remedy ARS fields.

   Note : Uncomment the lines (lines that begin with the hash sign (#)) for these `REPLICATE` command clauses if you want alert details. If you do not remove the commented lines for these `REPLICATE` command clauses, the gateway ignores the `DetailsMap` specified in the `bmc_remedy.map` file.

   See "Mapping" on page 21 for more information.

FILTER WITH examples

The `bmc_remedy.rdrwtr.tblrep.def` file is supplied with the following `FILTER WITH` clause:

```
FILTER WITH "Text1 NOT LIKE 'Generated by Remedy Gateway*'";
```

The following examples show other possible `FILTER WITH` clauses. Create `FILTER WITH` clauses that are suitable for your environment. See the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* for information on the `FILTER WITH` clause.

Note : When considering which characteristics of alerts to use as a filter, use those fields from the `alerts.status` table whose values do not change over time. For example, `ServerName`, `Class`, `Manager`, and so forth.

```
FILTER WITH 'Acknowledged>0'; 1

FILTER WITH "Text1 NOT LIKE 'GW%'" 2

FILTER WITH 'Text1 NOT LIKE \'GW%\' ' 3

FILTER WITH 'Node != \'localhost\' 4
FILTER WITH 'ServerName IN (\'NCOMS_US\',\'NCOMS_CA\')' 5
```

The following descriptions explain each of the numbered items in the `FILTER WITH` clause examples. The gateway uses these `FILTER WITH` clauses to filter the database rows that are selected for replication as follows:

1. Uses the > (greater than) comparison operator to test the ObjectServer `alerts.status` table field called `Acknowledged` to determine if the value is greater than 0 (zero). The result is the gateway selects only those alerts that have been acknowledged (that is, `Acknowledged` is equal to the value 1).

   Note : The `Acknowledged` field is either 0 (Not Acknowledged) or 1 (Acknowledged).
2. Uses the NOT LIKE operator to instruct the gateway to perform a string match between the pattern `GW %` and the table column represented by the string `Text1`. (The `%` (percent) operator is used as a wildcard in matching operations.) The result is the gateway replicates all rows in the target table in which `Text1` does not contain the string `GW%`.
3. Uses the `!=` (not equal to) comparison operator to test the ObjectServer `alerts.status` table field called `Node` to determine if the value is not equal to `'localhost'`. The result is the gateway selects all `Node` alerts except those that originate from the local host.
4. Uses the `IN` list comparison operator to compare the ObjectServer `alerts.status` table field called `ServerName` to the list of values `NCOMS_US` and `NCOMS_CA`. The result is the gateway selects only those alerts that originate from the ObjectServers named `NCOMS_US` and `NCOMS_CA`.

   Note : This `FILTER WITH` clause shows that if a filter contains quotation marks (single or double), an escape (the \ (backslash)) character must be specified before each quotation mark to escape it.

Mapping

Mapping defines how the Java Gateway for BMC Remedy ARS maps fields in ObjectServer tables to fields in a BMC Remedy ARS request. The mapping is used during replication of ObjectServer to BMC Remedy ARS data.

The map definition file

The mapping of fields in ObjectServer tables to fields in a BMC Remedy ARS request is defined in the map definition file. The gateway is supplied with a map definition file named `bmc_remedy.map` in the directory `$OMNIHOME/gates/bmc_remedy`. You can modify this file to tailor the data mapping to suit your environment.

The map definition file contains a number of `CREATE MAPPING` commands each of which maps specific ObjectServer table fields to fields in a BMC Remedy ARS request.

The information that follows describes the `bmc_remedy.map` definition file. See the *IBM Tivoli Netcool/ OMNIbus Probe and Gateway Guide* for more information, including information on the structure of the map definition file, the `CREATE MAPPING` command, and the optional `CREATE MAPPING` command clauses.

DEDUPLICATION

Deduplication is a mechanism by which the gateway reduces the amount of redundant updates sent to BMC Remedy ARS.

The gateway computes a cryptographic hash (SHA1) based on fields selected for deduplication in the specified map, and stores the resulting hash in the history of the alert in the gateway cache. If subsequent data results in a hash already stored in the alert history, the gateway drops this data because it existed in the hash.

The gateway maintains the main alert values (values from `alerts.status`) as a current hash, with no other history maintained.

The gateway maintains alert journals and details (from `alerts.journal` and `alerts.details` respectively) as a history, with multiple hashes per alert. This allows the gateway to record multiple journals and details for each alert.

Note : Without a `DEDUPLICATE` clause, all fields in a map are considered for deduplication purposes. The `DEDUPLICATE` clause specifies the fields to be included in the row hash used for deduplication in the specified map. Conversely, the `DO NOT DEDUPLICATE` clause means ignore the listed fields when computing the row hash used for deduplication.

See "Explanation of the StatusMap" on page 23 for an example of how to use the `DEDUPLICATE` clause.

The CREATE MAPPING command

The `CREATE MAPPING` command creates a map that defines how the gateway maps fields in ObjectServer tables to fields in a BMC Remedy ARS request. The `CREATE MAPPING` command has the following syntax:

```
CREATE MAPPING mappingname
(
'ArsFieldId' = 'value' [ ON INSERT ONLY ] [ CONVERT TO type ]
[ , 'ARSFieldId' = 'value' [ ON INSERT ONLY ] [ CONVERT TO type ] ]...
) ;
```

where:

- *`mappingname`* specifies the name of the map to be created for a specific mapping of ObjectServer table fields to fields in a BMC Remedy ARS request.
- `'`*`ArsFieldId`*`'` specifies the integer field ID for the destination field in the BMC Remedy ARS request.

  Note : Non-integer field IDs, such as `Chrono` and `UID` in the journal map, are used to include the corresponding ObjectServer alerts fields in the deduplication. ObjectServer alerts fields that do not resolve to BMC Remedy ARS field IDs are dropped by the gateway.
- *`value`* specifies the name of a field in the ObjectServer table.

Note : It is assumed that you are familiar with the fields in the ObjectServer tables and the corresponding fields in a BMC Remedy ARS request.

At a minimum, the gateway must pass across the `@Serial` and `@Severity` fields to BMC Remedy ARS in order for the replication to take place.

The optional `ON INSERT ONLY` controls the updating of the BMC Remedy ARS field during the life of its associated ObjectServer alert; when omitted, the BMC Remedy ARS field is updated for any change in the state of the alert. When included, the BMC Remedy ARS field is only set when the alert is created.

The optional `CONVERT TO` *`type`* allows 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` to force the source field to be converted to an integer, string, or date type.

Default maps supplied with the map definition file

The `bmc_remedy.map` definition file delivered with the gateway contains the following default maps:

- `StatusMap` - Is the map for `alerts.status` entries, and maps Tivoli Netcool/OMNIbus alert fields to their corresponding BMC Remedy ARS request fields.
- `JournalMap` - Contains the mapping from `alerts.journal` journal entries to BMC Remedy ARS diary fields. By default, only a single diary field is set, which appends the journal text to a BMC Remedy ARS diary field. The `JournalMap` also contains meta-data fields for the journal entry (user and timestamp of the journal entry in Tivoli Netcool/OMNIbus) for the purpose of deduplication.
- `DetailsMap` - Contains the mapping of alert details from the fields in the `alerts.details` table to BMC Remedy ARS request fields. By default, adds an entry to the same BMC Remedy ARS diary field as journals, with an entry per name/value pair in the details for an alert.

Note : The mapping file (`bmc_remedy.map`) defines the maps, but does not determine whether the maps are used. You specify which maps to use in the table replication definition file (`bmc_remedy.rdrwtr.tblrep.def`).

See "Table Replication" on page 19 for more information.

Explanation of the StatusMap

The `StatusMap` is one of the maps contained in the `bmc_remedy.map` delivered with the gateway. As discussed previously, the `StatusMap` is the main map for `alerts.status` entries, and maps Tivoli Netcool/OMNIbus alert fields to their corresponding BMC Remedy ARS request fields. The `StatusMap` is defined as follows:

```
CREATE MAPPING StatusMap  1
(
        '536870914' = '@Identifier'       ON INSERT ONLY, 2
        '536870913' = '@Serial'           ON INSERT ONLY,
        '536870944' = '@Node'             ON INSERT ONLY,
        '536870915' = '@NodeAlias'        ON INSERT ONLY,
        '536870932' = '@Manager'          ON INSERT ONLY,
        '536870939' = '@Agent'            ON INSERT ONLY,
        '536870918' = '@AlertGroup'       ON INSERT ONLY,
        '536870931' = '@AlertKey'         ON INSERT ONLY,
                '7' = '@Severity',  3
        '536870916' = '@Summary'          ON INSERT ONLY,
        '536870946' = '@StateChange',  3
        '536870912' = '@FirstOccurrence'  ON INSERT ONLY,
        '536870917' = '@LastOccurrence',
        '536870935' = '@InternalLast',  3
        '536870938' = '@Poll'             ON INSERT ONLY,
        '536870941' = '@Type'             ON INSERT ONLY,
        '536870940' = '@Tally',  3
        '536870945' = '@Class'            ON INSERT ONLY,
        '536870942' = '@Grade'            ON INSERT ONLY,
        '536870934' = '@Location'         ON INSERT ONLY,
        '536870933' = '@Acknowledged',  3
        '536870922' = '@ServerSerial'     ON INSERT ONLY,
        '536870923' = '@ServerName'       ON INSERT ONLY
) DEDUPLICATE ('7','536870917','536870940','536870933'); 4
```

The following list describes each of the numbered items in the `StatusMap`:

1. Uses the `CREATE MAPPING` command to create a map called `StatusMap`. As described previously, the `StatusMap` is the main map for alerts.status entries, and maps Tivoli Netcool/OMNIbus alert fields to their corresponding BMC Remedy ARS fields.
2. Maps the BMC Remedy ARS field identifier (the string value `536870914`) to the ObjectServer table field called `@Identifier`. The BMC Remedy ARS field id is defined for the form in the provided `omnibus.def` file.

   The string value on the left side of the expression in each mapping clause is the BMC Remedy ARS field identifier. Most fields are written to specially defined records within the BMC Remedy ARS alert form, with the exception of `Severity` (written to the ARS core field `Status`) and `Summary` (written to the ARS core field `Summary Description`).

   The string value on the right side of the expression in each mapping clause refers to a field in the ObjectServer table. The string values on the right side of the expressions use the format `@fieldname`. For example, in this mapping clause `@Identifier` is used.

   The use of `ON INSERT ONLY` instructs the gateway to set the `@Identifier` field only when the ObjectServer creates the corresponding alert.
3. These mapping clauses define fields that do not use `ON INSERT ONLY`. In this case, because `ON INSERT ONLY` is omitted, the gateway updates these fields during the life of their associated ObjectServer alerts.
4. The `DEDUPLICATE` clause specifies the fields to be included in the row hash used for deduplication. Conversely, the `DO NOT DEDUPLICATE` clause means ignore the listed fields when computing the row hash used for deduplication.

The StatusMap shows that the following fields are included in the row hash used for deduplication:

- BMC Remedy ARS field '7' ('@Severity')
- BMC Remedy ARS field '536870917' ('@LastOccurrence')
- BMC Remedy ARS field '536870940' ('@Tally')
- BMC Remedy ARS field '536870933' ('@Acknowledged')

Should any of these fields change, the resulting hash will also change, and the gateway knows the update should be forwarded. If the hash remains the same, then the update has not changed any of these fields, and the update will be dropped.

Explanation of the JournalMap

The JournalMap is one of the maps contained in the bmc_remedy.map delivered with the gateway. As discussed previously, the JournalMap contains the mapping from alerts.journal journal entries to BMC Remedy ARS diary fields. The JournalMap is defined as follows:

```
CREATE MAPPING JournalMap  1
(
        'UID'           = JOURNAL.UID INTERNAL ONLY, 2
        'Chrono'        = JOURNAL.CHRONO INTERNAL ONLY, 2
        '536870921'     = JOURNAL.TEXT  3

);
```

The following list describes each of the numbered items in the JournalMap:

1. Uses the CREATE MAPPING command to create a map called JournalMap. As described previously, the JournalMap contains fields that are included for deduplication purposes.
2. Fields marked INTERNAL ONLY are not sent to BMC Remedy ARS, but included for deduplication. If not included, journal entries with the same text, but made at different times, will be deduplicated and the latter entries dropped.
3. Maps the BMC Remedy ARS field identifier 536870921 to the ObjectServer alerts.journal table fields Text1...Text16.

Note : Without a DEDUPLICATE clause, all fields in a map are considered for deduplication purposes. The DEDUPLICATE clause specifies the fields to be included in the row hash used for deduplication in the specified map. Conversely, the DO NOT DEDUPLICATE clause means ignore the listed fields when computing the row hash used for deduplication.

Explanation of the DetailsMap

The DetailsMap is one of the maps contained in the bmc_remedy.map definition file delivered with the gateway. As discussed previously, the DetailsMap contains the mapping of alert details from the fields in the alerts.details table to BMC Remedy ARS request fields. The DetailsMap is defined as follows:

```
CREATE MAPPING DetailsMap  1
(
        '536870921'     = '@Name' + ':' + '@Detail'  2

);
```

The following list describes each of the numbered items in the DetailsMap:

1. Uses the CREATE MAPPING command to create a map called DetailsMap.
2. Maps the BMC Remedy ARS field identifier (the integer value 536870921) to the ObjectServer table fields called @Name and @Detail. The BMC Remedy ARS field id is defined for the form in the provided omnibus.def file.

Note : You must uncomment the REPLICATE command clauses for the DetailsMap specified in the bmc_remedy.rdrwtr.tblrep.def file. Otherwise, the gateway ignores the DetailsMap.

See "Table Replication" on page 19 for more information.

Note : Without a DEDUPLICATE clause, all fields in a map are considered for deduplication purposes. The DEDUPLICATE clause specifies the fields to be included in the row hash used for deduplication in the specified map. Conversely, the DO NOT DEDUPLICATE clause means ignore the listed fields when computing the row hash used for deduplication.

Specifying alternative values in the map definition file

When specifying alternative values for fields in the bmc_remedy.map file, ensure that the field does not exceed the maximum length of the target field in the BMC Remedy ARS request.

Controlling the types of data to forward in the map definition file

You can control which types of data that the gateway forwards to the BMC Remedy ARS server. For example, you might want to only create a request in BMC Remedy ARS and ignore further update and delete field activity related to that request. You can accomplish this by using the DEDUPLICATE clause. The following example shows how to specify the fields in the StatusMap to suppress all update and delete field activity after the gateway creates the request:

```
CREATE MAPPING StatusMap
(
.
.
.
'536870922' = '@ServerSerial',
'536870923' = '@ServerName'
) DEDUPLICATE ('536870922','536870923')
```

In this example, because all update and delete events will have matching values for @ServerSerial and @ServerName, they will be deduplicated with the original alert. This ensures that the gateway creates a request in BMC Remedy ARS, and all subsequent events will be dropped.

Historic journal forwarding

You can configure how the gateway handles historic journal entries for an ObjectServer alert by setting the **Gate.HistoricResync** property.

By default, the gateway forwards historic journals and details (those created in the ObjectServer before the alert is sent for request creation) when the alert is created. Setting the **Gate.HistoricResync** property to false prevents the gateway from sending historic journals and details to BMC Remedy ARS. Note, however, that the gateway sends any subsequent journals and details to BMC Remedy ARS after the ticket is created.

Store and forward capability

The Java Gateway for BMC Remedy ARS provides a store and forward capability to minimize any data loss should the gateway lose communication with BMC Remedy ARS. The store and forward capability is permanently enabled.

Operation of store and forward

The gateway creates a batch file for the following operations (depending on what is configured in the bmc_remedy.rdrwtr.tblrep.def file):

- Alert (alerts.status table) insert, update, or delete
- Alert journals (alerts.journal table) insert
- Alert details (alerts.details table) insert

When the gateway reestablishes communication with BMC Remedy ARS, it sends the data that it has stored in the batch files.

Startup behavior

When the gateway starts, it attempts to log in to BMC Remedy ARS. Once logged in the gateway determines if there are any store-and-forward batch files from a previous run and, if present, processes these batch files.

If the gateway cannot log in to BMC Remedy ARS for any reason, it shuts down. The gateway does not attempt to gather data from the ObjectServer and store it in store-and-forward batch files.

FIPS mode and encryption

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

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

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

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

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

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

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

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

AES encryption

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

Note : AES encryption is supported on all supported versions of Tivoli Netcool/OMNIbus on all UNIX and Linux operating systems.

nco_aes_crypt

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

```
@datalength:encrypted_data@
```

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

The encrypted values appear in single quotes in the properties file. The following example shows the server password in encrypted format for the **Gate.Remedy.Password** property in the G_BMC_REMEDY.props file:

```
# Remedy gateway specific properties
.
.
.
Gate.Remedy.Password  :
'@64:lHBLuIPLNye8zCWhykFVFY7y90V9kCjGK5GSWu5VBdSlgQOqarq6T4UK4xk5Vqix@'
.
.
.
```

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

Using the nco_aes_crypt tool

Property values in the properties file must be encrypted using the nco_aes_crypt tool.

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

```
nco_aes_crypt [-d] [-o outfile] [-c cipher] -k keyfile -f filename
nco_aes_crypt [-d] [-o outfile] [-c cipher] -k keyfile data
```

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

The following table describes the options available with nco_aes_crypt:

Table 6. nco_aes_crypt command line options

Command line option	**Description**
-d	Use this option to specify the mode in which the nco_aes_crypt tool runs: d - decrypt mode The default is encrypt mode.
-o *string*	Use this option to specify the output file to which the encrypted or decrypted data will be written.
-c *string*	Use this option to specify the cipher to use: • AES - Specifies the non-FIPS cipher. • AES_FIPS - Specifies the FIPS cipher. The default is AES (non-FIPS).
-k *string*	Use this option to specify the path of the file containing the key data. This option is mandatory.
-f *string*	Use this option to specify the path of the file containing data requiring encryption or decryption.
data	Use this option to specify the data to be encrypted or decrypted.

Encryption key file

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

The format of the key file is as follows:

```
key_length:key_data
```

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

For example:

```
128:1234567812345678
```

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

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

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

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

Table 7. Encrytion key file command line options

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

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

Using encrypted data

To use encrypted data, you set the **`ConfigKeyFile`** property in the `G_BMC_REMEDY.props` file to the path of the file that contains the encryption key. For example:

```
# Generic Omnibus Properties
#
ConfigKeyFile : 'key_file_path'
.
.
.
```

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

Running the ObjectServer in a secure mode

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

To enable the encryption, the location of the key file must be specified using the **`ConfigKeyFile`** property in the `G_BMC_REMEDY.props` file, as described previously. You also need to specify the encrypted username and password required for authentication using the **`Gate.RdrWtr.Username`** and **`Gate.RdrWtr.Password`** properties in the `G_BMC_REMEDY.props` file.

The following example shows the three fields that need to be specified in the `G_BMC_REMEDY.props` file when the ObjectServer runs in a secure mode:

```
# Generic Omnibus Properties
#
ConfigKeyFile  : '/HOME/74/solaris/omnibus/keyflie_name'
.
.
.
# Gateway Framework properties
.
.
.
Gate.RdrWtr.Password   : '@44:mdyEb8VTh+2wALnNlR7dnGnxRZ3BkMOQbR5IgxLlHuc=@'
.
.
.
Gate.RdrWtr.Username  : '@44:2yXgd6fp9q1Ey4sSAb2RibzA3+PpCZmhAZXo6nNdkvQ=@'
```

BMC Remedy ARS encryption

Use this information to learn how the Java Gateway for BMC Remedy ARS interacts with the standard security encryption that is built into the BMC Remedy ARS 8.1 API.

The BMC Remedy ARS 8.1 API comes with a standard security encryption that provides a 56-bit Data Encryption Standard (DES) using Cipher Block Chaining (CBC) mode. The algorithm uses a 512-bit RSA modulus to exchange keys and MD5 MAC to authenticate messages.

Note : You do not purchase or install the standard security encryption separately. By default, the BMC Remedy ARS 8.1 API standard security encryption is disabled. See the BMC Remedy ARS documentation for instructions on how to enable on the BMC Remedy ARS server the standard security encryption that comes with the BMC Remedy ARS 8.1 API.

BMC Remedy ARS also provides the following optional encryption products that must be purchased and installed separately from the BMC Remedy ARS 8.1 API. See the BMC Remedy ARS documentation for descriptions of these products and for instructions on how to install:

- BMC Remedy Encryption Performance Security (BMC Remedy Encryption Performance)
- BMC Remedy Encryption Premium Security (BMC Remedy Encryption Premium)

The following are some configuration tasks that you need to perform on both the BMC Remedy ARS server and the gateway server to configure the standard security encryption that is built into the BMC Remedy ARS 8.1 API.

Configuring the standard security encryption on the BMC Remedy ARS server

To configure the standard security encryption on the BMC Remedy ARS server, follow the steps in the BMC Remedy ARS documentation.

Configuring the standard security encryption on the gateway server

To configure the standard security encryption on the gateway server, follow these steps:

Note : The gateway requires the Bouncy Castle encryption library to communicate with the standard security encryption on the BMC Remedy ARS server. The Bouncy Castle encryption library jar file for the BMC Remedy ARS 8.1 API is called `bcprov-jdk15-145.jar` and it resides on the BMC Remedy ARS server. You add the Bouncy Castle encryption library jar file as an additional security provider to the Java Runtime Environment (JRE) by editing the `java.security` file.

1. Copy the Bouncy Castle encryption library jar file for the BMC Remedy ARS 8.1 API, `bcprov-jdk15-145.jar`, from the directory on the BMC Remedy ARS server to the *$JAVA_HOME*`/lib/jre/ext` directory on the gateway server.

   Where: *$JAVA_HOME* specifies an environment variable that defines the path to the JRE. You must ensure that this path picks up the JRE (`/lib/jre/ext`) that the gateway uses.
2. Go to the *$JAVA_HOME*`/lib/security` directory on the gateway server.
3. Make a backup copy of the `java.security` file before modifying it.
4. Open the `java.security` file for editing with a text editor of your choice and do the following:
   a. Locate the following property in the `java.security` file:

   ```
   security.provider.#=
   ```

   Where: # specifies a number (for example, 11) that identifies the security provider, in this case Bouncy Castle.

   Note : If Bouncy Castle is not listed as a security provider, you can add it by specifying a new entry after the list of existing security providers.

   b. Add the `BouncyCastleProvider` security provider to the property as follows (making sure to replace the value 11 with the number that identifies the Bouncy Castle security provider in your environment):

   ```
   security.provider.11=org.bouncycastle.jce.provider.BouncyCastleProvider
   ```
5. When finished, save and close the `java.security` file.

   Note : If you do not correctly configure the standard security encryption on the gateway server, the gateway will fail to connect to the BMC Remedy ARS server and it will print an `ERROR (9006)` message in the logfile.

Gateway operation

Use this information to learn how the Java Gateway for BMC Remedy ARS interacts with the BMC Remedy Action Request System. This includes information on how the gateway interacts with BMC Remedy ARS with regard to request operations.

This section covers the following subject areas:

- "Creating, updating, and deleting requests in BMC Remedy ARS" on page 30
- "Avoiding duplicate requests in BMC Remedy ARS" on page 31
- "Forwarding Journals" on page 31
- "Retrieving events" on page 32
- "Controlling how the gateway handles notifications" on page 32

Creating, updating, and deleting requests in BMC Remedy ARS

When a new alert is created in the ObjectServer, the gateway creates a corresponding request in BMC Remedy ARS. The mapping file defines how the fields in the BMC Remedy ARS request are filled with data from the ObjectServer. "Mapping" on page 21 contains more information on the mapping file and its content.

When an alert in the ObjectServer is updated, the gateway updates the corresponding BMC Remedy ARS request. The gateway updates fields in the request depending on their definition in the mapping file.

Fields with map entries qualified with `ON INSERT ONLY` are set only when the gateway creates the original request. Those fields are not updated when the ObjectServer alert is updated.

You set the action of the gateway when an alert is deleted from the ObjectServer through the value of the **`Gate.Remedy.DeleteEntries`** property. When the value of the property is `true`, the gateway deletes the request in BMC Remedy ARS. When the value of the property is `false`, the gateway sets the status of the request to `Clear`.

Avoiding duplicate requests in BMC Remedy ARS

As discussed, when a new alert is created in the ObjectServer, the gateway creates a corresponding request in BMC Remedy ARS. When the request creation operation succeeds BMC Remedy ARS assigns a unique request ID. However, there are situations when the gateway creates duplicate requests in BMC Remedy ARS, that is, requests with identical field values, but different request IDs. These duplicate requests result from the fact that the gateway makes two calls to the create request function in BMC Remedy ARS with the same set of data, when in fact, the gateway should have created only one request.

The following describes an example scenario where the gateway calls the create request function twice, thus causing duplicate requests with the same set of data to be created:

- A new alert is created in the ObjectServer.
- The gateway creates a corresponding request in BMC Remedy ARS by calling the create request function in BMC Remedy ARS.
- BMC Remedy ARS creates a request ID.
- The gateway crashes before it can receive the request ID from BMC Remedy ARS. (Receiving the request ID from BMC Remedy ARS indicates that the create request operation succeeded.)
- After the gateway restarts, the ObjectServer sends the gateway an update request on the alert that was created prior to the crash. The ObjectServer indicates that no request was created for this alert.
- The gateway creates a corresponding request in BMC Remedy ARS by calling the create request function in BMC Remedy ARS. This request has the same data as specified for the previous request. This action results in the creation of duplicate requests because the create request that occurred prior to the crash was successful. Thus, there are two requests in BMC Remedy ARS with the same data and different IDs.

Use the **`Gate.Remedy.ReclaimARS`** and **`Gate.Remedy.CreationErrorCode`** properties to control how the gateway handles request create and update operations to avoid duplicate requests in BMC Remedy ARS. The **`Gate.Remedy.ReclaimARS`** property allows you to specify whether and how to reclaim a request ID from BMC Remedy ARS when creating and updating an alert. The property takes one of the following values:

`0`: Never query BMC Remedy ARS for a request ID.

`1`: Query BMC Remedy ARS for a request ID when an error occurs during the creation of the BMC Remedy ARS request.

`2`: Always query BMC Remedy ARS for a request ID.

The **`Gate.Remedy.CreationErrorCode`** property sets the error code used to indicate failure to create a request in BMC Remedy ARS. If the gateway cannot create a request, it stores this value in the `TargetIdField` of the corresponding ObjectServer alert. The default value for the error code is `CREATION_FAILURE`.

Note : Use the **`Gate.TargetIdField`** property to specify the `alerts.status` column to use for target identifier tracking (that is, the identifier in BMC Remedy ARS). The default is `'Location'`

Forwarding Journals

Once the request exists in BMC Remedy ARS, updates to the journal data are handled by the table replication mechanism, in a similar way to updates in the fields of the request itself.

"Table Replication" on page 19 contains more information about table replication between the ObjectServer and BMC Remedy ARS.

Retrieving events

`Alert Events` is a standard form in BMC Remedy ARS that allows the gateway to retrieve update notifications from the BMC Remedy ARS server. For example, whenever a user updates the severity of a request in BMC Remedy ARS, a notification will be created in the `Alert Events` form to indicate the specific modification that has been made. The contents of this notification is specified in the filters `Omnibus-Close` and `Omnibus-Modify` (as defined in the `omnibus.def` configuration file). The gateway regularly queries BMC Remedy ARS for new notifications. The gateway uses the **`Gate.Remedy.UpdateForm`** property to identify the BMC Remedy ARS form (which is always `Alert Events`) to use to retrieve update notifications from the BMC Remedy ARS server.

The gateway uses the following properties in event retrieval operations:

- **`Gate.Remedy.NotificationQuery`**

  The gateway uses this property to identify the set of updates on which to report. The default event qualifier is `'7'<\"X\" AND '2'=\"USER\" AND '701'=\"FORM\"'`.

  The default qualifier has three parts, as follows:

  – `'7'<\"X\"` - Specifies that the gateway has not yet read the notification.

  – `'2'=\"USER\"` - Specifies that the notification results from a modification of a request that was originally created by the user identified by `USER`. This qualifier ensures that notifications come only from requests that the gateway creates. The gateway automatically substitutes `USER` with the value specified for the **`Gate.Remedy.Username`** property.

  – `'701'=\"FORM\"` - Specifies that the notification is the result of an action on the form identified by `FORM`. The gateway automatically substitutes `FORM` with the value specified for the **`Gate.Remedy.Form`** property.

- **`Gate.Remedy.UpdateQueryInterval`**

  The gateway uses this property to determine how often (in seconds) to query BMC Remedy ARS for updates. The default query interval is 3 seconds.

- **`Gate.Remedy.DeleteUpdates`**

  The gateway uses this property to delete a notification from the `Alert Events` form. If set to `true`, the gateway deletes a notification from the `Alert Events` form after it reads the notification. If set to `false`, the gateway leaves the notification in the `Alert Events` form even after it reads the notification and marks it as read. The default value is `false`.

Controlling how the gateway handles notifications

The gateway typically discards notifications that are generated by its own actions. For example, if the gateway updates a request to `Critical`, it discards the notification for this request. The gateway identifies notifications that result from its own actions by checking the user information contained in the notification. If the user in the notification matches the value of the **`Gate.Remedy.Username`** property, then the gateway generally discards it.

You can use the **`Gate.Remedy.OwnNotifications`** property to control how the gateway handles notifications that are generated by its own actions. If the **`Gate.Remedy.OwnNotifications`** property is set to `FIRST` (the default), the gateway processes the first notification associated with a BMC Remedy ARS request, even if the gateway user modifies the request. To instruct the gateway to process all notifications associated with a BMC Remedy ARS request, regardless of which user performs the update action on that BMC Remedy ARS request, set this property to `ALL`. To instruct the gateway to process no notifications associated with this BMC Remedy ARS request even if the user performs an update action on that BMC Remedy ARS request, set this property to `NONE`.

Resynchronization

The gateway can perform two types of resynchronization: unidirectional and bidirectional. The **`Gate.Cache.ResyncMode`** property provides three resynchronization modes to control how the gateway performs resynchronization operations.

Note : The gateway resynchronizes the alerts (`alerts.status` table) as well as journals and details (`alerts.journal` and `alerts.details` tables) if enabled in the table replication definition file.

The following resynchronization modes are available:

1. Unidirectional (UNI)

   The gateway performs a resynchronization operation from Tivoli Netcool/OMNIbus to BMC Remedy ARS.
2. Bidirectional (BI)

   The gateway operates in bidirectional resynchronization mode. In this mode the gateway retrieves open requests from BMC Remedy ARS and compares them with the open requests on the ObjectServer.
3. Automatic (AUTO)

   This is the default resynchronization mode. In this mode, the gateway operates in unidirectional resynchronization mode by default. However, if the gateway cache is empty on startup (which normally only occurs the first time that the gateway is run), it causes the gateway to operate in bidirectional resynchronization mode.

Resynchronization and Deduplication

The gateway deduplicates updates to alerts and journals. This results in efficient resynchronization, because only alerts that have changed are forwarded to BMC Remedy ARS. It also means that updates to fields that are not included in the map definition file, or fields that are not required in BMC Remedy ARS are ignored.

See "Explanation of the StatusMap" on page 23 for an example of the `DEDUPLICATE` command used in the `StatusMap` (one of the maps contained in the `bmc_remedy.map` delivered with the gateway).

Suppressing Deletes on and after Resynchronization

Use this information to learn how gateway framework properties **`Gate.Cache.ResyncInitialDeletes`** and **`Gate.Cache.ResynceSuppressDeletes`** interact with each other.

ResyncSuppressDeletes and ResynceSuppressInitialDeletes

When migrating to the gateway, previous instances of the gateway have not closed tickets, which is denoted by `Severity = 0`. When this occurs, the first resynchronization results in a large number of simulated deletes.

When you bring up the gateway after a long period of downtime, it may result in an alert churn requiring several deletes.

This is controlled with the addition of two new properties to the underlying gateway framework. **`Gate.Cache.ResyncSuppressDeletes`** and **`Gate.Cache.ResyncSuppressInitialDeletes`**.

The default behavior for **`Gate.Cache.ResyncSuppressDeletes`** and **`Gate.Cache.ResyncSuppressInitialDeletes`** is FALSE.

When **`Gate.Cache.ResyncSuppressDeletes`** is TRUE and **`Gate.Cache.ResyncSuppressInitialDeletes`** is TRUE all deletes are suppressed during resynchronization.

When **`Gate.Cache.ResyncSuppressDeletes`**is FALSEand **`Gate.Cache.ResyncSuppressInitialDeletes`**is TRUE deletes are only suppressed on initial resynchronization.

When **`Gate.Cache.ResyncSuppressDeletes`** is TRUE and **`Gate.ResyncSuppressInititalDeletes`** is FALSE deletes are only suppressed after initial resynchronization.

Running the gateway

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

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

1. Edit the `omnibus.def` file (which contains the schema) as follows:
   a. Change the references of `'__USER__'` to the actual user required by your BMC Remedy ARS configuration.
   b. Change the references of `'__VERSION__'` to the version of Tivoli Netcool/OMNIbus you are running.

   Note : Ensure to coordinate the previous edits with these gateway properties: **`Gate.Remedy.Username`** and **`Gate.Remedy.Form`**.
2. Import the `omnibus.def` file to the BMC Remedy ARS server.
3. Create the `conversions.remedy` table in the ObjectServer by running the `createConversionsTable.sql` file.

   The following example shows how to run the `createConversionsTable.sql`file on UNIX:

   ```
   $OMNIHOME/omnibus/bin/nco_sql -user username -password password
   -server servername < directory_path createConversions.sql
   ```

   The following example shows how to run the `createConversionsTable.sql`file on WINDOWS:

   ```
   "%OMNIHOME%\omnibus\bin\isql" -U username -P password
   -S servername -i directory_path createConversions.sql
   ```
4. Edit the `G_BMC_REMEDY.props` property file and then copy it under the `$OMNIHOME/etc` directory. The following properties must be updated:
   - **`Gate.Remedy.Server`** - Specify the IP address or hostname of the BMC Remedy ARS server.
   - **`Gate.Remedy.Username`** - Specify the username of the gateway user on the BMC Remedy ARS server.
   - **`Gate.Remedy.Password`** - Specify the password of the gateway user on the BMC Remedy ARS server.
   - **`Gate.RdrWtr.Server`** - Specify the name of the ObjectServer the gateway is running against.
   - **`Gate.Remedy.Form`** - Specify the name of the BMC Remedy ARS form used to create requests from ObjectServer alerts. Set this property to the name that you defined for the BMC Remedy ARS form in the `omnibus.def` file.

To run the gateway on UNIX operating systems, use the following command:

`$OMNIHOME/bin/nco_g_bmc_remedy`

To run the gateway on Windows operating systems, use the following commands:

```
%NCHOME%\platform\win32\jre_1.x.y\jre\bin\java.exe
-Dlog4j.configuration=file:\\\%OMNIHOME%\gates\bmc_remedy\log4j.properties
-jar %OMNIHOME%\java\jars\nco_g_java.jar -type bmc_remedy
```

Note : With no command line arguments, the gateway expects to find its properties file `G_BMC_REMEDY.props` located in the `$OMNIHOME/etc` directory.

Note : For additional information on running gateways and for running a second instance of a gateway on the same host, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Properties and command line options

The following tables describe the properties and command line options that are applicable to the Java Gateway for BMC Remedy ARS. They cover the properties specific to the gateway and those provided by the gateway's operational environment. The descriptions make reference to the BMC Remedy Action Request System.

Table 8 on page 35 lists and describes the properties specific to the Java Gateway for BMC Remedy ARS .

Table 8. Gateway specific properties

Property name	Command Line option	Description
`Gate.Remedy.ClearTicketId` *boolean*	`-remedyclearticketid` *boolean*	Use this property to specify whether the gateway clears the field defined by the **`Gate.TargetIdField`** property when the request is closed in BMC Remedy ARS. For example, an alert results in a request. The issue is resolved and the request is cleared. If the problem occurs again, and if you want a new request for the reoccurrence instead of reopening the existing request, setting this property to `true` will clear the association between the ObjectServer alert and the BMC Remedy ARS request when the request is cleared. The property takes one of the following values: • `true`: Instructs the gateway to clear the field defined by the **`Gate.TargetIdField`** property. • `false`: Instructs the gateway to not clear the field defined by the **`Gate.TargetIdField`** property. The default value is `false`.
`Gate.Remedy.Connections` *integer*	`-remedyconnections` *integer*	Use this property to set the number of simultaneous connections the gateway makes with BMC Remedy ARS. The default value is 2.

Table 8. Gateway specific properties (continued)

Property name	**Command Line option**	**Description**
`Gate.Remedy. CreationError Code` *string*	`remedycreationerrorcode` *string*	Use this property to set the error code used to indicate failure to create a request in BMC Remedy ARS. If the gateway cannot create a request, it stores this value in the `TargetIdField` of the corresponding ObjectServer alert. The default value is `'CREATION_FAILURE'`.
`Gate.Remedy. DeleteEntries` *boolean*	`-remedydeleteentries` *boolean*	Use this property to specify whether the gateway deletes BMC Remedy ARS requests when the corresponding alerts are deleted in the ObjectServer. The property takes one of the following values: • `true`: Instructs the gateway to delete BMC Remedy ARS requests when the corresponding alerts are deleted in the ObjectServer. • `false`: Instructs the gateway to not delete BMC Remedy ARS requests when the corresponding alerts are deleted in the ObjectServer. The default value is `false`.
`Gate.Remedy. DeleteUpdates` *boolean*	`-remedydeleteupdates` *boolean*	Use this property to specify whether the gateway deletes BMC Remedy ARS update notifications from the "Alert Events" form after processing the update notification. The property takes one of the following values: • `true`: Instructs the gateway to delete BMC Remedy ARS update notifications from the "Alert Events" form. • `false`: Instructs the gateway to simply mark the BMC Remedy ARS notifications as being read. The default value is `false`.

Table 8. Gateway specific properties (continued)

Property name	Command Line option	Description
`Gate.Remedy.Form` *string*	`-remedyform` *string*	Use this property to specify the name of the BMC Remedy ARS form used to create requests from ObjectServer alerts. Set this property to the name that you defined for the BMC Remedy ARS form in the `omnibus.def` file. The default value is `''`.
`Gate.Remedy.HopefulClose` *boolean*	`-remedyhopefulclose` *boolean*	Use this property to specify whether the gateway should reclaim unknown request IDs from BMC Remedy ARS when closing an alert. The property takes one of the following values: • `true`: Instructs the gateway to reclaim unknown request IDs from BMC Remedy ARS when closing an alert. • `false`: Instructs the gateway to not reclaim unknown request IDs from BMC Remedy ARS when closing an alert. The default value is `false`.
`Gate.Remedy.MaxElementsRetrieve` *integer*	`-remedymaxelementsretrieve` *integer*	Use this property to specify the maximum number of elements to retrieve when querying BMC Remedy ARS for updates. The default value is `'100'` elements (that is, 100 updates from the form defined in the **`Gate.Remedy. UpdateForm`** property).
`Gate.Remedy.NotificationQuery` *string*	`-remedynotificationquery` *string*	Use this property to define the query that the gateway uses to select entries in BMC Remedy ARS. The default value is `''7'<\"X \" AND '2'=\"USER\" AND '701'=\"FORM\"'`.

Table 8. Gateway specific properties (continued)

Property name	Command Line option	Description
`Gate.Remedy.Own Notifications` *string*	`-remedyownnotifications` *string*	Use this property to specify whether the gateway should process its own notifications. Specify one of the following values: `FIRST`: The gateway processes the first notification associated with a BMC Remedy ARS request, even if it is the gateway user modifying the request. `ALL`: The gateway processes all notifications associated with a BMC Remedy ARS request, regardless of which user modifies the BMC Remedy ARS request. `NONE`: The gateway discards all notifications resulting from an action of the gateway user associated with this BMC Remedy ARS request. The default value is `'FIRST'`.
`Gate.Remedy.Password` *string*	`-remedypassword` *string*	Use this property to set the password to connect to BMC Remedy ARS for the user specified in the **`Gate.Remedy.Username`** property. The default value is `''`.
`Gate.Remedy.Port` *integer*	`-remedyport` *integer*	Use this property to specify the remote TCP port (instead of the BMC Remedy ARS port mapper) to which the gateway connects. Specify one of the following values: `0`: Do not specify the port for the gateway to connect with BMC Remedy ARS. Let BMC Remedy ARS resolve the port using the port mapper. Otherwise, if BMC Remedy ARS is not configured to use a port mapper, specify the actual port for the gateway to use The default value is `0`.

Table 8. Gateway specific properties (continued)

Property name	Command Line option	Description
`Gate.Remedy.ReclaimARS` *integer*	`-remedyreclaimars` *integer*	Use this property to specify whether and how to reclaim a request ID from BMC Remedy ARS when creating and updating an alert. This property takes one of the following values: `0`: Never query BMC Remedy ARS for a request ID. `1`: Query BMC Remedy ARS for a request ID when an error occurs during the creating and updating of the BMC Remedy ARS request. `2`: Always query BMC Remedy ARS for a request ID. The default value is `1`.
`Gate.Remedy.RetryCodes` *string*	`-remedyretrycodes` *string*	Use this property to specify the error codes that BMC Remedy ARS can return that result in the gateway retrying an operation. Use a comma to separate each of the error codes. The default value is `'90, 91, 92, 93, 95, 99, 394, 566, 590, 591, 9425'`
`Gate.Remedy.RetryWait` *integer*	`-remedyretrywait` *integer*	Use this property to specify the number of seconds the gateway should wait before retrying an operation (for example, forwarding an event to BMC Remedy ARS) that failed. The default value is 2 seconds.
`Gate.Remedy.Server` *string*	`-remedyserver` *string*	Use this property to set the name of the server on which BMC Remedy ARS is running and to which the gateway connects. The gateway will connect to this BMC Remedy ARS server to create requests and to receive events from it. The default value is `'localhost'`.

Table 8. Gateway specific properties (continued)

Property name	Command Line option	Description
`Gate.Remedy.UpdateForm` *string*	`-remedyupdateform` *string*	Use this property to specify the BMC Remedy ARS form used to store updates made within BMC Remedy ARS and which the gateway queries in order to retrieve updates for sending back to the ObjectServer. The default value is `'Alert Events'`.
`Gate.Remedy.UpdateQueryInterval` *integer*	`-remedyupdatequeryinterval` *integer*	Use this property to specify how often (in seconds) to query BMC Remedy ARS for updates. The default value is 3 seconds.
`Gate.Remedy.Username` *string*	`-remedyuser` *string*	Use this property to set the username for the user connecting to the BMC Remedy ARS. The default value is `''`.

Table 9 on page 40 lists and describes the properties provided by the gateway framework.

Table 9. Gateway framework properties

Property name	Command Line option	Description
`Gate.Cache.NotificationMap` *string*	`-cachenotificationmap` *string*	Use this property to specify the JavaScript file to use for alert notifications. The default value is `'$OMNIHOME/gates/bmc_remedy/bmc_remedy.notification.js'`

Table 9. Gateway framework properties (continued)

Property name	Command Line option	Description
`Gate.Cache.ResyncMode` *string*	`-cacheresyncmode` *string*	Use this property to specify a resynchronization mode for the gateway. This property takes one of the following values: `UNI`: The gateway performs a resynchronize operation from Tivoli Netcool/OMNIbus to BMC Remedy ARS. `BI`: The gateway operates in bidirectional resynchronization mode. In this mode the gateway retrieves open requests from BMC Remedy ARS and compares them with the open alerts on the ObjectServer. `AUTO`: The gateway operates in unidirectional resynchronization mode by default. If its alert cache is empty on startup, which normally only occurs the first time it is run, the gateway operates in bidirectional resynchronization mode. The default value is `AUTO`.
`Gate.Cache. ResyncSuppressDeletes` *string*	`-cacheresyncsuppress deletes` *string*	Use this property to suppress all resynchronization deletes. This property takes the following values: If true, suppress all resynchronization deletes. The default value is `FALSE`.
`Gate.Cache.Resync SuppressInitialDeletes` *string*	`-cacheresyncinitial deletes` *string*	Use this property to suppress initial (gateway startup) resynchronization deletes. This property takes the following values: If true, suppress initial (gateway startup) resynchronization deletes. The default value is`FALSE`.

Table 9. Gateway framework properties (continued)

Property name	**Command Line option**	**Description**
`Gate.ConversionsDirection` *string*	`-conversionsdirection` *string*	Use this property to specify how the gateway converts field values. For BMC Remedy ARS, the property uses the table `conversions.remedy`. This property takes the following values: `BOTH`: The gateway converts in both forward and reverse directions. Conversions generally convert integer ObjectServer values to and from target specific string values, typically converting Severity ("Clear", "Indeterminate", "Warning", "Minor", "Major", "Critical") from its integer representation to a string. `FORWARD`: The gateway only converts from OMNIbus integers to target string. REVERSE: The gateway only converts from target strings to OMNIbus integers. The default value is REVERSE. **Note :** BMC Remedy ARS only uses REVERSE because BMC Remedy ARS internally performs the forward direction conversion of Severity.
`Gate.ConversionsTableName` *string*	`-conversionstablename` *string*	Use this property to specify the name of the conversations table that the gateway uses for value conversion. The default value is `conversions.remedy`.

Table 9. Gateway framework properties (continued)

Property name	Command Line option	Description
`Gate.HistoricResync` *boolean*	`-historicresync` *boolean*	Use this property to configure how the gateway handles historic journal entries for an ObjectServer alert. The property takes one of the following values: • `true`: Instructs the gateway to transfer existing journal entries when creating a request in BMC Remedy ARS. • `false`: Instructs the gateway to not transfer existing journal entries when creating a request in BMC Remedy ARS.
`Gate.MapFile` *string*	`-mapfile` *string*	Use this property to specify the mapping file for the gateway to use. The default value is `'$OMNIHOME/gates/bmc_remedy/bmc_remedy.map'`.
`Gate.PackageBase` *string*	`-packagebase` *string*	Use this property to specify the Java gateway package. The default value is `'com.ibm.tivoli.netcool.integrations.gateway'` **Note :** This is an internal property and its value should not be changed.
`Gate.RdrWtr.DetailsTableName` *string*	`-detailstablename` *string*	Use this property to specify the name of the details table that the gateway reads. The default value is `'alerts.details'`. **Note :** The value for this property must be `'alerts.details'` and should not be changed.

Table 9. Gateway framework properties (continued)

Property name	Command Line option	Description
`Gate.RdrWtr.IducFlushRate` *integer*	`-iducflushrate` *integer*	Use this property to specify the rate (in seconds) of the granularity of the gateway's 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 value is `0` **Note :** If you set this property to a value greater than 0, the reader issues automatic IDUC flush requests to the ObjectServer with this period. This enables the reader to run at a faster granularity than that of the ObjectServer, thus enabling the gateway to capture more detailed event changes in systems where the ObjectServer itself has high granularity settings.
`Gate.RdrWtr.JournalTableName` *string*	`-journaltablename` *string*	Use this property to specify the name of the journal table that the gateway reads. The default value is `'alerts.journal'` **Note :** The value for this property must be `'alerts.journal'` and should not be changed.
`Gate.RdrWtr.Password` *string*	`-password` *string*	Use this property to specify the password associated with the user specified by the **`Gate.RdrWtr.Username`** property. The default value is `''` **Note :** The gateway uses this property only when the ObjectServer runs in secure mode.
`Gate.RdrWtr.ReconnectTimeout` *string*	`-reconnecttimeout` *value* `-noreconnecttimeout`	Use this property to specify the time (in seconds) between each reconnection poll attempt that the gateway makes if the connection to the ObjectServer is lost. The default value is `30` seconds. You can use `-noreconnecttimeout` to set the value to `0` to indicate that the gateway does not try to reconnect.

Table 9. Gateway framework properties (continued)

Property name	Command Line option	Description
`Gate.RdrWtr.Server` *string*	`-server` *string*	Use this property to specify the name of the ObjectServer from which the gateway reads alerts. The name can either be an interface name (for example, `NCOMS`) or the `<host>: <port>` details of the ObjectServer. The default value is `'localhost:4100'`
`Gate.RdrWtr.StatusTableName` *string*	`-statustablename` *string*	Use this property to specify the name of the status table to which the gateway reads and writes. The default value is `'alerts.status'`
`Gate.RdrWtr.TblReplicateDefFile` *string*	`-tblrepfile` *string*	Use this property to specify the path to the table replication definition file. The path includes the name of the table replication definition file. The default value (including the name of the table replication definition file supplied with the gateway) is `'$OMNIHOME/gates/bmc_remedy/bmc_remedy.rdrwtr.tblrep.def'`
`Gate.RdrWtr.Username` *string*	`-username` *string*	Use this property to specify the username used to authenticate the ObjectServer connection. The default value is `'root'`. **Note :** The gateway uses this property only when the ObjectServer runs in secure mode.
`Gate.TargetIdField` *string*	`-targetidfield` *string*	Use this property to specify the `alerts.status` column to use for target identifier tracking (that is, the identifier in BMC Remedy ARS). The default value is `'Location'`. **Note :** The identifier is the BMC Remedy ARS request id of the request in the main form (as specified in the **`Gate.Remedy.Form`** property) used by the gateway.

Table 9. Gateway framework properties (continued)

Property name	Command Line option	Description
`Gate.TraceFile` *string*	`-tracefile` *string*	Use this property to specify the path of the file used to store trace information. The default value is `''`. **Note :** Typically, this property is used by IBM support to debug any issues with gateway operations.
`Gate.TraceMode` *string*	`-tracemode` *string*	Use this property to specify the gateway trace mode. This property takes the following values: `READ`: Enables trace reading. `WRITE`: Enables trace writing. `NONE`: Does not enable trace reading or writing. The default value is `'NONE'`. **Note :** Typically, this property is used by IBM support to debug any issues with gateway operations.
`Gate.Type` *string*	`-type` *string*	Use this property to specify the gateway type. The default value is `'bmc_remedy'`. **Note :** The value for this property must be `'bmc_remedy'` and should not be changed.

Table 10 on page 46 lists and describes the generic OMNIbus properties that the gateway uses.

Table 10. Generic OMNIbus properties

Property name	Command Line option	Description
`ConfigKeyFile`		See the *IBM Tivoli Netcool/ OMNIbus Probe and Gateway Guide* for information on this property.
`HttpServer.Password` *string*		This property is not currently used.
`HttpServer.Port` *integer*		This property is not currently used.
`HttpServer.Username` *string*		This property is not currently used.
`HttpServer.Realm` *string*		This property is not currently used.

Table 10. Generic OMNIbus properties (continued)

Property name	Command Line option	Description
`HttpServer.Root` *string*		This property is not currently used.
`MessageLevel`		See the *IBM Tivoli Netcool/ OMNIbus Probe and Gateway Guide* for information on this property. **Note :** The default message level specified in the `G_BMC_REMEDY.props` properties file is `'warn:debug'`.
`MessageLog`		See the *IBM Tivoli Netcool/ OMNIbus Probe and Gateway Guide* for information on this property. **Note :** The default message log file specified in the `G_BMC_REMEDY.props` properties file is `'$OMNIHOME/log/ G_BMC_REMEDY.log'`
`MaxLogFileSize`		See the *IBM Tivoli Netcool/ OMNIbus Probe and Gateway Guide* for information on this property. **Note :** The default maximum size for the message log file specified in the `G_BMC_REMEDY.props` properties file is `'0'` bytes.
`MessageLogRollTime` *string*	`-messagelogrolltime` *string*	Use this property to specify at what time a daily log file should roll. Typically, this is done at midnight, to generate daily logs. However, you can specify any time. Specify the time in *HHMM* format. Where: *HH* specifies the hour and *MM* specifies the minutes of the time you want to roll the daily log. For example, the value `'0000'` specifies that the log file rolls at midnight. The value `'0430'` specifies that the log file rolls at 04:30. The default daily log file roll time specified in the `G_BMC_REMEDY.props` properties file is `''`.

Table 10. Generic OMNIbus properties (continued)

Property name	**Command Line option**	**Description**
`Name`		See the *IBM Tivoli Netcool/ OMNIbus Probe and Gateway Guide* for information on this property. **Note :** The default name of the current gateway instance specified in the `G_BMC_REMEDY.props` properties file is `'bmc_remedy'`.
`Props.AllowUnknown` *boolean*	`-propsallowunknown` *boolean*	Use this property to instruct the gateway on how to handle unknown properties in the properties file. Specify one of the following values: `true`: Instructs the gateway to allow unknown properties in the properties file and to ignore these unknown properties. `false`: Instructs the gateway to generate an error if there are unknown properties in the properties file. Use this setting to catch such errors as typos in property names and to avoid the diagnostics that follow typographical related errors. The default value is `'false'`.
`PropsFile`		See the *IBM Tivoli Netcool/ OMNIbus Probe and Gateway Guide* for information on this property. **Note :** The default location and name specified in the `G_BMC_REMEDY.props` properties file is `'$OMNIHOME/etc/ G_BMC_REMEDY.props'`.

Error messages

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

The following table describes the error messages that the Gateway for BMC Remedy ARS generates. Many of these errors relate to the gateway's interaction with the BMC Remedy Action Request System.

Table 11. Error messages

Error	Description	Action

`Alert reclaim from ARS failed:`
 	
`Unable to reclaim alert`
	The gateway cannot retrieve events from the BMC Remedy ARS system. Specifically, this error message appears if the gateway receives an Exception when it calls the `getListEntryObjects` method on the BMC Remedy ARS system.
	
	See the BMC Remedy Action Request System 8.1 API documentation for more information on the `getListEntryObjects` method.
		Check the BMC Remedy ARS logs for details.

`Cannot start alert events listener`	The gateway cannot start the Event listening thread. Typically, this error message appears if the **`Gate.Remedy.NotificationQuery`** property has a value that the BMC Remedy ARS server rejects.	Check that the **`Gate.Remedy.NotificationQuery`** property is correctly set.
`Disable ARS reclaim - No fields defined for ServerName and/or ServerSerial`	The gateway cannot use the ARS reclaim function because the relevant fields in the `bmc_remedy.map` are not mapped to their corresponding fields in BMC Remedy ARS. Specifically, the ARS reclaim function cannot work if the `@ServerName` and `@ServerSerial` fields in the `bmc_remedy.map` file are not mapped to their corresponding fields in BMC Remedy ARS. If this is the case, the gateway overwrites the **`Gate.Remedy.ReclaimARS`** property to the value 0 (zero), thus causing the ARS reclaim function to be disabled. **Note :** The value 0 (zero) specified for the **`Gate.Remedy.ReclaimARS`** property signifies that the gateway never queries BMC Remedy ARS for a request ID.	Ensure that the following fields in the `StatusMap` (contained in the `bmc_remedy.map` file) are mapped to their corresponding fields in BMC Remedy ARS: • **`@ServerName`** • **`@ServerSerial`**

Table 11. Error messages (continued)

Error	**Description**	**Action**

`Dropping current Entry due to format error`	The gateway attempted to process an entry read from `Alert Events` (the standard form in BMC Remedy ARS). However, the gateway failed to process this entry due to a format error.	The contents of the entries in `Alert Events` is defined by the filters provided in the `omnibus.def` file. The following list identifies these filters:
		• `Omnibus-Close` - This filter is used when the user closes a BMC Remedy ARS request.
		• `Omnibus-Modify` - This filter is used when the user updates a BMC Remedy ARS request.
		Verify the definitions of these filters. The expected format of the `Omnibus-Close` filter is as follows:
		`[C] [user] field = value`
		`field = value ...`
		The expected format of the `Omnibus-Modify` filter is as follows:
		`[M] [user] field = value`
		`field = value ...`

`Exception raised while handling update from Remedy.`	The gateway encountered a problem while trying to update an entry in `Alert Events` (the standard form in BMC Remedy ARS). The gateway can update entries in `Alert Events` in two ways: • By marking the entry as "read". • By deleting the entry. Thus, the gateway encountered the problem while attempting to perform one of these actions.	Check the BMC Remedy ARS logs for details.

Table 11. Error messages (continued)

Error	Description	Action

`Exception while extracting data from the Entry object`	The gateway attempted to process an entry read from `Alert Events` (the standard form in BMC Remedy ARS). However, the gateway failed to process this entry due to a format error.
	
	Typically, this error message appears when one or several fields in the `Alert Events` entry contains whitespace. For example, the field `Ticket number = 27` causes a parsing error because of the whitespace between `Ticket` and `number`.
		The contents of the entries in `Alert Events` is defined by the filters provided in the `omnibus.def` file. The following list identifies these filters:
		
		`Omnibus-Close`: This filter is used when the user closes a BMC Remedy ARS request.
		
		`Omnibus-Modify`: This filter is used when the user updates a BMC Remedy ARS request.
		
		Verify the definitions of these filters. The expected format of the `Omnibus-Close` filter is as follows:
		
		`[C] [user] field = value field = value ...`
		
		The expected format of the `Omnibus-Modify` filter is as follows:
		
		`[M] [user] field = value field = value ...`
		
		In this example, to correct the parsing error due to the whitespace between `Ticket` and `number`, you would modify the `Omnibus-Modify` filter in the `omnibus.def` file to ensure that all field names (or field values) that included or might include whitespace were enclosed within quotes. For example: `"Ticket number" = 27`.
		
		Note : It is best practice to enclose all field names and field values specified in the `omnibus.def` file within quotes to avoid the error described here.

Table 11. Error messages (continued)

Error	**Description**	**Action**

`One of the following three key properties has no value:`
 	
`Property 'Gate.Remedy.Username' :` *`username`*
 	
`Property 'Gate.Remedy.Password' :` *`password`*
 	
`Property 'Gate.Remedy.Server' :` *`server`*
 	
`Therefore, the Gateway won't be able to connect to the Remedy server.`
	The gateway requires valid values for the **`Gate.Remedy.Username`**, **`Gate.Remedy.Password`**, and **`Gate.Remedy.Server`** properties to successfully connect to the BMC Remedy ARS server. The gateway has detected that one of these properties has no value specified.	Ensure that the **`Gate.Remedy.Username`**, **`Gate.Remedy.Password`**, and **`Gate.Remedy.Server`** properties contain valid values so that the gateway can connect to BMC Remedy ARS server.

`NotificationQuery property has an empty value, resetting it to its default value`
 	
`NotificationQuery is reset to its default value : '7' < "X" AND '2'="USER" AND '701'="FORM`
	The gateway logs these error messages when the **`Gate.Remedy.NotificationQuery`** property contains an empty value.	Ensure that the **`Gate.Remedy.NotificationQuery`** property contains a valid query that the gateway uses to select entries in BMC Remedy ARS.

`Parsing failed on text:` *`text`*	The gateway's fallback parser has detected an error in the *text* associated with the specified notification. For example:
	
	`Location = Cedar City`
	
	In the example, the space in the value string `Cedar City` causes the gateway's fallback parser to write this error message to the log file.
		The gateway has two string parsers: a new parser and a fallback parser. This error message is the result of the fallback parser detecting name/value pairs in the notification that do not adhere to the following syntax:
		
		`name` `=` *`value_no_spaces`* - For example: `CedarCity`
		
		`name` `= "`*`value`*`"` - For example: `"Cedar City"`
		
		`name` `= '`*`value`*`'` - For example: `'Cedar City'`
		
		Use the previously described syntax in the name/value pairs in notifications to eliminate these types of error message from the log file.

Table 11. Error messages (continued)

Error	Description	Action
`Remedy unable to find the form. Please double-check if the form exists in remedy.`	BMC Remedy ARS cannot find the form specified for creating requests, specified in the **`Gate.Remedy.Form`** property.	Ensure that the value of **`Gate.Remedy.Form`** is the valid name of a form in BMC Remedy ARS and is a form suitable for creating requests.
`Something went wrong while processing the notification`	An error occurred while the gateway tried to update an alert in the ObjectServer (after processing an alert event (or notification) read from BMC Remedy ARS).	Check that the ObjectServer is up and running. To update the entry, the gateway will most likely use the `update` and `clear` functions as defined in the `bmc_remedy.notification.js` file. Verify the contents of those functions.

`The Gateway cannot find the mapping associated with the Status table.`
`Check the value of the 'Gate.RdrWtr.StatusTable Name' property is correct.`
	The gateway was unable to validate the data mapping for the ObjectServer status table.	Check the following:
		• The mapping file contains a **`CREATE MAPPING`** statement for the status table.
		• The value of the **`Gate.Mapfile`** property is the path for the correct mapping file.
		• The value of the **`Gate.RdrWtr.StatusTable Name`** property is the correct name for the status table in the ObjectServer.

`Unable to connect to Remedy`	The gateway cannot connect to the BMC Remedy ARS server.	Check that the BMC Remedy ARS server is available. Ensure that the following gateway properties are correctly set: • **`Gate.Remedy.Password`** • **`Gate.Remedy.Port`** • **`Gate.Remedy.Server`** • **`Gate.Remedy.Username`**
`Truncated value to` *`maxSize`* `" characters: "` *`newValue`*	The gateway has determined that a BMC Remedy ARS field has exceeded its maximum length. The gateway truncates the length of this BMC Remedy ARS field to its valid maximum size and returns the new value.	Ensure that BMC Remedy ARS fields do not exceed maximum length of the field.

Table 11. Error messages (continued)

Error	**Description**	**Action**

`Unterminated quoted string in notification!`	The gateway's fallback parser has detected an unterminated quoted string in a name/value pair in the specified notification. For example:
	`Location = "Cedar City`
	`Location = 'Cedar City`
	In the examples the missing double quote (") and the missing single quote (') causes the gateway's fallback parser to write this error message to the log file.
		The gateway has two string parsers: a new parser and a fallback parser. This error message is the result of the fallback parser detecting name/value pairs in the notification that do not adhere to the following syntax:
		• *name* = *value_no_spaces* - For example: `CedarCity`
		• *name* = "*value*" - For example: `"Cedar City"`
		• *name* = '*value*' - For example: `'Cedar City'`
		Use the previously described syntax in the name/value pairs in notifications to eliminate these types of error message from the log file.

Messages related to events

The ObjectServer sends events to the BMC Remedy Action Request System. You can see these events in a log file using DEBUG mode.

Overview

The Java Gateway for BMC Remedy ARS uses the mapping file, `bmc_remedy.map`, to determine how to map fields in ObjectServer tables to fields in a BMC Remedy ARS request. The `bmc_remedy.map` file contains a number of `CREATE MAPPING` commands each of which maps specific ObjectServer table fields to fields in a BMC Remedy ARS request.

See"Mapping" on page 21 for more information on mapping and the `bmc_remedy.map` file.

Setting the level of debug messages

Use the **`MessageLevel`** property to specify the reporting level for gateway log file messages. Use the **`MessageLog`** property to specify the location of the gateway message log file.

See the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* for more information on these properties.

Example log file containing debug messages related to events (BMC Remedy ARS field ID numbers)

The following example shows a sample log file that contains debug messages related to the events that the ObjectServer sends to BMC Remedy ARS. These debug messages are the result of the gateway using the `bmc_remedy.map` file to map specific ObjectServer table fields to fields in a BMC Remedy ARS request. This example shows the debug messages with the BMC Remedy ARS ID numbers.

The numbers that follow provide explanations of specific lines of the sample log file:

```
14/09/01 10:20:14: Debug: [Request consumer 1] Calling createEntry for alert  1
14/09/01 10:20:14: Debug: [Request consumer 1] SOGValues :  2
536870914=LondonMachineMon4Systems;  3
536870913=5510;  4
```

```
536870944=London;
536870915=;
536870932=Simnet Probe;
536870939=MachineMon;
536870918=Systems;
536870931=;7=4;
536870916=Machine has gone offline;
536870946=1409563197;
536870912=1409563197;
536870917=1409563197;
536870935=1409563197;536870938=0;
536870941=0;536870940=1;
536870945=3300;
536870942=0;
536870934=;
536870933=0;
536870922=5510;
536870923=RAVINE;
```

1. This debug message indicates that the gateway called the `createEntry` method, which creates a new entry in the specified BMC Remedy ARS form on the specified BMC Remedy ARS server.

   Note : The `createEntry` method is part of the AR System API.
2. This debug message indicates that the values that follow are the Simple OMNIbus Gateway (SOG) values associated with BMC Remedy ARS fields in the request. Specifically, the `StatusMap`, defined in `bmc_remedy.map`, is the main map for alerts.status entries. The `StatusMap` maps Tivoli Netcool/OMNIbus alert fields to their corresponding BMC Remedy ARS fields.

   See "Mapping" on page 21 for more information on the `StatusMap`.
3. The 536870914 BMC Remedy ARS field maps to the ObjectServer table field called `@Identifier` (as defined in the `StatusMap` in the `bmc_remedy.map` file). In this alert example, the SOG value for this field is `LondonMachineMon4Systems`.

   Note : It is not possible to know that the 536870914 BMC Remedy ARS field maps to the ObjectServer table field called `@Identifier` just by looking at the debug messages. However, it is possible to make the appropriate mapping between the fields by checking the `StatusMap` in the `bmc_remedy.map` file.
4. The 536870913 BMC Remedy ARS field maps to the ObjectServer table field called `@Serial` (as defined in the `StatusMap` in the `bmc_remedy.map` file). In this alert example, the SOG value for this field is `5510`.

   Note : It is not possible to know that the 536870913 BMC Remedy ARS field maps to the ObjectServer table field called `@Serial` just by looking at the debug messages. However, it is possible to make the appropriate mapping between the fields by checking the `StatusMap` in the `bmc_remedy.map` file.

Example log file containing debug messages related to events (BMC Remedy ARS field names)

The following example shows a sample log file that contains debug messages related to the events that the ObjectServer sends to BMC Remedy ARS. These debug messages are the result of the gateway using the `bmc_remedy.map` file to map specific ObjectServer table fields to fields in a BMC Remedy ARS request. This example shows the debug messages with the BMC Remedy ARS field names (as defined in the `omnibus.def` file) instead of the BMC Remedy ARS field ID numbers as in the previous example.

The numbers that follow provide explanations of specific lines of the sample log file:

```
14/09/16 13:12:14: Debug: [Request consumer 2] Entry contents :  1 
Identifier=TokyoMachineStats4Stats;  2 
Serial=957;  3 
Node=Tokyo;
Node Alias=;
Manager=Simnet Probe;
Agent=MachineStats;
Alert Group=Stats;
Alert Key=99% full;
Status=4;
Summary=Diskspace alert;
State Change=1410859573;
FirstOccurance=1410451826;
```

```
LastOccurance=1410859573;
InternalLast=1410859573;
Poll=0;
Type=0;
Tally=13710;
Class=3300;
Grade=0;
Location=;
Acknowledged=0;
ServerSerial=957;
ServerName=NCOMS;
```

1. This debug message indicates that the values that follow are the Simple OMNIbus Gateway (SOG) values associated with BMC Remedy ARS fields in the request. Specifically, the `StatusMap`, defined in `bmc_remedy.map`, is the main map for alerts.status entries. The `StatusMap` maps Tivoli Netcool/OMNIbus alert fields to their corresponding BMC Remedy ARS fields.

   See "Mapping" on page 21 for more information on the `StatusMap`.
2. The `Identifier` BMC Remedy ARS field maps to the ObjectServer table field called `@Identifier` (as defined in the `StatusMap` in the `bmc_remedy.map` file). In this alert example, the SOG value for this field is `TokyoMachineStats4Stats`.

   Note : It is not possible to know that the `Identifier` BMC Remedy ARS field maps to the ObjectServer table field called `@Identifier` just by looking at the log messages. However, it is possible to make the appropriate mapping between the fields by checking the `StatusMap` in the `bmc_remedy.map` file.
3. The `Serial` BMC Remedy ARS field maps to the ObjectServer table field called `@Serial` (as defined in the `StatusMap` in the `bmc_remedy.map` file). In this alert example, the SOG value for this field is `957`.

   Note : It is not possible to know that the `Serial` BMC Remedy ARS field maps to the ObjectServer table field called `@Serial` just by looking at the log messages. However, it is possible to make the appropriate mapping between the fields by checking the `StatusMap` in the `bmc_remedy.map` file.

GatewayWatch messages

During normal operation, the gateway generates GatewayWatch messages and sends them to the ObjectServer. These messages inform the ObjectServer how the gateway is running.

The following table describes the GatewayWatch messages that the Gateway for BMC Remedy ARS generates. These GatewayWatch messages relate to the gateway's interaction with the BMC Remedy Action Request System.

Table 12. GatewayWatch messages

GatewayWatch message	Description	Triggers/causes
`Successfully queried Remedy for updates.`	While the gateway is running, it connects to and queries the BMC Remedy ARS server for alerts, in this case updates to BMC Remedy ARS requests.	This GatewayWatch message gets sent to the ObjectServer when the gateway update query to the BMC Remedy ARS server succeeds.

Table 12. GatewayWatch messages (continued)

GatewayWatch message	Description	Triggers/causes
`Problem while querying Remedy for updates:` *`exception`*	While the gateway is running, it connects to and queries the BMC Remedy ARS server for alerts, in this case updates to BMC Remedy ARS requests.	This GatewayWatch message gets sent to the ObjectServer when the gateway update query to the BMC Remedy ARS server fails. The *exception* specifies a message that describes why this particular update query failed. For example: `Problem while querying Remedy for updates: ERROR (90): 90; Connection refused connect 9.70.60.40.`
`Gateway connected to Remedy`	The gateway establishes a connection to the BMC Remedy ARS server and performs a number of initialization and validation operations.	This GatewayWatch message gets sent to the ObjectServer when the gateway successfully connects to the BMC Remedy ARS server. This message indicates that the gateway can perform its initialization and validation operations.
`Unable to connect to Remedy:` *`exception`*	The gateway fails to establish a connection to the BMC Remedy ARS server.	This GatewayWatch message gets sent to the ObjectServer when the gateway is unable to connect to the BMC Remedy ARS server. This message indicates that the gateway did not perform its initialization and validation operations. The *exception* specifies a message that describes why this particular connection attempt failed. For example: `Unable to connect to Remedy: ERROR (90): 90; Connection refused connect 9.70.60.40.`

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

Trademarks

Part Number:

(1P) P/N:

SC27-6553-02