IBM® Tivoli® Netcool/OMNIbus Generic Log File Java Probe
1.0

Reference Guide
July 20, 2017

SC27-8704-01

Notice

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 19.

Edition notice

This edition (SC27-8704-02) applies to version 1.0 of IBM Tivoli Netcool/OMNIbus Generic Log File Java Probe and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-8704-01.

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 Generic Log File Java Probe documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM® Tivoli® Information Center:

https://www.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/common/Probes.html

Table 1. Document modification history

Document version	Publication date	Comments
SC27-8704-00	December 10, 2015	First IBM publication. This probe addresses the following Enhancement Request: **RFE 13074**: Request to support target log files of 4Gb or above using the Netcool/OMNIbus Generic Log File Probe on Windows and UNIX platforms.
SC27-8704-01	July 20, 2017	"Summary" on page 1 updated to include note about the Generic Log File Java Probe unable to be managed under Process Agent control. "Example usage" on page 5 added.
SC27-8704-02	December 13, 2019	"Known issues" on page 17 updated.

Conventions used in this guide

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

Operating system-dependent variables and paths

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

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

```
$OMNIHOME/probes
```

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

```
%OMNIHOME%\probes
```

For probes supported on UNIX, Linux, and Windows operating systems, probe guides use the standard UNIX conventions for specifying environment variables and describing directory paths. When using the

Windows command line with these probes, 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. Where such variables are described in the guide, both the UNIX and Windows conventions will be used.

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/probes/`*`arch`*

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

Note : This probe 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

Probes 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. Generic Log File Java Probe

The IBM Tivoli Netcool/OMNIbus Generic Log File Java Probe is a multi-platform probe that reads a flat log file and parses the values using specified element delimiters and name-value pair separators. The probe creates dynamic, numbered elements according to the resulting parsed data.

The Generic Log File Java Probe is only supported on Netcool/OMNIbus V8.1 and above. If you are using an earlier version of Netcool/OMNIbus, use the legacy Generic Log File Probe.

This guide contains the following sections:

Summary

Each probe works in a different way to acquire event data from its source, and therefore has specific features, default values, and changeable properties. Use this summary information to learn about this probe.

The following table provides a summary of the IBM Tivoli Netcool/OMNIbus Generic Log File Java Probe.

Table 3. Summary

Probe target	Log files
Probe executable name	`nco_p_glf_java`
Package version	1.0
Probe supported on	For details of supported operating systems, see the following Release Notices on the IBM Software Support website: http://www-01.ibm.com/support/docview.wss?uid=swg21970416
Properties file	`$OMNIHOME/probes/`*`arch`*`/glf_java.props`
Rules file	`$OMNIHOME/probes/`*`arch`*`/glf_java.rules`
Requirements	For details of any additional software that this probe requires, refer to the `description.txt` file that is supplied in its download package.
Connection method	The probe accesses log files using Java NIO.2 in the GLF stream component. The probe implements the GLF stream component using Apache Camel as a routing and processing engine.

Table 3. Summary (continued)

Remote connectivity	No
Multicultural support	Available
Peer-to-peer failover functionality	Available
IP environment	IPv4 and IPv6
Federal Information Processing Standards (FIPS)	IBM Tivoli Netcool/OMNIbus uses the FIPS 140-2 approved cryptographic provider: IBM Crypto for C (ICC) certificate 384 for cryptography. This certificate is listed on the NIST website at http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2004.htm. For details about configuring Netcool/OMNIbus for FIPS 140-2 mode, see the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*.

Note : The Generic Log File Java Probe cannot be managed under Process Agent control. To workaround this issue you can run the probe as a Windows service. For details about configuring the probe to run as a Windows service, see

https://www.ibm.com/support/knowledgecenter/en/SSSHTQ/omnibus/probes/all_probes/wip/reference/running_windows_service.html.

Installing probes

All probes are installed in a similar way. The process involves downloading the appropriate installation package for your operating system, installing the appropriate files for the version of Netcool/OMNIbus that you are running, and configuring the probe to suit your environment.

The installation process consists of the following steps:

1. Downloading the installation package for the probe from the Passport Advantage Online website.

   Each probe has a single installation package for each operating system supported. For details about how to locate and download the installation package for your operating system, visit the following page on the IBM Tivoli Knowledge Center:

   http://www-01.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/all_probes/wip/reference/install_download_intro.html
2. Installing the probe using the installation package.

   The installation package contains the appropriate files for all supported versions of Netcool/OMNIbus. For details about how to install the probe to run with your version of Netcool/OMNIbus, visit the following page on the IBM Tivoli Knowledge Center:

   http://www-01.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/all_probes/wip/reference/install_install_intro.html
3. Configuring the probe.

   This guide contains details of the essential configuration required to run this probe. It combines topics that are common to all probes and topics that are peculiar to this probe. For details about additional configuration that is common to all probes, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Migrating to the Generic Log File Java Probe

This topic describes how to migrate from the legacy Generic Log File Probe to the Generic Log File Java Probe.

The following topics describe the steps required to migrate to the Generic Log File Java Probe:

- "Identifying new and changed features of the Generic Log File Java Probe" on page 3.
- "Customizing the rules file" on page 4.

Identifying new and changed features of the Generic Log File Java Probe

When migrating from the legacy probe to the Generic Log File Java Probe, there are some important points to note about the features that are available and the properties that are required to configure them:

- Some properties form the legacy probe have been deprecated in the Generic Log File Java Probe.
- Some properties form the legacy probe have been renamed and now take different values.
- Some features, and the properties required to configure them, have been added to the Generic Log File Java Probe.

The following table compares the features, and the properties required to configure them, of the legacy Generic Log File Probe and the Generic Log File Java Probe.

Table 4. Comparison of properties between the legacy Generic Log File Probe and the Generic Log File Java Probe

Legacy Generic Log File Probe property	Generic Log File Java Probe property	Functional description
`ValueSeparator`	**`ParserElementDelimiter`**	Separator/delimiter that separates tokens/elements.
`LineSeparator`	This property has been deprecated.	Separator that indicates the end of each line.
Not available in the legacy probe.	**`ParserNVPDelimiter`**	Separator/delimiter that separates each name-value pair.
`QuoteCharacter`	**`ParserQuoteCharacter`**	Character that the probe should treat as a quotation mark.
`MaxBufferSize`	**`ParserMaxEventSize`**	Maximum size allowed for a single event.
`IgnoreNullFields`	**`ParserIgnoreEmptylFields`**	Indicates whether the probe allows fields to be empty.
`StripCharacter`	**`ParserStripCharacter`**	Indicates which characters should be stripped out of a token value.
`StripCharacterFormat`	**`ParserReplaceStripCharWith`**	Character with which the probe should replace the strip character.

Table 4. Comparison of properties between the legacy Generic Log File Probe and the Generic Log File Java Probe (continued)

Legacy Generic Log File Probe property	Generic Log File Java Probe property	Functional description
MaxNumTokens	This property has been deprecated.	Maximum number of tokens allowed for a single event.
MaxTerminators	This property has been deprecated.	Maximum number of terminators allowed for a single event.
CleanStart	**CleanStart**	Indicates whether the probe ignores the recovery file and starts reading new events.
Not available in the legacy probe.	**DSLLogConfig**	Indicates the path to the configuration that defines framework Apache Camel context advanced logging for troubleshooting purpose.
Not available in the legacy probe.	**EncodingStandard**	Indicates the character set used when encoding the content of the input log file.
LogFileName	**LogFileName**	Indicates the path to the log file that the probe reads.
RecoveryFile	**RecoveryFile**	Indicates the path to the recovery file.
ReplayFile	This property has been deprecated.	Indicates whether the probe replays the log file.
RetryPeriod	**RetryInterval** This property is provided by the probe framework version 8.0.	Indicates the number of retries that the probe makes before shutting down.
Not available in the legacy probe.	**RetryCount** This property is provided by the probe framework version 8.0.	Indicates the number of retries that the probe makes before shutting down.
Not available in the legacy probe.	**Inactivity** This property is provided by the probe framework version 8.0.	Indicates the length of time (in seconds) that the probe allows the port to receive no incoming data before disconnecting.

Customizing the rules file

When migrating from the legacy probe to the Generic Log File Java Probe, there are some important points to note about the rules file:

- The rules file supplied with the Generic Log File Java Probe no longer generates tokens with the name: `FieldVal`. It uses instead: `Token`.
- The Generic Log File Java Probe no longer generates the token named `Overflow`. There is no replacement for this token.
- The ProbeWatch messages have been modified. Refer to the rules file supplied with the Generic Log File Java Probe for details of the new statements that deal with the updated ProbeWatch messages.

If you want to migrate custom rules from the legacy probe, you must update the rules accordingly.

Running the probe

Probes can be run in a variety of ways. The way you chose depends on a number of factors, including your operating system, your environment, and the any high availability considerations that you may have.

For details about how to run the probe, visit the following page on the IBM Tivoli Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/all_probes/wip/concept/running_probe.html

Example usage

The Generic Log File Java Probe obtains events by reading a flat log file. You can run the probe in either Recovery Mode or Clean Start Mode.

In Clean Start Mode, the probe reads all new events created in the log file since the probe started.

In Recovery Mode, the probe stores recovery data in a recovery file. When the probe starts, it references this recovery file to determine the position within the log file from which to start reading events.

The following steps describe how to run the probe in Recovery Mode:

1. Set the **`CleanStart`** property to false.
2. Set the **`RecoveryFile`** property to the location of the recovery file that the probe reads.
3. The default location of this recovery file is `/opt/IBM/tivoli/netcool/omnibus/var/glf_java.reco`. If the probe is installed outside of this default location, update the setting of the **`RecoveryFile`** property accordingly.

   Note : To instruct the probe to re-read an existing log file, delete the recovery file. When set, the probe incorporates the values set for the **`Name`** and **`Mode`** properties in the file name of the recovery file.

   For example, if you set the following properties:

   ```
   Name : 'my_glf_java'
   Mode : 'standard'
   ```

   The probe creates a recovery file with the name `reco.my_glf_java.standard`.

Data acquisition

Each probe uses a different method to acquire data. Which method the probe uses depends on the target system from which it receives data.

The IBM Tivoli Netcool/OMNIbus Generic Log File Java Probe accesses the log file specified by the **`LogFileName`** property using a GLF stream component. It reads the stream to retrieve alarms.

The probe parses the alarms retrieved from log file stream according to the values set for the **`ParserElementDelimiter`**, **`ParserNVPDelimiter`**, and **`ParserQuoteCharacter`** properties. The probe parses the stream of name-value pairs by searching for the character specified by

ParserElementDelimiter (this identifies where each name-value pair ends) and **ParserNVPDelimiter** (this separates each token name from its corresponding value).

The probe assigns each string separated by a `ParserElementDelimiter` as `$Token[`*n*`]`, where *n* is an incremental value, starting at `[1]`. The probe also concatenates all `$Token[`*n*`]` elements, for each event, into an element called `$FullDetails`.

The probe also generates `$TokenCount`, which is the total number of tokens, calculated as follows: total number of name-value pairs plus the `$FullDetails` token.

Data acquisition is described in the following topics:

- "Modes of operation" on page 6
- "Backoff strategy" on page 6
- "Support for Unicode and non-Unicode characters" on page 7
- "HTTP/HTTPS command interface" on page 8
- "Peer-to-peer failover functionality" on page 9
- "Running multiple probes to monitor more than one log file" on page 9

Modes of operation

The modes of operation determine how the probe reads the log file and the recovery file on startup.

You can run the probe in one of the following modes:

- **Recovery**: The probe stores recovery data in a recovery file. When the probe starts, it references the recovery file to determine the point within the log file from which to start reading alarms.

  If a recovery file is not present (for example, when the probe starts for the first time), the probe creates one and sets the current position in the log file; it then reads events as they are received.

  If a recovery file is present, the probe replays the log file from the position set in the recovery file and continues to read the log file as new events are received.

  If the stored file point in the recovery file is not valid (for example, the file has been overwritten or recreated), the probe reads the log file from the start, and adds a new file point to the recovery file.

  To run the probe in recovery mode, set the **CleanStart** properties to `false` and specify a recovery file using the **RecoveryFile** property. The default value for **RecoveryFile** is `/opt/IBM/tivoli/netcool/omnibus/var/glf_java.reco`. You must check that this is a valid location before running probe, and update the property value if necessary,
- **Cleanstart**: The probe ignores the recovery file as well as any existing file content and starts reading new events created in the log file since the probe started.

  To run the probe in cleanstart mode, set the **CleanStart** property to `true`.

If the **CleanStart** property is set to `true`, cleanstart mode takes precedence over recovery mode. However, when the probe starts in cleanstart mode, the probe will first change the store pointer in the recovery file specified by the **RecoveryFile** property to the end of the file, and then will update the pointer when subsequently reading the content of the log file. So if you set the **CleanStart** property to `true`, you must still specify a recovery file using the **RecoveryFile** property.

Backoff strategy

The **RetryInterval** and **RetryCount** properties allow you configure the probe's backoff strategy.

The **RetryInterval** property allows you to specify the length of time (in seconds) that the probe leaves between successive reconnection attempts. If the **RetryInterval** property is set to `0`, the probe tries to reestablish a connection after one second, then two seconds, then four seconds, then eight seconds, and so forth.

When the number of times that the probe has attempted to reconnect reaches that specified by the **RetryCount** property, the probe shuts down. If you set **RetryCount** to 0 the probe makes no retry attempts.

The following scenarios will trigger the probe to shutdown, making retry attempts as specified by the **RetryInterval** and **RetryCount** properties:

- There are file opening issues.
- The file is deleted, with or without the file name being taken over by another file.
- The file is moved to another location (and so is considered deleted).
- The file is renamed (and so is considered deleted).

The following scenarios will not trigger probe to shutdown. Instead it writes an error message to the log file and attempts to reopen the file:

- The file is truncated.
- The file is overwritten with another file content. For example, file overwriting using the **mv** command in Linux will cause the inode number to change and so will trigger the probe to attempt to reopen the file.
- The file creation date has changed (and so is considered overwritten).

Support for Unicode and non-Unicode characters

The probe can process multibyte characters and so can display both Unicode and non-Unicode characters.

Use the following procedure to set up the probe to process multibyte characters:

1. Ensure that the file contains data in the required format, for example, UTF-8.
2. Set the required locale on the system running the probe:

Table 5. Setting the locale for multibyte characters

Operating system	Procedure to set the locale
Linux and Unix	Set the locale by changing the values of the **LANG** and **LC_ALL** environment variables. For example, to set the locale to simplified Chinese in UTF-8, use the following commands: `export LANG=zh_CN.UTF-8` `export LC_ALL=zh_CN.UTF-8`
Windows	a. Open the **Control Panel** and double click on **Regional and Language**. b. On the **Formats** tab, select the language from the list in **Format**. c. On the **Administrative** tab, click **Change system locale**. d. Select the language from the list in **Current System Locale**. e. Click **OK**. f. Click **OK**.

3. Configure the ObjectServer to enable the insertion of data that uses the required character set. The *IBM Tivoli Netcool/OMNIbus Administration Guide* shows how to create, configure, and run an ObjectServer in UTF-8 mode or using another character set.
4. Run the probe. If it is already running, restart it.

   When running the probe on a Windows system using a UTF-8 character set, always specify the `-utf8enabled` command line option with a setting of TRUE. For all other character sets, omit the `-utf8enabled` command line option.

HTTP/HTTPS command interface

IBM Tivoli Netcool/OMNIbus Version 7.4.0 (and later) includes a facility for managing the probe over an HTTP/HTTPS connection. This facility uses the **nco_http** utility supplied with Tivoli Netcool/OMNIbus.

The HTTP/HTTPS command interface replaces the Telnet-based command line interface used in previous version of IBM Tivoli Netcool/OMNIbus.

The following sections show:

- How to configure the command interface.
- The format of the **nco_http** command line.
- The format of the individual probe commands.

For more information on the HTTP/HTTPS command interface and the utilities it uses, see the chapter on remotely administering probes in the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Configuring the command interface

To configure the HTTP/HTTPS command interface, set the following properties in the probe's property file:

NHttpd.EnableHTTP: Set this property to `True`.
NHttpd.ListeningPort: Set this property to the number of the port that the probe uses to listen for HTTP commands.

Optionally, set a value for the following property as required:

NHttpd.ExpireTimeout: Set this property to the maximum time (in seconds) that the HTTP connection remains idle before it is disconnected.

The *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* contains a full description of these and all properties for the HTTP/HTTPS command interface.

Format of the nco_http command line

The format of the **nco_http** command line to send a command to the probe is:

```
$OMNIHOME/bin/nco_http -uri probeuri:probeport/probes/generic_3gpp -datatype
application/json -method post -data '{"command":"command-name","params":
[command-parameters]}'
```

Where:

- *probeuri* is the URI of the probe.
- *probeport* is the port that the probe uses to listen for HTTP/HTTPS commands. Specify the same value as that set for the **NHttp.ListeningPort**.
- *command-name* is the name of the command to send to the probe. For the Generic Log File Java Probe, only the following command name is currently available:

  shutdown_probe
- *command-parameters* is a list of zero or more command parameters. For commands that have no parameters, this component is empty. The command descriptions in the following section define the parameters that each takes.

Probe commands

The following section defines the structure of the JavaScript Object Notation (JSON)-formatted commands that you can send to the probe.

The example uses a probe URI of `http://localhost` and a HTTP listening port of `8080`.

shutdown_probe

The **shutdown_probe** command allows you to shut down the probe.

The format of the `-data` option for the **shutdown_probe** command is:

```
-data '{"command":"shutdown_probe","params":[]}'
```

The following example shuts down the Generic Log File Java Probe:

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/glf_java -datatype
application/JSON -method POST -data '{"command":"shutdown_probe","params":[]}'
```

Peer-to-peer failover functionality

The probe supports failover configurations where two probes run simultaneously. One probe acts as the `master` probe, sending events to the ObjectServer; the other acts as the `slave` probe on standby. If the master probe fails, the slave probe activates.

While the slave probe receives heartbeats from the master probe, it does not forward events to the ObjectServer. If the master probe shuts down, the slave probe stops receiving heartbeats from the master and any events it receives thereafter are forwarded to the ObjectServer on behalf of the master probe. When the master probe is running again, the slave probe continues to receive events, but no longer sends them to the ObjectServer.

Example property file settings for peer-to-peer failover

You set the peer-to-peer failover mode in the properties files of the master and slave probes. The settings differ for a master probe and slave probe.

Note : In the examples, make sure to use the full path for the property value. In other words replace $OMNIHOME with the full path. For example: `/opt/IBM/tivoli/netcool`.

The following example shows the peer-to-peer settings from the properties file of a master probe:

```
Server      :    "NCOMS"
RulesFile   :    "master_rules_file"
MessageLog  :    "master_log_file"
PeerHost    :    "slave_hostname"
PeerPort    :    6789 # [communication port between master and slave probe]
Mode        :    "master"
PidFile     : "master_pid_file"
```

The following example shows the peer-to-peer settings from the properties file of the corresponding slave probe:

```
Server      :    "NCOMS"
RulesFile   :    "slave_rules_file"
MessageLog  :    "slave_log_file"
PeerHost    :    "master_hostname"
PeerPort    :    6789 # [communication port between master and slave probe]
Mode        :    "slave"
PidFile     : "slave_pid_file"
```

Running multiple probes to monitor more than one log file

You can run multiple instances of the Generic Log File Java Probe on a single machine allowing you to monitor more than one log file.

To do so, you must ensure that each instance specifies a unique value for the following properties so that they are able to access a different set of log files:

- **Name**
- **PropsFile**

- **`RulesFile`**
- **`PidFile`**

The following example shows the settings from the properties file of the first probe:

```
Name :       "glf_java1"
PropsFile :  "/opt/tivoli/netcool/omnibus/probes/<arch>/glf_java1.props"
RulesFile :  "/opt/tivoli/netcool/omnibus/probes/<arch>/glf_java1.rules"
MessageLog : "/opt/tivoli/netcool/omnibus/log/glf_java1.log"
PidFile :    "/opt/tivoli/netcool/omnibus/var/Stream1"
```

The following example shows the settings from the properties file of the second probe:

```
Name :       "glf_java2"
PropsFile :  "/opt/tivoli/netcool/omnibus/probes/<arch>/glf_java2.props"
RulesFile :  "/opt/tivoli/netcool/omnibus/probes/<arch>/glf_java2.rules"
MessageLog : "/opt/tivoli/netcool/omnibus/log/glf_java2.log"
PidFile :    "/opt/tivoli/netcool/omnibus/var/Stream2"
```

The following example shows the settings from the properties file of the third probe:

```
Name :       "glf_java3"
PropsFile :  "/opt/tivoli/netcool/omnibus/probes/<arch>/glf_java3.props"
RulesFile :  "/opt/tivoli/netcool/omnibus/probes/<arch>/glf_java3.rules"
MessageLog : "/opt/tivoli/netcool/omnibus/log/glf_java3.log"
PidFile :    "/opt/tivoli/netcool/omnibus/var/Stream3"
```

Properties and command line options

You use properties to specify how the probe interacts with the device. You can override the default values by using the properties file or the command line options.

The following table describes the properties and command line options specific to this probe. For information about default properties and command line options, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide,* (SC23-6373).

Table 6. Properties and command line options

Property name	Command line option	Description
`CleanStart` *string*	`-cleanstart` *string*	Use this property to specify whether the probe runs in cleanstart mode. This property takes the following values: `false`: The probe runs in recovery mode, which means that when the probe starts, it references the recovery file to determine the point within the log file from which to start reading alarms. `true`: The probe runs in non-recovery mode, which means it ignores the recovery file as well as any existing file content and starts reading new events created in the log file since the probe started. For more details see "Modes of operation" on page 6. The default is `false`.

Table 6. Properties and command line options (continued)

Property name	Command line option	Description
`DSLLogConfig` *string*	`-dsllogconfig` *string*	Use this property to specify the log file configuration file that defines the DSL framework Apache Camel context advanced logging for troubleshooting purposes. The default is `""`.
`EncodingStandard` *string*	`-encodingstandard` *string*	Use this property to specify the character set used in the input log file to encode the content to be read. If no value is specified for this property, the probe checks and uses the character set configured on the host. The default is `""`.
`LogFileName` *string*	`-logfilename` *string*	Use this property to specify the path to the log file from which the probe reads alarm data. The default is `""`. **Note :** When a log file is opened using a symbolic link, change detection is performed on the symbolic link itself, not on the content of the linked file. Therefore you should avoid using symbolic links when specifying a value for the **`LogFileName`** property.
`ParserElementDelimiter` *string*	`-parserelementdelimiter` *string*	Use this property to specify the Java regular expression that indicates the delimiter for the individual token in a line of the log file. The default is `" "` (a space character). **Note :** You must use a Java regular expression to specify a value for this property. For example `"[ |\\t]"` would delimit by either a space, pipe character, or a tab.
`ParserIgnoreEmptyFields` *string*	`-parserignoreemptyfields` *string*	Use this property to specify whether the probe discards tokens that contain no content This property takes the following values: `false`: The probe does not discard tokens that contain no content `true`: The probe discards tokens that contain no content The default is `true`. **Note :** Discarded tokens are not visible in the rules file.

Table 6. Properties and command line options (continued)

Property name	**Command line option**	**Description**
`ParserMaxEventSize` *integer*	`-parsermaxeventsize` *integer*	Use this property to specify the maximum buffer size (in bytes) that the probe allows for a single event. If this size is exceeded, the probe writes a warning message to the log file and breaks the event into smaller chunks, which it sends to the ObjectServer. The default is `4098`.
`ParserNVPDelimiter` *string*	`-parsernvpseparator` *string*	Use this property to specify the string that indicates the delimiter that separates the token name from the token value in the name-value pairs of an event attribute. The default is `""`.
`ParserQuoteCharacter` *string*	`-parserquotecharacter` *string*	Use this property to specify the quotation character, (this will either be the single or the double quote). The probe processes directly all event data that appears between quotation characters without attempting to interpret it. This property takes the following values: `"`: The probe treats the double quote mark as the quotation mark. `'`: The probe treats the single quote mark as the quotation mark. By default, no value is specified for this property. **Note :** If you do not specify a value for this property, the probe treats all quotation marks as ordinary literal characters.
`ParserReplaceStripCharWith` *string*	`-parserreplacestripcharwith` *string*	Use this property to specify that character with which the probe replaces the strip character when it is encountered in the data stream. The default is `""`.
`ParserStripCharacter` *string*	`-parserstripcharacter` *string*	Use this property to specify the characters that should be stripped out of the token value. The default is `""`. **Note :** You must use a Java regular expression to specify a value for this property. For example `"[|]"` would strip all pipe characters. To disable this feature, do not specify a value for this property.

Table 6. Properties and command line options (continued)

Property name	Command line option	Description
`RecoveryFile` *string*	`-recoveryfile` *string*	Use this property to specify the name of the recovery file that the probe references when running in recovery mode. For more details see "Modes of operation" on page 6. The default is `/opt/IBM/tivoli/netcool/omnibus/var/glf_java.reco`. **Note :** You must always specify the full path to the recovery file regardless of the mode in which the probe is running.

Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 8.0

All probes can be configured by a combination of generic properties and properties specific to the probe.

The following table describes the properties and command line options that are provided by the Java Probe Integration Library (probe-sdk-java) version 8.0.

Note : Some of the properties listed may not be applicable to your probe.

Table 7. Properties and command line options

Property name	Command line option	Description
`CommandPort` *integer*	`-commandport` *integer*	Use this property to specify the port to which users can Telnet to communicate with the probe using the Command Line Interface (CLI) supplied. The default is `6970`.
`CommandPortLimit` *integer*	`-commandportlimit` *integer*	Use this property to specify the maximum number of Telnet connections that can be made to the probe. The default is `10`.
`DataBackupFile` *string*	`-databackupfile` *string*	Use this property to specify the path to the file that stores data between probe sessions. The default is `""`. **Note :** Specify the path relative to `$OMNIHOME/var`.
`HeartbeatInterval` *integer*	`-heartbeatinterval` *integer*	Use this property to specify the frequency (in seconds) with which the probe checks the status of the host server. The default is `60`.

Table 7. Properties and command line options (continued)

Property name	Command line option	Description
`Inactivity` *integer*	`-inactivity` *integer*	Use this property to specify the length of time (in seconds) that the probe allows the port to receive no incoming data before disconnecting. The default is `0` (which instructs the probe to not disconnect during periods of inactivity).
`InitialResync` *string*	`-initialresync` *string*	Use this property to specify whether the probe requests all active alarms from the host server on startup. This property takes the following values: `false`: The probe does not request resynchronization on startup. `true`: The probe requests resynchronization on startup. For most probes, the default value for this property is `false`. If you are running the JDBC Probe, the default value for the **`InitialResync`** property is `true`. This is because the JDBC Probe only acquires data using the resynchronization process.
`MaxEventQueueSize` *integer*	`-maxeventqueue size`*integer*	Use this property to specify the maximum number of events that can be queued between the non native process and the ObjectServer. The default is `0`. **Note :** You can increase this number to increase the event throughput when a large number of events is generated.
`ResyncInterval` *integer*	`-resyncinterval` *integer*	Use this property to specify the interval (in seconds) at which the probe makes successive resynchronization requests. The default value for this property is `0` (which instructs the probe to not make successive resynchronization requests).
`RetryCount` *integer*	`-retrycount` *integer*	Use this property to specify how many times the probe attempts to retry a connection before shutting down. The default is `0` (which instructs the probe to not retry the connection).

Table 7. Properties and command line options (continued)

Property name	Command line option	Description
RetryInterval *integer*	-retryinterval *integer*	Use this property to specify the length of time (in seconds) that the probe waits between successive connection attempts to the target system. The default is 0 (which instructs the probe to use an exponentially increasing period between successive connection attempts, for example, the probe will wait for 1 second, then 2 seconds, then 4 seconds, and so forth).

Elements

The probe breaks event data down into tokens and parses them into elements. Elements are used to assign values to ObjectServer fields; the field values contain the event details in a form that the ObjectServer understands.

The following table describes the elements that the IBM Tivoli Netcool/OMNIbus Generic Log File Java Probe generates. Not all the elements described are generated for each event; the elements that the probe generates depends upon the event type.

Table 8. Elements

Element name	Element description
$FullDetails	This element contains a concatenation of all $FieldVal[*n*] elements for each event.
$Token[*n*]	The probe derives these elements by parsing the log file. [*n*] is the element number from the beginning of the line. [*n*] starts at 1.
$NVP_*[token_name]*	The probe derives these elements by parsing the log file stream into its constituent name-value pairs, and creates a element for each token in the stream.
$TokenCount	This element contains the total number of all name-value pair tokens generated plus the $FullDetails token.

Error messages

Error messages provide information about problems that occur while running the probe. You can use the information that they contain to resolve such problems.

The following table describes the error messages specific to this probe. For information about generic error messages, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide,*

Table 9. Error messages

Error	Description	Action
`The encoding: character_set is not supported`	The character set specified by the **EncodingStandard** property is not supported by the Java runtime running the probe.	Change the value set by the **EncodingStandard** property.
`Error starting camel context`	The probe processing and routing engine failed to start due to a configuration error.	Check the values set in the configuration file specified by the **EncodingStandard** property.
`java.nio.file. NoSuchFileException: <path> glf_java.reco. glf_java.standard`	The probe could not find the recovery file location.	Set the **RecoveryFile** property to a valid file path.
`File log_filename was deleted`	The log file was deleted by another program while the probe was reading it.	Restore the log file and restart the probe.
`File is truncated or overwritten, thus it needs to be re-opened`	The log file was truncated or overwritten while the probe was reading the file.	Probe will reopen log file.
`Unable to process stream content`	An error occurred during reading of log file.	
`Unable to open file stream`	The probe failed to open log file for reading.	Check the permissions set for the log file.
`Unhandled exception at GLF endpoint`	The probe encountered an error while reading the log file, and does not know how to proceed.	

ProbeWatch messages

During normal operations, the probe generates ProbeWatch messages and sends them to the ObjectServer. These messages tell the ObjectServer how the probe is running.

The following table describes the raw ProbeWatch error messages that the probe generates. For information about generic ProbeWatch messages, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Table 10. ProbeWatch messages

ProbeWatch message	Description	Triggers or causes
`Log file truncated: reopening`	The probe detected that the size of the log file decreased, so it is reopening the file.	The log file has decreased in size. This ProbeWatch message is generated on UNIX operating systems only.

Table 10. ProbeWatch messages (continued)

ProbeWatch message	Description	Triggers or causes
`Log file disappeared: reading operation was aborted`	The probe detected that there is no longer a log file with the configured filename. The probe will shut down if no retries have been specified.	The log file was deleted by another program.
`Corrupted recovery file, proceeding with beginning of log file`	The probe could not read the recovery file or its data was unrecognizable. The probe will read from the beginning of log file.	The recovery file has become corrupted.
`Unable to read or write to recovery file`	The probe failed to write recovery information to the recovery file.	The recovery file has become corrupted.
`Going Down`	The probe is shutting down.	The probe is performing the shutdown routine and shutting down.
`Running ...`	The probe is running normally.	The probe has just started up.

Known issues

At the time of release, there is known issue that you should be aware of when running the probe.

This section covers the following known issue:

- "Shutting down the probe" on page 17
- "Probe does not respond to log files stored on shared drive or mounted disk when appended on another server" on page 17
- "Probe cannot be run with commands that parse an empty parameter on Windows" on page 18
- "Master probe re-sends events sent by the slave probe" on page 18
- "Probe cannot start on RHEL 7.7 or 8.0" on page 18

Shutting down the probe

There is a known issue of potentially corrupting the recovery file if you use **Control-C** or PA to shut down the probe. The current workaround is to use the **shutdown_probe** command to shut down the probe.

The format of the `-data` option for the **shutdown_probe** command is:

```
-data '{"command":"shutdown_probe","params":[]}'
```

Probe does not respond to log files stored on shared drive or mounted disk when appended on another server

The probe does not work if the log file is stored in on a shared directory and is then modified by another server (one on which the probe is not running).

Probe cannot be run with commands that parse an empty parameter on Windows

There is a known Windows limitation whereby a command cannot be parsed if it contains an empty parameter.

If you need to parse an empty string as parameter on the command line on Windows, you must use an escape character. For example, the following command will not work:

```
.\nco_p_glf_java.bat -parserelementdelimiter ''
```

Use instead the following command:

```
.\nco_p_glf_java.bat -parserelementdelimiter \'\'
```

Probe cannot be run with `-recoveryfile ''` in command line on Windows

On Windows, if you set the `-recoveryfile` option to `''` when running the probe, the recovery file function is not disabled.

The workaround is to set the **`RecoveryFile`** property in the `glf_java.props` file to `''`.

Master probe re-sends events sent by the slave probe

When running in peer-to-peer failover mode, the master probe re-sends the events sent by the slave probe when it was down, based on its recovery file pointer.

Probe cannot start on RHEL 7.7 or 8.0

There is a known issue with this probe whereby the probe cannot start on RHEL 7.7 or 8.0.

You will see the following message in the error log:

```
Problem running java using /opt/IBM/tivoli/netcool/platform/linux2x86/jre_1.8.0/jre/bin/java.
Check the binary is accessible and has correct permissions
```

This was fixed in the probe dependency: DSL Factory Framework (`probe-dsl-framework`). To resolve this issue, update your DSL Framework to version 6.0.

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

Trademarks

SC27-8704-01