Edition notice

This edition (SC23-7931-07) applies to version 10.0 of IBM Tivoli Netcool/OMNibus Socket Probe and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC23-7931-06.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
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 Socket Probe documentation is provided in softcopy format only. To obtain the most recent version, please visit the IBM Tivoli Netcool Knowledge Center:


Table 1. Document modification history

<table>
<thead>
<tr>
<th>Document version</th>
<th>Publication date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC23-7931-01</td>
<td>September 25, 2007</td>
<td>First IBM publication.</td>
</tr>
</tbody>
</table>
Patch common-libnetcool-3_2 added.  
Timeout property added. |
IPv6 and FIPS information updated.  
Support for Linux on zSeries added. |
| SC23-7931-04     | October 31, 2009  | Summary table information updated with new patch number and platforms supported. |
Table 1. Document modification history (continued)

<table>
<thead>
<tr>
<th>Document version</th>
<th>Publication date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC23-7931-05</td>
<td>June 4, 2010</td>
<td>Summary table updated. Descriptions for the following properties added to the “Properties and command line options” on page 9: Host, Inactivity, PollInterval, Retry, SocketServerMode. Other property descriptions were updated to reflect support for operating the probe in Socket Client mode. “ProbeWatch messages” on page 22 added. Information about the distinction between Socket mode and Client mode added in “Running the probe in either Server mode or Client mode” on page 3.</td>
</tr>
<tr>
<td>SC23-7931-06</td>
<td>September 30, 2011</td>
<td>Installation section replaced by “Installing probes” on page 2. Updated the following section: “Timeout” on page 7. Updated the Timeout property in the following section: “Properties and command line options” on page 9.</td>
</tr>
</tbody>
</table>
Table 1. Document modification history (continued)

<table>
<thead>
<tr>
<th>Document version</th>
<th>Publication date</th>
<th>Comments</th>
</tr>
</thead>
</table>
| SC23-7931-07     | November 30, 2012| Summary table updated with new package version. Updated “Running the probe in either Server mode or Client mode” on page 3 with the following information:  
• Added ChatinString and ChatoutString to list of properties for Client mode.  
• Added KeepIdleConnection to list of properties for Server mode. Added “Chat in and chat out scripts” on page 5.  
Added the following properties to “Properties and command line options” on page 9  
• ChatinString  
• ChatoutString  
• KeepIdleConnection |

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>
<thead>
<tr>
<th>Operating system</th>
<th>Directory name represented by <code>arch</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX® systems</td>
<td>aix5</td>
</tr>
<tr>
<td>Red Hat Linux and SUSE systems</td>
<td>linux2x86</td>
</tr>
<tr>
<td>Linux for System z</td>
<td>linux2s390</td>
</tr>
<tr>
<td>Solaris systems</td>
<td>solaris2</td>
</tr>
<tr>
<td>Windows systems</td>
<td>win32</td>
</tr>
</tbody>
</table>

### 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`. 

IBM Tivoli Netcool/OMNIbus Socket Probe

The IBM Tivoli Netcool/OMNIbus Socket Probe can be configured to work in either Client mode or Server mode. In Server mode, the probe is a multiheaded probe that acts as a single process concurrent server; the probe can handle a limited number of multiple connections to it through a TCP port. In Client mode, the probe connects to a socket to which a device writes its events.

This guide contains the following sections:

- “Summary”
- “Installing probes” on page 2
- “Properties and command line options” on page 9
- “Data acquisition” on page 3
- “Elements” on page 21
- “Error messages” on page 21
- “Frequently Asked Questions” on page 22

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 summarizes the probe.

**Table 3. Summary**

<table>
<thead>
<tr>
<th>Probe target</th>
<th>Socket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probe executable name</td>
<td>nco_p_socket (UNIX)</td>
</tr>
<tr>
<td></td>
<td>nco_p_socket.exe (Windows operating systems)</td>
</tr>
<tr>
<td>Package version</td>
<td>10.0</td>
</tr>
<tr>
<td>Probe supported on</td>
<td>For details of supported operating systems, see the following Release Notice on the IBM Software Support website:</td>
</tr>
<tr>
<td>Properties file</td>
<td>$OMNIHOME/probes/arch/socket.props</td>
</tr>
<tr>
<td>Rules file</td>
<td>$OMNIHOME/probes/arch/socket.rules</td>
</tr>
<tr>
<td>Requirements</td>
<td>For details of any additional software that this probe requires, refer to the description.txt file that is supplied in its download package.</td>
</tr>
<tr>
<td>Connection method</td>
<td>TCP/IP (acting in client or server mode)</td>
</tr>
<tr>
<td>Remote connectivity</td>
<td>The probe can receive events from a remote system or can connect to a socket to which a device writes its event data.</td>
</tr>
</tbody>
</table>
Table 3. Summary  (continued)

| Licensing | Electronic licensing was deprecated with the release of IBM Tivoli Netcool V7.2.0. All IBM Tivoli Netcool V7.2.0 (and later) products use the IBM software licensing process. |
| Peer-to-peer failover functionality | 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 | For communications between the probe and IBM® Tivoli® Netcool/OMNIbus V7.2, the IPv6 environment is supported on UNIX platforms only. |
| For communications between the probe and IBM Tivoli Netcool/OMNIbus V7.2.1, the IPv6 environment is supported on all platforms. |
| For communications between the probe and the device, the probe supports the IPv6 environment on all platforms except Windows XP and Windows 2003. |
| Federal Information Processing Standards (FIPS) | The IBM Tivoli Netcool/OMNIbus SNMP Probe is compatible with FIPS. |

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:
   

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:
   

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.

Data acquisition

Each probe uses a different method to acquire data. The method that a probe uses is based on the target system from which the probe receives data.

The IBM Tivoli Netcool/OMNibus Socket Probe can be configured to operate in either Server mode or Client mode by setting the SocketServerMode property. By default, the probe operates in Server mode.

Note: When the probe operates in Server mode ensure that the event source information sent to the probe does not contain an asterisk (*).

The mode that you chose determines how the probe acquires data. Data acquisition is described in the following topics:

- “Running the probe in either Server mode or Client mode”
- “Event stream parsing as a single line” on page 4
- “Line parsing with quotation marks” on page 5
- “Line parsing with consecutive unquoted white spaces” on page 5
- “Chat in and chat out scripts” on page 5
- “Name resolution” on page 6
- “Data stream capture” on page 7
- “Event buffering” on page 7
- “Read Timeout” on page 7
- “Timeout” on page 7
- “Peer-to-peer failover functionality” on page 8

Running the probe in either Server mode or Client mode

The probe can be configured to operate in either Server mode or Socket Client mode by setting the SocketServerMode property. By default, the probe operates in Server mode.

In Server mode, the probe operates as a multi-headed server that can accept connections from multiple devices using TCP/IP and receive events sent to it. To run the probe in Server mode, set the SocketServerMode property to 1. In Client mode, the probe connects to the target system and listens on the socket to which the target system writes its event data; the probe receives ASCII data in the form of an element value followed by a carriage return. To run the probe in Client mode, set the SocketServerMode property to 0.

When in operating in Client mode, you can specify that the probe sends a ProbeWatch message to the ObjectServer if it fails receives data from the target system for the period of time specified by the Inactivity property.

Most properties are common to both Server and Client probe modes of operation. However, the way certain properties are set is determined by the mode in which you are running the probe. For example, the ThreadPoolSize property cannot be set when the probe is operating in Client mode.
The following properties are only configurable when the probe is running in Client mode:

- ChatinString
- ChatoutString
- Host
- PortNumber
- Inactivity
- PollInterval
- Retry

The following properties are only configurable when the probe is running in Server mode:

- EventReadString
- FdSetSize
- KeepIdleConnection
- LoginScript
- MaxEvents
- PortNumber
- ReadTimeOut
- ThreadPoolSize
- Timeout

For details of all these properties, see “Properties and command line options” on page 9.

Note: The PortNumber property is used when the probe is running in either Server mode or Client mode. However, the way in which it works depends upon the mode in which the probe is running.

**Event stream parsing as a single line**

The probe parses the event stream treating the character specified by the Delimiter property as the field delimiter instead of generating name-value pairs. The character specified by the EventTerminator property indicates the end of each event.

You can configure the probe to parse the event stream as a single line with tokens separated by token delimiters instead of generating standard name-value pairs. To specify that the probe parses the event stream as a single line, set the SingleLines property to 1 and specify the delimiter that separates tokens using the Delimiter property.

You can also specify how the probe parses lines that contain null fields (that is, adjacent delimiters with no data in between) using the ParseNullFields property.

For example, suppose the SingleLines property is set to 1, the Delimiter property is set to |, and the probe receives the following line:

"Hostname|PortNumber|Summary|||Severity"

This line contains two null fields between Summary and Severity.
If the ParseNullFields property is set to 0, the probe ignores the null fields and so parses the line into the following tokens:
Token000="Hostname",Token001="PortNumber",Token002="Summary",Token003="Severity"

If the ParseNullFields property is set to 1, the probe generates tokens with no values for the null fields and so parses the line into the following tokens:
Token000="Hostname",Token001="PortNumber",Token002="Summary",Token003="",Token004="", Token005="Severity"

Line parsing with quotation marks
If a line contains single or double quotation marks (' or "), respectively, the probe can treat them as either standard characters or special characters. If a line contains both single and double quotation the probe treats the first quotation in that line as a special character only and treats the subsequent quotation as a standard character.

As standard characters, quotation marks have no significant meaning and are parsed as any other character. As special characters, quotation marks indicate that the probe should treat characters falling between them (including any spaces) as a single token. To specify that the probe treats quotation marks as special characters, set the IgnoreQuotes property to 1. Otherwise, set the IgnoreQuotes property to 0.

Note: This property can only be set to 1 if the default values are used for the ParseAsLines and Delimiter properties.

Line parsing with consecutive unquoted white spaces
If a line contains consecutive unquoted white spaces (that is, consecutive tabs or spaces in a line that do not fall within a pair of quotation marks), the probe can either maintain all the white spaces in the line or reduce the consecutive white spaces to a single space.

To specify that the probe maintains consecutive unquoted white spaces, set the PreserveLine property to 1. Otherwise, set the PreserveLine property to 0.

Chat in and chat out scripts
Chat in and chat out scripts control probe login and logout. These scripts are on a single line in the expect-send format (for chat in scripts) or send-expect format (for chat out scripts). You can specify chat in and chat out strings using the ChatinString and ChatoutString properties or the -chatinstring and -chatoutstring command line options.

The format is:
ChatinString : expect send expect send....
ChatoutString : send expect send expect....

Note: Each element in the chat strings is separated by white space. In order to send or expect a sequence that includes white space, surround the sequence with single quotes.

A typical chat in script might be:
.*login.*:.*anu\r\n.*password.*:.*anu\r\n
The expect text can use any regular expression, while the send text can send any characters, including control characters using the standard UNIX/C escape
sequences described in "Escape codes."

**Escape codes**

You can use C-style escape codes in the `ChatinString` and `ChatoutString` properties. This allows you to easily define whether to send escape code sequences after commands.

For example, the following chatin string sends a carriage return character (\r) after the user name and password:

`ChatinString : ".*: user\r.*: passwd\r.*:"

The following table explains the character sequences that are recognized.

**Table 4. ChatinString escape codes**

<table>
<thead>
<tr>
<th>Escape code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>\b</td>
<td>This escape code specifies the backspace character.</td>
</tr>
<tr>
<td>\f</td>
<td>This escape code specifies the form-feed character.</td>
</tr>
<tr>
<td>\n</td>
<td>This escape code specifies the new-line character.</td>
</tr>
<tr>
<td>\r</td>
<td>This escape code specifies the carriage return character.</td>
</tr>
<tr>
<td>\t</td>
<td>This escape code specifies the tab character.</td>
</tr>
<tr>
<td>\</td>
<td>This escape code specifies the backslash character.</td>
</tr>
<tr>
<td>'</td>
<td>This escape code specifies the single quote character.</td>
</tr>
<tr>
<td>&quot;</td>
<td>This escape code specifies the double quote character.</td>
</tr>
</tbody>
</table>

**Note:** Due to the way in which the above properties are parsed, the escape sequences for backslash, single quote and double quote must be double-escaped. For example, to send a backslash character (\), use `\\`.

**Name resolution**

If you are using a naming service, such as Domain Name System (DNS) or Network Information Service (NIS), the IP address of each host can be written in either dotted quad format (for example, 193.131.98.3) or can be assigned a human-readable host name.

By default, the probe resolves all dotted quad IP addresses before sending events to the ObjectServer. If you set the `NoNameResolution` property to 1, the probe sends events to the ObjectServer without resolving the host name first. This option can improve the performance of the probe.
**Data stream capture**

The probe can capture the data stream sent from a device. This data is stored in a log file and can be used for debugging purposes, to develop new features for the probe, or to pass to other management systems that require the same data. To enable stream capture, set the `StreamCapture` property to 1 and specify a log file to store the data using the `StreamCaptureFilename` property.

The probe creates a stream capture file with the following naming convention:

```
streamcapturefile name.date_time
```

By appending the filename with a timestamp, the probe avoids overwriting the old stream capture file.

**Note:** The data stream capture function generates a lot of data and should be used with caution. When you no longer require data for debugging, set the `StreamCapture` property to 0 to disable the stream capture function.

**Event buffering**

Buffering is a standard feature in all probes. The probe buffers events and sends them to the ObjectServer in a batch when the buffer reaches a predetermined size. To specify that the probe uses buffering, set the `Buffering` property to 1. To specify the size of the buffer, use the `BufferSize` property.

The probe allows you to specify the frequency with which the probe flushes the events to the ObjectServer. This overrides the buffer size specification. To specify the frequency with which the probe flushes events to the ObjectServer, use the `BufferTimer` property. The probe flushes the events to the ObjectServer with this frequency regardless of how much of the buffer has been used.

**Read Timeout**

The `ReadTimeout` property specifies how long the probe waits to read alarm data before timing out. Each time the probe attempts to read an alarm, this is the allotted time that it waits to receive data. If nothing is received, the probe moves on to the next alarm.

**Timeout**

The probe has a timeout facility that allows it to close idle connections once the limit specified using the `Timeout` property is reached.
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"
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 more information about generic Netcool/OMNibus properties and command line options, see the IBM Tivoli Netcool/OMNibus Probe and Gateway Guide.

<table>
<thead>
<tr>
<th>Property name</th>
<th>Command line option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BufferTimer integer</td>
<td>-buffertimer integer</td>
<td>Use this property to specify the time (in seconds) that the probe waits before flushing the events in the buffer to the ObjectServer. The default is 60. Note: The probe uses this property only when the Buffering property is set to 1. The minimum values for this property is 10. If you set the BufferTimer property to a value less then 10, the probe resets this value to 10 automatically.</td>
</tr>
<tr>
<td>ChatinString string</td>
<td>-chatinstring string</td>
<td>Use this property to specify the chat in script for connection to the host system. The default is &quot;&quot;. Note: The probe uses this property only when running in Client mode (that is, when the SocketServerMode property is set to 0).</td>
</tr>
<tr>
<td>ChatoutString string</td>
<td>-chatoutstring string</td>
<td>Use this property to specify the chat out script for disconnection from the host system. The default is &quot;&quot;. Note: The probe uses this property only when running in Client mode (that is, when the SocketServerMode property is set to 0).</td>
</tr>
</tbody>
</table>
Table 5. Properties and command line options (continued)

<table>
<thead>
<tr>
<th>Property name</th>
<th>Command line option</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Delimiter** string| -delimiter string   | Use this property to specify the characters that indicate token delimiters. When set, the probe parses the event stream treating the specified characters as the field delimiter instead of generating standard name-value pairs.  
The default is "".  
**Note:** This property is used in conjunction with the **SingleLines** and **ParseNullFields** properties. For details, see ["Event stream parsing as a single line"](on page 4). |
| **EventReadString** string | -eventreadstring string | Use this property to specify the string the probe sends to the host to indicate that it has received an event. This string can act as an acknowledgment that prompts the host to send the next alarm.  
The default is "".  
**Note:** The probe uses this property only when running in Server mode (that is, when the **SocketServerMode** property is set to 1). |
| **EventTerminator** string | -eventterminator string | Use this property to specify the string that indicates the end of an event.  
The default is \n\n. |
| **FdSetSize** integer | -fdsetsize integer | Use this property to specify the maximum number of file descriptors that the probe allows to be opened. This limits the number of client connections.  
The default is 1024.  
**Note:** This property is invalid for the Windows version of the probe.  
The probe uses this property only when running in Server mode (that is, when the **SocketServerMode** property is set to 1). |
Table 5. Properties and command line options  (continued)

<table>
<thead>
<tr>
<th>Property name</th>
<th>Command line option</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Footer** string | -footer string | Use this property to specify the regular expression that indicates the footer of an event.  
The default is "".  
**Note:** Specify a value for both the Footer and Header properties so that the probe can send events to the ObjectServer. |
| **Header** string | -header string | Use this property to specify the regular expression that indicates the header of an event.  
The default is "".  
**Note:** Specify a value for both the Footer and Header properties so that the probe can send events to the ObjectServer. |
| **Host** string | -host string | Use this property to specify the name of the host to which the probe connects when running in Client mode.  
The default is localhost.  
**Note:** The probe uses this property only when running in Client mode (that is, when the SocketServerMode property is set to 0). |
| **IgnoreQuotes** integer | -ignorequotes integer | Use this property to specify how the probe handles quotation marks. This property takes the following values:  
0: The probe treats quotation marks as standard characters.  
1: The probe treats quotation marks as special characters.  
The default is 1.  
**Note:** For details about using this property, see "Line parsing with quotation marks" on page 5. |
<table>
<thead>
<tr>
<th>Property name</th>
<th>Command line option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inactivity</strong></td>
<td>-inactivity integer</td>
<td>Use this property to specify the time (in seconds) that a connection can stay idle before the probe sends an inactivity ProbeWatch message to the ObjectServer. You must set this property to a positive number greater than 0. If you specify a negative value or 0, the probe shuts down. If this period is exceeded, the probe sends the following ProbeWatch message to the ObjectServer: Inactivity alert: Idle Connection. The default is 20. <strong>Note</strong>: The probe uses this property only when running in Client mode (that is, when the <code>SocketServerMode</code> property is set to 0).</td>
</tr>
<tr>
<td><strong>KeepIdleConnection</strong></td>
<td>-keepidleconnection integer</td>
<td>Use this property to specify whether the probe retains or discards idle connections when the maximum number of file descriptors, specified in the <code>FdSetSize</code> property, has not been reached. This property takes the following values: 0: The probe always discards an idle connection. 1: The probe retains idle connections as long as the maximum number of file descriptors in use has not been reached. The default is 0. <strong>Note</strong>: The probe uses this property only when it is running in Server mode (that is, when the <code>SocketServerMode</code> property is set to 1).</td>
</tr>
<tr>
<td><strong>LineTerminator</strong></td>
<td>-lineterminator string</td>
<td>Use this property to indicate the end of a line within the data stream. All data that the probe encounters prior to receiving this code is considered to be contained within a single line. The default is \n.</td>
</tr>
<tr>
<td>Property name</td>
<td>Command line option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>LoginScript</strong> string</td>
<td>There is no command line equivalent for this property.</td>
<td>Use this property to control the prompt that the probe sends to any client that connects to it, and the expected reply from the client. The prompt you specify must consist of two words separated by a space. The first word is the prompt the probe sends to the client and the second word is the expected reply from the client. If the reply from the client matches the reply specified by this property the probe continues processing events, otherwise the probe disconnects from the client. The default is &quot;&quot;. <strong>Note:</strong> The probe uses this property only when running in Server mode (that is, when the <code>SocketServerMode</code> property is set to 1).</td>
</tr>
<tr>
<td><strong>MaxEvents</strong> integer</td>
<td><code>-maxevents integer</code></td>
<td>Use this property to specify the maximum number of events the probe attempts to read simultaneously from a given source. The default is 10. <strong>Note:</strong> You can increase this number to increase the event throughput when a large number of events is generated.</td>
</tr>
<tr>
<td><strong>MaxEventSize</strong> integer</td>
<td><code>-maxeventsize integer</code></td>
<td>Use this property to specify the maximum size (in bytes) of an event that the probe can read. The default is 8142.</td>
</tr>
<tr>
<td>Property name</td>
<td>Command line option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **NoNameResolution** integer | -nonameresolution integer | The way this property functions depends on the mode in which the probe is operating. When the probe is running in Server mode, use this property to specify whether the probe performs name resolution on IP addresses. This property takes the following values:  
   0: The probe performs name resolution.  
   1: The probe does not perform name resolution.  
   The default is 0.  

When running the probe in Client mode, use this property to specify whether the probe performs name resolution, but only if the **Host** property is configured as an IP address. If the **Host** property is configured as a host name, it sets the fully qualified domain name accordingly.  

This property takes the following values:  
   0: The probe does not perform name resolution.  
   1: The probe performs name resolution.  
   The default is 0.  |
| **NoTrim** integer | -notrim integer | Use this property to specify whether the probe strips leading and trailing white spaces and new lines from the elements that it generates. This property takes the following values:  
   0: The probe strips these items from the elements that it generates.  
   1: The probe does not strip these items from the elements that it generates.  
   The default is 0.  |
Table 5. Properties and command line options (continued)

<table>
<thead>
<tr>
<th>Property name</th>
<th>Command line option</th>
<th>Description</th>
</tr>
</thead>
</table>
| `ParseAsLines integer` | `-parseaslines integer` | Use this property to specify whether the probe parses all entries between the header and the footer one line at a time as individual elements. This property takes the following values:  
0: The probe does not parse the entries one line at a time.  
1: The probe parses the entries one line at a time.  
The default is 0.  
**Note:**  
You can use this field only if you have specified values for the *Header* and *Footer* properties.  
The probe uses this property only when running in Server mode (that is, when the *SocketServerMode* property is set to 1). |
| `ParseNullFields integer` | `-parsenullfields integer` | Use this property to specify how the probe parses lines that contain null fields (adjacent delimiters with no data in between). This property takes the following values:  
0: The probe only generates tokens for fields that contain information.  
1: The probe generates tokens for all fields including null fields.  
The default is 0.  
**Note:** This property is used in conjunction with the *SingleLines* and *Delimiter* properties. For details, see “Event stream parsing as a single line” on page 4. |
Table 5. Properties and command line options (continued)

<table>
<thead>
<tr>
<th>Property name</th>
<th>Command line option</th>
<th>Description</th>
</tr>
</thead>
</table>
| PollInterval  | -pollinterval integer | Use this property to specify the time (in seconds) to attempt connection before sending a Connection Failure ProbeWatch message to the ObjectServer.  
You must set this property to a positive number greater than 0. If you specify a negative value or 0 the probe shuts down.  
Connection failure is detected when the probe detects possible errors with the connection. If a connection failure occurs, the probe sends the following ProbeWatch message to the ObjectServer: Connection failure alert: Lost Connection.  
If Retry is enabled, the probe attempts to reconnect with the frequency specified by the PollInterval property and re-sends the ProbeWatch message if the connection failure recurs.  
The default is 5.  
**Note:** The probe uses this property only when running in Client mode (that is, when the SocketServerMode property is set to 0). |
### Table 5. Properties and command line options (continued)

<table>
<thead>
<tr>
<th>Property name</th>
<th>Command line option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PortNumber</strong></td>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td><strong>PreserveLine</strong></td>
<td>integer</td>
<td>integer</td>
</tr>
</tbody>
</table>

IBM Tivoli Netcool/OMNibus Socket Probe 17
### Table 5. Properties and command line options (continued)

<table>
<thead>
<tr>
<th>Property name</th>
<th>Command line option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ReadTimeOut</strong> integer</td>
<td><code>-readtimeout integer</code></td>
<td>Use this property to specify the number of milliseconds that the probe allows a socket to be silent before releasing the socket servicing the thread back to the thread pool. The default is 1. <strong>Note:</strong> The probe interprets values less than or equal to 10 as seconds. The probe uses this property only when running in Server mode (that is, when the <strong>SocketServerMode</strong> property is set to 1).</td>
</tr>
<tr>
<td><strong>ReportStatus</strong> integer</td>
<td><code>-reportstatus integer</code></td>
<td>Use this property to specify whether the probe sends a ProbeWatch message to the ObjectServer when clients connect to or disconnect from the socket. This property takes the following values: 0: The probe does not send ProbeWatch messages. 1: The probe sends ProbeWatch messages. The default is 1. <strong>Note:</strong> Using the command line option <code>-reportstatus</code> (This is equivalent to <strong>ReportStatus</strong> with a value of 1; omitting this command line option is equivalent to <strong>ReportStatus</strong> with a value of 0.)</td>
</tr>
<tr>
<td><strong>Retry</strong> integer</td>
<td><code>-retry integer</code></td>
<td>Use this property to specify whether the probe attempts to reconnect upon disconnection. When set to 1, the probe attempts to reconnect indefinitely with the frequency specified by the <strong>PollInterval</strong> property. When set to 0, the probe does not attempt to reconnect and shuts down. The default is 1. <strong>Note:</strong> The probe uses this property only when running in Client mode (that is, when the <strong>SocketServerMode</strong> property is set to 0).</td>
</tr>
<tr>
<td>Property name</td>
<td>Command line option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SingleLines</code></td>
<td><code>-singlelines integer</code></td>
<td>Use this property to specify whether the probe parses each line as a single event. This property takes the following values: 0: The probe does not parse each line as a single event. 1: The probe parses each line as a single event. The default is 0. <strong>Note:</strong> Set this property to 1 if the event data consists of a single line. It is used in conjunction with the <code>Delimiter</code> and <code>ParseNullFields</code> properties. For details, see “Event stream parsing as a single line” on page 4.</td>
</tr>
<tr>
<td><code>SocketBuffer</code></td>
<td><code>-socketbuffer integer</code></td>
<td>Use this property to specify the maximum buffer size for each socket. The default is 1024. <strong>Note:</strong> You can increase this number if the probe needs to read a large number of events quickly.</td>
</tr>
<tr>
<td><code>SocketServerMode</code></td>
<td><code>-socketservermode integer</code></td>
<td>Use this property to specify whether the probe runs in server or client mode. This property takes the following values: 0: The probe operates in Client probe mode. 1: The probe operates in Server probe mode. The default is 1.</td>
</tr>
<tr>
<td><code>StreamCapture</code></td>
<td><code>-streamcapture integer</code></td>
<td>Use this property to specify whether the stream capture feature is enabled for debugging purposes. This property takes the following values: 0: This disables the stream capture feature. 1: This enables the stream capture feature. The default is 0. <strong>Note:</strong> If you set this property to 1, specify the file in which to store the data using the <code>StreamCaptureFilename</code> property. When you no longer require data for debugging, set the <code>StreamCapture</code> property to 0 to disable the stream capture function.</td>
</tr>
</tbody>
</table>
Table 5. Properties and command line options (continued)

<table>
<thead>
<tr>
<th>Property name</th>
<th>Command line option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StreamCaptureFilename</td>
<td>-streamcapturefilename</td>
<td>Use this property to specify the file that the probe uses to store the input data stream. The default is $OMNIHOME/var/socket.stream. Note: This property is required if you set the StreamCapture property to 1.</td>
</tr>
<tr>
<td>StripChars</td>
<td>-stripchars string</td>
<td>Use this property to indicate characters that should be stripped out of the data stream to remove carriage returns. For example, if the probe accepts events from both UNIX and Windows NT sources, set this property to \r. The default is &quot;&quot;.</td>
</tr>
<tr>
<td>ThreadPoolSize</td>
<td>-threadpoolsz integer</td>
<td>Use this property to specify the number of socket servicing threads allowed in the thread pool. The default is 10. Note: The probe uses this property only when running in Server mode (that is, when the SocketServerMode property is set to 1).</td>
</tr>
<tr>
<td>Timeout</td>
<td>-timeout integer</td>
<td>Use this property to specify the time (in seconds) that the probe allows connections to remain idle before closing the connection. The default is 0 (probe does not disconnect the idle connections). Note: The probe uses this property when running in Server mode (that is, when the SocketServerMode property is set to 1).</td>
</tr>
</tbody>
</table>
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 probe generates. Not all the elements described are generated for each event; the elements that the probe generates depend on the event type.

Table 6. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Element description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FQDN</td>
<td>This element contains the fully qualified domain name (FQDN) of the client.</td>
</tr>
<tr>
<td>$HOST</td>
<td>This element contains the IP address of the client.</td>
</tr>
</tbody>
</table>

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 Netcool/OMNIbus error messages, see the IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide.

Table 7. Error messages

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error adding socket node to List</td>
<td>The probe has insufficient memory.</td>
<td>Make more memory available.</td>
</tr>
<tr>
<td>Error adding list element to node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error allocating memory for buffer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error allocating memory for ProbeWatchString</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error allocating memory for SockDataList</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error in parsing and additional lines added when there is * in the events sent (Server Mode)</td>
<td>The probe received an asterisk (*) in the event source information.</td>
<td>Configure the target system to send event source information that does not contain the asterisk (*) character.</td>
</tr>
</tbody>
</table>
## 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, (SC14-7530).

**Table 8. ProbeWatch messages**

<table>
<thead>
<tr>
<th>ProbeWatch message</th>
<th>Description</th>
<th>Triggers/causes</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Events received for inactivity seconds</td>
<td>The probe did not receive events for the specified period.</td>
<td>The probe has exceeded the time out period already set.</td>
</tr>
<tr>
<td>Connection failure alert: Lost Connection</td>
<td>The probe failed to connect to the target system in the specified period.</td>
<td>The remote server might be unavailable because of a network outage. Check the log files for more information.</td>
</tr>
<tr>
<td>Inactivity alert: Idle Connection</td>
<td>The probe has not read data for the specified period.</td>
<td>The probe has exceeded the time specified by the Inactivity property.</td>
</tr>
</tbody>
</table>

## Frequently Asked Questions

Various questions arise as users work with the probe. Answers to these questions are provided for your reference.

### Property settings

The following questions concern property settings.

**What does the MaxEvents property do?**

The **MaxEvents** property controls how many events the probe reads from a single connection before relinquishing control of the thread to another socket connection. You should not set this value too low or too high as it can degrade the performance of the probe. The property should be set to the number of events reasonably expected to be read from any one connection. In most cases, this value should be in the order of hundreds.

The **MaxEvents** property should be used in conjunction with the **ReadTimeOut** property. If the full number of events cannot be read from a connection, the probe waits for the time specified by the **ReadTimeOut** property before moving on to the next connection regardless of whether the probe has read the maximum number of events.

**Note:** Both **MaxEvents** and **ReadTimeOut** properties are not supported in Socket Client mode.

**What does the ReadTimeOut property do?**

This property controls the timeout value for a socket. For backward compatibility, the probe interprets values greater than 10 as milliseconds, and values less than 10 as seconds.
as seconds. For performance reasons, this value should either be less than or equal to 10, or greater than 300.

**What does the ThreadPoolSize property do?**

This property controls how many socket servicing threads exist in the thread pool. The more threads you have the less likely it is that a socket will not be serviced immediately. However, the more threads you have, the more resources the probe consumes.

**What are the ParseAsLines and Delimiter properties for?**

The lines that are not part of the event demarcation, can be parsed in two ways:

- If the **ParseAsLines** property is used, each line defined by the **LineTerminator** property is tokenized.
- If the **Delimiter** property is set to ; (semicolon), the probe expects the contents to be in the format Name1; Value1; Name2; Value2; . If it is set to a value of =; (equals sign followed by a semicolon), the probe expects the event contents to be in the format Name1=Value1; Name2=Value2; .

**What are the Headers/Footer and EventTerminator properties for?**

The probe needs a mechanism to demarcate alarms. This is implemented in two ways: An event terminator demarcates the events or a regular expression match is done for headers and footers. The probe will use the latter approach only if both the **Header** and **Footer** properties are set.
Appendix. Notices and Trademarks

This appendix contains the following sections:

• Notices
• Trademarks

Notices

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

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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

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

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

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

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

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

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

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

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

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

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

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

### Trademarks

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

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

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

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

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

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

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