

IBM® Tivoli® Netcool/OMNIbus Probe
Integration for Juniper Contrail Alerts
Version 4.1

Reference Guide
January 31, 2020



Note

Before using this information and the product it supports, read the information in [Appendix A, “Notices and Trademarks,”](#) on page 25.

Edition notice

This edition (SC28-3114-00) applies to version 11.0 of IBM Tivoli Netcool/OMNIBus Probe for Message Bus and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2020.**

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

Contents

About this guide.....	v
Document control page.....	v
Conventions used in this guide.....	v
 Chapter 1. Probe Integration for Juniper Contrail Alerts.....	1
Summary.....	1
Installing probes.....	2
Configuring the Probe for Message Bus to integrate with Juniper Contrail Alerts.....	3
Adding triggers for event deduplication.....	6
Running the probe.....	12
Additional properties for the probe integration.....	12
Properties and command line options.....	12
Properties and command line options provided by the Java Probe Integration Library (probe- sdk-java) version 11.0.....	18
Elements.....	21
Error messages.....	22
ProbeWatch messages.....	23
 Appendix A. Notices and Trademarks.....	25
Notices.....	25
Trademarks.....	26

About this guide

The following sections contain important information about using this guide.

Document control page

Table 1. Document modification history		
Document version	Publication date	Comments
SC28-3114-00	January 31, 2020	First IBM publication.

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.

<i>Table 2. Directory names for the arch variable</i>	
Operating system	Directory name represented by <i>arch</i>
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. Probe Integration for Juniper Contrail Alerts

Juniper Networks Contrail provides network visualization and service automation for federated networks. It can interact with other systems using northbound APIs. The Message Bus Probe integrates with Juniper Networks Contrail Alert API to monitor contrail alarms.

This guide contains the following sections:

- [“Summary” on page 1](#)
- [“Configuring the Probe for Message Bus to integrate with Juniper Contrail Alerts” on page 3](#)
- [“Installing probes” on page 2](#)
- [“Running the probe” on page 12](#)
- [“Additional properties for the probe integration” on page 12](#)
- [“Elements” on page 21](#)
- [“Error messages” on page 22](#)
- [“ProbeWatch messages” on page 23](#)

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 integration.

Probe Integration for Nokia NSP.

Table 3. Summary	
Probe target	Juniper Contrail Alert API
Probe executable name	nco_p_message_bus
Installation package	omnibus_arch_probe_nco_p_message_bus_version
Package version	11.0
Probe supported on	For details of supported operating systems, see the following Release Notice on the IBM Software Support website: http://www-01.ibm.com/support/docview.wss?uid=swg21970413
Properties files	The probe is supplied with the following properties file installed in the \$OMNIHOME/probes/arch directory: message_bus_contrail.props
Rules file	The probe is supplied with the following rules file installed in the \$OMNIHOME/probes/arch directory: message_bus_contrail.rules

Table 3. Summary (continued)	
Transport properties files	The probe supports the following transport configuration files installed in the \$OMNIHOME/java/conf/ directory: <code>eventSourceTransport.properties</code>
Transport type	EventSource This supports the <code>eventSourceTransport.properties</code> file.
Requirements	For details of any additional software that this probe requires, refer to the README file that is supplied in its download package.
Connection method	REST/API, Server-Sent Events (SSE)
Multicultural support	Available For information about configuring multicultural support, including language options, see the <i>IBM Tivoli Netcool/OMNIBus Installation and Deployment Guide</i> .
Peer-to-peer failover functionality	Available
IP environment	IPv4 and IPv6 For communications between the probe event source, the probe supports the IPv6 environment on all operating systems except Windows XP and Windows 2003.

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*.

Configuring the Probe for Message Bus to integrate with Juniper Contrail Alerts

The Message Bus Probe can be used to integrate with Juniper Contrail Alerts using the EventSource transport and the HTTPS REST transport. In the EventSource transport, there is an additional multi channel HTTPS REST transport component that can be enabled for Juniper Contrail. The probe supports HTTP REST and HTTPS REST for integrating with Juniper Contrail to stream live alarms with or without using an access token.

The following configuration files are supplied with the probe and are required by the integration with Juniper Contrail Alerts:

- message_bus_contrail.props
- message_bus_contrail.rules
- message_bus_contrail.lookup
- message_bus_contrail_parser.json
- contrailMultiChannelHttpTransport.json
- contrail-param-token-request.json
- eventSourceTransport.properties

To integrate with Juniper Contrail Alerts, use the following steps:

1. Edit the following line in the message_bus_contrail.rules file to include the correct path to the message_bus_contrail.lookup file.

```
include "/opt/IBM/tivoli/netcool/omnibus/probes/linuxx86/message_bus_contrail.lookup"
```

2. Configure **message_bus_juniper_contrail.props** for Juniper Contrail Alerts.

The Probe integration for Juniper Contrail Alerts uses the Probe for Message Bus configured by the message_bus_juniper_contrail.props file supplied with the probe. The following default probe properties are provided for the integration with Juniper Contrail Alerts. The EventSource transport is selected as the main transport type for this integration.

```
#=====
# SETTING PROBE LOGS, PROPS, RULES
#=====
Server : 'NCOMS'
Manager : 'JuniperContrailSSE'
MessageLog : '$OMNIHOME/log/message_bus_juniper_contrail_sse.log'
PropsFile : '$OMNIHOME/probes/message_bus_contrail.props'
RulesFile : '$OMNIHOME/probes/message_bus_contrail.rules'
#=====
# SETTING TRANSPORT TYPE
#=====
TransportType : 'EventSource'
TransportFile : '$OMNIHOME/java/conf/eventsourceTransport.properties'
#=====
# SETTING PARSER CONFIGURATIONS. (SUPPORTS JSON OR XML)
#=====
# FOR PARSING JSON DATA
TransformerFile : '$OMNIHOME/probes/contrailMultiChannelHttpTransport.json'
MessagePayload : 'JSON'

#=====
# SETTING CREDENTIALS FOR ACQUIRING ACCESS TOKEN
#=====
Username : 'contrail_username'
Password : 'contrail_password'

#=====
# SETTING SSL & KEYSTORE
#=====
EnableSSL : 'true'
KeyStore : '/keystore.jks'
KeyStorePassword : 'keystore_password'
```

```

#=====
# SETTING FOR CONNECTION RETRY
#=====
RetryCount: 5
RetryInterval : 30

```

3. Configure **eventSourceTransport.properties** for the Juniper Contrail Alerts integration:

```
httpConnectionPropertiesFile=$OMNIHOME/java/conf/contrailMultiChannelHttpTransport.json
```

Note: If **httpConnectionPropertiesFile** is set to an empty string, it will not enable the support for HTTP REST communication which is required in the Juniper Contrail Alerts integration. You must ensure that this property is properly configured.

4. Configure **contrailMultiChannelHttpTransport.json** for settings on REST requests. Default settings have been provided.

Alarm streaming with access token:

- Access token request takes place in the operation set for GET_ACCESS_TOKEN.
- The HTTP REST body for access token request should be specified in either content or content_file.
- The field content has precedence over content_file: A configured “content” will be used in the HTTP REST body.
- A configured content_file must point to a valid file whose content will be flattened to a single line as the body in HTTP REST.

```

{
  "GLOBAL":
  {
    "httpVersion": "1.1",
    "httpHeaders": "",
    "responseTimeout": "10",
    "securityProtocol": "TLSv1.2",
    "keepTokens": "X-Subject-Token",
    "tokenEndpointURI": "",
    "autoReconnect": "OFF",
    "gatherSubsTopicInfo": "false"
  },
  "LOGIN":
  {
    "GET_ACCESS_TOKEN":
    {
      "uri": "http://<HOST>:<PORT>/v3/auth/tokens",
      "method": "POST",
      "headers": "Accept=application/json,Content-Type=application/json,User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
      "content": "",
      "contentFile": "<PATH>/param-token-request.json",
      "interval": "0",
      "attempts": "0",
      "requireSSL": "false",
      "asEventStream": "false"
    }
  },
  "SUBSCRIBE":
  {
    "GET_SUBSCRIPTION":
    {
      "uri": "http://<HOST>:<PORT>/analytics/uve-stream?tablefilt=control-node",
      "method": "GET",
      "headers": "Connection=Keep-Alive,X-Auth-Token=++X-Subject-Token++,Accept=text/event-stream,Use-Cookie=true,User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
      "content": "",
      "contentFile": "",
      "interval": "0",
      "attempts": "0",
      "requireSSL": "false",
      "asEventStream": "true"
    }
  },
  "RESYNC":
  {
  },
  "LOGOUT":
  {
  }
}

```

REST body for access token:

The probe will replace ++Username++ and ++Password++ tokens in the json messages with the configured value of probe properties: **Username** and **Password**.

```
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "++Username++",
          "password": "++Password++",
          "domain": {
            "id": "default",
            "name": "Default"
          }
        }
      }
    },
    "scope": {
      "project": {
        "name": "admin",
        "domain": {
          "id": "default",
          "name": "Default"
        }
      }
    }
  }
}
```

Alarm streaming without access token:

Note: LOGIN is empty as there is no access token request required.

```
{
  "GLOBAL": {
    {
      "httpVersion": "1.1",
      "httpHeaders": "",
      "responseTimeout": "10",
      "securityProtocol": "TLSv1.2",
      "keepTokens": "X-Subject-Token",
      "tokenEndpointURI": "",
      "autoReconnect": "OFF",
      "gatherSubsTopicInfo": "false"
    },
    "LOGIN": {
      {
      },
    },
    "SUBSCRIBE": {
      {
        "GET_SUBSCRIPTION": {
          {
            "uri": "<HOST>:<PORT>/analytics/uve-stream?tablefilt=control-node",
            "method": "GET",
            "headers": "Connection=Keep-Alive,X-Auth-Token=++X-Subject-Token++,Accept=text/event-stream,Use-Cookie=true,User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
            "content": "",
            "contentType": "",
            "interval": "0",
            "attempts": "0",
            "requireSSL": "false",
            "asEventStream": "true"
          }
        },
      },
      "RESYNC": {
        {
        },
      },
      "LOGOUT": {
        {
        },
      },
    },
  }
}
```

Adding triggers for event deduplication

Processing Juniper Contrail UVEAlarms requires probe rules, custom tables and triggers specific for contrail data.

Configuration files

The triggers required for event deduplication are set up in the following files:

Table 4. Files for setting up deduplication	
File	Purpose
message_bus_contrail.rules	The probe rules for Contrail Stream-sent-event (SSE) UVEAlarms.
message_bus_contrail.lookup	The lookup file for Contrail Stream-sent-event (SSE) UVEAlarms.
create_agg_dedup_trigger.sql	This file performs the following tasks: • Creates default_triggers.agg_deduplication_contrail. • Disables default_triggers.deduplication.
create_contrail_tables.sql	This file performs the following tasks: • Creates new columns in the alerts.status table. • Creates the nc_triggers.monitor table. • Creates the custom.Contrail_problem_events table.
create_contrail_triggers.sql	This file performs the following tasks: • Creates Contrail_triggers.custom_contrail_generic_clear. • Creates Contrail_trigger.dedup_contrail_problem_events.

The SQL files create the following new objects in the ObjectServer.

Table 5. Objects created by the SQL files		
Type	Name	Purpose
Column	The following columns are created in the alerts.status table: ClearTally, AckTime, ClearTime, ClearedBy, RTTSsev, SuppressEscl, Int2, Int9, Int11, TTVariable, TTFlag, TTOriginatorUser, TTReferredGroup	These customized columns are used for holding values.

Table 5. Objects created by the SQL files (continued)		
Type	Name	Purpose
Table	nc_triggers.monitor	This table holds the name of the monitor trigger that is activated in the deployment.
Table	custom.Contrail_problem_events	This table holds the entry of a JuniperContrail node. Row insertion to this table is executed in the agg_contrail_deduplication trigger.
Trigger	default_triggers.agg_contrail_deduplication	This trigger updates a JuniperContrail alarm and inserts a new Node row record in the custom.Contrail_problem_events table. See the Contrail_trigger.dedup_contrail_problem_events trigger for the update operation on the custom.Contrail_problem_events table.
Trigger	Contrail_triggers.custom_contrail_generic_clear	This trigger periodically performs event clearing on the alarms that are incidentally not in the latest JuniperContrail update. To clear the JuniperContrail alarms in the alerts.status table, the inference logic is looking for alarms from the same Node having AlertGroup of the value not in the aggregated URL of the custom.Contrail_problem_events table for the Node.
Trigger	Contrail_trigger.dedup_contrail_problem_events	This trigger updates the record of the JuniperContrail node's alarms, prepping the aggregated URL list for next event clearing operation.

When the Message Bus Probe is no longer needed in the environment, run the following files in conjunction with the probe uninstallation:

Table 6. Files to remove triggers	
File	Purpose
remove_contrail_agg_dedup.sql	This file performs the following tasks: Removes the default_triggers.agg_deduplication_contrail trigger. Enables the default_triggers.deduplication trigger.
remove_contrail_tables.sql	This file performs the following task: Removes the objects created by the create_contrail_tables.sql file.

Table 6. Files to remove triggers (continued)	
File	Purpose
remove_contrail_triggers.sql	This file performs the following task: Removes the object created by the create_contrail_triggers.sql file.

Juniper Contrail UVEAlarms alarm

Alarms of type UVEAlarms are for probe consumption.

The UVEAlarms json has the following structure:

```
{
  "type": "UVEAlarms",
  "value": {
    "alarms": [ { alarm_1 }, { alarm_2 }, { alarm_3 } ],
    "___T": 1574156963194879
  },
  "key": "ObjectVRouter: dcr01-contrail-dpdk-13"
}
```

A UVEAlarms json message contains a list of alarms of a node in the array field named alarms. Each alarm reports the state change of a threshold. The probe pushes each alarm to the ObjectServer to become a Netcool event.

Member alarms in alarm list (identifiable by threshold name) may be constant or changing in every UVEAlarms json update. In the case of a reduced alarm list, for a threshold alarm previously reported but currently unavailable in the UVEAlarms json, its absence is an indication of the alarm concerned having been removed or resolved in the Contrail system.

In this case the alarm has the following structure:

```
{
  "severity": 2,
  "alarm_rules": { ... }, => Containing more data.
  "timestamp": 1577837594000011,
  "ack": false,

  "token": "eyJ0aW1lc3RhbXAiOiAxNTczNDcxN2MzI1MDQ5LCAiaHR0cF9wb3J0IjogNTk5NSwgImhvc3RfaXAiOiAiMTkyLjIwIn0=",
  "type": "default-global-system-config:system-defined-phyif-bandwidth",
  "description": "Physical Bandwidth usage anomaly."
}
```

When a node is cleared of all alarms in the Contrail system, the resultant UVEAlarms json message has the following structure:

```
{
  "type": "UVEAlarms",
  "value": null,
  "key": "ObjectVRouter: dcr01-contrail-dpdk-13"
}
```

Every UVEAlarms json subject to the probe's parsing will become one or more batches of event tokens for probe rules processing. For more details, see the following *UVEAlarms Event tokens* section.

UVEAlarms event tokens

The probe json parser converts every alarm in the alarms array into a Netcool event. The token format is driven by the probe json parser configuration.

Each set of Netcool event tokens consists of the following alarm tokens and data tokens:

Alarm tokens

The alarm tokens come from the json parent fields and the json level defined in the messagePayload. This subset of tokens is a key constituent for forming a Netcool event.

Table 7. Alarm tokens	
Token	Description
\$ack	This indicates whether the alarm has been acknowledged.
\$description	This displays a description of the alarm.
\$severity	This displays the severity of the alarm. It is mapped to Netcool severity.
\$timestamp	This shows the time at which the alarm occurred.
\$token	This contains the Contrail token.
\$type	Alarm threshold name.
\$(MSGHEADER.key	This indicates the node from which the alarms originated.
\$(MSGHEADER.type	This displays the alarm type. It will always be UVEAlarms.
\$(MSGHEADER.value._T	This shows the time at which the message was generated.
\$(alarm_rules.*	This contains the Contrail alarm rules that define the threshold boundary. Example: <pre>alarm_rules.or_list.0.and_list.0.condition.operand1: NodeStatus.process_status.state alarm_rules.or_list.0.and_list.0.condition.operand2.json_value: "Functional" alarm_rules.or_list.0.and_list.0.condition.operation: != alarm_rules.or_list.0.and_list.0.condition.variables.0: NodeStatus.process_status.module_id alarm_rules.or_list.0.and_list.0.condition.variables.1: NodeStatus.process_status.instance_id alarm_rules.or_list.0.and_list.0.match.0.json_operand1_value: "Non-Functional" alarm_rules.or_list.0.and_list.0.match.0.json_variables.NodeStatus.process_status.instance_id: 0 alarm_rules.or_list.0.and_list.0.match.0.json_variables.NodeStatus.process_status.module_id: "contrail-collector"</pre>

All data tokens

These tokens come from all fields in the UVEAlarms json inclusive of the data of all member alarms available.

The tokens are notably within a header (defined in `headerPrefix`) and inline indices (namely, for enumerating values from all elements).

For example:

```
MSGHEADER.value.alarms.0.description:    Process(es) reporting as non-functional
MSGHEADER.value.alarms.1.description:    Process Failure.
MSGHEADER.value.alarms.2.description:    System Info Incomplete
```

The probe rules collect threshold names from these tokens to form a comma-delimited list that will be stored in the URL field of the `custom.Contrail_problem_events` table, and will in turn be referenced by the `custom.contrail_generic_clear` trigger to identify and clear the Netcool events associated with the alarms no longer present in the UVEAlarms update.

Table 8. All data tokens	
Token	Description
<code>\$(MSGHEADER.value.alarms).<index>.ack)</code>	This is the same as <code>\$ack</code> .
<code>\$(MSGHEADER.value.alarms).<index>.description)</code>	This is the same as <code>\$description</code> .
<code>\$(MSGHEADER.value.alarms).<index>.severity)</code>	This is the same as <code>\$severity</code> .
<code>\$(MSGHEADER.value.alarms).<index>.timestamp)</code>	This is the same as <code>\$timestamp</code> .
<code>\$(MSGHEADER.value.alarms).<index>.token)</code>	This is the same as <code>\$token</code> .

ObjectServer Database

The following `alerts.status` table columns are pivotal to the Event Clear operation:

Table 9. All data tokens	
Column	Description
@URL	This token stores an aggregated list of the monitored thresholds. The comma-delimited values are extracted from <code>\$(MSGHEADER.value.alarms.*.type)</code> tokens.

Table 9. All data tokens (continued)

Column	Description
@AlertGroup	This token stores the name of the monitored threshold. The value is extracted from the second-half of \$type token, delimited by a colon. Examples of type value: <pre> default-global-system-config:system-defined-address-mismatch-compute default-global-system-config:system-defined-address-mismatch-control default-global-system-config:system-defined-bgp-connectivity default-global-system-config:system-defined-conf-incorrect default-global-system-config:system-defined-core-files default-global-system-config:system-defined-disk-usage-critical default-global-system-config:system-defined-disk-usage-high default-global-system-config:system-defined-node-status-failure default-global-system-config:system-defined-package-version-mismatch default-global-system-config:system-defined-partial-sysinfo default-global-system-config:system-defined-pending-cassandra-compaction-tasks default-global-system-config:system-defined-phyif-bandwidth default-global-system-config:system-defined-process-connectivity default-global-system-config:system-defined-process-status default-global-system-config:system-defined-vrouter-interface default-global-system-config:system-defined-xmpp-connectivity </pre>
@Node	This token stores the origin of the alarms, the parent key field, namely \$(MSGHEADER.key) token. This value is also used as primary key to the custom.Contrail_problem_events table. See the default_triggers.agg_contrail_deduplication trigger for the row insertion.

How event clear works

Every contrail alarm processed in the ObjectServer provides updates to the alerts.status table and the custom.Contrail_problem_events table.

The following is a-level description of the possible outcomes for a Contrail alarm in the alerts.status table:

- A Netcool event will be created for a node's alarm that is new to the alerts.status table.
- Existing Netcool events associated with the alarm present in UVEalarms will be updated.
- Existing Netcool events associated with the alarm absent in UVEalarms will be cleared.

For example, "value": null in UVEalarms denotes all alarms having been cleared, consequently the corresponding Netcool events will be cleared.

In the Contrail use case, event clear is a periodic operation run by a trigger, therefore clear in the Event List is not immediately seen but occurs in every trigger's cycle.

The custom.Contrail_problem_events table holds aggregated data of Contrail alarms grouped by their respective nodes. In conjunction with the clear trigger, the node's aggregated data facilitates alerts.status event clear, the crucial fields are Node and URL. Node is alarm's origin; URL is comma-delimited threshold list.

Every contrail alarm will bring one of these in the custom.Contrail_problem_events table:

- A new entry for previously unavailable node will be created.
- Existing entry (based on node) will be deduplicated – updated with new threshold list.

Every new or updated row is marked for the periodic clear trigger to pick up in its next active cycle.

The custom_contrail_generic_clear trigger periodically performs Event Clear by going through freshly updated rows in the custom.Contrail_problem_events table. When processing a row, the trigger uses the threshold list from the URL field as the reference to select the alarms of the node concerned in the alerts.status table that contains a threshold name (AlertGroup column) not found in the threshold list. In the trigger, the selected alarms will be set with 0 severity among other things as to effect event clear.

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

Additional properties for the probe integration

The Probe integration for Juniper Contrail Alerts uses the Probe for Message Bus configured by the message_bus_contrail.props supplied with the probe.

The remaining properties supported by the probe use the default settings described in “Properties and command line options” on page 12 and “Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 11.0” on page 18.

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*.

Table 10. Probe properties and command line options

Property name	Command line option	Description
Cookie <i>string</i>	-cookie <i>string</i>	Use this property to specify the HTTP cookie name to be retrieved from the probe store. The probe uses the value retrieved from the cookie to replace ++<i>property_setting</i>++ in the <code>restWebSocketTransport.properties</code> file. You can specify multiple values for this property by separating each string with a comma (,). The default is "". The XML or JSON event source sends the cookie in response to the probe's login request. The default setting for this property instructs the probe to replace the ++<i>property_setting</i>++ token in the <code>restWebSocketTransport.properties</code> file with the cookie value.
EnableSSL <i>string</i>	-noenables1 (This is equivalent to EnableSSL with a value of false.) -enables1 (This is equivalent to EnableSSL with a value of true.)	Use this property to specify whether SSL connectivity between the probe and the EMS server is enabled or disabled. This property takes the following values: false: SSL connectivity between the probe and the EMS server is disabled. true: SSL connectivity between the probe and the EMS server is enabled. The default is false. Note: This property is only used by the probe if you are using the WebSocketTransportType.
Host <i>string</i>	-host <i>string</i>	Use this property to specify the host name or IP address of the instance of the XML or JSON event source to which the probe connects. This property is only used by the probe if you are using the WebSocketTransportType. The default is "". Note: The probe also uses this value to replace the ++Host++ token in the <code>restWebSocketTransport.properties</code> file.

Table 10. Probe properties and command line options (continued)

Property name	Command line option	Description
JsonMessageDepth <i>integer</i>	-jsonmessagedepth <i>integer</i>	Use this property to specify the number of levels in the message to traverse during parsing. This enables you to prevent the probe from having to traverse all sub-trees exhaustively. The default is 3.
JsonNestedHeader <i>string</i>	-jsonnestedheader <i>string</i>	Use this property to specify either XML or the JSON tree structure to the nested message header. Note: The message header is included in the events generated by the probe. The default is "".
JsonNestedPayload <i>string</i>	-jsonnestedpayload <i>string</i>	Use this property to specify whether nested parsing on JSON data is enabled. To enable, specify either XML or JSON tree structure to the nested message payload in the JSON string values in the JSON array as specified by the MessagePayload property. This property has the same semantics as MessagePayload except that the default value is blank (an empty string), which turns off nested parsing. The default is "".
JsonParserName <i>string</i>	-jsonparsername <i>string</i>	Use this property to specify the parser type. This property takes the following values: DEFAULT: Generic parser for all target systems. AWS: Specific parser for the AWS integration. The default is "DEFAULT".
KeyStore <i>string</i>	-keystore <i>string</i>	Use this property to specify the location of the keystore file that contains the client certificate for the SSL and trusted authority certificate. The default is "".

Table 10. Probe properties and command line options (continued)

Property name	Command line option	Description
KeyStorePassword <i>string</i>	-keystorepassword <i>string</i>	Use this property to specify the password required to access the certificate specified by the Keystore property. The default is "". Note: You can encrypt this password using the <code>nco_aes_crypt</code> utility within Netcool/OMNIBus.
MessageHeader <i>string</i>	-messageheader <i>string</i>	Use this property to specify either XML or the JSON tree structure to the message header. Note: The message header is included in the events generated by the probe. The default is "".
MessagePayload <i>string</i>	-messagepayload <i>string</i>	Use this property to specify either XML or the JSON tree structure to the message payload. The default is <code>xml</code>. If this property is set to <code>xml</code>, the TransformerFile property must be set to the XML data transformer configuration file. For JSON object parsing, consider migrating to use the new JSON parser configuration file. Note: If you specify a JSON tree structure, it must start with <code>json</code> to indicate that the message is a JSON object. A probe event is derived from a JSON object pointed by message payload. The message payload object consists of name-value data pairs. The probe processes the message payload object to generate probe name-value pair elements.

Table 10. Probe properties and command line options (continued)

Property name	Command line option	Description
Password <i>string</i>	-password <i>string</i>	Use this property to specify the password associated with the Username property for logging into the XML or JSON event source. The default is "". Note: The probe uses this value to replace the ++Password++ token (if it is specified) in the <code>restWebSocketTransport.properties</code> file or in the <code>restWebHookTransport.properties</code> file.
Port <i>integer</i>	-port <i>integer</i>	Use this property to specify the host port of the instance of the XML or JSON event source to which the probe connects. This property is only used by the probe if you are using the WebSocketTransportType. It is used in conjunction with the Host property. The default is 0. Note: The probe also uses this value to replace the ++Port++ token in the <code>restWebSocketTransport.properties</code> file.
RecordData <i>string</i>	-recorddata <i>string</i>	Use this property to specify a comma-separated list of attributes from the event to be recorded in the file specified by the DataBackupFile property. The data recorded can be used by the probe to resolve transport properties using tokens with the prefix "RecordData.". For example, if the event generated by the probe has a URL attribute that should be recorded, set the RecordData property to URL. To use this attribute to resolve a property in the probe's transport property file, set the property with the following token: <code>WebSocketURL=++URL++</code>
StreamCapture <i>string</i>	-streamcapture <i>string</i>	Use this property to specify whether or not the probe stores the XML or JSON event data in a stream capture file. The default is false.

Table 10. Probe properties and command line options (continued)

Property name	Command line option	Description
StreamCaptureFile <i>string</i>	<code>-streamcapturefile <i>string</i></code>	Use this property to specify the location of the stream capture file. On UNIX and Linux operating systems, the default is \$OMNIHOME/var/message_bus.stream. On Windows operating systems, you must specify the full directory path to the file. For example: C:\\IBM\\Tivoli\\Netcool\\omnibus\\var\\ message_bus.stream
TransformerFile <i>string</i>	<code>-transformerfile <i>string</i></code>	Use this property to specify the location of the transformer properties file. This property can be used to specify the transformer configuration file for XML event data transformation, or the JSON parser configuration file for parsing different JSON object structures. On UNIX and Linux operating systems, the default is \$OMNIHOME/java/conf/transformers.xml. On Windows operating systems, you must specify the full directory path to the file. For example: 'C:\\IBM\\Tivoli\\Netcool\\omnibus\\java\\conf\\transformers.xml'
TransportFile <i>string</i>	<code>-transportfile <i>string</i></code>	Use this property to specify the location of the transport properties file. On UNIX and Linux operating systems, the default is \$OMNIHOME/java/conf/jmsTransport.properties. On Windows operating systems, you must specify the full directory path to the file. For example: 'C:\\IBM\\Tivoli\\Netcool\\omnibus\\java\\conf\\jmsTransport.properties'

Table 10. Probe properties and command line options (continued)

Property name	Command line option	Description
TransportType <i>string</i>	-transporttype <i>string</i>	Use this property to either specify the transport method to be used or to define the name of the transport module class to use. This property takes the following values: • Cometd • EventSource • File • HTTP • JMS • KAFKA • MQTT • Socket • Webhook • WebSocket The default is JMS.
Username <i>string</i>	-username <i>string</i>	Use this property to specify the user account for logging into the XML or JSON event source. This property is only used by the probe if you are using the WebSocket TransportType. The default is "". Note: The probe uses this value to replace the ++Username++ token (if it is specified) in the restWebSocketTransport.properties file or in the restWebHookTransport.properties file.

Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 11.0

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 11.0.

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

Table 11. Properties and command line options

Property name	Command line option	Description
CommandPort <i>integer</i>	<code>-commandport <i>integer</i></code>	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 <i>integer</i>	<code>-commandportlimit <i>integer</i></code>	Use this property to specify the maximum number of Telnet connections that can be made to the probe. The default is 10.
DataBackupFile <i>string</i>	<code>-databackupfile <i>string</i></code>	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 <i>integer</i>	<code>-heartbeatinterval <i>integer</i></code>	Use this property to specify the frequency (in seconds) with which the probe checks the status of the host server. The default is 1.
Inactivity <i>integer</i>	<code>-inactivity <i>integer</i></code>	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).
InactivityAction <i>string</i>	<code>-inactivityaction <i>string</i></code>	Use this property to specify the action the probe takes when the inactivity timeout is reached: SHUTDOWN: The probe sends a ProbeWatch message to notify the user and then shuts down. CONTINUE: The probe sends a ProbeWatch message to notify the user, but does not shut down. The default is SHUTDOWN.

Table 11. Properties and command line options (continued)

Property name	Command line option	Description
InitialResync <i>string</i>	-initialresync <i>string</i>	Use this property to specify whether the probe performs resynchronization 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 <i>integer</i>	-maxeventqueuesize <i>integer</i>	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 <i>integer</i>	-resyncinterval <i>integer</i>	Use this property to specify the interval (in seconds) at which the probe makes successive resynchronization requests. For most probes, the default value for this property is 0 (which instructs the probe to not make successive resynchronization requests). If you are running the JDBC Probe, the default value for the ResyncInterval property is 60. This is because the JDBC Probe only acquires data using the resynchronization process.
RetryCount <i>integer</i>	-retrycount <i>integer</i>	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 11. Properties and command line options (continued)

Property name	Command line option	Description
RetryInterval <i>integer</i>	<code>-retryinterval <i>integer</i></code>	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).
RotateEndpoint <i>string</i>	<code>-rotateendpoint <i>string</i></code>	Use this property to specify whether the probe attempts to connect to another endpoint if the connection to the first endpoint fails. This property takes the following values: <code>false</code>: The probe does not attempt to connect to another endpoint if the connection to the first endpoint fails. <code>true</code>: The probe attempts to connect to another endpoint if the connection to the first endpoint fails. The default is <code>false</code>.

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.

During installation of the probe, several rules files are installed in addition to the main `message_bus.rules` file. Specific rules for the Probe Integration for Juniper Contrail Alerts are contained in the `message_bus_contrail.rules` file.

Live Alarm elements

The following elements are generated from Juniper Contrail SSE live alarms:

Table 12. Live alarm elements

Element name	Element description
<code>\$ack</code>	This element indicates whether the alarm has been acknowledged.
<code>\$description</code>	This element displays a description of the alarm.
<code>\$resync_event</code>	This element indicates the alarm phase. This is always <code>false</code> for live alarms.

Table 12. Live alarm elements (continued)

Element name	Element description
\$severity	This element displays the severity of the alarm: 0: CRITICAL 1: MAJOR 2: WARNING
\$timestamp	This element shows the time at which the alarm occurred.
\$token	This element contains the access token.
\$type	This element indicates the alarm type.
\$MSGHEADER.key	This element displays the node from which the alarm originated.
\$MSGHEADER.type	This element displays the alarm type.
\$MSGHEADER.value_T	This element displays the date and time of the alarm occurrence.
\$alarm_rules.or_list.0.and_list.0.condition.operand1	This element displays the alarm rules or list operand 1.
\$alarm_rules.or_list.0.and_list.0.condition.operand2.json_value	This element displays the alarm rules or list operand 1.
\$alarm_rules.or_list.0.and_list.0.condition.operation	This element displays the alarm rules or list operation.
\$alarm_rules.or_list.0.and_list.0.condition.variables.0	This element displays the alarm rules or list variable indexed 0.
\$alarm_rules.or_list.0.and_list.0.condition.variables.1	This element displays the alarm rules or list variable indexed 0.
\$alarm_rules.or_list.0.match.*	This element displays the alarm rules or list matches. Example: \$alarm_rules.or_list.0.and_list.0.match.0.json_operand1_value

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 13. Error messages

Error	Description	Action
Failed to startup probe	The probe failed to start, probably due to an invalid combination of properties set in the <code>message_bus.props</code> file.	Check the values set for the Host , Port Username , and Password properties.
Failed to transform	The probe is unable to transform the XML into name-value pairs.	Check the entries in the transformer file. Then test the XSLT file created for the event source.
Failed to parse message	The probe could not parse the event data.	Check the format of the event generated by the XML event source.
Failed to record data into backup file	The probe could not write record data into the backup file.	Check that the backup file is specified correctly by the DataBackupFile property in the <code>message_bus.props</code> file and that the file has the appropriate permissions set.
Failed to start Transport module for connection	The transport module failed to start.	Check the value set for the TransportType property and the details specified in the TransportFile .
Failed to subscribe Transport module to the interface	The transport module failed to subscribe to the event source.	Check the details specified in the TransportFile .
Failed to get active alarms during resync	The probe failed to received active alarms during resynchronization with the event source.	Check the details specified in the TransportFile .
Exception caught in WebSocketClientHandler: Queue full Note: This error message only applies when the Probe for Message Bus runs with the WebSocket transport.	The WebSocket transport event queue has reached its limit and has started to discard events. This usually occurs in a flooding scenario or if the event processing is slow or blocked.	Verify that no other error occurred in the probe log or ObjectServer logs that could potentially slow down or block the probe event processing. Verify that the probe is not under a flood or denial-of-service attack.

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 14. ProbeWatch messages

ProbeWatch message	Description	Triggers/causes
Connection to source lost	The connection to the XML source has been lost.	The target system might have disconnected or gone down.
Failed to open stream capture file Failed to write to stream capture file Failed to close stream capture file	The probe is unable to use the specified stream capture file.	Check the permissions set for the file and the directory in which it is being written. Then check the value specified for the StreamCaptureFile property.
Start resynchronization	The resynchronization process started.	The probe started with the InitialResync property set to true.
Finish resynchronization	The resynchronization process ended.	The probe completed the resynchronization process.

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

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

Trademarks

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

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

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

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

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

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

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



Part Number:

SC28-3114-00



(1P) P/N: