IBM® Tivoli® Netcool/OMNIbus Probe Integration for Nokia Network Services Platform Release 22.x

Reference Guide
December 14, 2022

SC27-9589-04

Note

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 33.

Edition notice

This edition (SC27-9589-04) applies to version 21.0 of IBM Tivoli Netcool/OMNIbus Probe for Message Bus and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-9589-03.

Contents

About this guide.. v

Document control page..v

Conventions used in this guide... vi

Chapter 1. Nokia Network Services Platform..1

Summary.. 1

Installing probes.. 2

Configuring the Probe for Message Bus to integrate with Nokia Network Services Platform.................. 3

Configuring the cross-launch application.. 7

Nokia NSP server redundancy...9

Performing partial resynchronization.. 10

Configuring SSL...11

Configuring persistent subscriptions...13

Enabling alarm correlation...14

Enabling JSON stream support...15

Running the probe.. 15

Additional properties for the probe integration...15

Properties and command line options..15

Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 12.0..20

Elements.. 23

Error messages.. 28

Error messages generated for the Probe Integration for Nokia NSP...29

ProbeWatch messages.. 29

Known issues... 30

Appendix A. Notices and Trademarks...33

Notices.. 33

Trademarks.. 34

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
SC27-9589-00	December 30, 2019	First IBM publication.
SC27-9589-01	April 3, 2020	Updated for version 11 of IBM Tivoli Netcool/OMNIbus Probe for Message Bus. The probe has been validated against Nokia NSP 19.11. The following sections updated: • Chapter 1, "Nokia Network Services Platform," on page 1 • "Summary" on page 1
SC27-9589-02	September 25, 2020	Updated for version 13 of IBM Tivoli Netcool/OMNIbus Probe for Message Bus. The probe has been validated against Nokia NSP 20.6. "Summary" on page 1 updated. Chapter 1, "Nokia Network Services Platform," on page 1 updated. Description for the **`PartialResync`** property added to "Properties and command line options" on page 15. The following sections added: • "Configuring the cross-launch application" on page 7 • "Nokia NSP server redundancy" on page 9 • "Performing partial resynchronization" on page 10 • "Configuring SSL connections" on page 11 • "Known issues" on page 30
SC27-9589-03	November 5, 2021	Updated for version 18 of IBM Tivoli Netcool/OMNIbus Probe for Message Bus. The probe has been validated against Nokia NSP 21.6. "Summary" on page 1 updated. The following sections added: • "Configuring persistent subscriptions" on page 13 • "Enabling alarm correlation" on page 14 • "Enabling JSON stream support" on page 15 "Known issues" on page 30 updated.

Table 1. Document modification history (continued)

Document version	**Publication date**	**Comments**
SC27-9589-04	December 14, 2022	Updated for version 21 of IBM Tivoli Netcool/OMNIbus Probe for Message Bus. "Summary" on page 1 updated. The Probe for Message Bus version 21 has been certified for Nokia NSP 22.x. "Known issues" on page 30 updated.

Conventions used in this guide

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

Operating system-dependent variables and paths

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

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

`$OMNIHOME/probes`

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

`%OMNIHOME%\probes`

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

Note: The names of environment variables are not always the same in Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX and Linux environments. Where such variables are described in the guide, both the UNIX and Windows conventions will be used.

Operating system-specific directory names

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

`$OMNIHOME/probes/`*`arch`*

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

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

Table 2. Directory names for the arch variable

Operating system	**Directory name represented by *arch***
AIX® systems	`aix5`

Table 2. Directory names for the arch variable (continued)

Operating system	**Directory name represented by *arch***
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. Nokia Network Services Platform

The Message Bus Probe can be used to integrate with Nokia Network Services Platform (NSP) using the Kafka transport and the HTTP REST transport. The probe supports all revisions of Nokia NSP 18.6 and above, and has been certified against Nokia NSP 18.6, Nokia NSP 19.11, Nokia NSP 20.6, Nokia NSP 21.6, and Nokia NSP 22.x.

Note: The probe is not backward compatible with NSP 18.3.

This guide contains the following sections:

- "Summary" on page 1
- "Configuring the Probe for Message Bus to integrate with Nokia Network Services Platform" on page 3
- "Installing probes" on page 2
- "Running the probe" on page 15
- "Additional properties for the probe integration" on page 15
- "Elements" on page 23
- "Error messages" on page 28
- "ProbeWatch messages" on page 29

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 Network Services Platform (NSP).

Table 3. Summary

Probe target	Nokia NSP Kafka notification payload
Probe executable name	`nco_p_message_bus`
Installation package	`omnibus_`*`arch`*`_probe_nco_p_message_bus_`*`version`*
Package version	21.0 **Note:** The Probe for Message Bus version 21 has been certified for Nokia NSP Release 22.x.
Probe supported on	For details of supported operating systems, see the following Release Notice on the IBM Software Support website: https://www.ibm.com/support/pages/node/270963
Properties files	The probe is supplied with the following properties file installed in the `$OMNIHOME/probes/`*`arch`* directory: `message_bus_nokia_nfmp.props`
Rules file	The probe is supplied with the following rules file installed in the `$OMNIHOME/probes/`*`arch`* directory: `message_bus_nokia_nfmp.rules`

Table 3. Summary (continued)

Transport properties files	The probe supports the following transport configuration files installed in the `$OMNIHOME/java/conf/` directory:
	`nokiaNspKafkaTransport.properties`
	`nokiaNspKafkaConnectionProperties.json`
	`nokiaNspKafkaClient.properties`
	`nokiaNspRestMulitChannelHttpTransport.json`

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	Kafka
Multicultural support	Available For information about configuring multicultural support, including language options, see the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*.
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 IBM Documentation:

   https://www.ibm.com/support/knowledgecenter/SSSHTQ_int/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 in IBM Documentation:

   https://www.ibm.com/support/knowledgecenter/SSSHTQ_int/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 Nokia Network Services Platform

The Message Bus Probe can be used to integrate with Nokia Network Services Platform (NSP) using the Kafka transport and the HTTPS REST transport. In the Kafka transport, there is an additional multi channel HTTPS REST transport component that can be enabled for Nokia NSP. The probe can be configured to integrate with Nokia NSP with the following configurations:

- HTTPS REST and SSL Kafka
- HTTPS REST and Plaintext Kafka

The following configuration files are supplied with the probe and are required by the integration with Nokia NSP:

- `message_bus_nokia_nfmp.props`
- `message_bus_nokia_nfmp.rules`
- `message_bus_nokia_nfmp_parser.json`
- `nokiaNspKafkaTransport.properties`
- `nokiaNspKafkaConnectionProperties.json`
- `nokiaNspKafkaClient.properties`
- `nokiaNspRestMulitChannelHttpTransport.json`

To integrate with Nokia NSP, use the following steps:

1. Configure **`message_bus_nokia_nfmp.props`** for the Nokia NSP integration.

   The Probe integration for Nokia NSP uses the Probe for Message Bus configured by the `message_bus_nokia_nfmp.props` file supplied with the probe. The following default probe properties are provided for the integration with Nokia NSP. Kafka transport is selected as the main transport type for this integration.

```
#======================================================================
# SETTING PROBE LOGS, PROPS, RULES
#======================================================================
Server : 'NCOMS'
Manager : 'Kafka'
MessageLog : '$OMNIHOME/log/message_bus_nokia_nfmp.log'
PropsFile : '$OMNIHOME/probes/<arch>/message_bus_nokia_nfmp.props'
RulesFile : '$OMNIHOME/probes/<arch>/message_bus_nokia_nfmp.rules'

#======================================================================
# SETTING TRANSPORT TYPE
#======================================================================
TransportType    : 'KAFKA'
TransportFile    : '$OMNIHOME/java/conf/nokiaNspKafkaTransport.properties'

#======================================================================
# SETTING PARSER CONFIGURATIONS. (SUPPORTS JSON OR XML)
#======================================================================
# FOR PARSING JSON DATA
TransformerFile  : '$OMNIHOME/probes/<arch>/message_bus_nokia_nfmp_parser.json'
MessagePayload   : 'JSON'

#======================================================================
# SETTING CREDENTIALS WHEN KAFKA TRANSPORT ALSO NEEDS A HTTP TRANSPORT
#======================================================================
Username                       : 'nfmp_username'
Password                       : 'nfmp_password'

#======================================================================
# SETTING SSL & KEYSTORE
#======================================================================
EnableSSL                     : 'true'
KeyStore                      : '<Path to KeyStore>/keystore.jks'
KeyStorePassword              : 'keystore_password'

#======================================================================
# SETTING RESYNC
```

```
    #==================================================================
    InitialResync                 : 'true'
    ResyncBatchSize               : 1000
```

2. Configure **nokiaNspKafkaTransport.properties** for the Nokia NSP integration:

```
    #==========================================================================
    # KAFKA CLIENT MODE AS CONSUMER
    #==========================================================================
    KafkaClientMode=CONSUMER

    #==========================================================================
    # LOCATION OF JSON FILE CONTAINING KAFKA & ZOOKEEPER CONNECTION PROPERTIES
    #==========================================================================
    ConnectionPropertiesFile=$OMNIHOME/java/conf/nokiaNspKafkaConnectionProperties.json

    #==========================================================================
    # LOCATION OF FILE CONTAINING REST CONNECTION PROPERTIES
    #==========================================================================
    httpConnectionPropertiesFile=$OMNIHOME/java/conf/
 nokiaNspRestMultiChannelHttpTransport.json
```

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

3. Configure **nokiaNspKafkaConnectionProperties.json** to enable either the Plaintext or SSL Kafka connection with the Nokia NSP.

   To enable the Plaintext Kafka connection with the Nokia NSP, apply the following settings:

```
   {
    "zookeeper_client" :
        {
            "target" : "",
            "properties" : "",
            "java_sys_props" : "",
            "topic_watch": false,
            "broker_watch": false
        },
    "brokers" : "PLAINTEXT://<Nokia SDN Host IP>:<Kafka Port>",
    "topics": "",
    "kafka_client" :
      {
       "properties" : "<Absolute Path To $OMNIHOME/java/conf/nokiaNspKafkaClient.properties",
       "java_sys_props" : ""
      }
 }
```

   Otherwise, to enable SSL Kafka connection with the Nokia NSP, apply the following settings:

```
 {
    "zookeeper_client" :
    {
       "target" : "",
       "properties" : "",
       "java_sys_props" : "",
       "topic_watch": false,
       "broker_watch": false
    },
    "brokers" : "SSL://<Nokia SDN Host IP>:<Kafka port>",
    "topics": "",
    "kafka_client" :
    {
       "properties" : "<Absolute Path To $OMNIHOME/java/conf/nokiaNspKafkaClient.properties",
       "java_sys_props" : ""
    }
 }
```

4. Configure the Kafka client settings in **nokiaNspKafkaClient.properties** by providing a location and password for the Keystore and Truststore. The following settings are required to enable the Plaintext Kafka connection:

```

 security.protocol=PLAINTEXT
```

```
ack=all
group.id=NokiaNsp

enable.auto.commit=true
auto.commit.interval.ms=1000
session.timeout.ms=30000
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
value.deserializer=org.apache.kafka.common.serialization.StringDeserializer
```

Otherwise, to enable the SSL Kafka connection with the Nokia NSP, apply the following settings:

```
security.protocol=SSL
ssl.enabled.protocols=TLSv1.2,TLSv1.1,TLSv1
ssl.keystore.location=<Path To Keystore File>/samserver.keystore
ssl.keystore.password=<keystore password>
ssl.keystore.type=JKS
ssl.truststore.location=<Path To Keystore File>/samserver.keystore
ssl.truststore.password=<truststore password>
ssl.truststore.type=JKS

ack=all
group.id=NokiaNsp
enable.auto.commit=true
auto.commit.interval.ms=1000
session.timeout.ms=30000
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
value.deserializer=org.apache.kafka.common.serialization.StringDeserializer
```

5. Configure **`nokiaNspRestMulitChannelHttpTransport.json`** for settings on REST requests. Default settings has been provided.

   a. Ensure following property has been added:

      `"keepTokens":"access_token, refresh_token, subscriptionId",`

   b. Check that the correct HOST and PORT of the target system is set for each request.

```
{
   "GLOBAL":
   {
      "httpVersion":"1.1",
      "httpHeaders":"",
      "responseTimeout":"60",
      "securityProtocol":"TLSv1.2",
      "keepTokens":"access_token, refresh_token, subscriptionId",
      "tokenEndpointURI":"",
      "autoReconnect":"OFF"
   },

   "LOGIN":
   {
        "GET_ACCESS_TOKEN":
        {
            "uri":"https://HOST:PORT/rest-gateway/rest/api/v1/auth/token",
            "method":"POST",
            "headers":"Authorization=Basic ++Username++:++Password++,Accept=application/json,
                       Content-Type=application/json,Use-Cookie=true,
                       User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
            "content":"{\"grant_type\":\"client_credentials\"}",
            "interval":"0",
            "requireSSL":"true"
        },

        "GET_REFRESH_ACCESS_TOKEN":
        {
            "uri":"https://HOST:PORT/rest-gateway/rest/api/v1/auth/token",
            "method":"POST",
            "headers":"Authorization=Basic ++Username++:++Password++,Accept=application/json,
                       Content-Type=application/json,Use-Cookie=true,
                       User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
            "content":"{\"grant_type\":\"refresh_token\",\"refresh_token\":\"++refresh_token++\"}",
            "interval":"60",
            "requireSSL":"true"
        }
   },

   "SUBSCRIBE":
   {
        "GET_SUBSCRIPTION":
        {
            "uri":"https://HOST:PORT/nbi-notification/api/v1/notifications/subscriptions",
            "method":"POST",
            "headers":"Authorization=Bearer ++access_token++,Accept=application/json,
                       Content-Type=application/json,Use-Cookie=true,
                       User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
            "content":"{\"categories\":[{\"name\":\"NSP-FAULT\"}]}",
            "interval":"0",
            "requireSSL":"true"
        },

        "GET_SUBSCRIPTION_REFRESH":
        {
            "uri":"https://HOST:PORT/nbi-notification/api/v1/notifications/subscriptions/
                   ++subscriptionId++/renewals",
            "method":"POST",
            "headers":"Authorization=Bearer ++access_token++,Accept=application/json,
                       Content-Type=application/x-www-form-urlencoded,Use-Cookie=true,
                       User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
            "content":"",
            "interval":"120",
            "requireSSL":"true"
        }
    },

    "RESYNC":
    {
        "RESYNC_FAULT_MANAGEMENT_ALARMS":
        {
            "uri":"https://HOST:PORT/FaultManagement/rest/api/v2/alarms/details",
            "method":"GET",
            "headers":"Authorization=Bearer ++access_token++,Accept=application/json,
                       Content-Type=application/x-www-form-urlencoded,Use-Cookie=true,
                       User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
            "content":"",
            "interval":"0",
            "requireSSL":"true"
        }
    },

    "LOGOUT":
    {
        "DELETE_SUBSCRIPTION_ID":
        {
            "uri":"https://HOST:PORT/nbi-notification/api/v1/notifications/subscriptions/
                   ++subscriptionId++",
            "method":"DELETE",
            "headers":"Authorization=Bearer ++access_token++,Accept=application/json,
                       Content-Type=application/x-www-form-urlencoded,Use-Cookie=true,
                       User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
            "content":"",
            "interval":"0",
            "requireSSL":"true"
        },

        "REVOKE_ACCESS_TOKEN":
        {
            "uri":"https://HOST:PORT/rest-gateway/rest/api/v1/auth/revocation",
            "method":"POST",
            "headers":"Authorization=Basic ++Username++:++Password++,Accept=application/json,
                       Content-Type=application/x-www-form-urlencoded,Use-Cookie=true,
                       User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
            "content":"token=++access_token++&token_type_hint=token",
            "interval":"0",
            "requireSSL":"true"
        }
    }
}
```

6. Modify the default OMNIbus deduplication triggers to process probe events from Nokia NSP NFM-P (KAFKA).

   On Unix, run the following command:

   ```
   $OMNIHOME/bin/nco_sql -server <objectserver_name> -user <username> -password <password>
   $OMNIHOME/java/conf/nokiaNsp_update_deduplication.sql
   ```

   On Windows, run the following command:

   ```
   %NCHOME%\bin\redist\isql.exe -S <objectserver name> -U <username> -P <password> -i %OMNIHOME%
   \java\conf\nokiaNsp_update_deduplication.sql
   ```

Configuring the cross-launch application

The probe is supplied with a launch-in-context feature that enables you to launch the Nokia NSP Fault Management Web app from Netcool/OMNIbus Web GUI Active Event List right-click tool using two separate methods. You can configure the GUI manually. You can also configure the Web GUI using WAAPI which requires `createCrossLaunchTool.xml` and `modifyAlertsMenu.xml`.

You must run `addCrossLaunchOSFields.sql` by running one of these commands according to the platform to add the `ObjectFullName` and `NokiaNspObjectId` fields in `alerts.status`.

The `createCrossLaunchTool.xml`, `modifyAlertsMenu.xml` and `addCrossLaunchOSFields.sql` files are located here:

`$OMNIHOME/probes/NOKIA_NSP_CrossLaunch/`

On Unix:

```
$OMNIHOME/bin/nco_sql –server <objectserver_name> –user –password <
<path_to_file>/addCrossLaunchOSFields.sql
```

On Windows:

```
%NCHOME%\bin\redist\isql.exe -S -U -P -i
<path_to_file>\addCrossLaunchOSFields.sql
```

After executing `addCrossLaunchOSFields.sql`, uncomment the following lines in the `message_bus_nokia_nfmp.rules` rules file:

```
#if (exists($objectFullName))
    #{
        #@ObjectFullName = $objectFullName
    #}
    #if (exists($nspObjectId))
    #{
    #  @NokiaNspObjectId = $nspObjectId
    #}
```

Note: `createCrossLaunchTool.xml` and `modifyAlertsMenu.xml` are not required if you are configuring Web GUI manually.

Configure the Web GUI manually

To configure the Web GUI manually, complete the following steps.

1. Login into the IBM Tivoli Netcool/OMNIbus WebGUI Dashboard Application Service Hub.

   https://<DASH IP or hostname>:16311/ibm/console

   Refer to the *IBM Tivoli Netcool/OMNIbus Web GUI Administration and User's Guide* to create a new tool of script type named `NFMP-CrossLaunch` with the following inputs.

   For example:

   a. Open the Event Management Tool (from the Dashboard's side bars)
   b. Select **Tool Configuration** and create a new tool named "NFMP-CrossLaunch"
   c. Choose **Type as script** and input the script as below:

      For the alarm impacted list, using `objectFullName`:

      ```
      var objName="{@ObjectFullName}";
      var sdnhost = "127.0.0.1:8544";
      var address = "https://" + sdnhost + ":8544/FaultManagement?
      view=alarmListImpacts&objectFullName=" + encodeURIComponent(objName);
      window.open (address,"NFM-P Cross Launch");
      ```

      For the alarm list, using `objectId/fdn`:

```
var alarmId='{@NokiaNspObjectId}';
var sdnhost = '127.0.0.1:8544';
var address = 'https://' + sdnhost + '/FaultManagement?view=alarmList&alarmId=' +
encodeURIComponent(alarmId);
window.open (address,"NSP Alarm List");
```

d. * replace the IP <127.0.0.1> with your Nokia NFM-P server hostname or IP address accordingly.

   Note: When SSL is enabled on NFM-P, ensure the following settings are correct:

   i) Ensure the protocol is changed from `http` to `https`.

   ii) Ensure the IP address contains a port number configured for the NFM-P Client; for example, `127.0.0.1:4321`

2. Refer to *IBM Tivoli Netcool/OMNIbus Web GUI Administration and User's Guide* to perform the menu configuration and modify alert menu to include NFMP-CrossLaunch that should be created in step 2 from available items.

   For example:

   a. **Open Event Management Tool** > **Menu Configurations**
   b. Click on "alerts" from the available menu and click on the "Modify" button.
   c. Add the **NFMP-CrossLaunch** tool that was previously created in step 2 from the available items to the current items.
   d. Click **Save**.
   e. Open **Active Event List**.
   f. Click **Refresh**.
   g. Right click on one of the alarms received from NFM-P. You should see **NSP-AlarmListImpacted** and **NSP-AlarmList** as an options.

Configure the Web GUI using WAAPI

To configure the Web GUI using the WAAPI, complete the following steps.

1. Refer to the *IBM Tivoli Netcool/OMNIbus Web GUI Administration API (WAAPI) User's Guide* to create a new tool by using `createCrossLaunchTool.xml`. **NSP-AlarmListImpacted** and **NSP-AlarmList** must not exist before this step.

   For example:

   a. Go to WebGUI WAAPI bin dir.
   b. Modify the `createCrossLaunchTool.xml` to change the `sdnhost` in the line below to your Nokia NFM-P server hostname or IP address and save the XML file:

```
<tool:script foreach="true"  command="var objName='{@ObjectFullName}'; var sdnhost =
'127.0.0.1:8544'; var address = 'https://' + sdnhost +
'/FaultManagement?view=alarmListImpacts&amp;objectFullName=' +
encodeURIComponent(objName); window.open (address,&quot;NSP Alarm Impacted
List&quot;);" />
```

   c. After reviewing and modifying `createCrossLaunch.xml` run this command:

```
$WAAPI_BIN_DIR/bin/runwaapi -file
$OMNIHOME/probes/<platforms>/NOKIA_NFMP_CrossLaunch/createCrossLaunchTool.xml
 -user <WAS_USER_ID> -password
<WAS_USER_PASSWORD>
```

2. Refer to the *IBM Tivoli Netcool/OMNIbus Web GUI Administration API (WAAPI) User's Guide* to modify the alert menu by using `modifyAlertsMenu.xml`.

   Note: Running the `modifyAlertsMenu.xml` will overwrite your existing alerts menu items. Make sure in `modifyAlertsMenu.xml` the content under modify.menu does not overwrite any of your existing items in the Alerts menu.

Make any changes if required, or you can run the manual step in above section to add in the newly created NFMP-CrossLaunch tool to the Active Event List alerts right-click tool menu.

3. After reviewing and modifying the `modifyAlertsMenu.xml` run this command:

```
$WAAPI_BIN_DIR/bin/runwaapi -file
$OMNIHOME/probes/<platforms>/NOKIA_NFMP_CrossLaunch/modifyAlertsMenu.xml
 -user <WAS_USER_ID> -password <WAS_USER_PASSWORD>
```

4. Open **Active Event List** and click **Refresh**.
5. Right click on one of the alarms received from NFM-P. You should see **NSP-AlarmListImpacted** and **NSP-AlarmList** as an option.

Nokia NSP server redundancy

Two Nokia NSP servers can run in a redundancy pair (that is, one runs as the primary server and the other as a backup server). This affects the way that you configure the probe.

If the primary NSP server is down while the probe is connected, then the probe will attempt to connect to the secondary NSP server which will take over the primary server role. The probe cannot connect to the secondary NSP server if the primary server is still operational.

To enable fail-over support, use the following steps:

1. Change probe properties as below:

```
RotateEndpoint: "true"
RetryCount = 100
RetryInterval = 10
```

`RotateEndpoint` required to be used together with `RetryCount` so that it can be fail-over to the secondary server settings during the retry/reconnect attempts.

2. Edit the `nokiaNspKafkaTransport.properties` to point to the `nokiaNspRestMultiChannelHttpTransportFailover.json` file which contains the fail-over primary and secondary server settings:

```
#===============================================================================
# LOCATION OF FILE CONTAINING REST CONNECTION PROPERTIES
#===============================================================================
httpConnectionPropertiesFile=/opt/IBM/tivoli/netcool/omnibus/java/conf/
nokiaNspRestMultiChannelHttpTransportFailover.json
```

3. Edit the `nokiaNspRestMultiChannelHttpTransportFailover.json` file to configure the primary and secondary server IP/hostname and port.

```
"FailOverServer":
      {
         "Primary":
         {
            "authenticationServer": "10.0.0.1:443",
            "restAPIServer": "10.0.0.1:8544"
         },
         "Secondary":
         {
            "authenticationServer": "10.0.0.2:443",
            "restAPIServer": "10.0.0.2:8544"
         }
      }
```

Check with Nokia NSP regarding the port if there is any changes that not using the default port in the deployment.

Leave the `++authenticationServer++` and `++restAPIServer++` as it is in the configuration file. The probe will replace these variables with your defined `FailOverServer.Primary|Secondary.authenticationServer|restAPIServer` during runtime.

4. Edit the `$OMNIHOME/java/conf/nokiaNspKafkaConnectionProperties.json` file for Nokia NSP Kafka brokers configuration:

   ```
   {
       "zookeeper_client" :
           {
               "target" : "<Nokia SDN 1 Host IP>:<Zookeeper port>,<Nokia SDN 2 Host
   IP>:<Zookeeper port>",
               "properties" : "",
               "java_sys_props" : "",
               "topic_watch": true,
               "broker_watch": true
           },
       "brokers" : "SSL://<Nokia SDN 1 Host IP>:<Kafka port>,SSL://<Nokia SDN 2 Host IP>:<Kafka
   port>",

       "topics": "",
       "kafka_client" :
           {
               "properties" : "<Absolute Path To $OMNIHOME>/java/conf/
   nokiaNspKafkaClient.properties",
               "java_sys_props" : ""
           }
   }
   ```

   Note: For SSL, the default Kafka port is 9192, and the zookeeper port is 2281. For plaintext, the default Kafka port is 9092, and the zookeeper port is 2181.

   If the Nokia SDN is deployed with high-availability (HA), the host IP here will be referred to the HA's virtual IP.

Edit `$OMNIHOME/java/conf/nokiaNspRestMultiChannelHttpTransportFailover.json` under `GLOBAL` properties to configure `errorResponseLimit` and `failOnHTTPResponseCodes`. You must configure `failOnHTTPResponseCodes` to react to a response error of `307`, `500`, and `503`, because these error codes indicate that the probe is not connected to the correct active server or that the NSP server is not yet ready. Thus when probe received these errors, the probe should fall back to retry the connection.

```
    "GLOBAL":
    {
       ...
       "errorResponseLimit": "5",
       "failOnHTTPResponseCodes": "307,500,503",
       ...
```

Performing partial resynchronization

When the probe is disconnected or restarted, it will perform a full resync at startup. This normally pulls a lot of active alarms from the Nokia NSP server and some of these events have already been received while the probe running previously. The probe can be configured to perform a partial resync at startup based on the last event received timestamp stored in a persistent file.

When partial resync is enabled, during resync, the probe will check whether **`DataBackupFile`** exists and has a valid value of the last event received time. If it does, the probe will retrieve the last event received timestamp from persistent file and append the partial resync uri with `alarmFilter=lastTimeDetected>=`*`resyncTimestamp`* in the resync request.

Every probe has a check subscription feature controlled by the **`HeartbeatInterval`** property. During check subscription, the probe stores the last received event timestamp into the **`DataBackupFile`**. Thus, the last event received time is based on the check subscription interval, for example: saved last event received time every 120s (if `HeartbeatInterval = 120`). When partial resync is enabled, the **`HeartbeatInterval`** property must be set to `120` (in seconds) or greater so that the databackup file will not be updated frequently which may impact probe performance and stability.

To enable the partial resync feature, use the following steps:

1. Configure probe property as below:

```
DataBackupFile: "/opt/IBM/tivoli/netcool/omnibus/var/nspPartialResync"
PartialResync: "true"
HeartbeatInterval: 120
```

2. Create an empty file as defined in **`DataBackupFile`**, for example:

   `/opt/IBM/tivoli/netcool/omnibus/var/nspPartialResync`

   Make sure the user that executes the probe start command has read/write permission to this file.
3. Check the resync request in `$OMNIHOME/java/conf/nokiaNspRestMultiChannelHttpTransport.json (RESYNC.RESYNC_FAULT_MANAGEMENT_ALARMS.uri)`

- If there is no alarm filter customization made in the resync uri, the probe will append the partial resync alarm filter in the URI, for example:

  ```
  ?alarmFilter=lastTimeDetected%2520%253E%2520<LastResyncTime>
  ```

- If there is alarm filter customization made in the resync uri, you need to append the following string to your resync URI for partial resync to work correctly:

  ```
  and%2520lastTimeDetected%2520%253E%2520<LastResyncTime>%2520
  ```

For example:

Change the following lines:

```
"RESYNC":
   {
        "RESYNC_FAULT_MANAGEMENT_ALARMS":
        {
            "uri":"https://host:port/FaultManagement/rest/api/v2/alarm
s/details?
alarmFilter=alarmName%2520like%2520'%2525Equipment%2525'and%2520severity%2520%253D%2520'major'%2
520",
      ...
```

to:

```
"RESYNC":
   {
        "RESYNC_FAULT_MANAGEMENT_ALARMS":
        {
            "uri":"https://host:port/FaultManagement/rest/api/v2/alarm
s/details?
alarmFilter=alarmName%2520like%2520'%2525Equipment%2525'and%2520severity%2520%253D%2520'major'%2
520and%2520lastTimeDetected%2520%253E%2520<LastResyncTime>%2520",
      ...
```

Partial resync is a configurable enhancement to utilize the Netcool internal heartbeat interval as `lastTimeDetected` attribute value to retrieve the latest alarms. Netcool and NSP systems must be synchronized for the feature to work accurately. The limitation is that if there is any drift between system times, data could be missed. The probe server time must be in synch with the NSP server time.

Configuring SSL connections

If the Nokia NSP server is using a Secure Socket Layer (SSL) connection to encrypt data exchanged over JMS and HTTP, you will need to configure the truststore for the HTTPS connection on the Netcool/OMNIbus probe server.

To configure the truststore, use the following steps:

1. Obtain the security certificate from the NSP server.
2. Import the security certificate from the NSP server.
3. Verify that the security certificate has been imported into the keystore.

Obtaining a certificate file into the truststore

There are two possible approaches:

1. Obtaining Nokia NSP security certificate from certificate authority (CA)
2. Exporting security certificate file from an existing keystore file from NSP server using the command:

   ```
   ./keytool -export -alias alias_name -keystore keystore_file -storepass
   password -file certificate_file
   ```

   Where:

   alias_name is the keystore alias specified during Nokia NSP keystore generation, for example: NSP_ALIAS.

   keystore_file is the path to and name of the Nokia NSP keystore file, for example: `/opt/nspserver.keystore`.

   password is the Nokia NSP keystore password, for example: the password of `nspserver.keystore`.

   certificate_file is the path to and name of the certificate file to be created, for example: `/opt/nspcert`.

Importing a security certificate into a new or an existing truststore on the Netcool/OMNIbus probe server

To import a certificate file into the truststore, use one of the following steps:

1. For importing the certificate into a new truststore, use the following command:

   ```
   ./keytool -import -trustcacerts -alias new_alias_name -file certificate_file
   -keystore truststore_file -storepass password
   ```

   Note: If the alias does not point to an existing key entry in a truststore file, then keytool assumes you are adding a new trusted certificate entry into truststore file. In this case, the alias should not already exist, otherwise importing fails.
2. For importing the certificate into an existing truststore, use the following command:

   ```
   ./keytool -import -trustcacerts -alias alias_name -file certificate_file
   -keystore truststore_file -storepass password
   ```

   Note: If the alias points to a key entry in a truststore file, then keytool assumes you are importing a certificate reply, replacing old certificate chain with new certificate chain in truststore file.

   Where:

   alias_name is the key entry of the certificate reply. The alias must be the same as that specified during keystore file generation in Nokia NSP server, for example: NSP_ALIAS.

   new_alias_name is the keystore alias of a new keystore, for example: NSP_ALIAS_NEW.

   certificate_file is the path to and name of the certificate file created earlier, for example: `/opt/nspcert`.

   truststore_file is the path to and name of the truststore file that will contain the imported certificate, for example: `/opt/nspserver.truststore`.

   password is the Nokia NSP keystore password, for example: the password of `nspserver.truststore`.

Verifying that the security certificate has been imported into the keystore

To verify that the certificate has been imported into the keystore, use the following command:

```
./keytool -list -v -keystore truststore_file
```

Where:

`truststore_file` is the path to and name of the truststore file generated, for example: `/opt/nfmpserver.trustStore`.

Note: For more details about configuring SSL security for the Nokia NSP server (including instructions about obtaining certificate files) refer to the NSP Installation and Upgrade Guide.

Configuring persistent subscriptions

Persistent subscription enables the probe to use the same subscription ID when it disconnects from Nokia NSP and subsequently restarts. This helps to prevent event loss.

To configure persistent subscriptions, set the probe properties as below and manually create a persistent file `$OMNIHOME/var/nokiaNSP.persistent` file where the `subscriptionId` will be saved:

```
PersistentSubscription: 'true'
PersistentSubscriptionFile    : '/opt/IBM/tivoli/netcool/omnibus/var/nokiaNSP.persistent'
```

Warning: Nokia NSP subscriptions have an expiry time of approximately 60 minutes. If the previous subscription session has expired, the probe cannot resubscribe to the previous session. In this case, the probe creates a new subscription session and there may be event loss while the probe is down. If `InitialResync` is enabled, the probe resyncs from the active alarms list.

Configure `$OMNIHOME/java/conf/nokiaNspKafkaTransport.properties` to use `nokiaNspRestMultiChannelHttpTransportPersistent.json` as shown below:

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

Configure `$OMNIHOME/java/conf/nokiaNspRestMultiChannelHttpTransportPersistent.json` to define the `HOST` and `PORT` as the Nokia SDN actual/virtual IP address and port number respectively.

Configure `$OMNIHOME/java/conf/nokiaNspRestMultiChannelHttpTransportPersistent.json` to remove `DELETE_SUBSCRIPTION_ID` during logout.

```
### REMOVE the DELETE_SUBSCRIPTION_ID block during LOGOUT
"LOGOUT":
   {
        "REVOKE_ACCESS_TOKEN":
        {
            "uri":"https://HOST:PORT/rest-gateway/rest/api/v1/auth/revocation",
            "method":"POST",
            "headers":"Authorization=Basic ++Username++:++Password++,Accept=application/
json,Content-Type=application/x-www-form-urlencoded,Use-Cookie=true,User-Agent=IBM Netcool/
OMNIBus Message Bus Probe",
            "content":"token=++access_token++&token_type_hint=token",
            "contentFile": "",
            "interval":"0",
            "requireSSL":"true",
            "asEventStream" : "false"
        }
   }
```

Note:

If persistent subscription is enabled, if when checking the subscription, the `ACTIVE` stage is returned, the probe will resubscribe to the previous session using the same `subscriptionId`.

If the stage returned is the `CREATING` or `RECREATING` stage, the probe will disconnect. It will then retry the connection or shut down depending on how `RetryCount` is configured. The `CREATING` or `RECREATING` stage indicates that the Nokia NSP subscription is not yet ready.

If the stage returned is `STOPPED`, or the subscription ID is not found, the probe will subscribe with a new subscription session ID because the previous session has expired or cannot be found.

Enabling alarm correlation

Nokia NSP 20.9 and later supports alarm correlation information of multiple level impact and root cause details in the NSP-FAULT topic.

```
"rootCauseAndImpactDetails": {
"status": "causalityDetailsAvailable",
"impacts": {
"alarmFdn": "fdn:model:fm:Alarm:1638",
"impacts": [
                            {
"alarmFdn": "fdn:model:fm:Alarm:1838",
"impacts": [
                                    {
"alarmFdn": "fdn:model:fm:Alarm:1535",
"impacts": []
                                    }
                                ]
                            }
                        ]
                    }
```

To enable root cause and impact detail correlation, set the probe property as below and enable the pre-parser script for the Nokia NSP integration in the `TransformerFile` file: `$OMNIHOME/probes/message_bus_nokia_nfmp_parser_rootCauseImpact.json`

```
TransformerFile                  : '/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/
message_bus_nokia_nfmp_parser_rootCauseImpact.json'
```

In `message_bus_nokia_nfmp_parser_rootCauseImpact.json` modify **`<arch>`** in the following line for your operating system:

```
"preparserScript" : "/opt/IBM/tivoli/netcool/omnibus/probes/<arch>/
message_bus_nokia_nfmp_preparser.js"
```

For example, for Linux, modify the line to:

```
"preparserScript" : "/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/
message_bus_nokia_nfmp_preparser.js"
```

Edit `$OMNIHOME/<arch>/message_bus_nokia_nfmp.rules` and uncomment the following line:

```
# Uncomment below line to enable alarm correlation root cause & impacts
$rootCauseImpactsEnabled = "true"
```

Edit `$OMNIHOME/java/conf/nokiaNspRestMultiChannelHttpTransport.json` to enable root cause and impacts for notification and active alarm resynchronization:

```
"SUBSCRIBE":
   {
        "GET_SUBSCRIPTION":
        {
            "uri":"https://HOST:PORT/nbi-notification/api/v1/notifications/subscriptions",
            "method":"POST",
            "headers":"Authorization=Bearer ++access_token++,Accept=application/json,
                       Content-Type=application/json,Use-Cookie=true,
                       User-Agent=IBM Netcool/OMNIBus Message Bus Probe",
            "content":"{\"categories\":[{\"advancedFilter\": \"{\\
\"includeRootCauseAndImpactDetails\\\":true}\",\"name\":\"NSP-FAULT\"}]}",
            "interval":"0",
            "requireSSL":"true"
        },
 ...

   "RESYNC":
   {
        "RESYNC_FAULT_MANAGEMENT_ALARMS":
        {
            "uri":"https://HOST:PORT/FaultManagement/rest/api/v2/alarms/details/?
includeRootCauseAndImpactDetails=true",
            "method":"GET",
            "headers":"Authorization=Bearer ++access_token++,Accept=application/
stream+json,Content-Type=application/stream+json,Use-Cookie=true,User-Agent=IBM Netcool/OMNIBus
Message Bus Probe",
            "content":"",
            "interval":"0",
            "requireSSL":"true",
            "enablePagination":"false"
        }
   },
...
```

Enabling JSON stream support

Nokia NSP 20.11 and later supports the content type `application/stream+json`. Support for this content type ensures that all alarms can be retrieved in a single GET request.

To support `stream+json`, the `enablePagination` flag has been added to `nokiaNspRestMultiChannelHttpTransport.json` and set to `false` by default to disable the resync bookmark pagination functionality.

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 in IBM Documentation:

https://www.ibm.com/support/knowledgecenter/SSSHTQ_int/omnibus/probes/all_probes/wip/concept/running_probe.html

Additional properties for the probe integration

The Probe integration for Nokia NSP uses the Probe for Message Bus configured by the `message_bus_nokia_nfmp.props` supplied with the probe.

Additional properties supported by the probe use the default settings described in "Properties and command line options" on page 15 and "Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 12.0" on page 20.

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 4. Probe properties and command line options

Property name	Command line option	Description
Cookie *string*	`-cookie` *string*	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 ++*property_setting*++ in the `restWebSocketTransport.properties` 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 ++*property_setting*++ token in the `restWebSocketTransport.properties` file with the cookie value.
EnableSSL *string*	`-noenablessl` (This is equivalent to **EnableSSL** with a value of `false`.) `-enablessl` (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 `WebSocket` **TransportType**.
Host *string*	`-host` *string*	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 `WebSocket`, `Webhook`, or `WebhookV2` **TransportType**. The default is `""`. **Note:** The probe also uses this value to replace the `++Host++` token in the `restWebSocketTransport.properties` file.
JsonMessageDepth *integer*	`-jsonmessagedepth` *integer*	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 *string*	`-jsonnestedheader` *string*	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 `""`.

Table 4. Probe properties and command line options (continued)

Property name	Command line option	Description
JsonNestedPayload *string*	-jsonnestedpayload *string*	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 *string*	-jsonparsername *string*	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 *string*	-keystore *string*	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 "".
KeyStorePassword *string*	-keystorepassword *string*	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 nco_aes_crypt utility within Netcool/OMNIbus.
MessageHeader *string*	-messageheader *string*	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 *string*	-messagepayload *string*	Use this property to specify either XML or the JSON tree structure to the message payload. The default is xml. If this property is set to xml, 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 json 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 4. Probe properties and command line options (continued)

Property name	Command line option	Description
`PartialResync` *string*	`-partialresync` *string*	Use this property to specify that the probe performs a partial resync on startup. If this property is set to `true`, the probe performs a partial resync based on the last event received timestamp stored in a persistent file. The default is `false`. **Note:** This property is only for use with the Probe Integration for Nokia Network Services Platform.
`Password` *string*	`-password` *string*	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 `restWebSocketTransport.properties` file or in the `restWebHookTransport.properties` file.
`Port` *integer*	`-port` *integer*	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 `WebSocket`, `Webhook`, or `WebhookV2` **`TransportType`**. The default is `0`. **Note:** The probe also uses this value to replace the `++Port++` token in the `restWebSocketTransport.properties` file.
`RecordData` *string*	`-recorddata` *string*	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: `WebSocketURL=++URL++`
`StreamCapture` *string*	`-streamcapture` *string*	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`.
`StreamCaptureFile` *string*	`-streamcapturefile` *string*	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`

Table 4. Probe properties and command line options (continued)

Property name	Command line option	Description
`TransformerFile` *string*	`-transformerfile` *string*	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 `$OMNIHOME/java/conf/transformers.xml`. On Windows operating systems, you must specify the full directory path to the file. For example: `'C:\\IBM\\Tivoli\\Netcool\\omnibus\\java\\conf\\transformers.xml'`
`TransportFile` *string*	`-transportfile` *string*	Use this property to specify the location of the transport properties file. On UNIX and Linux operating systems, the default is `$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'`
`TransportQueueSize` *integer*	`-transportqueuesize` *integer*	Use this property to increase the queue size for raw events to prevent events from being discarded due to a full queue. The default is `2000`.
`TransportType` *string*	`-transporttype` *string*	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: • `EventSource` • `File` • `HTTP` • `JMS` • `KAFKA` • `MQTT` • `PULSAR` • `Socket` • `Webhook` • `WebhookV2` • `WebSocket` The default is `JMS`.

Table 4. Probe properties and command line options (continued)

Property name	Command line option	Description
`Username` *string*	`-username` *string*	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 12.0

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 12.0.

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

Table 5. Properties and command line options

Property name	Command line option	Description
`DataBackupFile` *string*	`-databackupfile` *string*	Use this property to specify the path to the file that stores data between probe sessions. The default is `""`. **Note:** Specify the path relative to `$OMNIHOME/var`.
`HeartbeatInterval` *integer*	`-heartbeatinterval` *integer*	Use this property to specify the frequency (in seconds) with which the probe checks the status of the host server. The default is `1`.
`Inactivity` *integer*	`-inactivity` *integer*	Use this property to specify the length of time (in seconds) that the probe allows the port to receive no incoming data before disconnecting. The default is `0` (which instructs the probe to not disconnect during periods of inactivity).

Table 5. Properties and command line options (continued)

Property name	**Command line option**	**Description**
`InactivityAction` *string*	`-inactivityaction` *string*	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`.
`InitialResync` *string*	`-initialresync` *string*	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` *integer*	`-maxeventqueuesize` *integer*	Use this property to specify the maximum number of events that can be queued between the non native process and the ObjectServer. The default is `0`. **Note:** You can increase this number to increase the event throughput when a large number of events is generated.

Table 5. Properties and command line options (continued)

Property name	Command line option	Description
`ResyncInterval` *integer*	`-resyncinterval` *integer*	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` *integer*	`-retrycount` *integer*	Use this property to specify how many times the probe attempts to retry a connection before shutting down. The default is `0` (which instructs the probe to not retry the connection).
`RetryInterval` *integer*	`-retryinterval` *integer*	Use this property to specify the length of time (in seconds) that the probe waits between successive connection attempts to the target system. The default is `0` (which instructs the probe to use an exponentially increasing period between successive connection attempts, for example, the probe will wait for `1` second, then `2` seconds, then `4` seconds, and so forth).
`RotateEndpoint` *string*	`-rotateendpoint` *string*	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: `false`: The probe does not attempt to connect to another endpoint if the connection to the first endpoint fails. `true`: The probe attempts to connect to another endpoint if the connection to the first endpoint fails. The default is `false`.

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 Nokia NSP are contained in the `message_bus_nokia_nfmp.rules` file.

Resync elements

The following elements are generated from Nokia NSP resync events:

Table 6. Resynch elements

Element name	**Element description**
`$resync_event`	This element indicates whether this is a resync event.
`$originalSeverity`	This element shows the original severity of the alarm.
`$lastTimeAcknowledged`	This element shows the last time at which the alarm was acknowledged.
`$neId`	This element displays the IP address of the network element to which the alarm applies.
`$acknowledged`	This element indicates whether the alarm has been acknowledged.
`$userText`	This element displays an text added to the alarm by the user.
`$sourceSystem`	This element displays the source system to which the alarm applies.
`$additionalText`	This element shows any additional text related to the alarm.
`$affectedObject`	This element displays the network ID of the object affected.
`$acknowledgedBy`	This element displays the name of the user who acknowledged the alarm.
`$lastTimeCleared`	This element shows the last time at which the alarm was last cleared.
`$neName`	This element displays the name of the network element to which the alarm applies.
`$deletedBy`	This element displays the name of the user who deleted the alarm.
`$probableCause`	This element shows the probable cause of the alarm.

Table 6. Resynch elements (continued)

Element name	**Element description**
`$firstTimeDetected`	This element shows the time at which the alarm was first detected.
`$adminState`	This element displays the administration state of the alarm.
`$rootCause`	This element shows the root cause of the alarm.
`$numberOfOccurrencesSinceAck`	This element shows the number of times that the alarm has occurred since it was acknowledged.
`$nodeTimeOffset`	This element shows the time offset of the node.
`$severity`	This element displays the severity of the alarm.
`$affectedObjectName`	This element displays the name of the object affected.
`$clearedBy`	This element displays the name of the user who cleared the alarm.
`$numberOfOccurrences`	This element shows the number of times that the alarm has occurred.
`$serviceAffecting`	This element indicates whether the problem reported is affecting service.
`$impact`	This element indicates the impact of the alarm.
`$implicitlyCleared`	This element indicates whether the alarm has been cleared implicitly.
`$alarmName`	This element displays the name of the alarm.
`$wasAcknowledged`	This element indicates whether the alarm was acknowledged.
`$numberOfOccurrencesSinceClear`	This element shows the number of times that the alarm has occurred since it was cleared.
`$objectFullName`	This element displays the full name of the object affected by the alarm.
`$previousSeverity`	This element shows the previous severity of the alarm.
`$highestSeverity`	This element shows the highest severity that has been reported for this alarm.
`$affectedObjectType`	This element displays the type of the object affected.
`$fdn`	This element displays the fully distinguished name of the object affected.

Table 6. Resynch elements (continued)

Element name	Element description
`$alarmType`	This element displays the type of the alarm.
`$specificProblem`	This element indicates the specific problem being reported.
`$sourceType`	This element indicates the type of the alarm source.
`$lastTimeSeverityChanged`	This element shows the time at which the severity last changed.
`$lastTimeDetected`	This element shows the time at which the alarm was last detected.

Notification elements

The following elements are generated from Nokia NSP notification events:

Table 7. Notification elements

Element name	Element description
`$resync_event`	This element indicates whether this is a resync event.
`$nsp-faultalarm-create.acknowledged`	This element indicates whether the alarm has been acknowledged.
`$nsp-faultalarm-create.acknowledgedBy`	This element displays the name of the user who acknowledged the alarm.
`$nsp-faultalarm-create.sourceSystem`	This element displays the source system to which the alarm applies.
`$nsp-faultalarm-create.previousSeverity`	This element shows the previous severity of the alarm.
`$nsp-faultalarm-create.objectId`	This element displays the fully distinguished name of the object.
`$nsp-faultalarm-create.additionalText`	This element shows any additional text related to the alarm.
`$nsp-faultalarm-create.numberOfOccurrencesSinceClear`	This element shows the number of times that the alarm has occurred since it was cleared.
`$nsp-faultalarm-create.affectedObjectType`	This element displays the type of the object affected.
`$nsp-faultalarm-create.implicitlyCleared`	This element indicates whether the alarm has been cleared implicitly.

Table 7. Notification elements (continued)

Element name	**Element description**
`$nsp-faultalarm-create.numberOfOccurrences`	This element shows the number of times that the alarm has occurred.
`$nsp-faultalarm-create.alarmName`	This element displays the name of the alarm.
`$nsp-faultalarm-create.serviceAffecting`	This element indicates whether the problem reported is affecting service.
`$nsp-faultalarm-create.severity`	This element displays the severity of the alarm.
`$nsp-faultalarm-create.firstTimeDetected`	This element shows the time at which the severity was first detected.
`$nsp-faultalarm-create.probableCause`	This element shows the probable cause of the alarm.
`$nsp-faultalarm-create.affectedObjectName`	This element displays the name of the object affected.
`$nsp-faultalarm-create.affectedObject`	This element displays the network ID of the object affected.
`$eventTime`	This element displays the time of the event.
`$nsp-faultalarm-create.clearedBy`	This element displays the name of the user who cleared the alarm.
`$nsp-faultalarm-create.numberOfOccurrencesSinceAck`	This element shows the number of times that the alarm has occurred since it was acknowledged.
`$nsp-faultalarm-create.lastTimeDetected`	This element shows the time at which the alarm was last detected.
`$nsp-faultalarm-create.sourceType`	This element indicates the type of the alarm source.
`$nsp-faultalarm-create.rootCause`	This element shows the root cause of the alarm.
`$nsp-faultalarm-create.originalSeverity`	This element shows the original severity of the alarm.
`$nsp-faultalarm-create.specificProblem`	This element indicates the specific problem being reported.
`$nsp-faultalarm-create.neId`	This element displays the IP address of the network element to which the alarm applies.
`$nsp-faultalarm-create.wasAcknowledged`	This element indicates whether the alarm was acknowledged.

Table 7. Notification elements (continued)

Element name	**Element description**
`$nsp-faultalarm-create.objectFullName`	This element displays the full name of the object affected by the alarm.
`$nsp-faultalarm-create.nodeTimeOffset`	This element shows the time offset of the node.
`$nsp-faultalarm-create.neName`	This element displays the name of the network element to which the alarm applies.
`$nsp-faultalarm-create.lastTimeSeverityChanged`	This element shows the time at which the severity last changed.
`$nsp-faultalarm-create.highestSeverity`	This element shows the highest severity that has been reported for this alarm.
`$nsp-faultalarm-create.lastTimeCleared`	This element shows the last time at which the alarm was last cleared.
`$nsp-faultalarm-create.lastTimeAcknowledged`	This element shows the last time at which the alarm was acknowledged.
`$nsp-faultalarm-create.adminState`	This element displays the administration state of the alarm.
`$nsp-faultalarm-create.alarmType`	This element displays the type of the alarm.
`$nsp-faultalarm-create.userText`	This element displays an text added to the alarm by the user.

Notification alarm change elements

The following elements are generated from Nokia NSP notification alarm change events:

Table 8. Notification alarm change elements

Element name	**Element description**
`$resync_event`	This element indicates whether this is a resync event.
`$nsp-faultalarm-change.numberOfOccurrences.old-value`	If the number of times that the alarm has occurred has changed, this element shows the old value reported.
`$nsp-faultalarm-change.numberOfOccurrencesSinceClear.old-value`	If the number of times that the alarm has occurred since it was cleared has changed, this element shows the old value reported.
`$nsp-faultalarm-change.lastTimeDetected.old-value`	If the time at which the alarm was last detected has changed, this element shows the old value reported.

Table 8. Notification alarm change elements (continued)

Element name	Element description
`$nsp-faultalarm-change.lastTimeDetected.new-value`	If the time at which the alarm was last detected has changed, this element shows the new value reported.
`$nsp-faultalarm-change.objectId`	This element displays the fully distinguished name of the object.
`$eventTime`	This element displays the time of the event.
`$nsp-faultalarm-change.numberOfOccurrencesSinceClear.new-value`	If the number of times that the alarm has occurred since it was cleared has changed, this element shows the new value reported.
`$nsp-faultalarm-change.numberOfOccurrences.new-value`	If the number of times that the alarm has occurred has changed, this element shows the new value reported.

Error messages

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

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

Table 9. Error messages

Error	Description	Action
`Failed to startup probe`	The probe failed to start, probably due to an invalid combination of properties set in the `message_bus.props` 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 `message_bus.props` file and that the file has the appropriate permissions set.

Table 9. Error messages (continued)

Error	**Description**	**Action**
`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.

Error messages generated for the Probe Integration for Nokia NSP

The following table describes the error messages specific to the Probe Integration for Nokia NSP.

Table 10. Error messages

Error	**Description**	**Action**

`KeeperException thrown when adding watcher for (/brokers/ids): KeeperErrorCode = NoNode for /brokers/ids`
 	
`KeeperException thrown when adding watcher for (/brokers/topics): KeeperErrorCode = NoNode for /brokers/topics`
	The given znode (`/brokers/ids` or `/brokers/topics`) cannot be not found in the system.	Check the configuration of the Kafka server.

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 11. 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.

Known issues

This section explains known issues with this probe.

NSP-PACKET-ALL topic not supported

The current rules file used in certification supports the `NSP-FAULT` topic, but it does not properly support the `NSP-PACKET-ALL` topic.

Probe behavior regarding the configurable connection retry interval

Currently the **`RetryInterval`** property is used for both failover and disconnection/reconnection, thus it is serving two different uses and you will need to adjust its setting accordingly.

During normal disconnection and reconnection, you might want the probe to reconnect in a shorter interval to avoid event loss. But during failover, the probe might require to wait longer for reconnection due to the waiting buffer before the NSP Server has successfully failed over.

Using cross-launch with Internet Explorer

The probe is unable to cross-launch using the Internet Explorer browser from the **Active Event List** because Internet Explorer is no longer supported by NSP 21.6. Use the **Event Viewer** for cross-launch instead.

Alarms from Nokia NSP integration showing timestamps incorrectly

This is due to Netcool interpreting timestamps as seconds while NSP sends them as milliseconds.

This issue will be fixed in the Probe for Message Bus version 21.

Workaround

Make the following changes in the `$OMNIHOME/probes/<arch>/message_bus_nokia_nfmp.rules` file:

```
# Find below lines and modify them to
@FirstOccurrence = real($firstTimeDetected)/1000
```

```
@LastOccurrence = real($lastTimeDetected)/1000

# Find below lines and modify them to
@FirstOccurrence = real($(nsp-fault:alarm-create.firstTimeDetected))/1000
@LastOccurrence = real($(nsp-fault:alarm-create.lastTimeDetected))/1000
```

Kafka connection requires TLS host verification

By default, the Kafka connection requires TLS host verification which means the certificate created on the Nokia SDN host must match the host IP address or hostname. If in the environment the certificate does not match with the IP or hostname, the probe write the following connection exception to the error log:

```
Error: E-JPR-000-000: [KafkaConsumerRunnable] Exception when kafka consumer is running:
org.apache.kafka.common.errors.SslAuthenticationException: SSL handshake failed
```

Workaround

If you want to disable the TLS host verification on Kafka, configure the following lines in `$OMNIHOME/java/conf/nokiaNspKafkaClient.properties`:

```
## Uncomment below lines if want to disable TLS host verification
ssl.endpoint.identification.algorithm=
```

Important: You should not disable TLS host verification in a production environment due to security concerns.

Alarm details truncated

Alarms from Nokia NSP the `additionalText` that displayed in the Event Viewer's **Details** tab will be truncated to 256 char. You can enlarge the field size of alerts.details table to 1024 or 2048 char.

The probe fails to record an error

If the probe receives an HTTP response of `200` but 0 rows of alarms during the get active alarm request, it will not treat this as an error and will proceed to the Kafka notification pull. The probe might miss the active alarm resynchronization in this use case. This may happen after a failover if the REST API component is available for requests before alarm data is ready, and will be dependent on the timing of the request.

Workaround

You should use persistent subscriptions. In this case, the probe can resubscribe to its previous persistent subscription. There will be no event lost or missed notification in this scenario, provided the subscription has not expired (this applies to all disconnect scenarios). However, you can also execute a dynamic resynchronization through the `nco_http` command interface in probe if required.

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

Trademarks

Part Number:

(1P) P/N:

SC27-9589-04