RSA

# RSA® NetWitness

Version 11.7

# NetWitness Export Connector - Installation and Configuration Guide

NETWITNESS

## Contact Information

RSA Link at https://community.rsa.com contains a knowledge base that answers common questions and provides solutions to known problems, product documentation, community discussions, and case management.

## Trademarks



## License Agreement



## Third-Party Licenses



## Note on Encryption Technologies



## Distribution



January 2022

# Contents


**Overview** ..... **4**
Work Flow of NetWitness Export Connector ..... 4
**Configuration Process** ..... **6**
**VM Sizing Recommendations** ..... **7**
Log Decoder ..... 7
Network Decoder ..... 8
**Install Logstash** ..... **10**
Troubleshooting ..... 10
**Install NetWitness Export Connector** ..... **11**
**Configure NetWitness Export Connector** ..... **12**
Position tracking and start session ..... 16
Configure SSL ..... 16
Certificate and Keystore ..... 17
Health and Wellness ..... 18
Download and Install Dashboard. ..... 20
Create a User in New Health & Wellness (Kibana) ..... 21
Enable the Logstash Plugin Metrics ..... 21
Enable the Logstash Host Metrics ..... 22
Configuring Custom Multi-valued Meta ..... 24
**Configure Logstash Filter Plugin (optional)** ..... **25**
**Configure Logstash Output Plugin** ..... **26**
Performance tuning for Kafka and Kafka Output Plugin ..... 26
**Known Issues** ..... **27**

# Overview

The NetWitness Export Connector is an input plugin for Logstash, used to export NetWitness Platform events and routes the data where you want, all in continuous, streaming fashion. Giving you the flexibility to unlock a variety of downstream use cases.

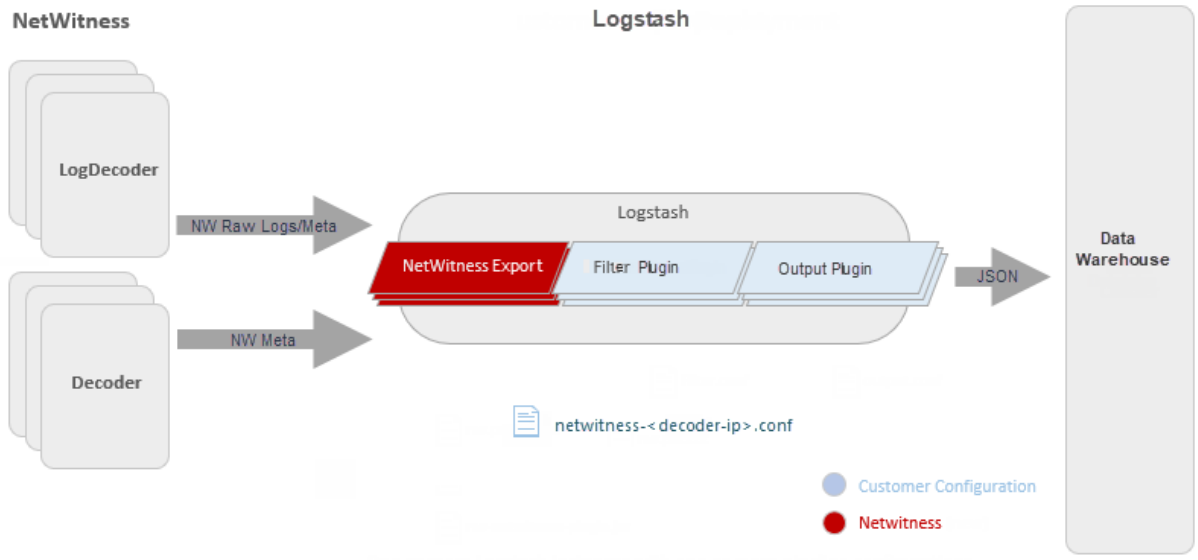
This plugin is installed on Logstash and integrates with NetWitness Decoders and Log Decoders. This plugin aggregates meta data and raw logs from the Decoder or Log Decoder and converts it to Logstash JSON object, which can easily integrate with numerous consumers such as Kafka, AWS S3, TCP, Elastic and others.

Install NetWitness Export Connector on the Logstash service. To activate the connector, restart the Logstash service.

**Note:** From 11.6 onwards, the Logstash server is packaged and supported along with the NetWitness Log Collector or Virtual Log Collector (VLC) service to provide easy access to Logstash. This is referred to as Managed Logstash and it eliminates the need for a separate Logstash server outside of the NetWitness Platform. For more information, see "Configure Logstash Event Sources in NetWitness" in the *Log Collection Configuration Guide*.

## Work Flow of NetWitness Export Connector

Following diagram shows how NetWitness Export Connector works.




There are of three plugins available that helps with export.

- Input plugin
- Filter plugin (optional)
- Output plugin

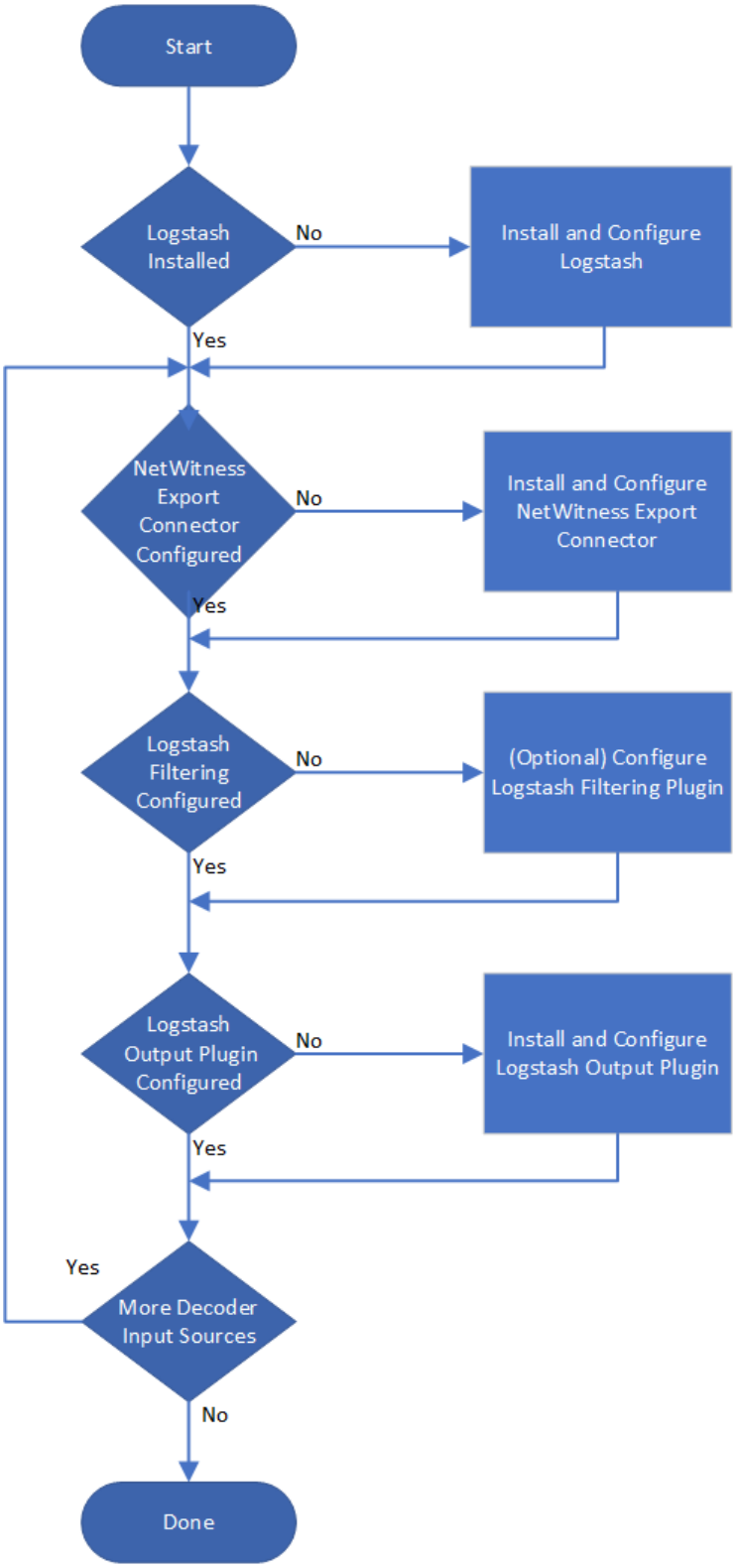
1. The Input plugin collects the events from the event sources. You must install the NetWitness Export Connector to collect events from Decoder or Log Decoder. The NetWitness Export Connector uses NetWitness API that collects the following data and forwards it as Logstash messages.
   - Meta data and raw log data from the Decoder
   - Meta data from Log Decoder

The data is then forwarded to the Filter plugin.

2. (Optional) The Filter plugin adds, removes, or modifies the received data and forwards it to the Output plugin. You can use the standard Logstash filter plugins to add, remove, or modify the data.
3. The Output plugin sends the processed event data to the data warehouse destinations. You can use the standard Logstash output plugins to send the data.

# Configuration Process

The following flowchart describes the steps to configure NetWitness Export Connector.



# VM Sizing Recommendations

It is recommended to install the Logstash and the NetWitness Export Connector in an independent VM (Virtual Machine) deployment. Following are the requirements for different variations of EPS (Events Per Second).

**Total VM Memory:** Set an additional 4GB memory for the JVM (Java Virtual Machine) memory mentioned in the following table for the respective EPS. For Example, for 5K EPS, the JVM memory required is 4GB, the total VM memory must be set to 8 GB.

**JVM Memory:** The Xms and Xmx for the Logstash service must be set to the same value as shown in the table. You can update the JVM settings for Logstash in the `/etc/logstash/jvm.options` file. For more information, see JVM Settings documentation.

For more info, https://www.elastic.co/guide/en/logstash/7.x/jvm-settings.html

**vCPUs:** The vCPUs values that are shown is required only for NetWitness Export Connector to be functional for respective EPS. It is recommended to add extra 4 CPUs for other OS (Operating System) services. The recommended vCPU specifications for all the EPS listed in the following tables are: Intel Xeon CPU @2.6 Ghz.

**Note:** The above recommendations is valid only if the VM instance is completely dedicated for Logstash deployment.
The resources consumption is dependent on cumulative EPS irrespective of number of sources. For Example, if you have three Decoders of 5000 EPS each, the cumulative EPS is 15000 as a result we can use the sizing option of 15000 EPS instead of 5000 EPS 3 times.

## Log Decoder

- Metadata over SSL (~50 meta keys)

| EPS | vCPUs | JVM Memory |
|---|---|---|
| 5000 | 4 | 4 GB |
| 10000 | 4 | 4 GB |
| 15000 | 8 | 8 GB |
| 20000 | 8 | 8 GB |
| 25000 | 14 | 12 GB |
| 30000 | 14 | 12 GB |
| 60000 | 16 | 32 GB |

- Metadata over SSL (~140 meta keys)

| EPS | vCPUs | JVM Memory |
|---|---|---|
| 5000 | 8 | 4 GB |
| 10000 | 11 | 4 GB |
| 15000 | 14 | 8 GB |
| 20000 | 17 | 12 GB |
| 25000 | 19 | 16 GB |
| 30000 | 19 | 16 GB |

- Meta data and raw logs over SSL (~50 meta keys)

| EPS | vCPUs | JVM Memory |
|---|---|---|
| 5000 | 4 | 4 GB |
| 10000 | 8 | 4 GB |
| 15000 | 10 | 8 GB |
| 20000 | 12 | 12 GB |
| 25000 | 15 | 24 GB |
| 30000 | 18 | 24 GB |
| 60000 | 23 | 48 GB |

- Meta data and raw logs over SSL (~140 meta keys)

| EPS | vCPUs | JVM Memory |
|---|---|---|
| 5000 | 8 | 4 GB |
| 10000 | 10 | 8 GB |
| 15000 | 17 | 10 GB |
| 20000 | 19 | 16 GB |
| 25000 | 20 | 32 GB |
| 30000 | 20 | 32 GB |

## Network Decoder

- Meta data over SSL

| EPS | vCPUs | JVM Memory |
|---|---|---|
| 500 Mbps | 4 | 2 GB |
| 1 Gbps | 4 | 2 GB |
| 1.5 Gbps | 8 | 4 GB |
| 2 Gbps | 8 | 4 GB |
| 3.4 Gbps | 12 | 8 GB |
| 6 Gbps (10G Decoder) | 12 | 24 GB |
| 8 Gbps | 15 | 32 GB |
| 9.4 Gbps | 15 | 48 GB |

# Install Logstash

**IMPORTANT:** Ensure that you follow all the security-related best practices and guidelines outlined in the Logstash documentation to avoid any potential security risks.

The NetWitness Export Connector is an input plugin for Logstash. Hence, installation of Logstash is important for NetWitness Export Connector to work. For more information, see the Overview section.

You can install the open source version of Logstash (OSS) or the paid version (Elastic). The supported version is Logstash 7.6.2.

Information on released versions of Logstash is available at Logstash Reference.

**Note:** It is recommended to install the Logstash service in CentOS operating system for better results.

Do the following steps to Install the Logstash.

1. Install the service: Installing Logstash
2. After installation, set Logstash to run as a service: Running Logstash
3. Enable Logstash to start when the system boots up: https://www.unix.com/man-page/centos/1/systemctl/

If you are using CentOS, make a note of the following:

- Logstash logs are stored in `/var/log/logstash/logstash-plain.log`
- If you install Logstash using rpm install, make sure it installs as logstash user and folders get created with the same user and **not** the root user.

## Troubleshooting

For Generic Troubleshooting Instructions for Logstash, see Logstash Troubleshooting.

# Install NetWitness Export Connector

**Note:** From 11.6 onwards, the Logstash server is packaged and supported along with the NetWitness Log Collector or Virtual Log Collector (VLC) service to provide easy access to Logstash. This is referred to as Managed Logstash and it eliminates the need for a separate Logstash server outside of the NetWitness Platform. For more information, see "Configure Logstash Event Sources in NetWitness" in the *Log Collection Configuration Guide*.

Do the following steps to install NetWitness Export Connector .

1. Download the offline installer from RSA Link in the following location: NetWitness Export Connector Installer.
2. Copy the downloaded NetWitness ZIP archive to the system where Logstash runs.
3. Open a command prompt and run the following command to change directory to Logstash home.
   `cd /usr/share/logstash`
4. Check the status of the Logstash service by running the following command.
   `systemctl status logstash`
5. Stop the Logstash service by running the following command.
   `systemctl stop logstash`
6. Install the NetWitness Export Connector by running the following command.
   `bin/logstash-plugin install file:///<path-to-file>/netwitness-export-connector-x.x.x.zip`
7. Make sure that all the required configuration files (`netwitness-<decoder-ip>-input.conf`) are available in the following folder.
   `/etc/logstash/conf.d/`
8. Start the Logstash service by running the following command.
   `systemctl start logstash`

# Configure NetWitness Export Connector

**Note:** Make sure you open the firewall of the Decoder or Log Decoder to establish connection with the Logstash. For more information, see "Network Architecture and Ports" in *Deployment Guide for RSA NetWitness*.

You must configure the Logstash configuration file to process the NetWitness events. Create a Logstash configuration file and add the NetWitness Export Connector plugin parameter settings for event processing. Save the file as `netwitness-<decoder-ip>-input.conf`. After adding the NetWitness Export Connector plugin parameter settings, place the configuration file in `/etc/logstash/conf.d/` location.

A Logstash configuration file can have three separate sections for each type of plugin that you want to add to the event processing pipeline. The first section is for Input plugin (NetWitness Export Connector), the second section is for Filter plugin (optional) and the third section is for Output plugin.

To configure the NetWitness Export Connector plugin, add the parameter settings in the first section the Logstash configuration file.

For multiple pipelines configuration, see Multiple Pipelines Configuration documentation.

The configuration of each NetWitness Export Connector plugin must consist of the plugin name followed by a block of parameter settings for that plugin. If the NetWitness Export Connector has multiple plugins with block of parameters, they are applied in the order of their appearance.

The following is an example of NetWitness Export Connector with one plugin instance with block of parameter settings which fetches data from a single decoder .

```
input {
 netwitness {
 host => "<host>" # Mandatory
 username => "<username>" # Mandatory
 password => "<password>" # Mandatory
 decoder_type => "logdecoder" # Mandatory
 }
}
```

The following is an example of NetWitness Export Connector with two plugin instances with block of parameter settings which fetches data from two different decoders. Each plugin in the configuration is applied in the order as shown.

```
input {
 netwitness {
 host => "<host>" # Mandatory
 username => "<username>" # Mandatory
 password => "<password>" # Mandatory
 decoder_type => "logdecoder" # Mandatory
 }
netwitness {
 host => "<host>" # Mandatory
 username => "<username>" # Mandatory
 password => "<password>" # Mandatory
 decoder_type => "logdecoder" # Mandatory
 }
}
```

**Note:** When configuring the Logstash, you may need to specify sensitive settings such as passwords. You can use the Logstash keystore to securely store secret values instead of file system permissions for using it in configuration settings. For more information, see Logstash keystore Documentation.

Following are the parameters accepted by NetWitness Export Connector.

| Parameter | Description | Parameter Type | Default Value |
|---|---|---|---|
| host | IP address or hostname of the Decoder or Log Decoder (mandatory) | String | N/A |
| username | Username used to access the Decoder or Log Decoder (mandatory) | String | N/A |
| password | Password of the user (mandatory) | String | N/A |
| decoder_type | Accepts only 'decoder' or 'logdecoder' (mandatory) | String | N/A |
| ssl_enable | Enable SSL connection between the Decoders and the NetWitness Export Connector. For more information, see Configure SSL | Boolean | false |
| ssl_certificate_path | Path of the CA certificate that is used for SSL and Trusted Connections | | /etc/pki/nw/trust/truststore.pem |
| ssl_certificate_password | Password of the certificate in use. Mandatory if SSL is enabled | String | N/A |
| ssl_client_certificate_path | Client's SSL certificate | | /etc/pki/nw/node/node-cert.pem |
| ssl_version | Version of the SSL connection | String | TLSv1.2 |
| plugin_metrics_enable | Enables metrics reporting to Elastic (New Health and Wellness), for more information see Health and Wellness | Boolean | false |
| elastic_host | IP address or hostname of the Elastic host. Mandatory if plugin metrics is enabled | String | N/A |
| elastic_port | Port number of the Elastic host | String | 9200 |

| Parameter | Description | Parameter Type | Default Value |
|---|---|---|---|
| elastic_username | The username that is used to access the Elastic host. Mandatory if plugin metrics is enabled | String | N/A |
| elastic_password | The password that is used to access the Elastic host. Mandatory if plugin metrics is enabled | String | N/A |
| meta_include | Aggregates only the meta keys that are added in this parameter setting. Accepts comma separated values (csv) format | String | nil |
| meta_exclude | Excludes the meta keys that are added in this parameter setting from aggregation. Accepts comma separated values (csv) format | String | nil |
| start_session | Session from which the aggregation starts. Setting the value to 0 starts the aggregation from `last.session.id` in the Decoder | Number | 0 |
| export_log | Includes the raw log with the meta in the session (applicable only for Log Decoder aggregation) | Boolean | false |
| aggregate_ sessions | Number of sessions ingested in a batch from the Decoders | Number | 1000 |
| aggregation_ interval | Time interval (in milliseconds) between two event cycles | Number | 1000 |
| prefetch_count | Controls the number of batches to be pulled into the buffer that is available for the plugin to collect | Number | 2 |

| Parameter | Description | Parameter Type | Default Value |
|---|---|---|---|
| compression | The number of bytes in each message before message is compressed. Setting the value to '0' does not allow compression. Ranges between 0 to 131071 | Number | 0 |
| compression_ level | The level of compression. Ranges between 0 to 9 where 1 is fastest and 9 has the better compression. A value of 0 selects the best balance between speed and compression | Number | 6 |
| buffer_size | Controls the number of records that the stream in the buffer before it is ingested | Number | 40000 |
| position_ tracking_path | Path where the last consumed session id is stored. Default storage location is `/var/lib/logstash`. For more information, see Position tracking and start session | String | `/var/lib/logstash` |
| custom_meta_ config_path | Path to the custom multivalued meta configuration file. For more information, see Configuring Custom Multi-valued Meta | String | nil |
| Query | Takes any NetWitness Platform query as Input<br>**Note:** Only Indexed meta key must be the part of the query. For example, select `* where device.type = 'rhlinux'` | String | Select * |

## Position tracking and start session

Position tracking or bookmarking is used to track the sessions that are aggregated by logstash and sent to the consumer. Position tracking initiates automatically and updates the tracking file every 60 seconds in the path mentioned in `position_tracking_path` parameter in the configuration file. The file consists of two parts `[sessionid,timestamp]`. Default location is `/var/lib/logstash`, if it is not mentioned in the configuration file.

The `start_session` parameter accepts a number (long: primitive datatype) and indicates which is the first `sessionid` the plugin should request from its corresponding source. if the `start_session` parameter is not mentioned in the configuration file or if the value is mentioned as 0, the first session requested by the plugin will be the `last.session.id + 1`, `last_session_id` as in the decoder's REST API `/database/stats/last.seesion.id`.

if the position tracking file exists for a source and `start_session` is not configured or the `start_session` is set as value 0, the plugin with initiate aggregation from the `sessionid` indicated in the position tracking file.

if the position tracking file exists for a source and the `start_session` is set as non-zero value, the `start_session` value will take precedence over the position tracking file. The plugin will request from the session mentioned in the `start_session` parameter onwards.

## Configure SSL

**Note:** When configuring the Logstash, you may need to specify sensitive settings such as passwords. You can use the Logstash keystore to securely store secret values instead of file system permissions for using it in configuration settings. For more information, see Logstash keystore Documentation.

To support trusted connections, the Decoder or Log Decoder has two ports, an unencrypted non-SSL port and an encrypted SSL port. Trusted connections require the encrypted SSL port.

To establish trusted connection for the Decoder or Log Decoder with Logstash, add the following parameters in the Logstash Configuration file (`netwitness-<decoder-ip>-input.conf`) to enable the SSL mode.

| Parameters | Settings | Parameter Type | Default Value |
|---|---|---|---|
| ssl_enable | Set the value to 'true' to enable SSL | Boolean | false |
| ssl_certificate_path | Enter the path of the Logstash keystore file. The keystore file must be in `.p12` format | String | N/A |
| ssl_certificate_password | Enter the password of the keystore | String | N/A |

## Certificate and Keystore

The NetWitness Export Connector must have a valid CA (Certificate Authority) certificate and a server certificate to establish a trusted connection with Decoders or Log Decoders.

**Create a CA certificate**

To create a CA certificate, do the following steps.

1. SSH to NW Logstash host.
2. Change the directory to /root by running the following command.
   `cd /root`
3. Create a private key of 2048 bits by running the following command.
   `openssl genrsa -out CA-key.pem 2048`
4. Create a CA certificate by running the following command.
   `openssl req -new -key CA-key.pem -x509 -days 1000 -out CA-cert.pem`

**Create Certificate Signing Request (CSR) and Keystore**

Do the following steps provided in this procedure to create a CSR for server and Keystore for the NetWitness Export Connector. You must submit the CSR to the Certificate Authority (CA) server to obtain a server certificate. Once the server certificate is created, do the following steps to package the private key and the signed certificate that must be uploaded to the Logstash keystore.

To create a CSR, do the following steps.

**Note:** You can skip till step 6 if you have PEM already available.

1. SSH to NW Logstash host.
2. Change the directory to /root by running the following command.
   `cd /root`
3. Create a private key of 2048 bits by running the following command.
   `openssl genrsa -out server-key.pem 2048`
4. Create a CSR by running the following command.
   `openssl req -new -key server-key.pem -out signingReq.csr`
5. Submit the CSR to the CA and get a signed server certificate by running the following command.
   `openssl x509 -req -days 1000 -in signingReq.csr -CA CA-cert.pem -CAkey CA-key.pem -CAcreateserial -out server-cert.pem`
6. Create a Keystore for the NetWitness Export Connector by running the following command.
   `- openssl pkcs12 -export -in server-cert.pem -inkey server-key.pem -certfile CA-cert.pem -out logstash-input-netwitness.p12`
7. Copy the `/etc/pki/nw/trust/truststore.pem` from the Decoder or Log Decoder to the Logstash host by running the following command.
   `scp /etc/pki/nw/trust/truststore.pem <logstash ssh username>@<logstash destination directory path>`
8. Import `truststore.pem` from the Decoder or Log Decoder to the Logstash keystore (`logstash-input-netwitness.p12`) by running the following command.

```
keytool -importcert -keystore logstash-input-netwitness.p12 -trustcacerts -
alias nw-inter -file truststore.pem -storetype PKCS12
```

**Note:** `truststore.pem` is same for all Decoders, if the Logstash is aggregating from same NetWitness setup. If the Logstash is aggregating from two different NetWitness setups, you must copy the `truststore.pem` for each Decoders.

9. Copy `logstash-input-netwitness.p12` file to the `/etc/logstash` directory by running the following command.
   ```
   cp logstash-input-netwitness.p12 /etc/logstash
   ```
10. Change the owner of `logstash-input-netwitness.p12` as Logstash by running the following command.
    ```
    chown logstash:logstash /etc/logstash/logstash-input-netwitness.p12
    ```

**Note:** You must use the same keystore "`logstash-input-netwitness.p12`" for all NetWitness hosts that is integrated with Logstash.

11. Connect to the Decoders to add the `server-cert.pem` to `trustpeer` and `caupload` APIs by running the following commands. You can also use to REST API port to connect.
    ```
    curl -X POST -d server-cert.pem http://<logdecoder or decoder
    IP>:<logdecoder or decoder REST port>/sys/trustpeer
    curl -X POST -d server-cert.pem http:// <logdecoder or decoder
    IP>:<logdecoder or decoder REST port>/sys/caupload
    ```

**Note:** If you are not able to use the REST API, follow the below steps to copy the certificates to `/sys peerCert` and `/sys caCert` APIs using NetWitness CLI .
1. SSH to Admin Server.
2. Type "help" for a list of commands or "man" for a list of manual pages.
3. Run the following command.
```
login localhost:50002 <username> <password>
```
3. Run the following commands.
```
send /sys peerCert op=add --file-data=/root/server-cert.pem
send /sys caCert op=add --file-data=/root/server-cert.pem
```

12. Restart the Logstash service running the following command.
    ```
    systemctl restart logstash
    ```

**Note:** If you are using an untrusted certificate, copy the `truststore.pem` file from the Decoder or Log Decoder and import it to `<JAVA_HOME>/lib/security/cacerts` for the Logstash service to trust the CA certificate of Decoder or Log Decoder. For example: `keytool -importcert -file </path/to/file/>truststore.pem -keystore /usr/java/jdk-X.Y.Z/lib/security/cacerts -alias nw-core-cert -storepass <password>`

## Health and Wellness

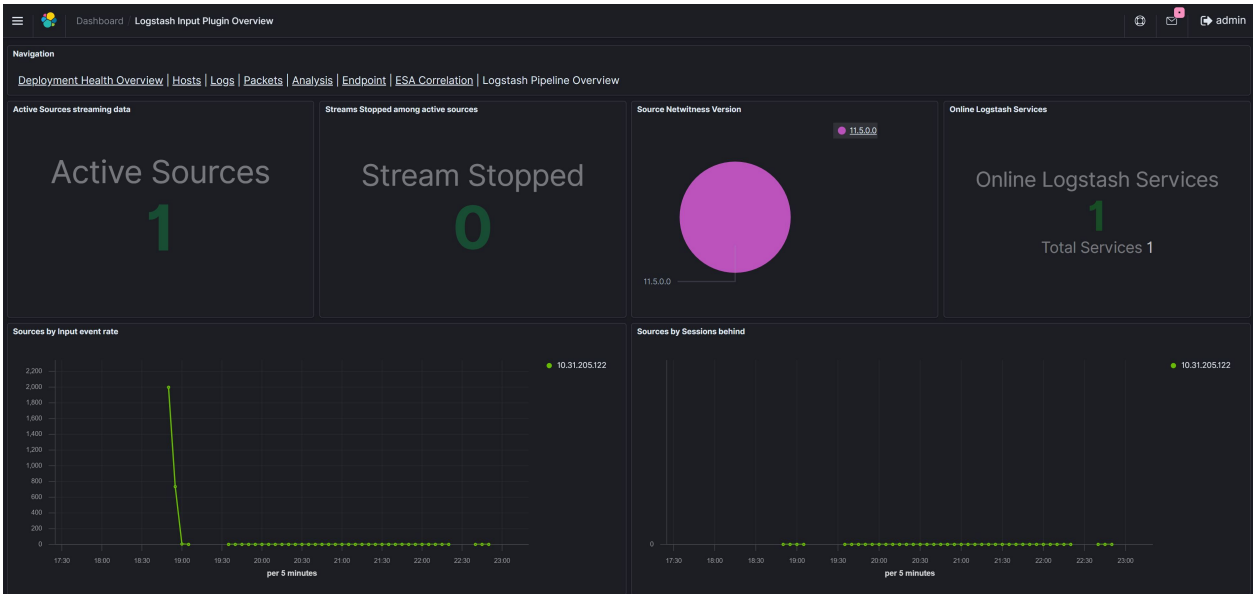
**Note:** This feature is available only from NetWitness version 11.5 and later. For more information, see "Monitor New Health and Wellness" in *System Maintenance Guide for RSA NetWitness*.

You can monitor the operational state of the Logstash service and details of sources configured in the New Health & Wellness tab available in NetWitness user interface. The New Health & Wellness is built on Elastic and Kibana. The metrics from the Logstash instance is sent to the Elastic service and visualized using Kibana. The tab displays the Logstash service status and the NetWitness Export Connector sources in a dashboard view. The New Health & Wellness service is referred as Elastic host in this document.

The Logstash service sends metrics for every 30 seconds. These metrics are used to view the operational status of the Logstash service and the NetWitness Export Connector. There are two types of metrics that are available for monitoring the Logstash status.

- Plugin metrics - For monitoring the NetWitness Export Connector status.
- Host metrics - For monitoring Logstash service instance (appliance or Virtual Machine) health status such as IO stats, CPU, memory usage and others.

By default, the plugin metrics are available in Logstash. To collect the host metrics, you must install Metricbeat, a third party software used for collecting the host metrics. It is recommended to enable both the plugin metrics and host metrics to view all the dashlets in the dashboard.




## Download and Install Dashboard.

You need to download and deploy the dashboards from live to view the Logstash service and the NetWitness Export Connector stats in Kibana. Do the following steps.

1. Log in to NetWitness Platform UI.
2. Click **(Configure) > LIVE CONTENT**.
3. In the **Search Criteria** panel, select the **Resource Types** as:
   - Health and Wellness Dashboards
   - Health and Wellness Monitors
4. Click **Search**.
5. In the **Matching Resources** view, select the **Logstash Input Plugin Overview** dashboard to deploy.
6. In the **Matching Resources** toolbar, click Deploy.
7. In the **Deployment Wizard > Resources** tab, click **Next**.
8. In the **Services** tab, select the Metrics Server service.
9. Click **Next**.
10. Click **Deploy**.
    The **Deploy** page is displayed. The Progress bar turns green when you have successfully deployed the resources to the selected services.
11. Click **Close**.

For more information, see Advanced Configuration section in "Monitor New Heath and Wellness" topic in the *System Maintenance Guide for RSA NetWitness*.

## Create a User in New Health & Wellness (Kibana)

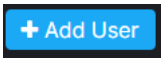
You must create user account in New Health & Wellness in NetWitness (Kibana), to send Logstash metrics to New Health & Wellness in NetWitness (Kibana). This user account is used in configuring the Logtstash plugin metrics and host metrics.

To create a user account in Kibana, do the following steps.

**Note:** This is applicable only for hosts or services that are not managed by NetWitness.

1. Go to  **(Admin) > HEALTH & WELLNESS > New Health & Wellness**.
   New Heath & Wellness panel view is displayed.
2. Click **Pivot to Dashboard**.
   Kibana Dashboard view is displayed in a new tab.
3. Click the Security icon  on the left.
   Kibana Security view is displayed.
4. In the Authentication Backends section, click **Internal User Database** and click the  icon to create new user.
5. Enter the username and the password in the respective columns.
6. In the Backend Roles section, click 

and add `nwservice-role`, and then click **Submit**.
   Username is displayed in the Internal Database section.
7. Click the 

icon to go back to Kibana Security view.
8. In the Permissions and Roles section, click **Role Mappings** and click **nwservice-role** to add the user name.
9. Click 

and add newly created username, and then click **Submit**.

## Enable the Logstash Plugin Metrics

To enable the Logstash plugin metrics, add the following parameters in the Logstash Configuration file (`netwitness-<decoder-ip>-input.conf`).

| Parameter | Setting | Parameter Type | Default Value |
|---|---|---|---|
| plugin_metrics_ enable | Set the value to 'true' to enable the plugin metrics | Boolean | false |

| Parameter | Setting | Parameter Type | Default Value |
|---|---|---|---|
| elastic_host | Enter the IP address or hostname of the Elastic host or IP address of the New Health & Wellness service | String | N/A |
| elastic_port | Enter the port number of the Elastic host | String | 9200 |
| elastic_username | Enter the username that is used to access the Elastic host (user account created in Kibana, see Create a User in New Health & Wellness (Kibana)) | String | N/A |
| elastic_password | Enter the password that is used to access the Elastic host | String | N/A |

## Enable the Logstash Host Metrics

**Note:** This is applicable only for hosts or services that are not orchestrated by NetWitness.

To enable the Logstash host metrics, follow the below steps.

1. Download Metricbeat software. For more information refer to Metricbeat.
   ```
   curl -L -O https://artifacts.elastic.co/downloads/beats/metricbeat/
   metricbeat-oss-7.8.0-x86_64.rpm
   ```
2. Install Metricbeat on the Logstash service by running the following command.
   ```
   sudo rpm -ivh metricbeat-oss-7.8.0-x86_64.rpm
   ```
3. Disable the default collection of Logstash monitoring metrics in the configuration file `logstash.yml` by changing `xpack.monitoring.enabled` parameter to `false`.
   ```
   xpack.monitoring.enabled: false
   ```

**Note:** Generally, the `x.pack.monitoring.enabled` parameter is commented. Remove the (#) character to activate it.

4. Enable the `logstash-xpack` module in Metricbeat to collect host status by running the following command.
   ```
   metricbeat modules enable logstash-xpack
   ```
5. In `/etc/metricbeat/modules.d/system.yml` file, add the following lines at the last and save it.

```
# Processors to parse process names of Netwitness Logstash Service.
- module: system
  period: 1m
  metricsets:
    - process
  processors:
    - add_fields:
        fields: {process.given.name: "logstash-service"} # Adding process
name to a new field
        when.contains:
          system.process.cmdline: "logstash-core.jar" # Condition on which
process name will be updated
```

**Note:** Use proper indentation while adding the lines. Refer an existing module in the `system.yml` file. If indentation is incorrect, the service may fail to start.

6. Configure the monitoring data to send metrics to Metricbeat. Follow the below steps.
   a. Update the `/etc/metricbeat/metricbeat.yml`file.
      - Replace the content in 'Elasticsearch output' section in `metricbeat.yml` with the following lines.

```
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["<ELASTIC_HOST>:9200"]
  index: "nw-logstash-metricbeats-%{+yyyy.MM.dd}"

  # Protocol - either `http` (default) or `https`.
  protocol: "https"
  ilm:
    enabled: false
    check_exists: false
  # Authentication credentials - either API key or username/password.
  username: "<ElASTIC_USERNAME>"
  password: "<ElASTIC_PASSWORD>"
  ssl.verification_mode: none

setup.ilm.enabled: false
setup.template:
   enabled: false # Disabled so that metricbeat auto-creates the
template everyday based on incoming metrics.
   name: "nw-logstash-metricbeats-%{+yyyy.MM.dd}"
   pattern: "nw-logstash-metricbeats-%{+yyyy.MM.dd}"
```

   b. Update the `ELASTIC_HOST`, `ELASTIC_USERNAME` and `ELASTIC_PASSWORD` details in `metricbeat.yml` file.

7. Start the Metricbeat service by running the following command.

```
sudo service metricbeat start
```

For more information visit the official documentation of Metricbeat.

# Configuring Custom Multi-valued Meta

Default multi-valued meta keys are `action, alias.host, alias.ip, alias.ipv6, email` and `username`, if the custom multi-valued path parameter is not set in Logstash configuration file (`netwitness-<decoder-ip>-input.conf`), then only default values are considered for multi-valued meta. Custom valued meta can be setup in generic (applicable for all the Netwitness Platform host where the json file is referenced) or specific to the Netwitness Platform host and can have custom multi-valued meta for multiple host in single json file. Both the `custom_multi_valued_meta_generic field` and `custom_multi_valued_meta_specific field` can be used in single file or at-least one 1 field must be set.

**Note:** Provide absolute path to the json file in the configuration (`netwitness-<decoder-ip>-input.conf`) and make sure Logstash user has access to this file. Meta name should be the same as the Netwitness Platform default name format. Do not use underscore format of the meta key.

For example, below shown is the json file which has both generic and specific field set in `custom_meta.json`

```
{
  "custom_multi_valued_meta_generic": [
    "meta.1,meta.2,meta.3"
  ],
  "custom_multi_valued_meta_specific": {
    "0.0.0.0": [
      "meta.4,meta.5,meta.6"
  ],
  "1.1.1.1": [
    "meta.4,meta.5,meta.7"
  ]
 }
}
```

In above example, the meta keys `meta.1, meta.2 and meta.3` are applied to all the NetWitness Platform host in the configuration file that has custom_meta_config path is set. The meta keys `meta.1, meta.2, meta.3, meta.4, meta.5, meta.6` are set as custom multi-valued meta for hosts `0.0.0.0` and the meta keys `meta.1, meta.2, meta.3, meta.4, meta.5, meta.7` are set as custom multi-valued meta keys for the host `1.1.1.1` along with default mutli-valued meta.

# Configure Logstash Filter Plugin (optional)

You can configure the Logstash Filter plugin to add, remove, or modify the specific input events from the Log Decoder or Decoder. To configure the Filter plugin, add the Filter plugin parameter settings in the second section of the Logstash configuration file (`netwitness-<decoder-ip>-input.conf`). This plugin modifies the events based on the parameter settings. You can use the existing standard Logstash filter plugins for adding the parameter settings to the configuration file. For more information on existing Logstash standard filter plugins, see Filter Plugin Documentation.

The configuration of the plugin must consist of the plugin name followed by a block of parameter settings for that plugin. The following is an example of Logstash mutate filter plugin configuration parameters that remove specific meta keys.

```
filter
 {
 mutate {
 remove_field => ["ip_dst", "ip_addr"]
 }
}
```

# Configure Logstash Output Plugin

You must configure the Logstash Output plugin to send the input events to a data warehouse destination. To configure the Output plugin, in the third section of the Logstash configuration file (`netwitness-<decoder-ip>-input.conf`), add the Output plugin parameter settings. You can use the existing Logstash standard output plugins for adding the parameter settings to the configuration file. For more information on existing Logstash standard output plugins, see Output Plugin Documentation.

The configuration of the plugin must consist of the plugin name followed by a block of parameter settings for that plugin. The following is an example of Logstash Output plugin configuration parameters.

```
output
{
 kafka {
 codec => json
 topic_id => "logstash"
 }
}
```

## Performance tuning for Kafka and Kafka Output Plugin

Following are the output plugin parameters for Kafka and Kafka Broker.

**Kafka Output Plugin:**
Following parameters require change in value for each parameters to add additional throughput in the deployment.
`batch_size, buffer_memory, compression_type, send_buffer_bytes`
For more information, see Kafka output plugin documentation.

**Kafka Broker:**
Following parameters require change in value for each parameters to add additional throughput in the deployment.
`num.network.threads, num.io.threads, socket.receive.buffer.bytes, socket.request.max.bytes, message.max.bytes, replica.fetch.max.bytes`
For more information, see Kafka Broker Performance documentation.

# Known Issues

| Com-ponents | Title, Problem & Workaround | Found In / Exists In | Fixed Version | Track-ing Number |
|---|---|---|---|---|
| NetWitness Export Connector | **Title:** Position tracking is not updated properly when query is used to filter the sessions.<br>**Problem:** Last `session.id` aggregated is not updated to the latest aggregated ID but is updated only to the last filtered session retrieved. For Example, If a decoder has 1000 sessions, and only first 800 sessions match the filter, the `last.session.id` in the position tracking shows 800 even though all 1000 sessions are processed. This does not cause any data loss but if the Logstash service is restarted, aggregation will begin from 801 `session.id`, processing the 200 sessions again.<br>**Workaround:** None | netwitness-export-connector-1.0.0 | | ASOC-101795 |
| NetWitness Export Connector | **Title:** Consumer does not consume all the events after the Logstash service is restarted.<br>**Problem:** While running at a high EPS or if there is a large session behind, the NetWitness Export Connector aggregates the sessions of the decoders. The Logstash pipeline sends to the output plugin which in turn sends it out to the consumer. At this point, if a Logstash service is stopped or restarted, the Logstash pipeline ensures the data is sent out of the output plugin to the consumer, however all the sessions are not consumed by the consumer. This leads to data loss on the consumer end even though Logstash has sent out all the data.<br>**Workaround:** Update the `start_session` parameter in the Logstash configuration file (`netwitness-<decoder-ip>-input.conf`) by referring to the latest received `session.id` on Kafka or any other consumer used. | netwitness-export-connector-1.0.0 | netwitness-export-connector-1.1.0 | ASOC-102440 |

| Com-ponents | Title, Problem & Workaround | Found In / Exists In | Fixed Version | Track-ing Number |
|---|---|---|---|---|
| NetWitness Export Connector | **Title:** Position tracking not saved for some Decoder sources after sometime in a multiple sources configuration.<br>**Problem:** When aggregating data from multiple sources in a single Logstash instance, after a period of time, the position tracking thread fails for some or all of the sources gradually and stops tracking the position (bookmarking) for those sources.<br>**Workaround:** None | netwitness-export-connector-1.0.0 | netwitness-export-connector-1.1.0 | ASOC-102411 |
| NetWitness Export Connector | **Title:** Unable to start session aggregation from `first.session.id` of Decoder when it is not 1.<br>**Problem:** If the Decoder has the `first.session.id` other than 1, which means the Decoder has data that has rolled over, the NetWitness Export Connector's `start_session` parameter will not start aggregation from that value, instead it would begin from the `last.session.id`.<br>**Workaround:** Add 1 to the value to the `start_session` parameter and the aggregation would start as expected dropping the first session only. For example, if `first.session.id` on a Decoder is '1105000' add 1 to it and set the value of the `start_session` parameter as '1105001'. | netwitness-export-connector-1.0.0 | netwitness-export-connector-1.1.0 | ASOC-101951 |