RSA NETWITNESS® PLATFORM

Decoder Configuration Guide

for RSA NetWitness® Platform 11.4

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

May 2020

Contents

Decoder and Log Decoder Quick Setup

A basic RSA NetWitness® Platform network includes at minimum Brokers, Concentrators, and Decoders. Brokers aggregate data from Concentrators, and Concentrators consume data from at least one Network Decoder or Log Decoder. The basic network may include both types of Decoders. Network Decoders are usually referred to as Decoders, and they capture network data in packet form. Log Decoders capture log data as events.

Adding a Decoder makes it visible and available for use with NetWitness Platform Administration, Live Services, and Investigate. To add a service in NetWitness Platform, you select the service type, provide service connection information, and validate that the service can be reached. The *Hosts and Services Getting Started Guide* provides the information you need to understand and install all NetWitness Platform services.

After the services are added, you need to configure each service. This is the preferred order for configuring your system:

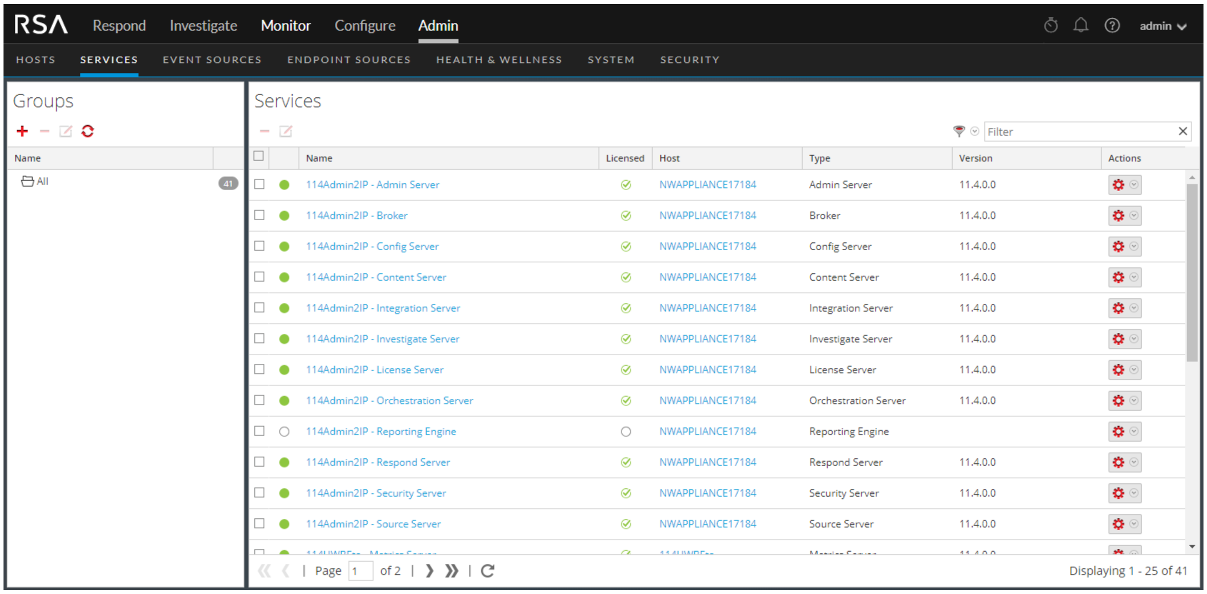
1. Decoders
2. Log Decoders
3. Concentrators (refer to the *Broker and Concentrator Configuration Guide*)
4. Brokers (refer to the *Broker and Concentrator Configuration Guide*)

Note: A Log Decoder is a special type of Decoder, which is configured and managed in a similar way to a Decoder. Most of the information in this guide refers to both types of Decoders. "Decoder" refers to both types of Decoders. Information that applies exclusively to Network Decoders or Log Decoders is clearly identified.

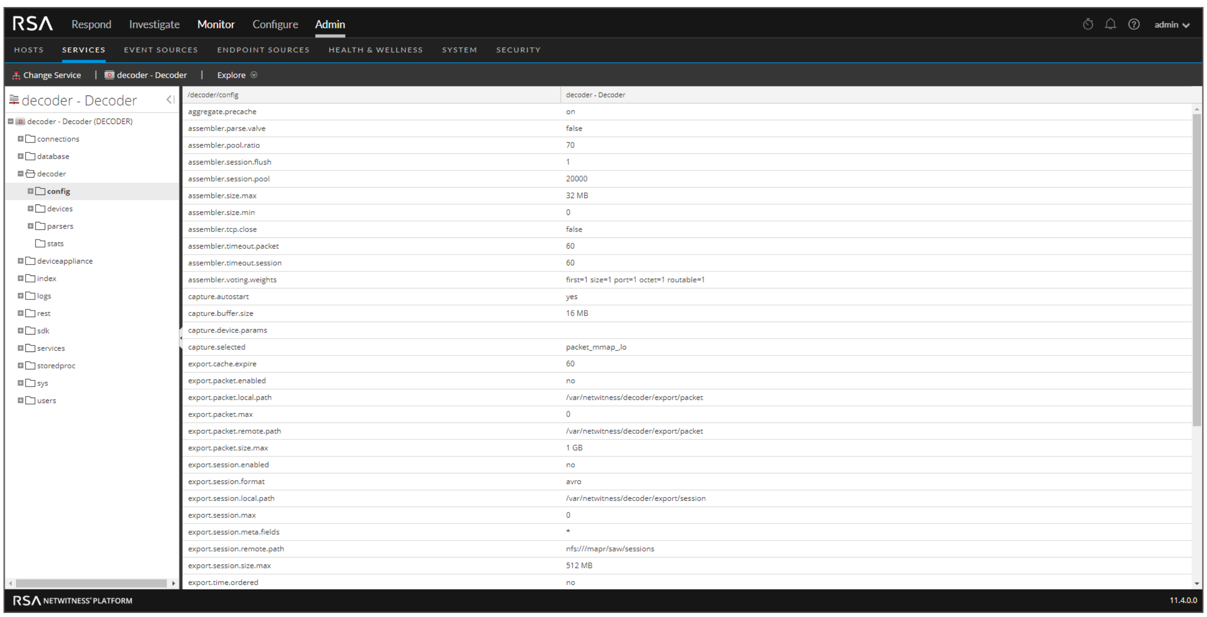
Basic configuration of the Decoder involves selecting a network adapter interface and starting data capture.

In addition, you can configure each Decoder to control the type of traffic captured using rules, feeds, and parsers. Advanced configuration tasks enable additional features that are relevant to specific applications. For example, configure a 10G Decoder, create custom meta keys, or decrypt incoming packets.

The easiest way to configure all of the required Decoder and Log Decoder settings is to use the options in the NetWitness Platform user interface. For the most part, configuration is performed in the Administration Services view (Admin > Services).



Administrators who feel comfortable working outside of the user interface can configure the basic parameters as well as advanced settings by editing database nodes in the Decoder node tree using the Services Explore view.



Perform Initial Quick Setup

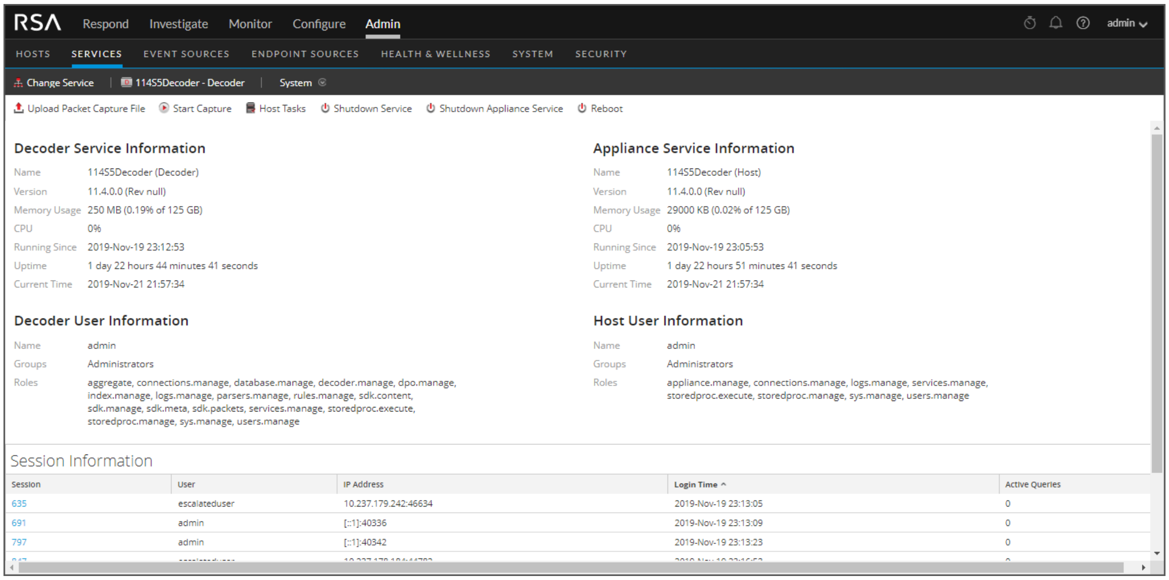
This procedure accomplishes the initial, basic configuration of a Decoder, and starts data capture. When the basic setup is complete, the Decoder begins capturing data for the Concentrator to consume.

To configure a Decoder and start capturing data:

1. Assign a network interface for capturing data. For details, see "Select a Network Adapter" in Configure Capture Settings.

2. Do one of the following:

   a. To start capture, select the Decoder and  > **View** > **System**. In the toolbar click Start Capture .



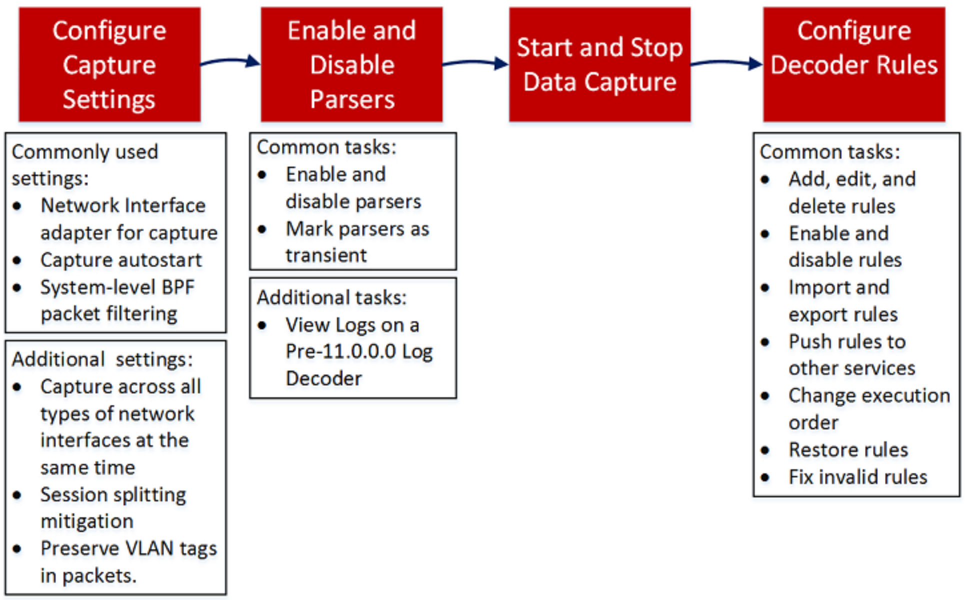
   b. To enable Capture Autostart, see "Configure a Decoder to Begin Capturing Data Automatically" in Configure Capture Settings.
   The Decoder begins capturing data for consumption by a Concentrator. For additional configuration options, refer to Configure Common Settings on a Decoder Configure Common Settings on a Decoder and Decoder and Log Decoder Additional Procedures

Configure Common Settings on a Decoder

This section introduces commonly used configuration settings on a Decoder with procedures and background information. After you have completed Decoder and Log Decoder Quick Setup, you can refine your configuration by using parsers, feeds, and rules to limit the captured data.

Note: A Log Decoder is a special type of Decoder, which is configured and managed in a similar way to a Decoder. Most of the information in this guide refers to both types of Decoders. "Decoder" refers to both types of Decoders. Information that applies exclusively to Network Decoders or Log Decoders is clearly identified.

The following workflow illustrates commonly used settings and breaks the configuration process into four steps.

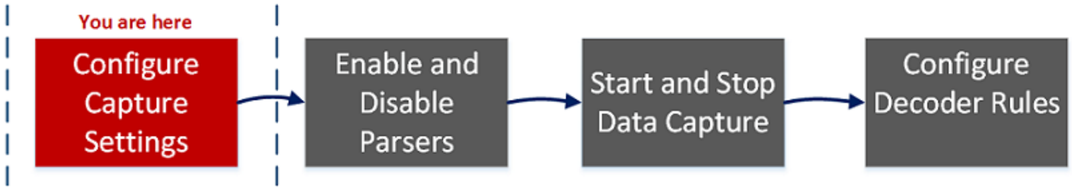


Configuration Step	Description
Configure Capture Settings	When initially setting up the Decoder, configuring the network adapter interface is required. Additional optional capture settings are available; one that is frequently used is Capture Autostart.
Enable and Disable Parsers and Log Parsers	View the parsers that have been downloaded and deployed from Live, and manage which ones are enabled or disabled.
Start and Stop Data Capture	When a Decoder starts up, it automatically begins aggregating data if Capture Autostart is enabled. When autostart is not enabled, you can start and stop data capture manually.

Configuration Step	Description
Configure Decoder Rules	Capture rules can add alerts or contextual information to sessions or logs. They can also define which data a Decoder or Log Decoder filters out. By default, no capture rules are defined when you first configure NetWitness Platform. Unless rules are specified and the rules are valid, the packets are not filtered. You can deploy the latest rules from Live as described in the *Live Services Management Guide*. You can define capture rules at any time, and you can fix rules that use invalid syntax (Fix Rules with Invalid Syntax).

Configure Capture Settings

When initially setting up the Decoder, configuring the network adapter interface is required. Additional optional capture settings are available; two that are frequently used are the Berkeley Packet Filter, and Capture Autostart.



Besides the basic network adapter interface setup, you may decide to use one of the special-purpose configurations described in Preserve VLAN Tags When Using the Packet MMAP Capture Interface or Configure a Decoder to Capture Data Across All Types of Network Interfaces.

The rest of the capture settings have default values chosen to be effective in most cases (see a detailed list in Services Config View - General Tab). You can adjust these in some circumstances, for example, if Customer Support advises a change. You can edit the capture settings at any time.

Select a Network Adapter

The table below describes the Network Adapter settings for a Decoder. The system administrator sets the default network adapters when the Decoder is installed. Consult your System Administrator for more information.

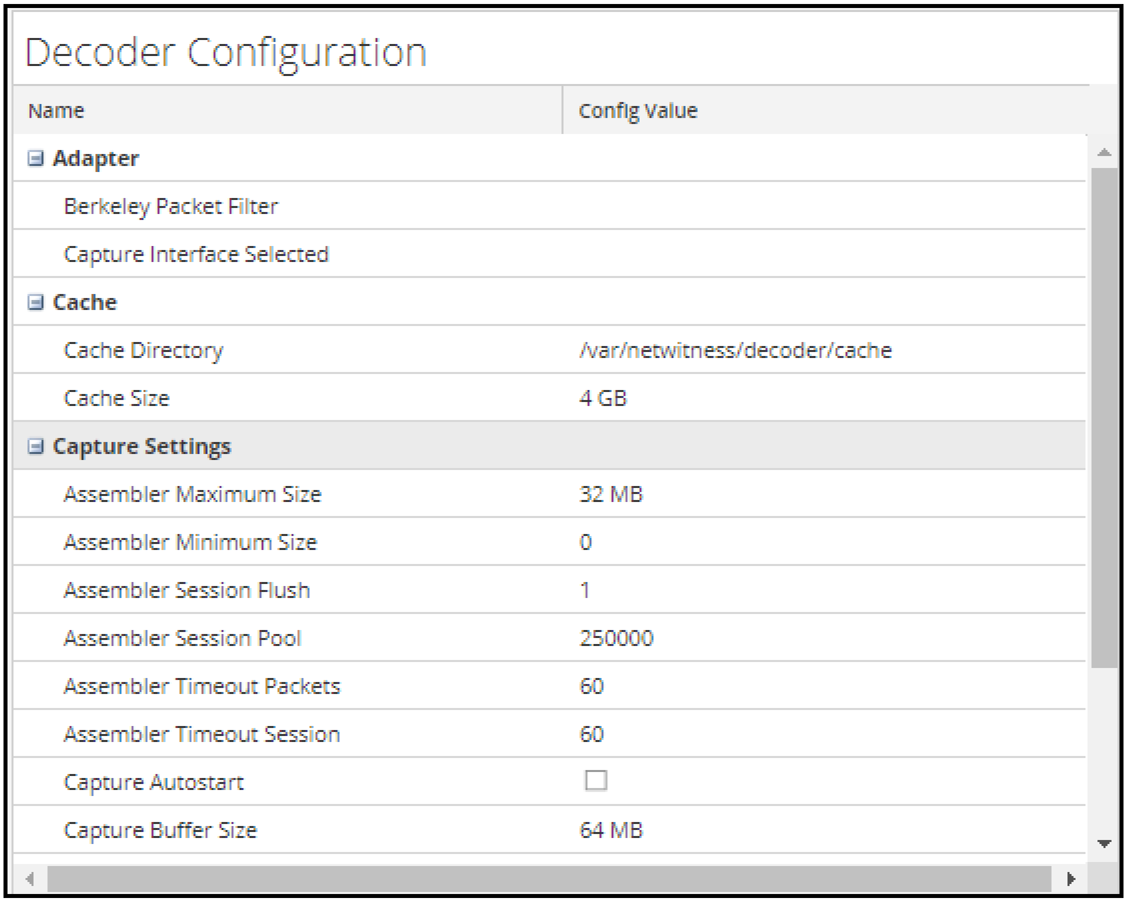
Adapter Parameter	Description
Berkeley Packet Filter	Berkeley Packet Filters (BPF) are applied to the packet stream before the packets are copied to the Decoder adapter for analysis. This allows unwanted traffic to be efficiently discarded. However, any packets discarded are not accounted for in any Decoder statistics (capture rate, packets dropped, and packets filtered and total packets).

Adapter Parameter	Description
Capture Interface Selected	Select an adapter through which the Decoder captures packets. For the lower speed internal capture interface, use the `packet_mmap_,7,eth1` adapter, which corresponds to the monitor port located on the motherboard. There are six additional capture ports: • `packet_mmap_,1,lo (bpf)` • `packet_mmap_,2,eth2 (bpf)` • `packet_mmap_,3,eth3 (bpf)` • `packet_mmap_,4,eth4 (bpf)` • `packet_mmap_,5,eth5 (bpf)` • `packet_mmap_,8,ALL (bpf)` There are three wireless capture services available: • `packet_netmon_ (Microsoft Netmon)` • `packet_mac80211_ (Linux mac80211)` • `packet_airport_ (Mac OS X AirPort)`
Capture Interface Selected for Log Decoder	The following capture service is available: • `log_events,Log Events`

To configure the network adapter on a Decoder:

1. Go to **Admin > Services**.
2. In the **Administration Services view**, select the Decoder and  **> View > Config.**

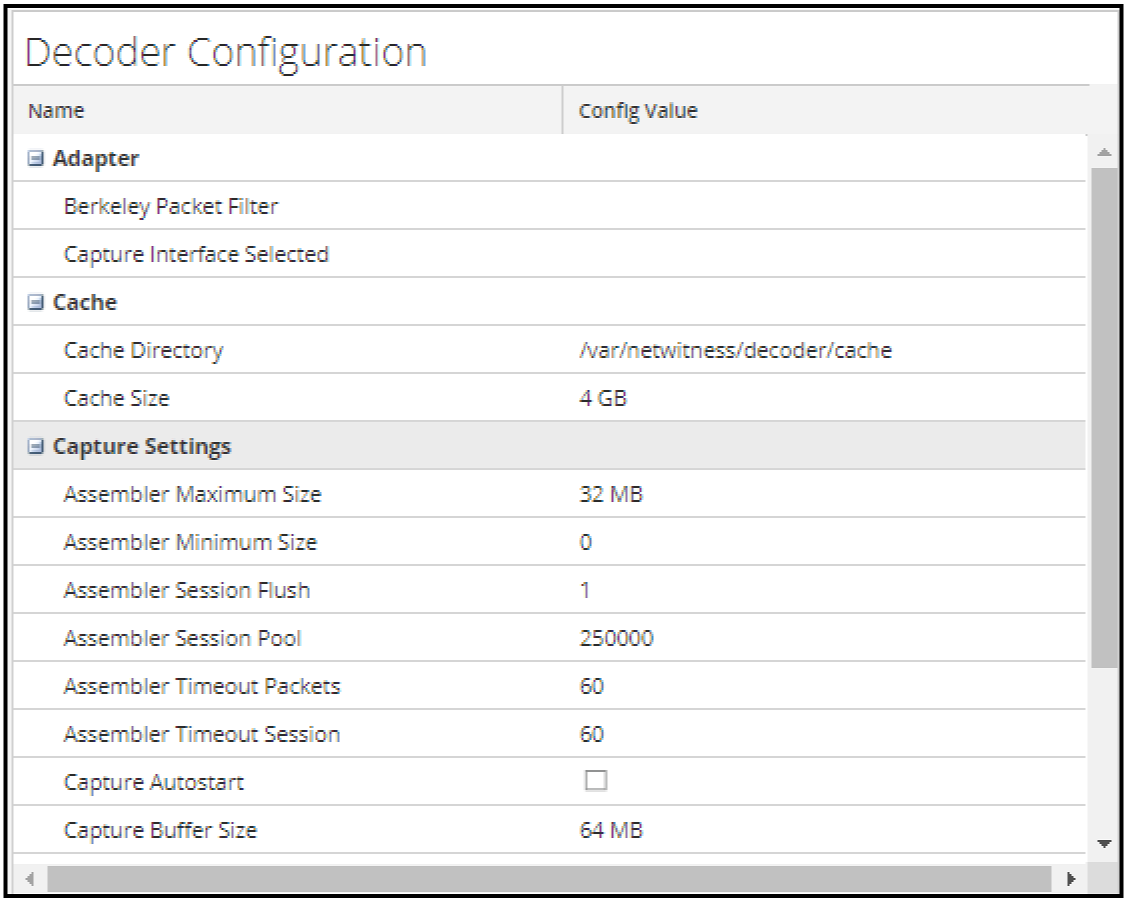
   The Services Config view is displayed with the General tab open.



3. In the **Capture Interface Selected** field, select the network adapter that best suits the Decoder.
4. To save the changes, click **Apply.**
5. If necessary to put the changes into effect, navigate back up to the **Administration Services view**, select the Decoder, and select > **Restart**.

Configure a Decoder to Begin Capturing Data Automatically

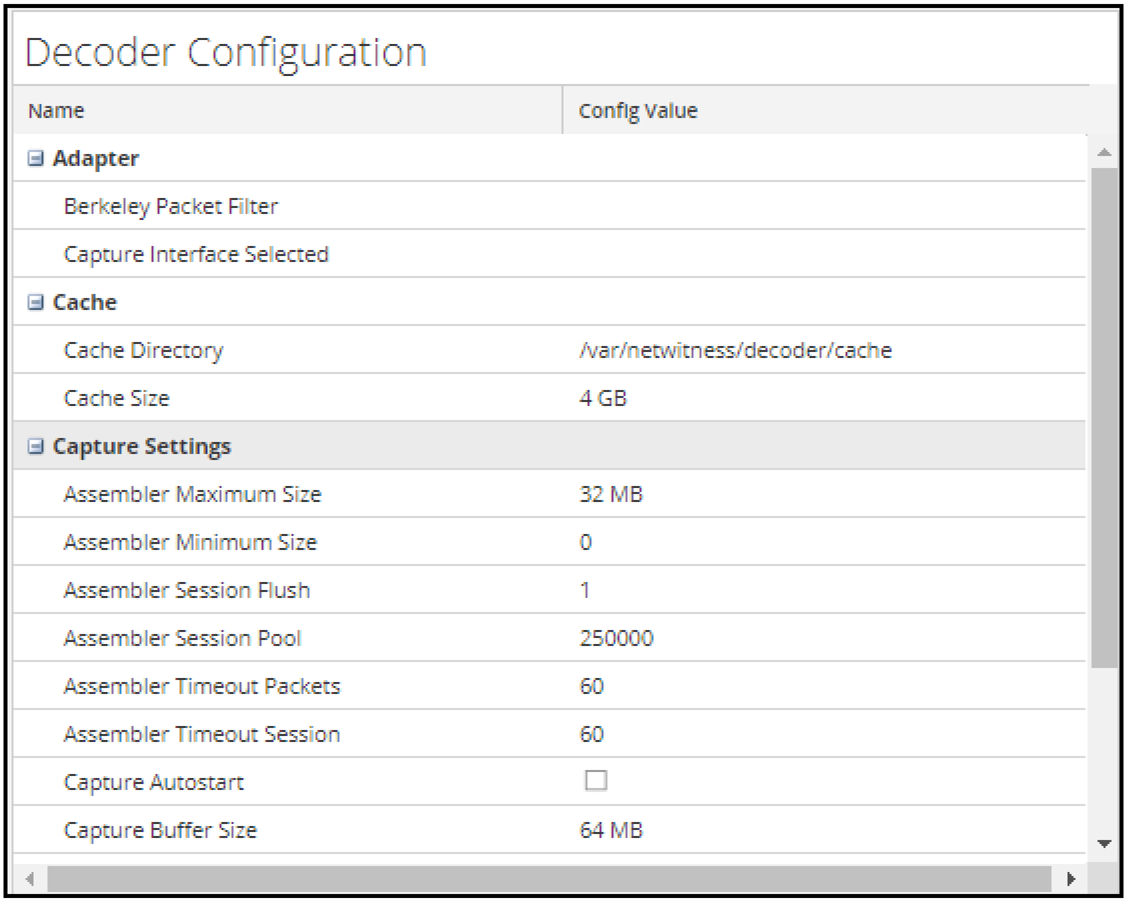
1. Go to **Admin** > **Services**.
2. In the **Administration Services view**, select the Decoder and > **View** > **Config.**
   The Services Config view is displayed with the General tab open.

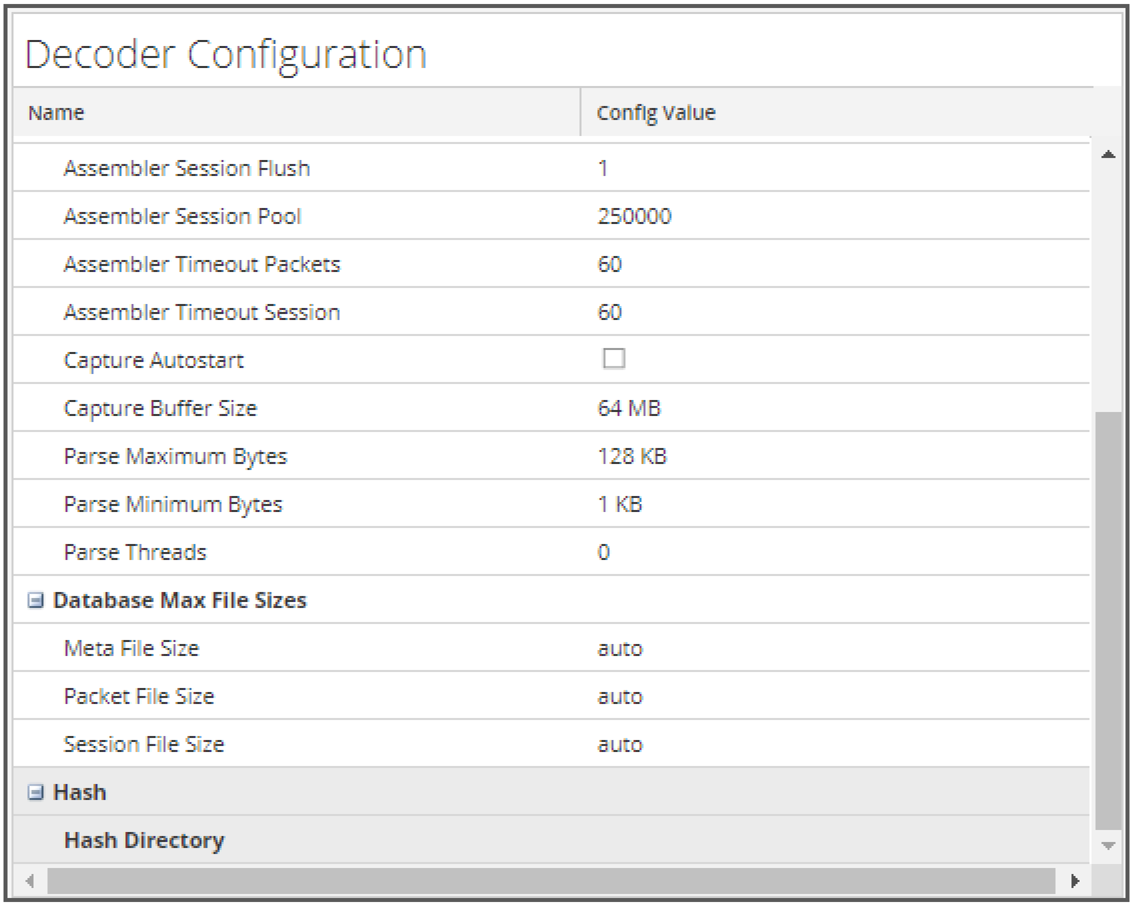


3. Under **Capture Settings**, select the **Capture Autostart** checkbox.
4. To save the changes, click **Apply.**
5. If necessary to put the changes into effect, navigate back up to the **Administration Services view**, select the Decoder, and select > **Restart**.

Configure Optional Capture Settings

1. Go to **Admin** > **Services**.
2. In the **Administration Services view**, select the Decoder and **> View > Config**.
   The Services Config view is displayed with the General tab open.


Decoder Configuration
Name
Config Value
Adapter
Berkeley Packet Filter
Capture Interface Selected
Cache
Cache Directory
/var/netwitness/decoder/cache
Cache Size
4 GB
Capture Settings
Assembler Maximum Size
32 MB
Assembler Minimum Size
0
Assembler Session Flush
1
Assembler Session Pool
250000
Assembler Timeout Packets
60
Assembler Timeout Session
60
Capture Autostart
Capture Buffer Size
64 MB



3. If you want to apply a system-level filter to the packet stream before the packets are copied to the Decoder adapter for analysis, configure the Berkeley Packet Filter as described in (Optional) Configure System-Level (BPF) Packet Filtering.

4. In the **Capture Settings** sections, review the default values. When a service is first added, default values are in effect and should be changed only in special circumstances, for example, if Customer Support advises a change. See Services Config View - General Tab for an explanation of these settings.

5. In the **Database Max File Sizes** section, review the default values. When a service is first added, default values are in effect and should be changed only in special circumstances, for example, if Customer Support advises a change. See Services Config View - General Tab for an explanation of these settings.

6. In the **Hash** section, define a directory for hash files if you are using this feature. See Services Config View - General Tab for an explanation of these settings.

(Optional) Configure System-Level (BPF) Packet Filtering

You can use Berkeley Packet Filters to control which packets and logs are processed by a Decoder.

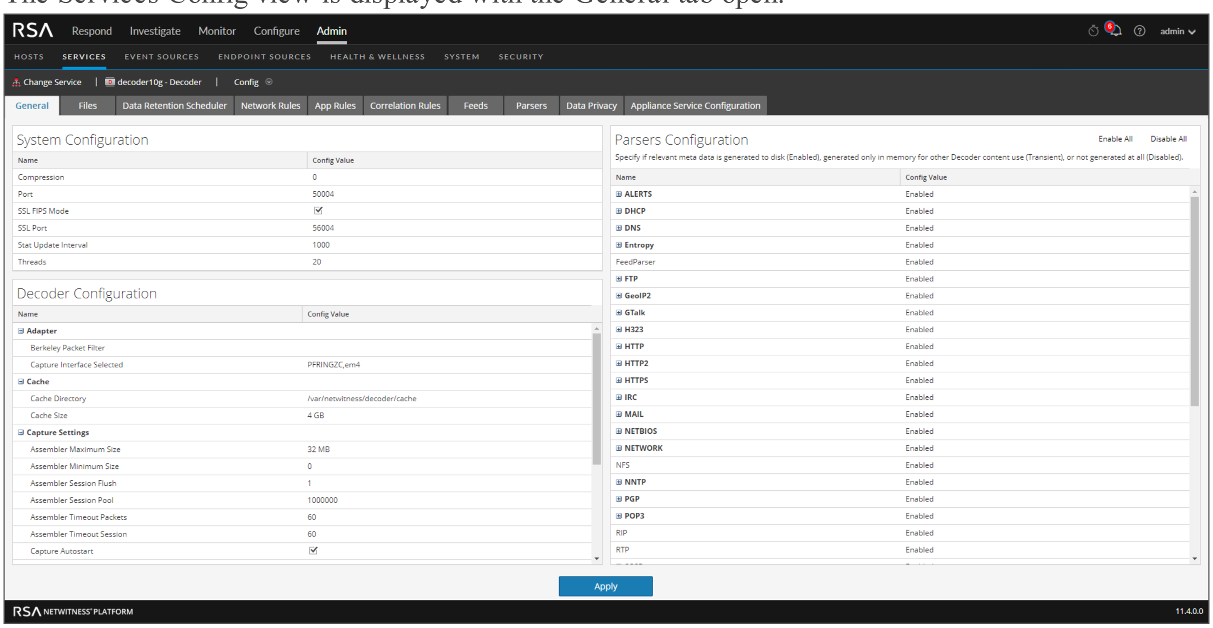
Berkeley Packet Filters (BPF) are applied to the packet stream before the packets are copied to the Decoder adapter for analysis. This allows unwanted traffic to be efficiently discarded. These discarded packets are not accounted for in any Decoder statistics (capture rate, packets dropped, and packets filtered and total packets).

The Decoder also supports system-level packet filtering defined using `tcpdump/libpcap` syntax. Specifying a `Libpcap` filter can efficiently reduce packet volume based on Layer 2 - Layer 4 attributes. A `Libpcap` filter is appropriate for use when a Decoder is receiving a traffic volume that is placing a load against the physical resources of the platform. In this scenario, the Decoder may consistently drop packets and have a large number of capture pages available(`/decoder/stats/capture.pagefree` is high).
The following is an example of a `libpcap` filter to keep only packets that do not have both source and destination addresses in the 10.21.0.0/16 subnet.
```
not (src net 10.21.0.0/16 and dst net 10.21.0.0/16)
```
For a full reference of the `Libpcap` filter syntax, see the main pages for:

- tcpdump (http://www.tcpdump.org/tcpdump_man.html).
- pcap-filter (http://www.unix.com/man-page/FreeBSD/7/pcap-filter/).

To add a system-level Berkeley Packet Filter:

1. Go to **Admin** > **Services.**
2. In the Administration Services view, select a Decoder service and  > **View** > **Config.**
   The Services Config view is displayed with the General tab open.



3. In the **Decoder Configuration Section**, under **Adapter**, click in the field next to **Berkeley Packet Filter**.

4. Type only one filter in the field. If you want to filter multiple items, join multiple expressions using `and`. Several examples are provided below.
   The user interface validates input at the time you enter your filter string.
5. To save the filter, click **Apply**.
   If the syntax is correct, a confirmation message is displayed.
   If the syntax is incorrect, a **Packet filter is not valid** message is displayed and a corresponding log message will follow in the log messages on the Decoder:
   `164474800 2015-May-01 19:03:08 warning Decoder Failed to parse filter 'example_badrule': syntax error`
6. To activate the filter, you must stop and start capture on the Decoder:
   a. Change the **Config** view to the **System** view.
   b. Click **Stop Capture**.
   c. Click **Start Capture**.
      The active filter will be displayed in the Decoder logs.

Examples

These are several filter examples:

- Drop packets to or from any address in the 10.21.0.0/16 subnet:
  `not (net 10.21.0.0/16)`
- Drop packets that have both source and destination addresses in the 10.21.0.0/16 subnet:
  `not (src net 10.21.0.0/16 and dst net 10.21.0.0/16)`
- Drop packets that are from 10.21.1.2 or are headed to 10.21.1.3.
  `not (src host 10.21.1.2 or dst host 10.21.1.3)`
- Combine both IP and HOST:
  `not (host 192.168.1.10) and not (host api.wxbug.net)`
- Drop all port 53 traffic, both TCP & UDP:
  `not (port 53)`
- Drop only UDP port 53 traffic:
  `not (udp port 53)`
- Drop all IP protocol 50 (IPSEC) traffic:
  `not (ip proto 50)`
- Drop all traffic on TCP ports 133 through 135.
  `not (tcp portrange 133-135`

The following filters combine some of the above to demonstrate how to put multiple directives into one filter:

- Drop any port 53(DNS) traffic sourced from 10.21.1.2 or destined to 10.21.1.3.
  `not (port 53) and not (src host 10.21.1.2 or dst host 10.21.1.3)`

- Drop any traffic using IP proto 50 or port 53 or any traffic from net 10.21.0.0/16 destined to net 10.21.0.0/16
  `not (ip proto 50 or port 53) or not (src net 10.21.0.0/16 and dst net 10.21.0.0/16)`

Caution: The use of parentheses can have a large and potentially disruptive effect on the use of Packet Filters. As a best practice, keep "not" operations outside of parentheses and always test your rules before deploying them. Failure to properly format your rules (despite input validation) can cause a packet filter to drop ALL traffic or behave in other unexpected ways. This is due to the way packet `Libpcap` filters work and is not the result of any logic within NetWitness Platform software.

Testing

BPF filters can and should be tested using either `tcpdump` or `windump` to ensure that they will provide the expected behavior before implementing them. This example shows a test of a filter using windump:

```
windump -nni 2 not (port 53 or port 443) or not (ip proto 50)
```

Conversions

If for the sake of performance, you have decided that an existing network rule filter would be better running as a System-Level Packet Filter, you can convert it. There are a few things to remember when doing conversions.

- `&&` becomes `and`
- `alias.ip` becomes `host` if a single host or **net** if a network.
- `ip.src` becomes `src host` if a single host or `src net` if a network.
- `ip.dst` becomes `dst host` if a single host or `dst net` if a network.
- Use CIDR notation when listing a network (that is, 10.10.10.0/24).
- `||` becomes `or`
- `!` becomes `not`
- Multiple rules must be joined with `and`.

The manual for TCPDump also gives examples of filters and strings that can be used: http://www.tcpdump.org/tcpdump_man.html

Additionally, the following site provides an excellent reference for BPF-style packet filters: http://biot.com/capstats/bpf.html

Caution: If you are capturing `vlan` tagged packets, above standard bpf filter may not work. For example, if you use `not (udp port 123)` to filter `vlan` tagged NTP traffic on udp port 123, it will not work. This is because the bpf filter machinery is simple and does not account for protocols not referenced in the rule. So the OS executing the bpf filter will look for the `udp port` values at the byte offset they would occur in a standard Ethernet/udp packet; but the optional `vlan` tag fields in the Ethernet header pushes those values by 4 bytes, thus the bpf filter rule will fail. To fix it, you need to change the bpf filter to: `not (vlan and udp port 123)`.

(Optional) Configure a Decoder to Capture Data Across All Types of Network Interfaces

The `packet_mmap_,ALL` adapter is capable of capturing across all types of network interfaces at the same time. For example, this can include things like physical network interfaces over different media types and tunnel interfaces.

The default behavior of the `ALL` adapter is to capture from all interfaces from the system, except for the hard-coded defaults of `lo`, `eth0`, and `em1`.

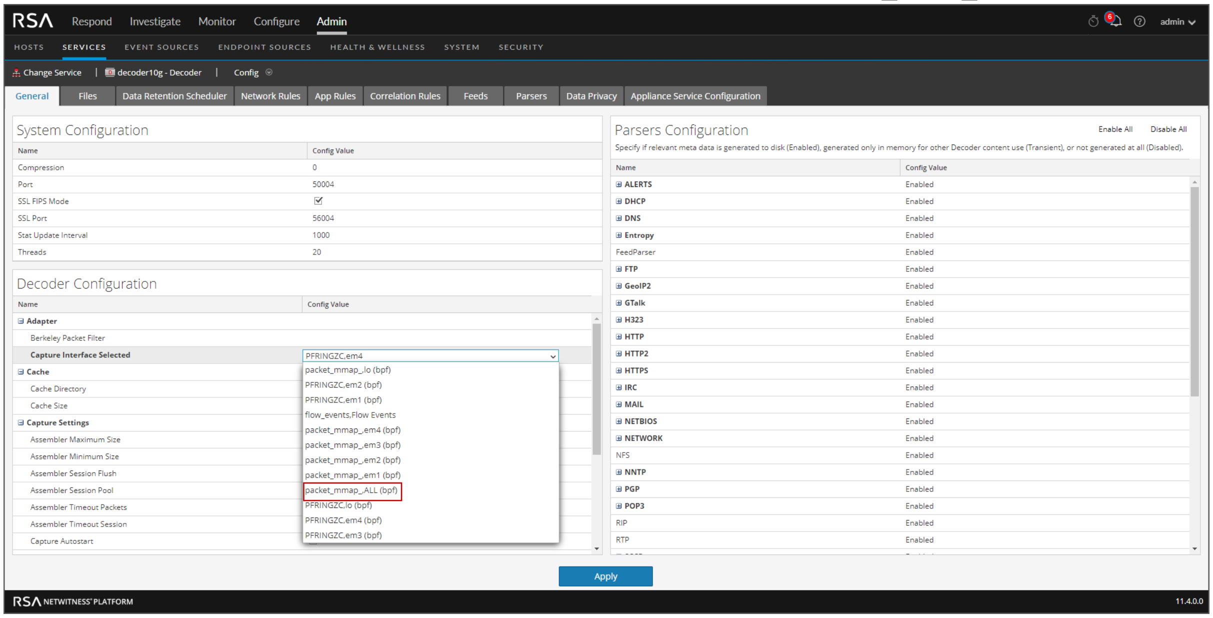
You can select any subset of the capture interfaces by editing the Decoder configuration node `/decoder/config/capture.device.params` to include an `interfaces=` parameter. The `interfaces` parameter contains a comma-separate list of interfaces that are used for capture. Instead of using all interfaces for capture, only the specified interfaces are used.

For example, if you want to force capture on interfaces `em1`, `em2`, and `em4`, and ignore `em3`, you can select the `packet_mmap_,ALL` adapter, and then add this line to `capture.device.params`:
`interfaces=em1,em2,em4`

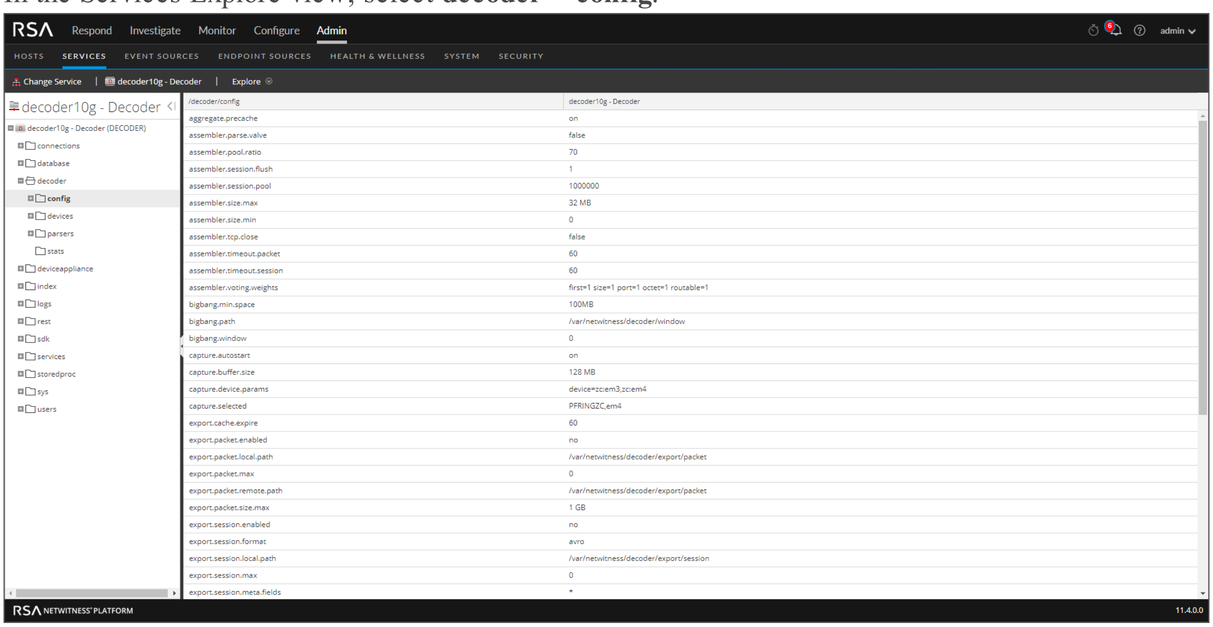
Note: Using the `interfaces` parameter to select `eth0`, `lo`, or `em1` overrides the default behavior, which is to drop traffic from those ports.

To configure the `packet_mmap_,ALL` adapter to capture from specific interfaces instead of all interfaces:

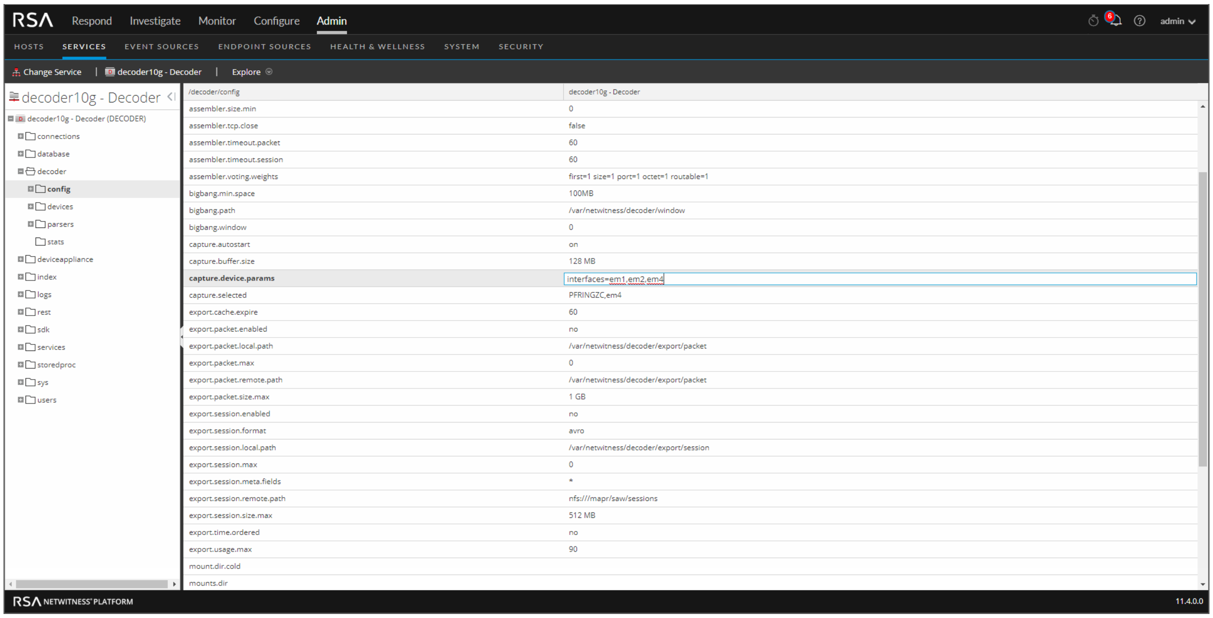
1. Go to **ADMIN** > **Services**, select the Decoder service and  > **View > Config.**
2. In the **Services Config view**, set **Capture Interface Selected** to `packet_mmap_,ALL` adapter.



3. To go to the Services Explore view, click **Config** in the toolbar and select **Explore** in the drop-down list.
4. In the Services Explore view, select **decoder** > **config**.



5. Click in the values column next to `capture.device.params`, type **`interfaces=em1,em2,em4`**, and press **Enter**.



The change goes into effect immediately; only traffic on `em1`, `em2`, and `em4` interfaces is captured.

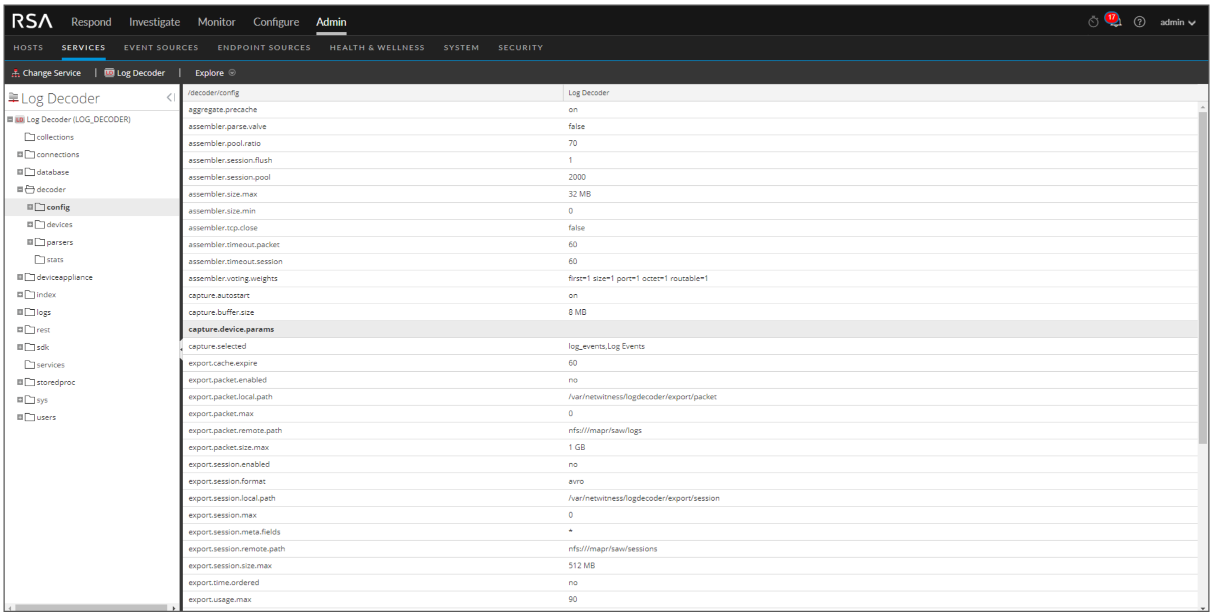
(Optional) Configure Custom Certificates on Log Decoders

You can configure custom certificates for the syslog listener on Log Decoders. This enables you to put your own trusted cert in place for the syslog listener, while all other functionality uses the pre-installed certs.

Note: This feature applies only to Log Decoders.

To configure custom certificates for the syslog listener:

1. Go to **ADMIN** > **Services**, select the Log Decoder service and  > **View > Explore.**
2. Select **Log Decoder** > **decoder** > **config**.



3. In the values column next to `capture.device.params`, use the following parameter to configure the custom cert:

   `cert-path=<folder>`

   The Log Decoder's SSL syslog connections will use the `logdecoder_cert.pem` and `logdecoder_key.pem` files in the `<folder>` specified.

4. Press **Enter**. The change goes into effect on capture restart.

(Optional) Configure a Decoder to Write Standard pcap-formatted Files

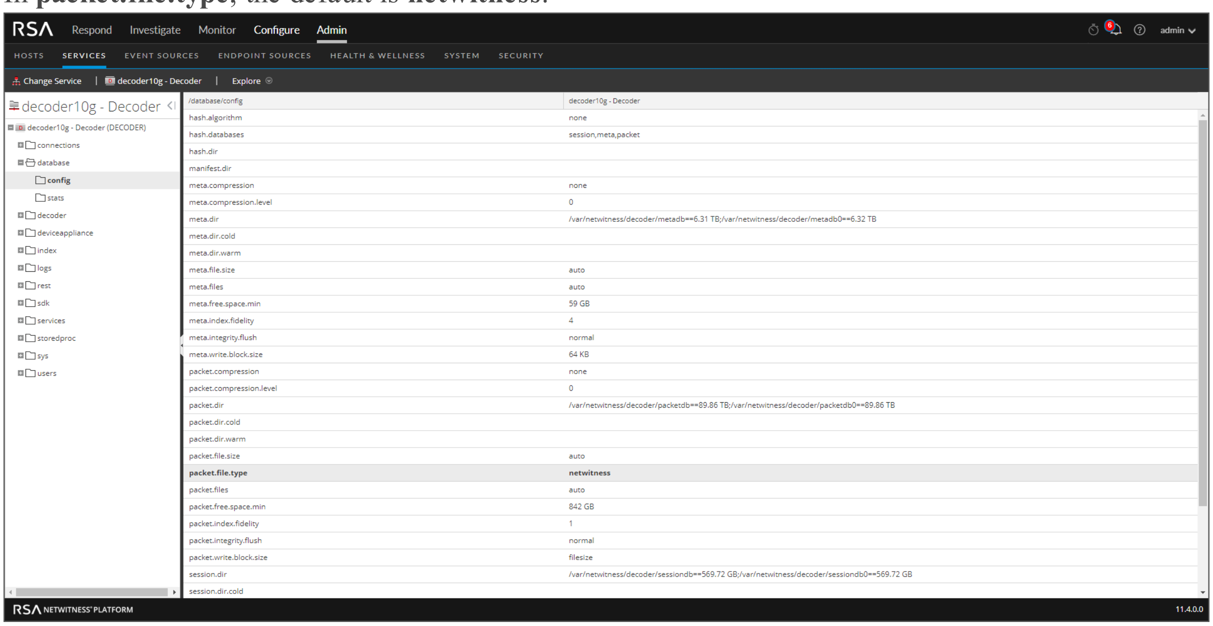
> **Note:** The information in this topic applies to RSA NetWitness® Platform Version 11.2 and later.

To provide a more open database format, the Network Decoder can now write standard pcap-formatted files. You can enable pcapng-formatted database files with the new configuration node: `/database/config/packet.file.type = 'netwitness' or 'pcapng'`

> **Note:** This capability is enabled by default if you install 11.2 directly. If you upgrade from a previous version to 11.2, you must enable pcapng-formatted database files manually, which can result in an approximate 4% decrease in disk space (as the pcapng files require more space than the nwdb files). You can also use the pcapng format with 10 Gbps capture, which does not decrease performance significantly (< 1%).

To enable writing standard pcap-formatted files:

1. Go to **ADMIN** > **Services**, select a Network Decoder service, and then select > **View > Explore**.
2. Go to **database** > **config**.
3. In **packet.file.type**, the default is **netwitness**.



4. To change the packet file type to standard pcap formatting, type **pcapng** and press **Enter**. This change will take effect immediately on the next packet file that is created.

> **Note:** In the pcapng database file format, the data is in clear text, and is not obfuscated by our proprietary format, which can improve security.

Caution: Please do not touch any files in the packet database directories! You must not read or edit any `pcapng` file in the packet database directories, as they are always in use while Decoder is running. Decoder always expects full and exclusive access to those files, and other processes reading those files prevent normal Decoder operation. The proper way to access the `pcapng` files is to set up a cold storage directory. This allows Decoder to copy `pcapng` files to the cold storage directory before deletion. At that point, you are responsible for managing the `pcapng` files, including making sure that the cold storage volume never fills up. Keep in mind that copying the `pcapng` files to cold storage requires a non-trivial amount of I/O and could interfere with packet capture. Cold storage for `pcapng` is not supported at 10G speeds.

(Optional) Preserve VLAN Tags When Using the Packet MMAP Capture Interface

When capturing traffic containing VLAN tags, you may need to configure the Packet MMAP capture interface to preserve the VLAN tags in the packets (VLAN fixup). By default, the network capture hardware removes the tags. Performing this procedure preserves the tags in the packets, and the tag values are parsed into VLAN meta data for further analysis.

There are two mechanisms for enabling the VLAN fixup.

- Option 1: Set `vlan-fix=true` within `capture.device.params`. This option performs the VLAN fixup on all traffic entering the Decoder. This option is appropriate in most cases, since it is assumed that all the traffic will be VLAN tagged.This mechanism works on either single-interface mode, or on all-interfaces mode. This option overrides the VLAN fixup settings on individual interfaces; even interfaces that are not configured to do VLAN fixup will have the feature enabled.
- Option 2: Use the `interfaces` parameter within `capture.device.params` on a per-device basis. The `interfaces` parameter accepts a comma-separated list of interface names on which to capture packets. By adding `:vlan` to an interface name, you can enable the VLAN fixup on individual interfaces. If the interface does not have the `:vlan` suffix added, then it will not perform the VLAN fixup.

After editing this parameter, you must restart capture on the Decoder in order for changes to `capture.device.params` to take effect.

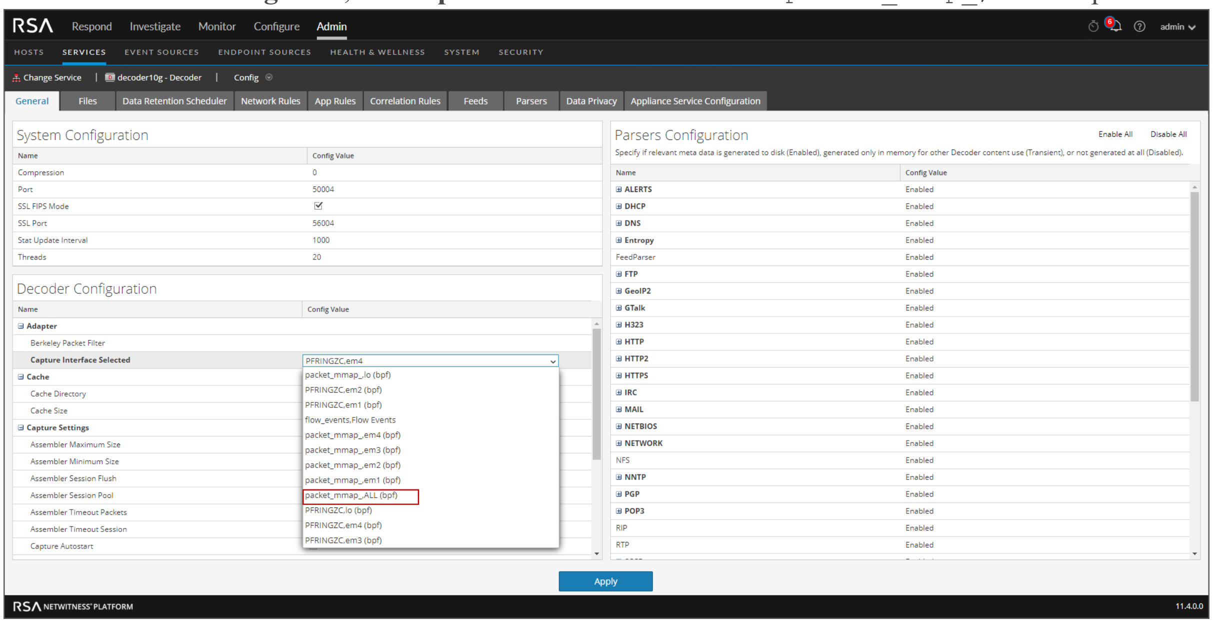
These are `vlan` examples of both options. If you need to pass multiple settings for `capture.device.params`, use the following synax. Notice that quotes are needed to delineate whitespace. For more information about how to use quotes for whitespace, see the "Connecting to a Service" topic in the *NwConsole User Guide for RSA NetWitness Platform*.
`name1="value1" name2="value2"`.

Parameter	Value	Effect
`capture.device.params`	`vlan-fix=true`	VLAN fixup always performed on all interfaces. The default value is `vlan-fix=false`.
`capture.device.params`	`interfaces=eth0:vlan,eth1`	VLAN fixup performed on traffic capture on `eth0` interface only

Parameter	Value	Effect
`capture.device.params`	`interfaces=eth0:vlan,eth1 vlan-fix=true`	VLAN fixup always performed because the `vlan-fix` setting overrides the interfaces setting.

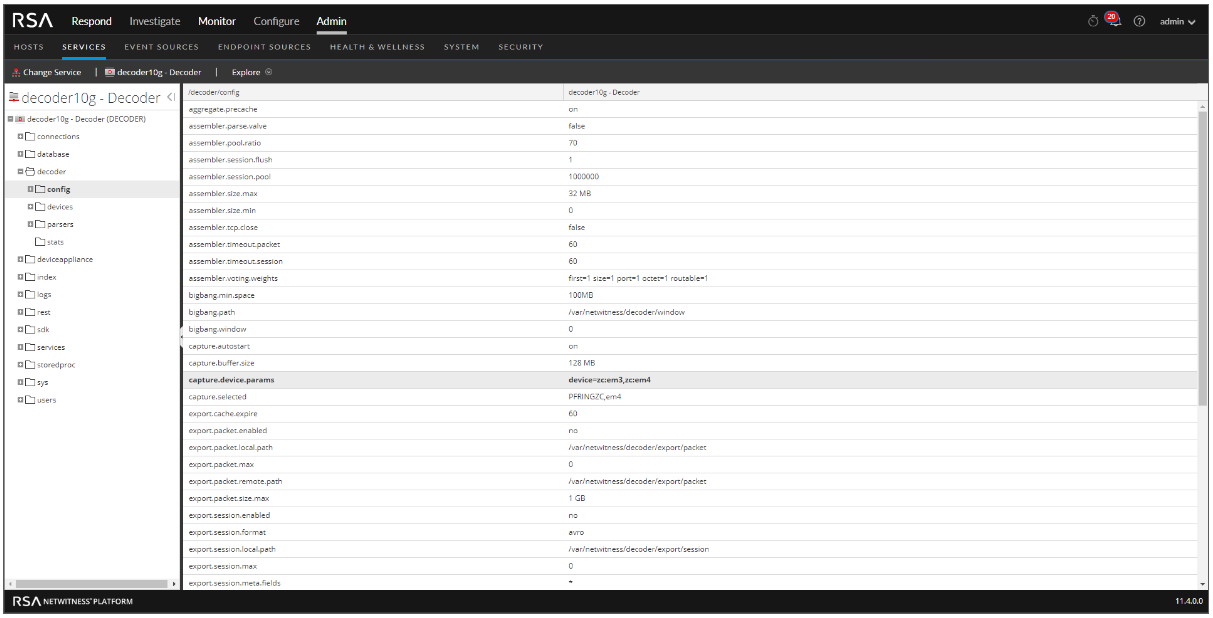
To configure the `packet_mmap_` adapter to preserve the VLAN tags in packets:

1. In the **Administration Services view**, select the Decoder service and  **> View > Config.**
2. In the **Services Config view**, set **Capture Interface Selected** to `packet_mmap_,ALL` adapter.



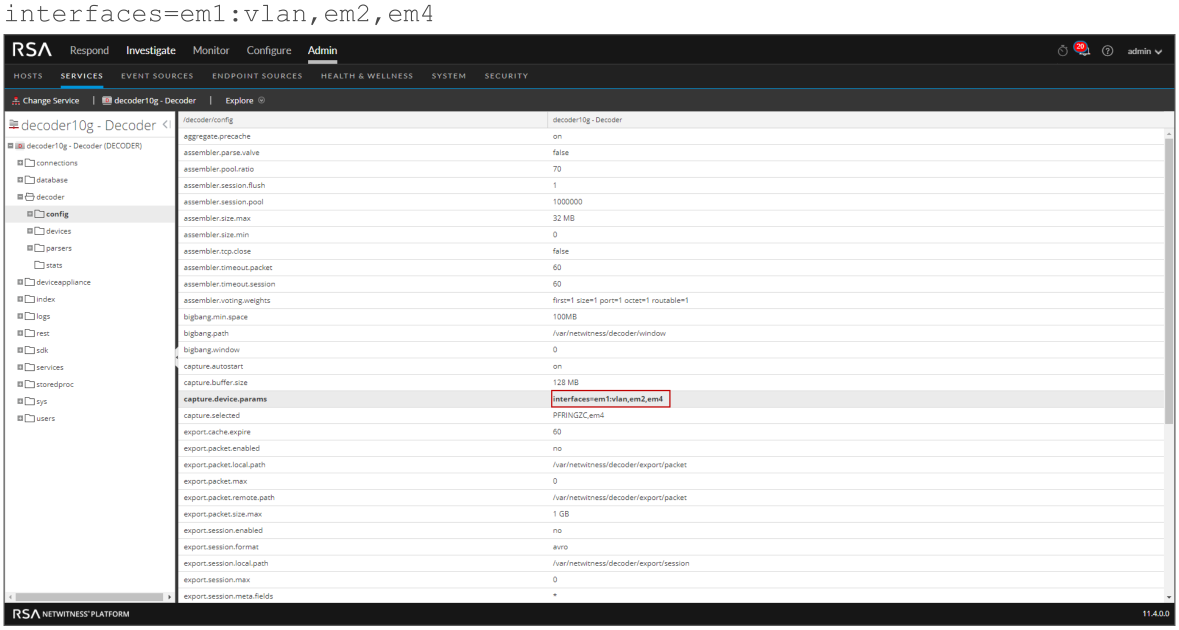
3. To go to the Services Explore view, click **Config** in the toolbar and select **Explore** in the drop-down list.

4. In the Services Explore view select **decoder** > **config**.



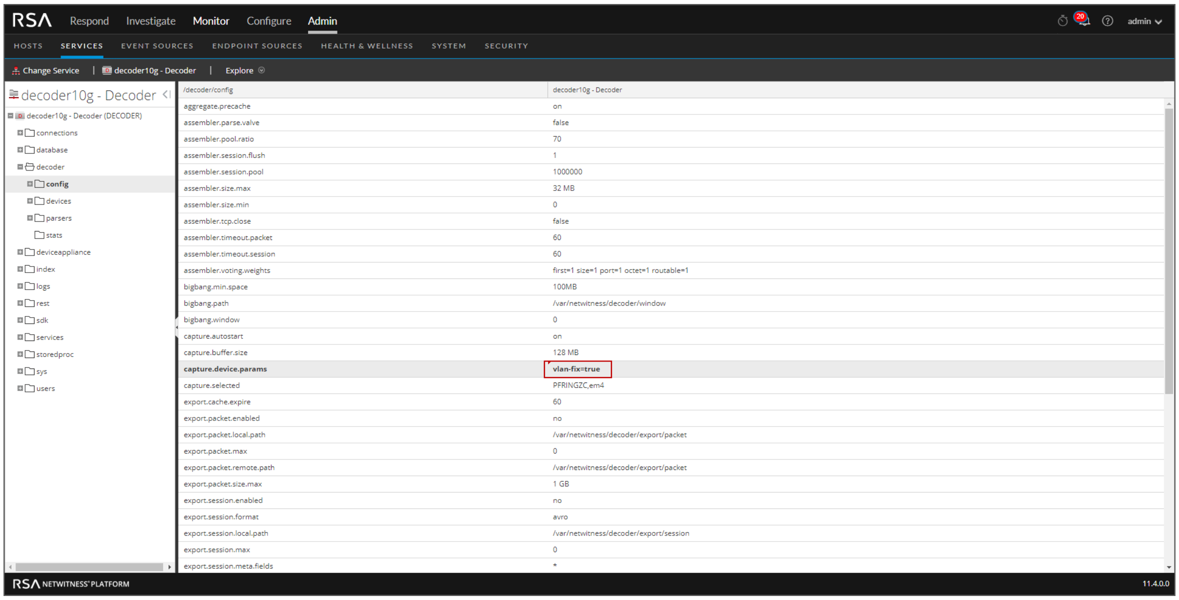
5. Click in the values column next to `capture.device.params`, and do one of the following:

- To preserve VLAN tags on an interface in the interfaces list, add **`:vlan`** after the interface name and press **Enter**. For example, this specifies that VLAN tags are preserved on `em1`, but not on `em2` and `em4`:
  `interfaces=em1:vlan,em2,em4`



  The change goes into effect immediately; only traffic on `em1` has the VLAN tags preserved.

- To preserve VLAN tags on all interfaces, enter the following and press **Enter**:
  `vlan-fix=true`

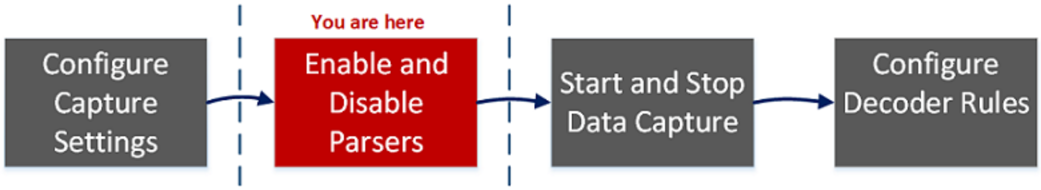


The change goes into effect immediately; VLAN tags are preserved on all capture interfaces.

Enable and Disable Parsers and Log Parsers

Administrators can see which parsers have been downloaded from Live and deployed on a Decoder or Log Decoder, see which of these have been enabled, and enable or disable parsers and log parsers.

The following figure illustrates commonly used settings on a Decoder. For a quick basic setup with only the required steps, see Decoder and Log Decoder Quick Setup.



You should only download and deploy the parsers you need for the following reasons:

- There is an impact on performance as you increase the number of deployed parsers.
- The more parsers you deploy, the more meta data created, which impacts data retention.
- Not having extra (unnecessary) log parsers deployed reduces the potential for mis-identification of messages.

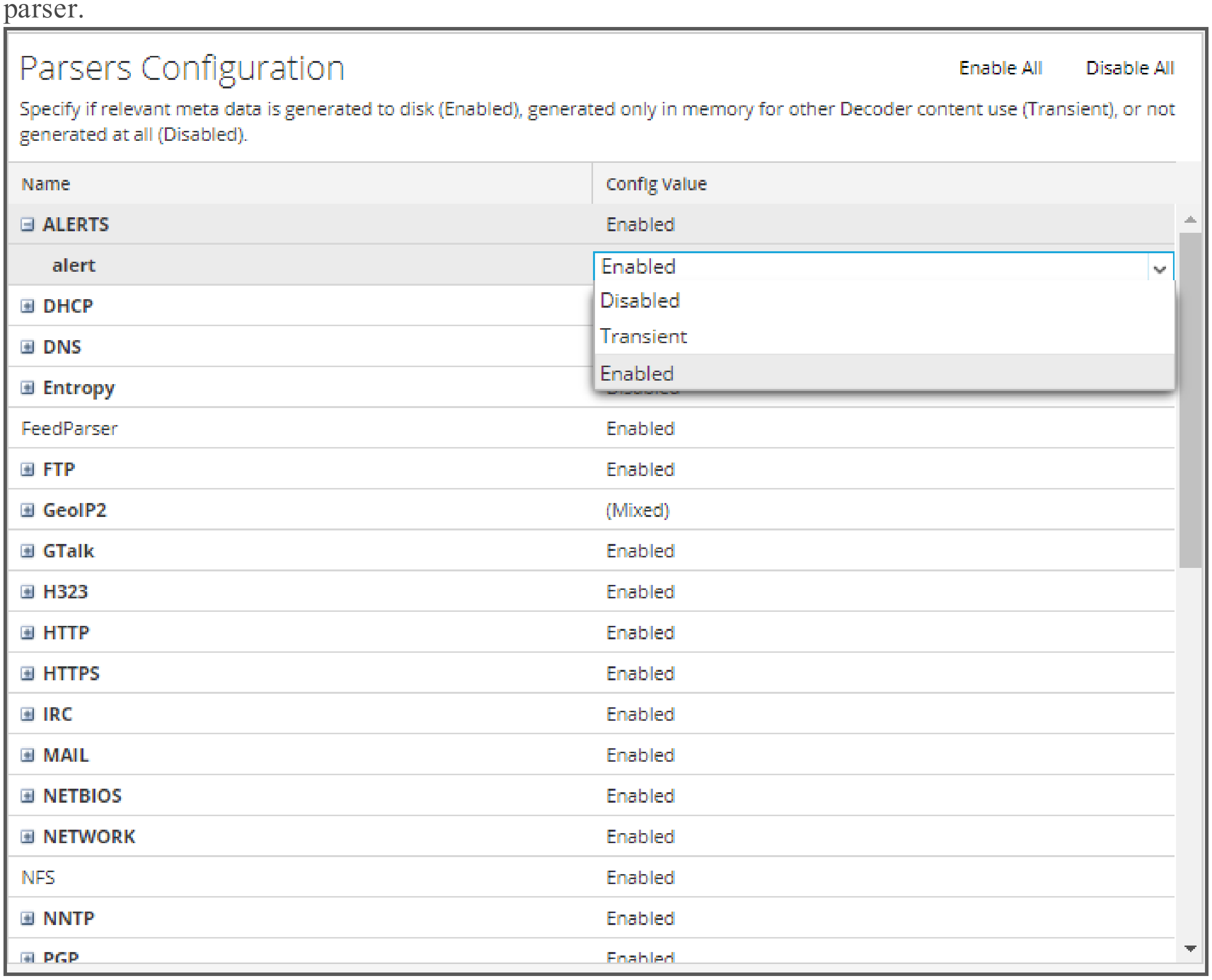
The Parsers Configuration panel provides a way to select parsers to use on the Decoder. Within some parsers, you can also configure the metadata that the parser creates. These are the options in the Parsers Configuration panel.

Option	Description
Enable All **Disable All**	These options provide a way to quickly select either all parsers or no parsers.
Name	The names of parsers available to the Decoder. A plus sign indicates that the metadata generated by the parser is configurable. Clicking the plus sign displays the metadata that the parser can create.
Config Value	A drop-down list changes the setting for the parser or metadata to **Enabled**, **Disabled**, or **Transient**. • When **Enabled**, the Decoder is using the parser to filter traffic. • When **Transient**, the Decoder is using the parser to filter traffic, and the generated metadata is not stored on disk. The transient metadata is available in memory to additional content (that is, parsers, feeds, and application rules) on that Decoder. This helps administrators to protect certain data and is usually done as part of a data privacy plan (see the *Data Privacy Management Guide*). • When **Disabled**, the Decoder is not using the parser. If the generated metadata for the parser is configurable, clicking the plus sign to expand the parser displays configurable meta keys and the same drop-down list selects the meta key the parser will create.

Note: For a Log Decoder, you must have previously deployed log parsers from Live. See the **Find and Deploy Live Resources** topic in the *Live Services Management* Guide for details. Go to the Master Table of Contents to find all RSA NetWitness Platform 11.x documents.

To enable or disable a parser, or to view the status for each parser:

1. Go to **Admin** > **Services**.
2. In the **Administration Services view**, select a Log Decoder or a Decoder, and  **>View** > **Config**.
3. In the **Parsers Configuration** panel, look for the Decoder parser or the Log Decoder event source parser.



4. In the **Config Value** column, note the current status for your parser.

   You can update the status of any individual parser by selecting its **Config Value** and selecting **Disabled**, **Transient**, or **Enabled** from the drop-down menu. Alternatively, you can select **Enable All** or **Disable All** to update the status for all of your log parsers at once.

5. Click **Apply**.

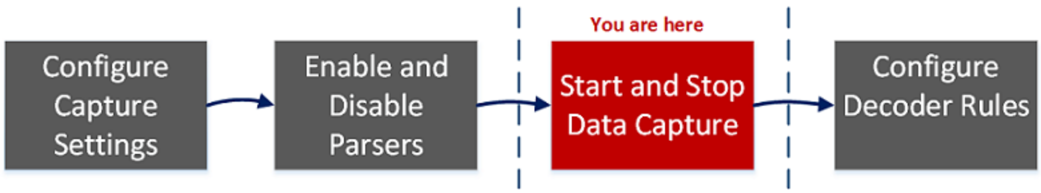
When you click **Apply**, note that all parsers are reloaded into NetWitness Platform. The status for each parser is updated, based on your selections.

Start and Stop Data Capture

When a Decoder starts up, it automatically begins aggregating data if **Capture Autostart** is enabled. When autostart is not enabled, you can start and stop data capture manually.

> **Note:** The Capture Configuration Settings in the Service Config view for a Decoder determine whether Capture Autostart is enabled.

The following figure illustrates commonly used settings on a Decoder. For a quick basic setup with only the required steps, see Decoder and Log Decoder Quick Setup. You may want to stop and start capture at other times, for example, before you shut down the service.



To start and stop capture:

1. Go to **ADMIN > Services.**
2. In the **Admin Services** view, select a Decoder or Log Decoder service, and select > **View > System**.
3. In the toolbar, click **Start Capture**.

   If the service is a Decoder, it begins capturing packets. If the service is a Log Decoder, it begins capturing logs.

   When packet or log capture is in progress, the option in the toolbar changes to **Stop Capture**, and the option to upload a file is unavailable.
4. Whenever you want to discontinue traffic capture on a Decoder, click **Stop Capture**.

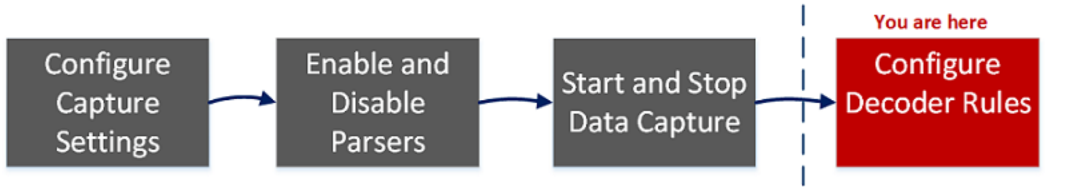
   Packet or log capture ceases, and the option to upload a file to the service is again available.

> **Note:** When you stop the Log Decoder service while capture is running, all events currently in Log Decoder memory will be processed and persisted. Should an issue arise where it is necessary to quickly shutdown the service, use the Services Explore view to stop capture (`/decoder stop`), passing the parameters `flush=false` before stopping the Log Decoder service. For further information, see the "Services Explore View" in the *Host and Services Getting Started Guide*.

Configure Decoder Rules

This topic provides procedures for creating and managing rules for Decoder or Log Decoder traffic capture in the **Services Config view** > **Rules** tabs . Services Config View - Rules Tabs provides details about the Rules tab options.

The following figure illustrates commonly used settings on a Decoder. For a quick basic setup with only the required steps, see Decoder and Log Decoder Quick Setup.



Capture rules can add alerts or contextual information to sessions or logs. They can also define which data is filtered out by a Decoder or Log Decoder. Rules are created for specific metadata patterns, which result in predefined actions when matches are found. For example, to keep all traffic that fits certain criteria, but discard all other traffic, you can create a rule to perform the necessary actions. When applied, rules affect both packet capture file importing, as well as live network capture.

Rule and Query Guidelines provides guidelines that all queries and rule conditions in NetWitness Platform Core Services must follow.

By default, no rules are defined when you first install NetWitness Platform. Until rules are specified, the packets are not filtered. You can deploy the latest rules from Live. You can define three types of rules: Network Rules, Application Rules, and Correlation Rules.

- Network rules are applied at the packet level and are made up of rule sets from Layer 2, Layer 3, and Layer 4. Multiple rules can be applied to the Decoder. Rules can be applied to multiple layers (for example, when a network rule filters out specific ports for a specific IP address). Network rules are only available on Network Decoders.
- Application rules are applied at the session level. If the first rule listed is not a match, the Decoder then attempts to match the next rule listed, until a match is found.
- Correlation rules are applied over a configurable sliding time window. When a match is found, the service creates a new super session that identifies other sessions that match the rule, then creates a session list for analysis.

The two most common uses of rules are:

- To alert, and thereby create a custom alert meta value, when certain conditions are found.
- To filter out certain types of traffic that do not add value to the analysis of the data.

Groups of capture rules form rule sets, which you can import and export. This feature enables use of multiple rule sets for various scenarios. You can import the exported rule set, in the form of an .nwr file, to other NetWitness Platform services, simplifying the deployment and configuration of multiple services.

Rule Processing

These are the principles governing capture rule processing:

- Multiple rules can be applied to the Decoder.
- Capture rules are executed one after the other, in sequence.
- Rule processing stops when all rules are processed or after a rule configured to stop rule processing is matched.
- A default rule can be used to either include or exclude all traffic not otherwise selected by a rule. A default rule, if used, must always be placed at the bottom of the rule list. Otherwise, rule processing stops as soon as the default rule is evaluated since, by definition, all traffic is selected by the default rule.
- When rule processing stops, the session is saved using the configured session options and debug options.

Rule and Query Guidelines

All queries and rule conditions in RSA NetWitness Core services must follow these guidelines:

All string literals and time stamps must be quoted. Do not quote numbers, MAC, or IP addresses.

- `extension = 'torrent'`
- `time='2015-jan-01 00:00:00'`
- `service=80`
- `ip.src = 192.168.0.1`

Note: The space on the right and the left of an operator is optional. For example, you can type a rule as `service=80` or `service = 80` .

Rule Examples

The following table shows examples of rule conditions. You can use rule conditions for log retention collections in an Archiver and for application, network, and correlation rules on a Decoder, Log Decoder, or Concentrator. Rule conditions are also used in all `WHERE` clauses in all Core database queries.

For detailed information on rule syntax in NetWitness Platform, see "`WHERE` Clauses" in the "Queries" section of the *Core Database Tuning Guide* .

Rule Name	Condition
ComplianceDevices	`device.group='PCI Devices' || device.group='HIPPA Devices'`
HighValueWindows	`device.group='Windows Compliance'`
MediumValueWindows	`device.type='winevent_nic' && msg.id='security_4624_security'`

Rule Name	Condition
LowValueWinLogs	`device.type='winevent_nic' && msg.id='security_4648_security'`
LowValueProxyLogs	`device.class='proxy' && msg.id='antivirus_license_expired'`
GeneralWindows	`device.type='winevent_nic'`

Invalid Rules

NetWitness Platform uses a rule parser that strictly defines valid syntax for rules and queries. When a Core service encounters invalid syntax, it writes a warning in the NetWitness Platform logs indicating the error.

Note: NetWitness Platform 11.x does not support parsing of legacy syntax rules (as version 10.6 did). After you update to NetWitness Platform 11.x, rules with invalid syntax are highlighted in the user interface, and no rules will be applied until the invalid rules are corrected. The Rule Editor provides additional tooltips. After you fix the rules, the highlights disappear. See Fix Rules with Invalid Syntax.

The `/decoder/config/rules/rule.errors` and `/concentrator/config/rules/rule.errors` stats, contain the count of rules with errors. If `rule.errors` is nonzero, NetWitness Platform generates a Health and Wellness alert to indicate that you need to fix the rules.

General Syntax Guidelines

- All text values must quote literal values. Example: `user = 'user1'`
- Quotes can use single or double quotes; but they must match. (You cannot start with a single quote and finish with a double quote.)
- If the literal value has a quote, you can escape it using double quotes. Example: `user = "User's"`

The following are valid syntax rules:

- All time values should use quotes for dates in this form: `time = 'YYYY-MM-DD HH:MM:SS'`
- All time values that are the number of seconds since EPOCH (Jan 1, 1970), should not be quoted. Example: `time = 1448034064`
- **Everything** else is unquoted: IP addresses, MAC addresses, numerics, and so on. Example: `service = 80 && ip.src = 192.168.1.1/16`

Capture Rule Syntax

Capture rules compare fields to values or to other fields. This is an example of a simple expression with a meta key on the left side of the operator and a value on the right side.

`ip.dst=192.168.1.1`

The syntax allows a meta key on the right side of the operator in Decoders and Log Decoders for application and network rules. Meta key comparison does not apply in the `where` clause in queries. This is an example of a simple expression with a meta key on the left side of the operator and a meta key on the right side.

`ip.src=ip.dst`

Rules that include a meta key comparison support renamed meta keys; if a rule queries a meta key that has been renamed, the rule is parsed for the renamed meta key. For example, if the meta key `ip_dst` is used in a rule, it is transparently mapped to the renamed meta key: `ip.dst`. Existing rules that include original keys will trigger alerts that include data for the renamed meta key.

This is an example of a rule that finds packets having the same `ip.src` address and `ip.dst` address on a Decoder, and generates an alert on the Concentrator.

`alert=ioc name=testRule8 rule="ip.src=ip.dst" order=38`

This rule would generate an error because `eth.src` and `ip.src` are incompatible formats.

`rule="eth.src=ip.src" name="testRule99" alert=ioc`

Values can be expressed as discrete values, a range of values, an upper or lower bound, or a combination of these three. You can create a greater than or less than comparison, and test equality or inequality against a range of values or an upper/lower bound.

`key 0-5` (a range of values)

`key = 0-u` is the same as `key >= 0` (upper bound, greater than or equal to)

The following table summarizes the operators on meta keys.

Left Operand Format	Operator	Right Operand Format	Description
any	`=`	compatible with left operand	Equality operator. You can use values or meta keys on the right side of the equality operator.
any	`!=`	compatible with left operand	Inequality operator. You can use values or meta keys on the right side of the inequality operator.
any	`<`	compatible with left operand	Less than operator. You can use values or meta keys on the right side of this operator.

Left Operand Format	Operator	Right Operand Format	Description
any	`<=`	compatible with left operand	Less than or equal to operator. You can use values or meta keys on the right side of this operator.
any	`>`	compatible with left operand	Greater than operator. You can use values or meta keys on the right side of this operator.
any	`>=`	compatible with left operand	Greater than or equal to operator. You can use values or meta keys on the right side of this operator.
text	`contains`	text	Find values that contain the right operand. You can use meta keys or values on the right side of this operator.
text	`begins`	text	Find values that begin with the right operand. You can use meta keys or values on the right side of this operator.
text	`ends`	text	Find values that end with the right operand. You can use meta keys or values on the right side of this operator.
text	`length`	integer	Find strings of a certain length. You can use meta keys or values on the right side of this operator.
any	`count`	integer	Find values with a specific number of occurrences within the session. You can use meta keys or values on the right side of this operator.
any	`ucount` and `unique`	integer	Finds a number of uniquely occurring values. You can use meta keys or values on the right side of this operator. For example, if the results include instances of a meta key with five unique values and three of the same value, the `ucount` is six.
N/A	`exists`	any	Finds any values for the meta key. You can use meta keys or values on the right side of this operator.
N/A	`!exists`	any	Finds any sessions in which the meta key does not occur. You can use meta keys or values on the right side of this operator.
text	`regex`	text	Finds values matching a regular expression. You can use values on the right side of this operator.

The following table summarizes other syntax elements used in rules.

Syntax element	Description
*	Default rule. By using an asterisk (*) as the sole character in a rule, that rule will select all traffic.
`u`	Upper bound of a range a range of times, IP addresses, or numeric formats. For example, to select all TCP ports above 40000, the syntax would be: `tcp.port = 40000-u`
`l`	Lower bound of a range of times, IP addresses, or numeric values. For example, to select all TCP ports below 40000, the syntax would be: `tcp.port = l-40000`
`-` (dash)	Denotes a range. This is only applicable to time values, IP or MAC addresses, or numeric values. Separate the lower and upper bounds of the range with a dash (-) character. For example, to select TCP ports between 25 and 443, the syntax would be: `tcp.port = 25-443`
`,` (comma)	Denotes a list of ranges or values or meta keys. Single values may be used as well as any combination of ranges and upper or lower bounds. Single meta keys may be used in a list. Meta keys and literal values cannot both appear on the right-hand side of an operator. For example, the following is valid syntax: `tcp.port = 1-10,25,110,143-225,40000-u`
`( )`	Grouping operator. An expression can be enclosed in parentheses to create a new logical expression. For example, the following would select traffic on port 80 to/from 192.168.1.1 OR traffic on port 443 to/from 10.10.10.1: `(alias.ip=192.168.1.1 && tcp.port=80) || (alias.ip=10.10.10.1 && tcp.port=443)`
`~`	Logical NOT operator, a negation of an expression.
`&&`	Logical AND operator, a conjunction of two expressions.
`||`	Logical OR operator, a disjunction of two expressions.

Configure Capture Rules

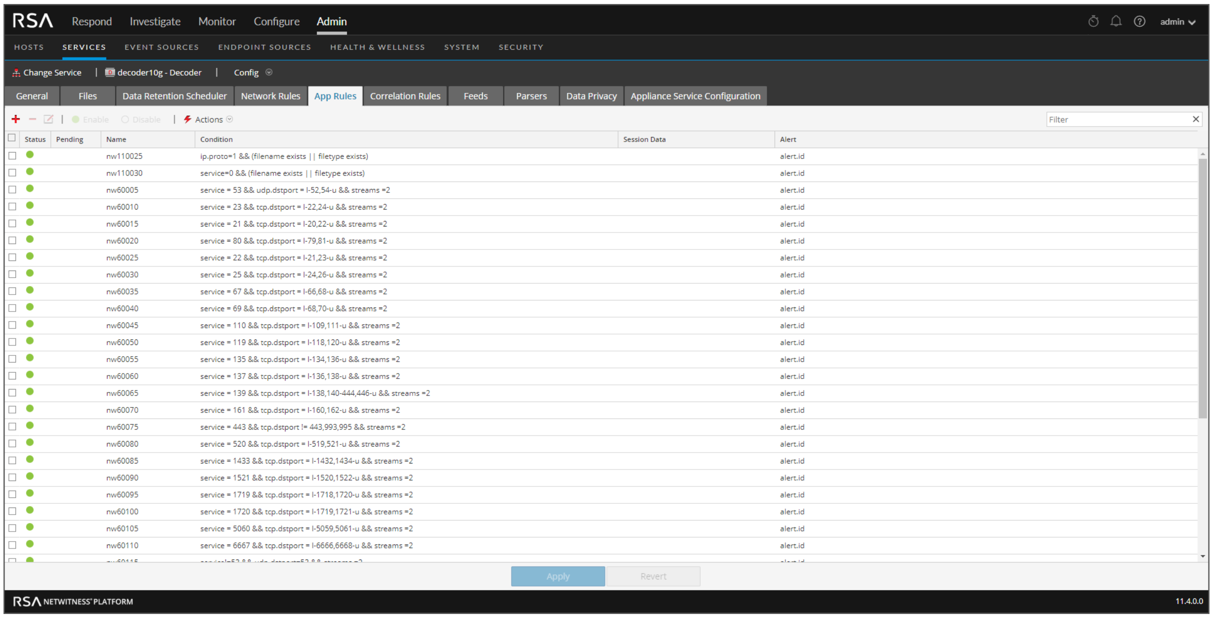
The Decoder and Log Decoder rules are editable in the Services Config view. While each type of rule (network, application, and correlation) has its own tab; the functions are similar for all types of rules. You can:

- Add, edit, and delete rules
- Enable and disable rules
- Change the execution sequence of rules
- Import rules from a file
- Export rules to a file

- Push rules to another service
- Revert or apply rule changes
- Restore one of the last ten rule configurations from a snapshot

To configure rules in the Rules tabs

1. Go to **Admin** > **Services**.
2. In the **Services** view, select a Decoder service and  > **View** > **Config**.
3. In the **Services Config** view, select one of the Rules tabs: Network Rules, App Rules, or Correlation Rules.
   The rules list for the selected rule type is displayed.



Each type of rule has a list with slightly different columns and different parameters. Several basic guidelines apply to all rule management activities:

- The rules are executed in the sequence they are displayed in the list. To change the execution sequence of rules, drag and drop rules to the appropriate location in the list or use the context menu options to arrange the rules in the list.
- To select a single row, click the row.
- To select a group of adjacent rows, click the first, then shift-click the row at the end of the group.
- To select multiple non-adjacent rows, click the first, then control-click the others.
- When editing rules in the Rules tab, you must apply the configuration changes in order to activate.
- Until changes are applied, you can discard edits to the list and revert to the unedited rules.
- Once rules are applied, you can recover the last ten rules configurations using the **History** option in the **Actions** menu.

To add a rule in any Rules tab, do one of the following:

- Click .
- Right-click a rule, and select **Insert Above** or **Insert Below** from the context menu.
  The Rule Editor dialog for that type of rule is displayed.

To remove a rule:

1. From any Rules tab, select the rules to remove from the rules list.
2. Click .
   The selected rules are removed from the list, but still exist on the service.

To edit a rule

1. From any Rules tab, select the rule to edit.
2. Click or double-click the rule row.
   The Rule Editor dialog for that type of rule is displayed.

To disable a rule:

1. From any Rules tab, select the rules to disable.
2. Click Disable.
   The status changes to disabled in the rules list, but the rule is still enabled on the service.

To enable a rule:

1. From any Rules tab, select the rules to enable.
2. Click Enable.
   The status changes to enabled in the rules list, but the rule is still disabled on the service.

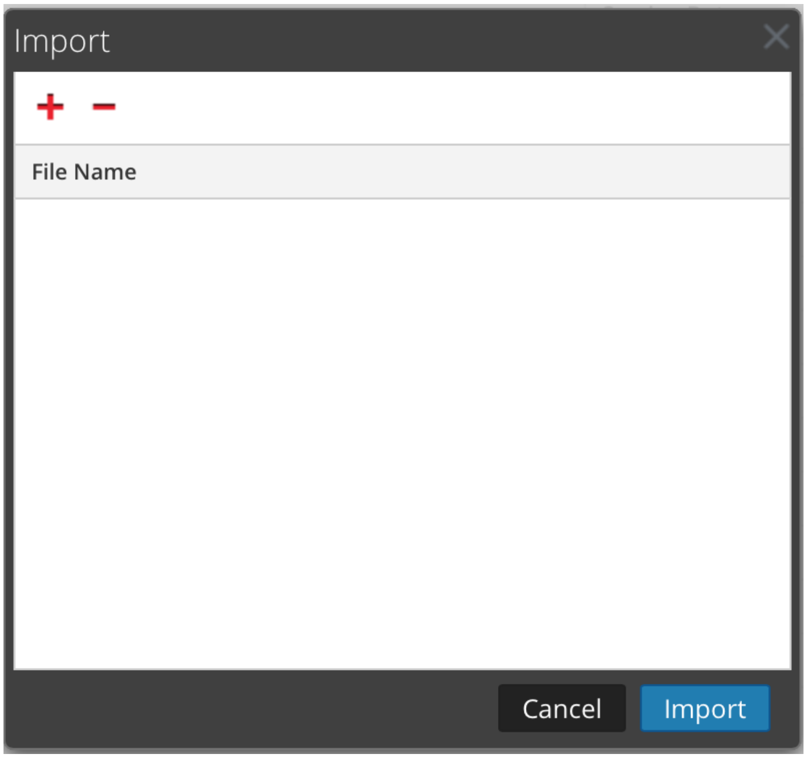
Import Rules from a File and Export Rules

You can import network, application, and correlation rules to a Decoder from a file that contains rules of the same type. After the rules are imported, you can edit and manage them as you would any other rules.

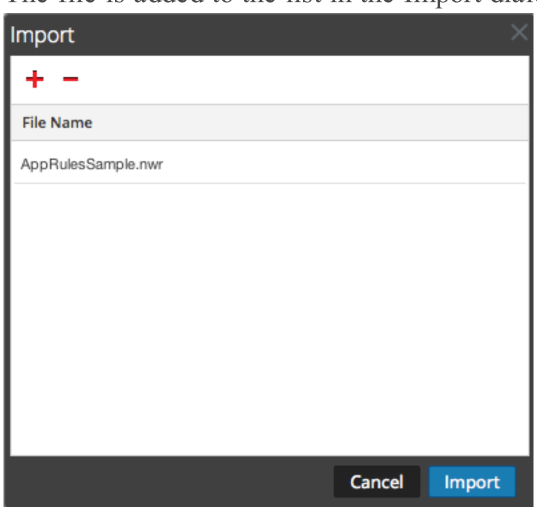
When you attempt to import a group of rules, NetWitness Platform Administration checks the type of rules imported. If you are successful, a message displays the number of rules imported. If the rule type differs from the active tab type, the rules are not imported. You must re-import the rules under the correct tab or select another file to import.

To import rules to a service:

1. From any Rules tab, select Actions > Import.
   The Import dialog is displayed.



2. Click .
   A view of the directory structure is displayed.
3. Choose one or more NetWitness rules (.nwr) files to import, and click **Open**.
   The file is added to the list in the Import dialog.



4. Click **Import**.
   The rules are imported into the user interface. Imported rules have a red corner in each edited column.
5. Edit or reorder the rules if needed.

6. To save the rules to the service, click **Apply**.
   The rules for the service are updated with the changes.

To export a rule to a file:

1. To export a subset of the rules, select the rules to be exported.
2. Do one of the following:
   - In the toolbar, select **Actions** > **Export** > **Selection**. (**Export** > **All** exports all rules in the rules list even if you have a subset selected for export.)
   - Right-click the selected rules and select **Export Selection**.

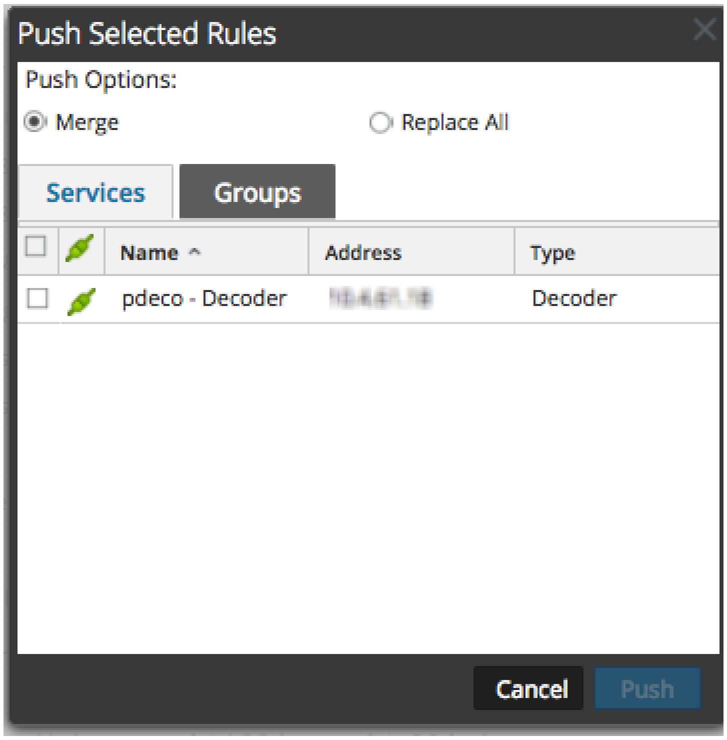
   A prompt for the filename is displayed.
3. Enter the filename and click **Export**.
   The .**nwr** file is downloaded.

Push Rules to Other Services

You can apply (push) rules or selected rules to other services (Decoders or Log Decoders) or service groups. When you push all rules to other services, all rules on the target services are removed and replaced with all of the rules on the source service.

To push selected rules from this Decoder to other Decoders:

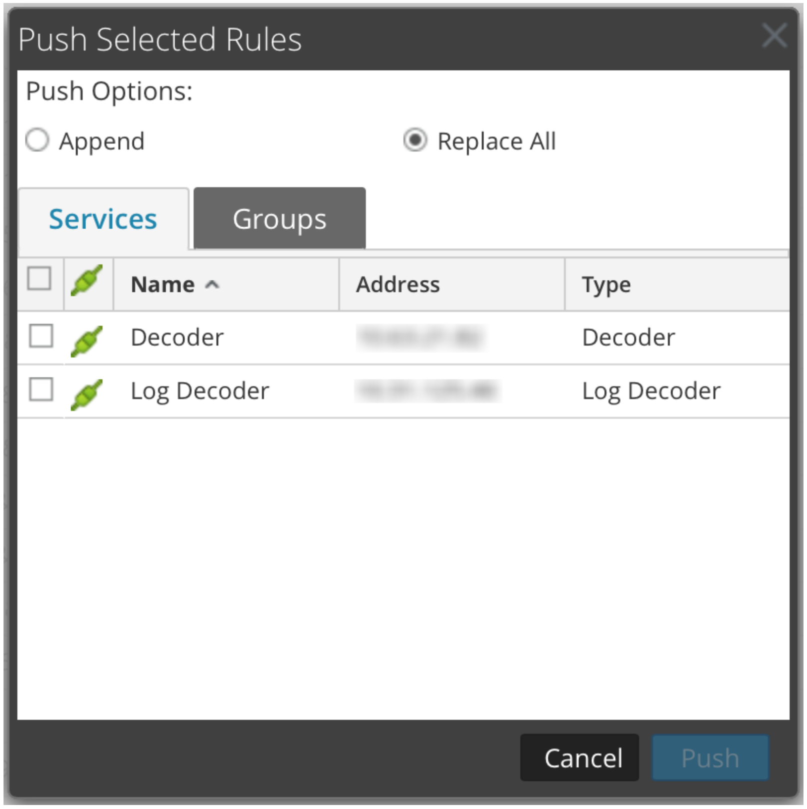
1. From any Rules tab, select the rules that you want to push to another Decoder.
2. Do one of the following:
   - Select **Actions** > **Push** > **Selection**.

- Right-click the selected rules and select **Push Selected Rules**.
  The Push Selected Rules dialog is displayed.
  

3. Select a Push Option:
   - Select **Replace All** to delete all rules on the target services and replace them with the selected rules. This is the default selection.
   - Select **Merge** to merge the selected rules with the existing rules on the target services.
4. On the **Services** tab, select the target services to receive the pushed rules, or select the groups of services from the **Groups** tab.
5. Click **Push**.
   The rules are pushed to the selected services and become effective immediately.

To push all rules from this Decoder to other Decoders:

1. From any Rules tab, select Actions > **Push** > **All**.
   (**Push** > **All** pushes all rules in the rules list even if you have a subset selected to push.) The Push Selected Rules dialog is displayed.



2. On the **Services** tab, select the target services to receive the pushed rules, or select the groups of services from the **Groups** tab.
3. Click **Push**.
   All rules from the target services are deleted and replaced with all of the rules from source service. The rules become effective immediately.

Change Execution Order of Rules

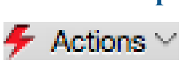
Capture rules are applied in the order they are displayed in the rules list. To reorder rules, use either of these methods:

- Drag and drop the rules in the appropriate location in the rules list.
- Right-click a rule to display the context menu, and use the **Cut** and **Paste** options.

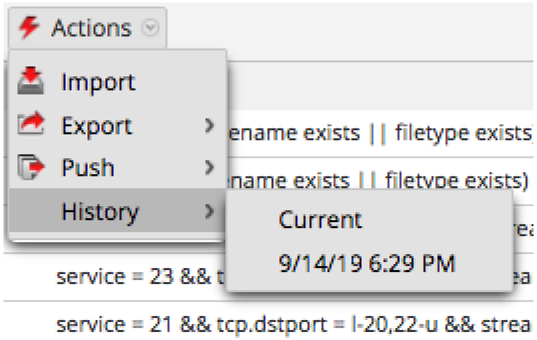
Restore a Rule Snapshot from History

NetWitness Platform keeps the last ten snapshots of rules applied to a service.

To restore a rules snapshot from history:

1. Select 

 > **History**.
   A submenu of snapshots is displayed.



2. Select the snapshot time from the submenu.
   The rules from the snapshot are loaded into the rules list, replacing the current set. But the current set is still in use on the service.
3. To apply the rules to the service, click **Apply**.
   The rules are applied to the service.

Configure Application Rules

Application layer rules are applied at the session level. The following are sample application rules.

To truncate packets carried via Server Message Block protocol (SMB), create a rule as follows:

- Rule Name: Truncate SMB
- Condition: `service=139`
- Rule Action: Truncate All

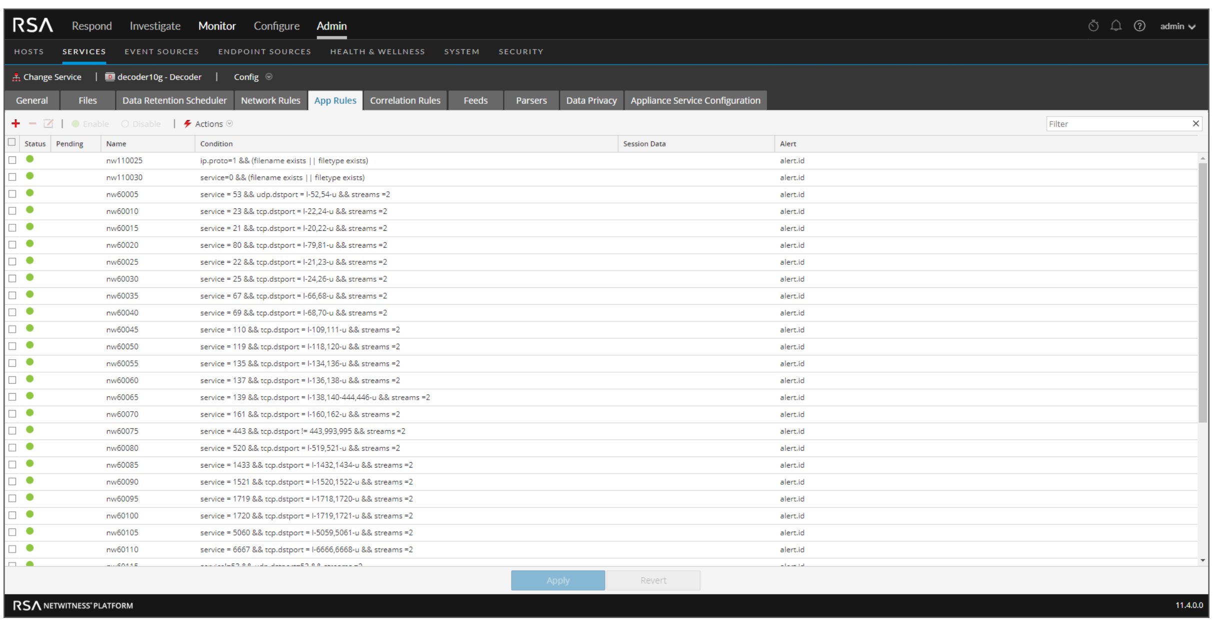
To retain email to and from a specific e-mail address, create a rule as follows:

- Rule Name: Retain Email Tom Jones
- Condition: `email='Tom.Jones@TheShop.com'`
- Rule Action: Keep

To add or edit an application rule:

1. Go to **ADMIN** > **Services**.
2. Select a Decoder or Log Decoder service and > **View** > **Config**.

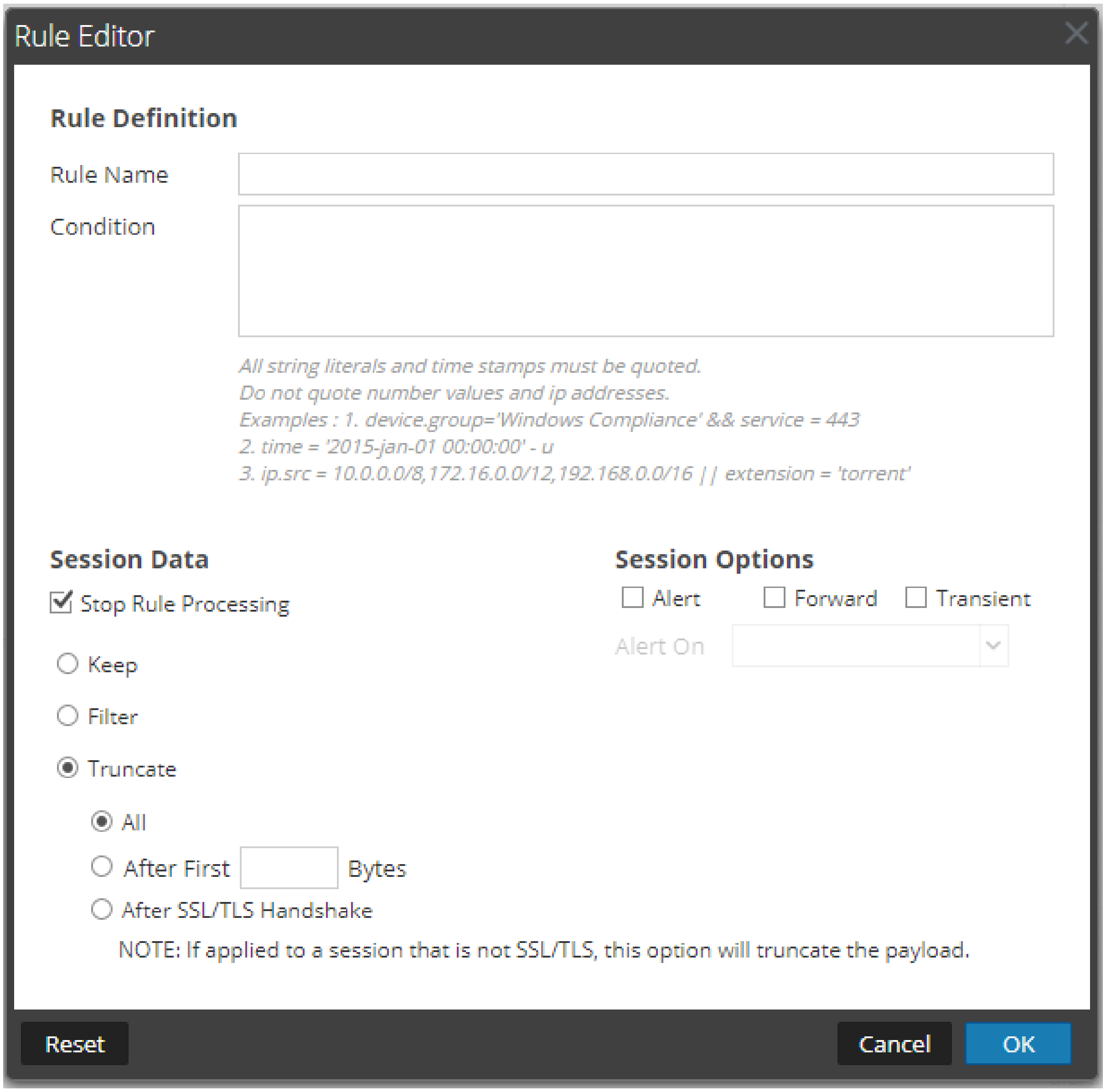
   The Systems Config view for the selected service is displayed.
3. Select the **App Rules** tab.



4. Do one of the following:

   - If adding a new rule, click .
   - If editing a rule, select the rule from the rules list and click .

5. The Rule Editor Dialog is displayed with application rule parameters.



a. In the **Rule Name** field, type a name for the rule. For example, for a rule that truncates all SMB, type **Truncate SMB**.

b. In the **Condition** field, build the rule condition that triggers an action when matched. You can type directly in the field or build the condition in this field using meta from the window actions. As you build the rule definition, NetWitness Platform displays syntax errors and warnings. For example, to truncate all SMB, type **service=139**.

   All string literals and time stamps must be quoted. Do not quote number values and IP addresses. Configure Decoder Rules provides additional details.

c. If you want rule evaluation to end with this rule, check the **Stop Rule Processing** checkbox.

d. In the **Session Data** section, choose one of the following actions to apply when a matching packet is found:

- **Keep**: The packet payload and associated meta are saved when they match the rule.
- **Filter**: The packet is not saved when it matches the rule.
- **Truncate**: Select a truncate option to execute when a packet matches the rule. The example uses the **All** option.
- **Truncate All** to save the packet headers and associated metadata, and do not save the packet payload.
- **Truncate After First <n> Bytes** to save the packet headers and associated metadata, and do not save the packet payload after the specified first <n> bytes, where <n> is a number of bytes.
- **Truncate SSL/TLS Handshake** to truncates the payload for all sessions except in the case of an SSL/TLS session, where the SSL exchange is preserved, but the rest of the payload is not saved. This option is for use with SSL parsers.

e. In the **Session Options** section, do any of the following:

- **To generate a custom alert** when a session metadata matches the rule, enable the Alert flag and select the name of the alert meta from the **Alert On** drop-down list.
- **To perform syslog forwarding** when the log matches the rule, enable the **Forward** flag. Make sure that:
  - You have enabled both the Alert and Forward flags to carry out syslog forwarding.
  - The name of the rule mentioned in the Rule Editor dialog matches the syslog forwarding destination name specified in the Log Decoder > View > Explore > `/decoder/config/logs.forwarding.destination` parameter
- **To prevent the alert metadata that is created from being written to the disk**, enable the **Transient** flag.

6. To save the rule and add it to the grid, click **OK**.

   The rule is added at the end of the grid or inserted where you specified in the context menu. The plus sign is displayed in the **Pending** column.

7. Check that the rule is in the correct execution sequence with other rules in the grid. If necessary, move the rule.
8. To apply the updated rule set to the Decoder or Log Decoder, click **Apply**.

NetWitness Platform saves a snapshot of the currently applied rules, then applies the updated set to the Decoder and removes the pending indicator from the rules that were pending.

Monitor Application Rules

The Decoder and Log Decoder keep track of how many times each application rule matches a session. These stats can be viewed by connecting to the Decoder or Log Decoder Explore view and viewing the properties on the `/decoder/config/rules/application` folder. Then, send the command "`statdump`" to that folder. The output of this message is a listing of the number of times each application rule is hit. The listing is ordered in the same order as the contents of the rule definitions in the `/decoder/config/rules/application` folder. For example, on a system with three application rules:

```
0001: hits=6543 loaded=true
0002: hits=9294 loaded=true
0003: hits=43 loaded=true
```

The hit counters for the application rules are reset whenever the parsers are reloaded.

Configure Correlation Rules

Basic Correlation Rules are applied at the session level and alert the user to specific activities that may be occurring in their environment. NetWitness Platform applies correlation rules over a configurable sliding time window. When the conditions are met, alert metadata is created for this activity and there is a visible indicator of the suspicious activity.

The following are sample correlation rules illustrating two use cases and the syntax.

Objective: In sessions where `port.dst` exists, if there is any combination of ip.src and ip.dst where the count of unique instances of `port.dst > 5` within one minute, then alert. To achieve this objective, create a rule as follows:

- Rule Name: IPv6 Vertical TCP Port Scan 5
- Rule: `port.dst exists`
- Instance Key: `ip.src,ip.dst`
- Threshold: `u_count(port.dst)>5`
- Time Window: 1 min

Objective: In sessions where `action==login` and `error==fail`, if there is any combination of `ip.src` and `ip.dst` that appears in more than 10 sessions within five minutes, then alert. To achieve this objective, create a rule as follows:

- Rule Name: IPv4 Potential Brute Force 10
- Rule: action=`'login' && error='fail'`
- Instance Key: `ip.src,ip.dst`
- Threshold: `count()>10`
- Time window: 5 mins

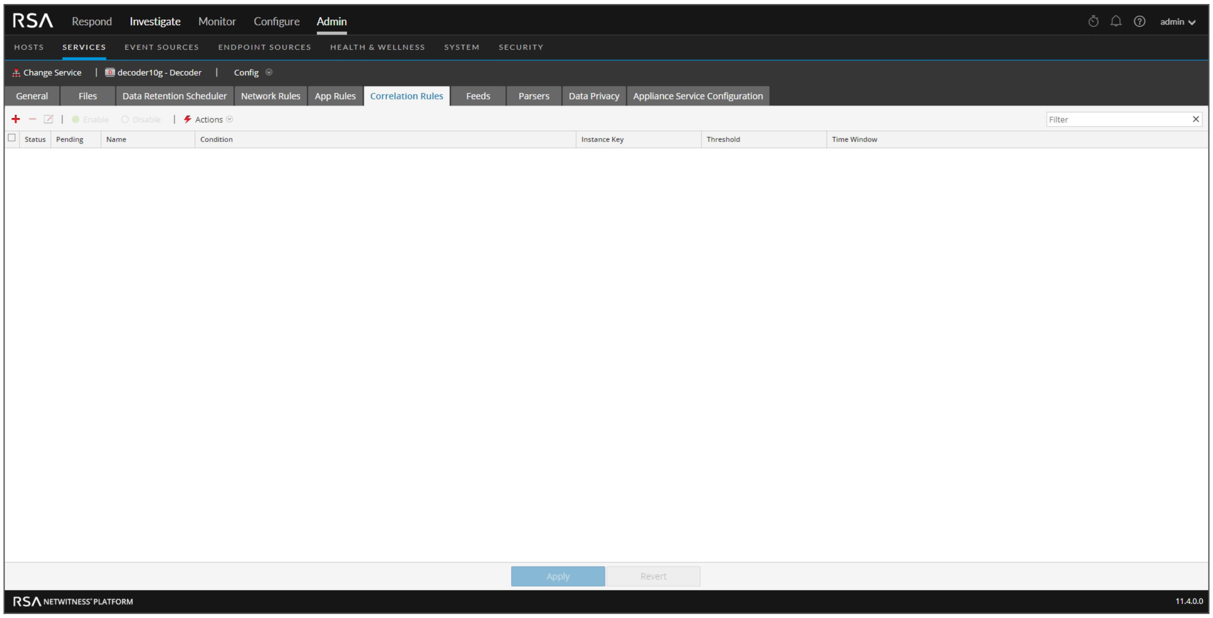
Both sample rules have the same instance key: `ip.src` and `ip.dst`. Because we are looking for unique combinations of `ip.src` and `ip.dst` that match the correlation condition, **`ip.src`** and **`ip.dst`** are **primary keys**.
Threshold can include an **associated key** that identifies the meta type that we are counting to determine if the condition is satisfied. In the first example, the associated key specified in Threshold is `port.dst`. We are counting unique instances of `port.dst` for every `ip.src/ip.dst` pair. In the second example, the associated key is not specified in the Threshold because it is merely a count of sessions. It is helpful to think of this scenario as counting unique session IDs and the associated meta is implicitly session.id. We are counting unique `session.id` for every `ip.src/ip.dst` pair.

Invalid use case: In sessions where (rule), if there is any combination of `ip.src` and `ip.dst` that have a unique count of `ipv6.dst > 5` within (time window), then alert. This case does not work because the associated key `ipv6.dst` is an IPv6 meta type. IPv4 and IPv6 meta types are not permitted to be used as associated keys.

To add or edit a correlation rule

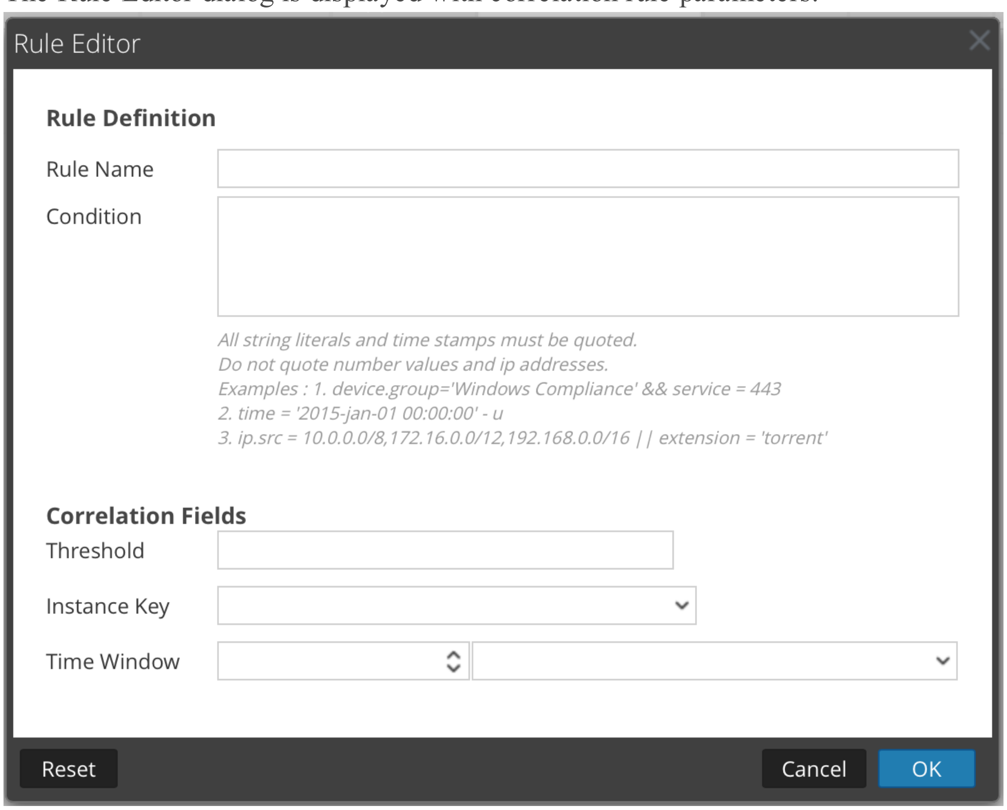
1. Go to **ADMIN > Services,** select a service, and **> View > Config**.
   The Service Config view for the selected service is displayed.

2. Select the **Correlation Rules** tab.



3. In the **Correlation Rules** tab, do one of the following:

- If adding a new rule, click .
- If editing a rule, select the rule from the rules grid and click .
  The Rule Editor dialog is displayed with correlation rule parameters.



4. In the **Rule Name** field, type a name for the rule. For example, to create the sample rule, **IPv6 Vertical TCP Port Scan 5**.
5. In the **Condition** field, build the rule condition that triggers an action when matched. You can type directly in the field or build the condition in this field using meta from the window actions. As you build the rule definition, syntax errors and warnings are displayed by NetWitness Platform. For example, to create the sample rule, type **tcp.dstport exists**. When this condition is matched, the session data action is performed.
   All string literals and time stamps must be quoted. Do not quote number values and IP addresses. Configure Decoder Rules provides additional details.
6. In the **Threshold** field, use one of the threshold parameters to specify the minimum number of occurrences required to create a correlation session and an associated key if required. The associated key cannot be an IPv4 or IPv6 meta type.

- `u_count(associated_key)` = the count of unique values of the specified key
- `sum(associated_key)` = the values of the specified key
- `count` = number of sessions (no associated key is specified)

7. In the **Instance Key** field, select the target indicator to base the event upon. This can be a single key or a compound key (two primary keys, separated by a comma).
8. In the **Time Window**, set the duration during which the threshold must be reached to create a correlation session.
9. To save the rule and add it to the grid, click **OK**.
   The rule is added at the end of the grid or inserted where you specified in the context menu. The plus sign is displayed in the **Pending** column.
10. Check that the rule is in the correct execution sequence with other rules in the grid. If necessary, move the rule.
11. To apply the updated rule set to the service, click **Apply**.
    NetWitness Platform saves a snapshot of the currently applied rules, then applies the updated set to the Decoder or Log Decoder.

Configure Network Rules

Network rules are applied at the packet level on a Decoder and are made up of rule sets from Layer 2, Layer 3, and Layer 4. Multiple rules can be applied at the packet level to a Decoder. Network rules can apply to multiple network layers (for example, when a network rule filters out specific ports for a specific IP address). Network rules do not apply to Log Decoders, they apply only to Network Decoders.

You can create and manage network rules in the Services Config view > Network Rules tab.

Note: Because network rules are applied on the packet level, you must specify both the source and destination meta keys in the rule condition. This is because the packet flow can occur on both sides at the packet level, while the directions are still not determined. For example, if you want to filter traffic on port 553 and port 55553, the condition should be written as follows:
`port.src=553 || port.dst=553 || port.src=55553 || port.dst=55553`
You must also specify both the source and destination meta keys in the conditions for `ip.src` and `ip.dst`. Specifying only source or destination in the condition will not work as expected.

Supported Meta Keys in Network Rule Conditions

The following table describes the meta keys that NetWitness Platform supports for use in network rule conditions.

Meta Key	Description
`eth.addr`	Ethernet source or destination address. Commonly known as the MAC address.
`eth.dst`	Destination Ethernet address. This is the same as the Ethernet address field except that it selects only packets where the destination address matches the selected value (s).
`eth.src`	Same as Ethernet destination except that it focuses on the source address.
`eth.type`	Ethernet frame type.
`hdlc.type`	Frame type of the HDLC frame.
`alias.ip`	IPv4 source or destination address in standard form. IP addresses can be entered in CIDR notation for subnets.
`ip.dst`	Destination IPv4 address in standard form. IP addresses can be entered in CIDR notation for subnets.
`ip.proto`	IPv4 protocol field.
`ip.src`	Source IPv4 address in standard form. IP addresses can be entered in CIDR notation for subnets.
`alias.ipv6`	IPv6 source or destination address in hex format. Generally IPv6 addresses are written as eight groups of four hex digits, thus expressing the entire 128 bit address length. Supports notation to represent multiple blocks of 0000 in an address. Does not support CIDR notation.

Meta Key	Description
`ipv6.dst`	Destination IPv6 address in hex format.
`ip.proto`	IPv6 protocol field. This maps to the Next Header field in the IPv6 header and uses the same values as the IPv4 protocol field.
`ipv6.src`	Source IPv6 address in hex format.
`port.dst`	Destination TCP port.
`tcp.port`	TCP source or destination port.
`port.src`	Source TCP port.
`port.dst`	Destination UDP port.
`udp.port`	UDP source or destination port.
`port.src`	Source UDP port.

The following are sample network rules.

To truncate all SSL from the source port, create a rule as follows:

- Rule Name: Truncate SSL
- Condition: `port.src=443`
- Rule Action: Truncate

To filter subnet traffic, create a rule as follows:

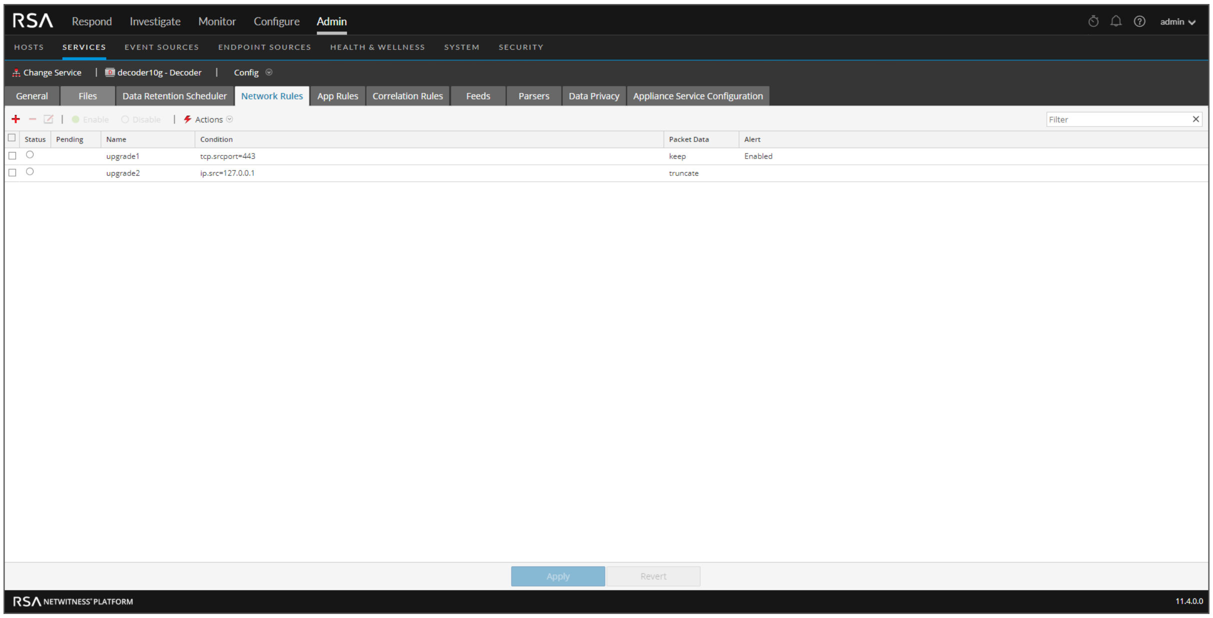
- Rule Name: Subnet Filter
- Condition: `alias.ip=192.168.2.0/24`
- Rule Action: Filter

Meta entities, which provide a way to work with several meta keys at the same time, can be used in application rules, but are not supported in network rules as the metadata available are too limited. For more information on meta entities, see the *Core Database Tuning Guide*.

To add or edit a network rule:

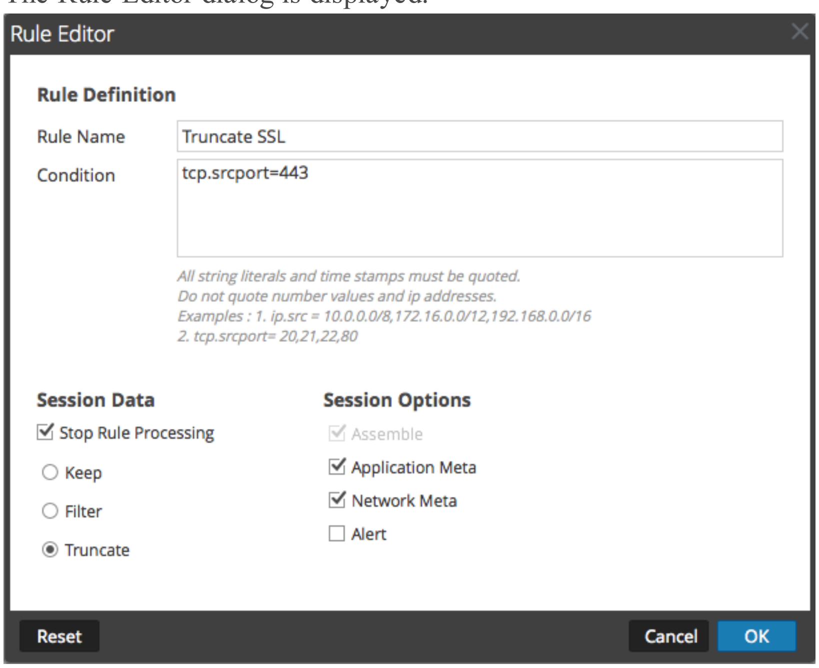
1. Go to **ADMIN > Services,** select a Decoder service, and > **View > Config**.
   The Services Config view for the selected service is displayed.

2. Select the **Network Rules tab**.
   The Network Rules tab is displayed.



3. In the **Network Rules** tab, do one of the following:

- If adding a new rule, click .
- If editing a rule, select the rule from the rules list and click .
  The Rule Editor dialog is displayed.



4. In the **Rule Name** field, provide a name for the rule. For example, for a rule that truncates all SSL from the source port, type **SSL Truncate**.

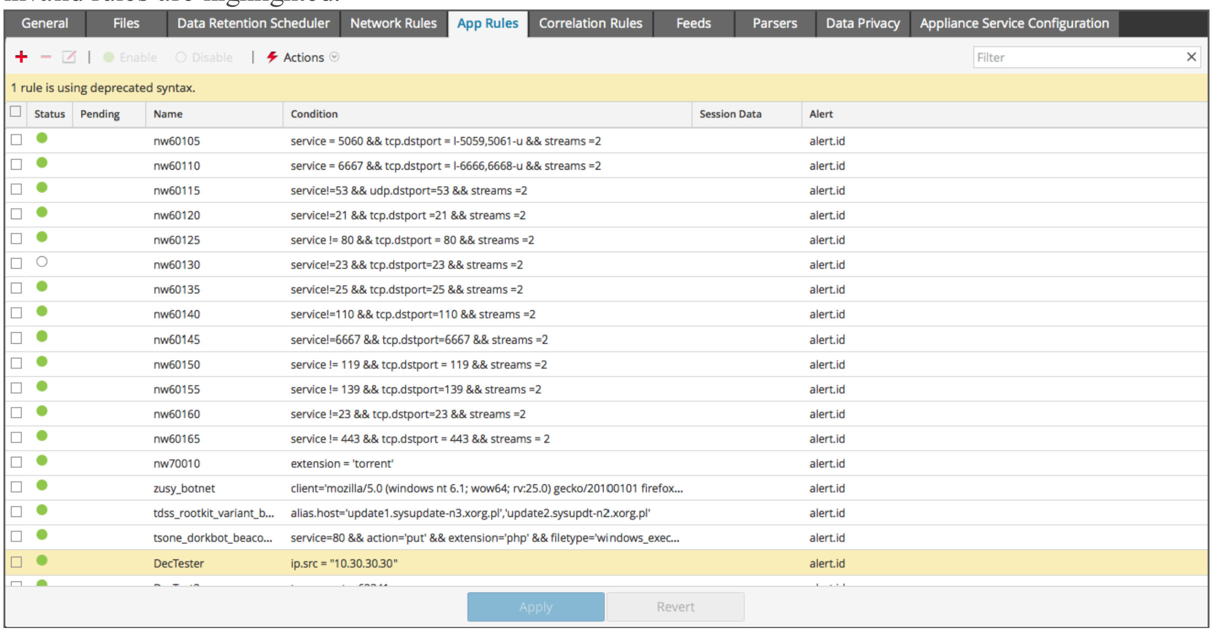
5. In the **Condition** field, build the rule condition that triggers an action when matched. You can type directly in the field or build the condition in this field using meta values from the window actions. As you build the rule definition, NetWitness Platform displays syntax errors and warnings. For example, to truncate all SSL from the source port, **`tcp.srcport=443`**.
   All string literals and time stamps must be quoted. Do not quote number values and IP addresses. Configure Decoder Rules provides additional details. Supported Meta Keys in Network Rule Conditions describes the meta keys that NetWitness Platform supports for use in network rule conditions.
6. If you want rule evaluation to end with this rule, select the **Stop Rule Processing** checkbox.
7. In the **Session Data** section, choose one of the following actions to apply when a matching packet is found:
   - **Keep**: The packet payload and associated metadata are saved when they match the rule.
   - **Filter**: The packet is not saved when it matches the rule.
   - **Truncate**: The packet payload is not saved when it matches the rule, but packet headers and associated metadata are retained.
8. In the **Session Options** section, select all options that apply of these four.
   - **Assemble**: The assembler assembles the packet chain when it matches the rule.
   - **Network Meta:** The packet generates network metadata when it matches the rule.
   - **Application Meta**: The packet generates application metadata when it matches the rule.
   - **Alert**: The packet generates a custom alert when metadata matches the rule.
9. To save the rule and add it to the rules list, click **OK**.
   The rule is added at the end of the list or inserted where you specified in the context menu.
10. Check that the rule is in the correct execution sequence with other rules in the list. If necessary, move the rule.
11. To apply the updated rule set to the Decoder, click **Apply**.
    NetWitness Platform saves a snapshot of the currently applied rules, then applies the updated set to the Decoder and removes the pending indicator from the rules that were pending.

Fix Rules with Invalid Syntax

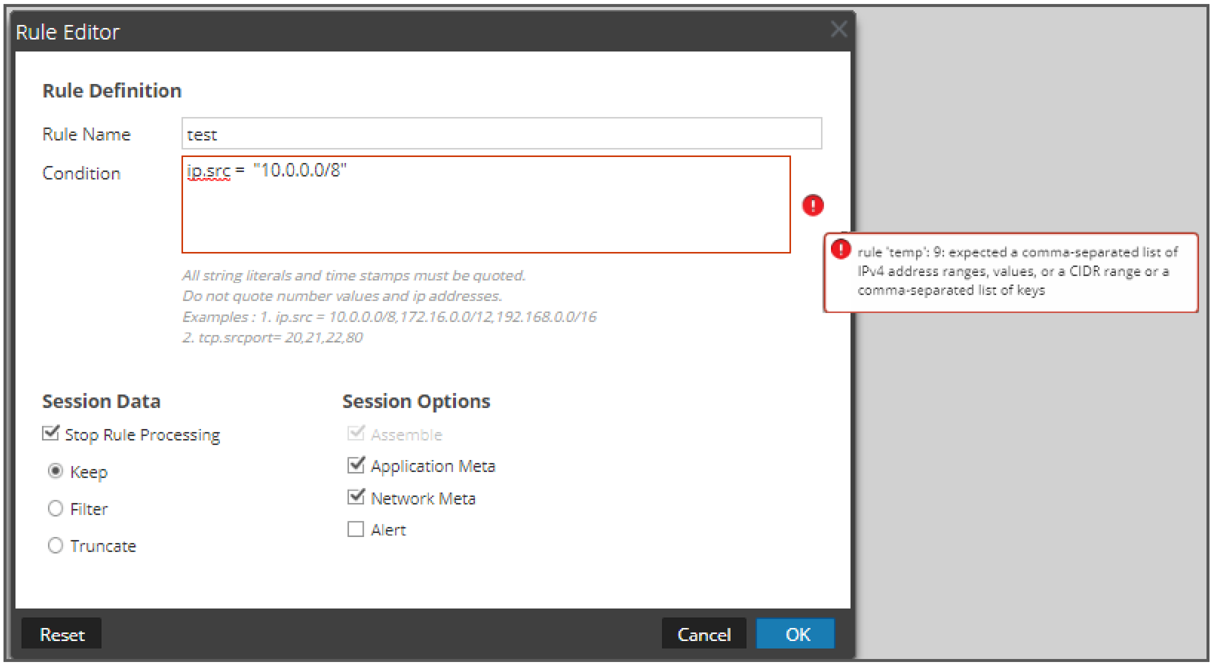
After an update to NetWitness Platform 11.x, the user interface highlights any rules with invalid syntax. The Rule Editor provides additional tooltips. After you fix the rules, the highlights disappear. Configure Decoder Rules provides guidelines that all queries and rule conditions in NetWitness Platform must follow.

To correct rules with invalid syntax:

1. Go to **ADMIN** > **Services**.
2. In the **Services** view, select a Decoder and > **View** > **Config**.
3. In the **Services Config** view, select one of the Rules tabs: Network Rules, App Rules, or Correlation Rules.
   The Rules tab for the selected rule type shows the number of rules using invalid syntax and the invalid rules are highlighted.



4. Select an invalid rule and click .
   The Rules Editor shows additional information for the invalid rule.



5. In the **Condition** field, correct the rule syntax.
   All string literals and time stamps must be quoted. Do not quote number values and IP addresses. Configure Decoder Rules provides additional details.
   For example, if the invalid rule condition is `ip.src="10.30.30.30"`, correct the syntax by removing the quotes: `ip.src=10.30.30.30`

6. Do one of the following:
   - To correct the rule individually, click **Save**.
     The corrected rule is applied independently to the Decoder. The corrected rule appears on the Rules tab without highlights.
   - To correct the rule and apply the rule to the Decoder later with other rules, click **OK**.
     The corrected rule appears on the Rules tab without highlights. The rule is not applied to the Decoder.

Decoder Commands for Managing Rules

In the NetWitness Core database, the Rules tree holds the main functionality related to managing rules for all Core services that have rules: Concentrators, Decoders, Log Decoders, and Archivers. Although you can manage rules in the NetWitness Platform user interface, advanced users may prefer to manage rules using a command line to add, merge, replace, delete, and validate rules on a service. This section provides a brief overview of the commands and their usage. These are the available commands:

- `add` - Adds a single rule at the specified position.
- `clear` - Deletes all existing rules in the current node on the service. For example, using the command in `/decoder/config/rules/application` node deletes all existing application rules on the Decoder.
- `delete` - Deletes one or more rules at a specified position and count.
- `merge` - Merges a pushed rule set with an existing rule set. Existing rules that match the incoming rules (by name or rule) are replaced; otherwise, rules are inserted by the position indicated as described in merge Command.
- `replace` - Deletes all existing rules and replaces them with the incoming rule set.
- `validate` - Validates the syntax of a rule, but does not validate the meta keys.

add Command

The `add` command adds the rule to the existing rule set. Formatting is important because the API uses double quotes in the rule language and also uses double quotes as parameters to all RSA NetWitness® Platform APIs. Therefore, you must escape any double quotes in the rule itself by preceding it with a backslash (\) character. This is the syntax of the command:

```
add rule=<string> name=<string> alert=<string, optional> atPos=<<uint32,
optional>
```

- `rule` is the rule to add. Be sure to place double quotes around any rules with white space and to escape any double quotes that are part of the rule with a backslash.
- `name` is the name of the rule.
- `alert` is the alert for the rule (if any).
- `atPos` is the position at which the rule should be added (1 based). Zero is the top of the list and any number larger than the current size of the list is appended to the list.

This is an example of command to add a rule using NwConsole

```
send /decoder/config/rules/application add rule="ip.src exists" order=1
alert=ioc name=testrule
```

For example, take the following rule:

```
alias.host = "myPC" && country.src="china","russian federation"
```

To add this as a rule, you would need to send the parameters as follows:

```
rule="alias.host = \"myPC\" && country.src=\"china\",\"russian federation\""
name=myRule filter
```

Notice how all the double quotes had to be escaped inside the rule parameter. A simple trick to make this more readable is to use single quotes inside the rule. Single and double quotes are interchangeable in the rule and query language, but not in parameters for the API (only double quotes are supported there). Therefore, this is more readable:

```
rule="alias.host = 'myPC' && country.src='china','russian federation'"
name=myRule filter
```

merge Command

The `merge` command is used to merge an incoming list of rules with the existing rules on the service. This is how it works:

- It finds existing rules that match via the name OR via a matching rule, updates the existing rule name, and keeps the same position.
- It inserts new rules into the rule list based on the NUMBER position. If the number is zero, it goes to the top of the list.
- It processes the rules in the order received so if you have two rules numbered zero, the second rule is processed after the first and claims the top spot. All existing rules are pushed down two places. Any numbers higher than existing rule positions are appended after the last existing rule and numbered in sequence.
- Any non-numbered rule is appended after the last existing rule and numbered in sequence.

This is the syntax of the merge command:

```
merge --file-data=<string> --file-format<string>
```

- `file-data` is the full path and name of the rules file to merge.
- `file-format` is the format of the rules file. Valid values are `params-list`, `string, params`, `binary`, and `params-binary`.

Methods of Sending a List of Rules to a Service

There are two ways to send a list of rules. You can send them as a `.nwr` (NetWitness Rule) file or as a numbered set of parameters, each number indicates the position to insert the rule at as well as the encoded rule. If you want to see the current list of rules on a service, you need to run the `ls` command on the rule category (for instance, application rules on a Decoder are found in `/decoder/config/rules/application`).

This is an example of commands to list the existing rules using NwConsole:

```
login <hostname>:50004 <username> <password>
cd /decoder/config/rules/application
ls
```

This is another example to list existing rules in NwConsole:

```
send /decoder/config/rules/application ls
```

This is an example of the command to point to network rules in the RESTful port, which supports a basic `admin HTML` app.

```
http[s]://<decoder>:50104/decoder/config/rules/network
```

Send a NetWitness Rule File

Let's start with an example `nwr` file, each rule must be on a separate line:

```
rule="ip.src=192.168.0.1" name=first keep
rule="ip.src=192.168.1.1" name=second alert=ioc
rule="ip.src=192.168.2.1" name=third filter
```

To push and merge rules using NwConsole, use the following commands:

```
login <hostname>:50004 <username> <password>
send /decoder/config/rules/application merge --file-data=/root/App_
Rules.nwr --file-format=params-list
```

To replace the existing rules with the rules in the file, instead of using the `merge` command, use the `replace` command.

```
send /decoder/config/rules/application replace --file-
data=<pathname> --file-format=params-list
```

To merge the rules in an `nwr` file using the RESTful port, you can use a `curl` command that pushes the rules:

```
curl -u "<username>:<password>" -H "Content-Type: application/octet-
stream" --data-binary @<pathname> -X POST
"http://<hostname>:50104/decoder/config/rules/application?msg=merge"
```

The examples are pushing application rules. To push network rules, send the rules to `/decoder/config/rules/network`. For correlation rules, send the rules to `/decoder/config/rules/correlation`.

Send Numbered Parameters

The other way to send a list of rules is to send them as numbered parameters. The difficulty with this method is remembering to escape the quotes within each numbered rule. Though it is only a problem if you are trying to do it by hand. For instance, to send the same rules above as parameters via NwConsole, use the following command:

```
send /decoder/config/rules/application merge
1="rule=\"ip.src=192.168.0.1\" name=first keep"
2="rule=\"ip.src=192.168.1.1\" name=second alert=ioc"
3="rule=\"ip.src=192.168.2.1\" name=third filter"
```

This command is hard to read because you have to escape the inner quotes with a backslash (\). Otherwise, these two commands accomplish the same thing. Merging or adding three rules in positions 1, 2 and 3. If you think the above was hard to read, this is what the equivalent `curl` command looks like:

```
 curl -u "<username>:<password>"
"http://<hostname>:50104/decoder/config/rules/application?msg=merge&1=rule%3D%
22ip.src%3D192.168.0.1%22%20name%3Dfirst%20keep&2=rule%3D%22ip.src%3D192.168.1
.1%22%20name%3Dsecond%20alert%3Dioc&3=rule%3D%22ip.src%3D192.168.2.1%22%20name
%3Dthird%20filter"
```

For more details on how to escape double quotes inside parameters, see add Command.

Ordering Rules When Pushing

Pushed rules are ordered in one of two ways. When passing as parameters, the number of each parameter determines the insertion order. If it is not actually a number, `merge` checks for an `order` parameter within the rule itself and uses that value if found.

> **Note:** Using `order` is the only way to set the order with a `.nwr` file. If neither a number nor an `order` parameter is found, there are no guarantees of the insertion order.

Example

A Decoder has the following application rules installed; notice the numbering is ALWAYS consecutive and starts at 1:

```
0001 : rule="ip.src = 192.168.0.1 || ip.dst = 192.168.0.1 || alias.host = 'My-PC'" name=first keep
0002 : rule="ip.src=192.168.1.1" name=second alert=ioc
0003 : rule="ip.src=192.168.2.1" name=third filter
```

And you want to merge the following four rules:

```
rule="ip.src=192.168.3.1" name=third keep
rule="ip.dst=192.168.4.1" name=NewRule filter order=0
rule="alias.host = 'pc1','pc2'" name=filterTheseNames filter order=append
rule="service=80,443" name=web filter order=3
```

Use any method to push your rules and this is what you end up with:

```
0001 : rule="ip.dst=192.168.4.1" name=NewRule filter order=1
0002 : rule="ip.src = 192.168.0.1 || ip.dst = 192.168.0.1 || alias.host = 'My-PC'" name=first keep order=2
0003 : rule="service=80,443" name=web filter order=3
0004 : rule="ip.src=192.168.1.1" name=second alert=ioc order=4
0005 : rule="ip.src=192.168.3.1" name=third keep order=5
0006 : rule="alias.host = 'pc1','pc2'" name=filterTheseNames filter order=6
```

Are there any surprises here? This is how each rule was processed.

`1. rule="ip.src=192.168.3.1" name=third keep`

This rule had the same name as an existing rule on the Decoder (third). So the rule updated the existing rule, `changing _filter_ to _keep_`.

`2. rule="ip.dst=192.168.4.1" name=NewRule filter order=0`

This rule is new and had `order=0` in it, which means insert at the very top.

`3. rule="alias.host = 'pc1','pc2'" name=filterTheseNames filter order=append`

This rule had a non-number `append` for `order`, therefore, it went to the end of the list. You can accomplish the same thing by giving a very large number, like `999999`.

`4. rule="service=80,443" name=web filter order=3`

This rule is last but has `order=3`, therefore, if it does not match an existing rule by name or the text of the rule itself, it should be placed in position 3. And there it is, the third rule in the list. Any rules that follow were pushed further down.

replace Command

The `replace` command removes all existing rules and replaces them with the incoming rule list. Refer to merge Command for details on how to format the incoming rule list and how ordering works.

This is an example of the `replace` command using a NetWitness Rule File :

```
send /decoder/config/rules/application replace --file-data=/root/Decoder-
AppRules.nwr --file-format=string
```

This is an example of the `replace` command using Numbered Parameters :

```
send /decoder/config/rules/application replace 1="rule=\"ip.src exists\"
name=\"test rule\" order=1 alert=ioc"
```

clear Command

The `clear` command removes all existing rules on the service. This is an example of the command:

```
send /decoder/config/rules/application clear
```

delete Command

The `delete` command deletes one or more rules on the service.

```
delete atPos <uint32> count <uint32, optional>
```

- `atPos` deletes the rule at the given position. Rules are numbered starting with 1 and go in sequential order.
- `count` deletes one or more rules starting `atPos`. This is an optional parameter defining the number of rules to delete starting `atPos`. The default value is 1.

This example of the command deletes four rules beginning at position 0003:

```
send /decoder/config/rules/application delete atPos=0003 count=4
```

validate Command

The `validate` command takes the provided rule and verifies that it parses correctly. Keep in mind that this command cannot verify whether language keys and entities are valid.

```
validate rule <string>
```

`rule` - is the name of the rule to validate. Make sure to place double quotes around any rules with white space.

Configure Parsers and Feeds

Parsers and feeds are responsible for analyzing the packets and logs when captured or imported in Decoders. Most commonly, they are used for static metadata extraction and service identification. The flexible definition allows custom extension of the core defined services to provide extra service type identification and metadata extraction. This is important due to the volume of custom applications that are used on networks. See the following topics for more information.

Configure Parsers

Configure Feeds

Configure Parsers

NetWitness Platform has a set of native parsers that are defined by the system, and also provides the option to add additional parsers. Each parser is configurable in the Services Config View - General Tab. The Parser Configuration panel provides a way to enable or disable parsers to use on Decoders in addition to limiting the metadata that the parser creates.

There are also several types of custom configurable parsers:

- GeoIP2 – This parser associates IP addresses with geographical locations. For new installations and upgrades, the GeoIP2 parser is enabled by default. For more information on these parsers, see GeoIP2 Parsers.
- Search – This parser is user-configured to generate metadata by scanning for pre-defined keywords and regular expressions.
- FLEXPARSE (deprecated) – This is a generic parser definition language for extending the existing application protocol support of the Decoder. By default this parser is disabled (see Enable or Disable Lua and Flex Parsing Systems).
- Lua – This parser is defined using the Lua scripting language for extending the existing application protocol support of the Decoder.
- Log – This application parser supports the Log Decoder and is configured to generate metadata by scanning log files.
- Snort® – This parser supports the payload detection capabilities of Snort IDS rules. Snort rules and configuration are added to the `parsers/snort` directory for Investigation and Decoder (see Snort Parsers).

In the Services Config view > Parsers tab, you can view deployed parsers on a Decoder, upload parsers, and delete deployed parsers. The user interface includes an Indicator if the parser originated from Live Services, installed through NetWitness Platform, or uploaded manually. Parsers can be added and removed while a Decoder is running without affecting capture.

In addition, you can download parsers using NetWitness Platform Live Services.

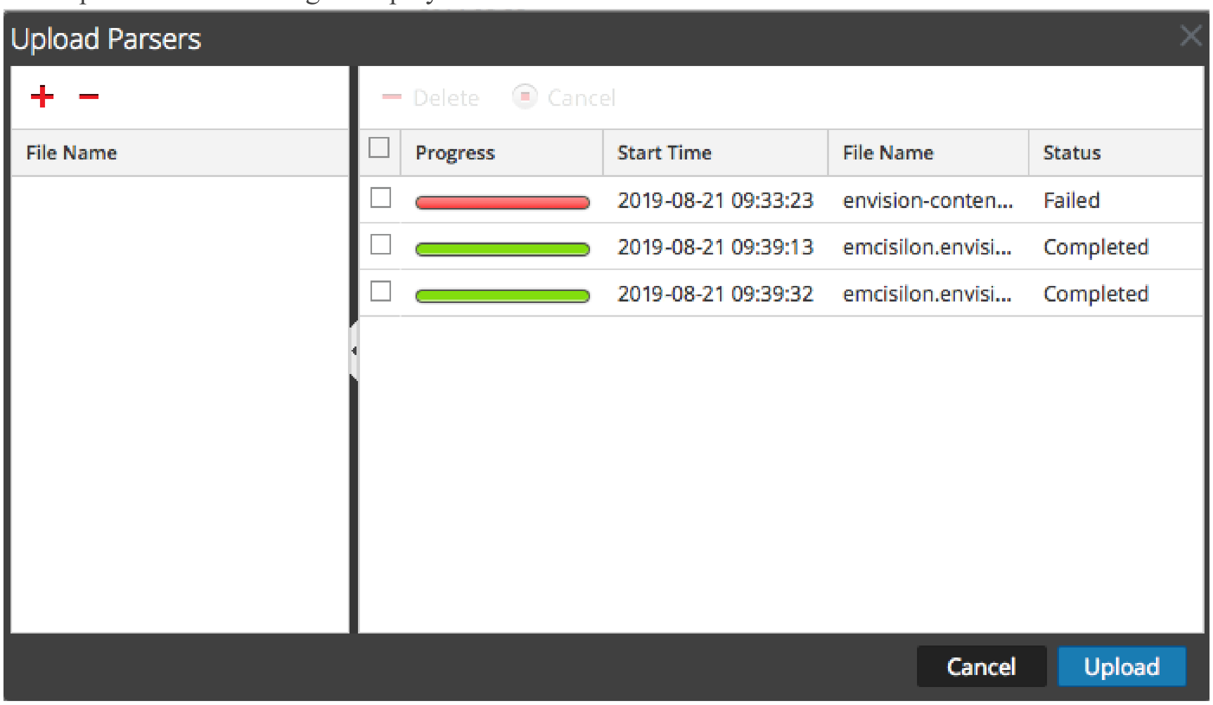
Upload and Delete Custom Parsers

RSA NetWitness Platform has the ability to upload parsers from your local system and delete these parsers.

Upload Parsers to a Decoder or Log Decoder

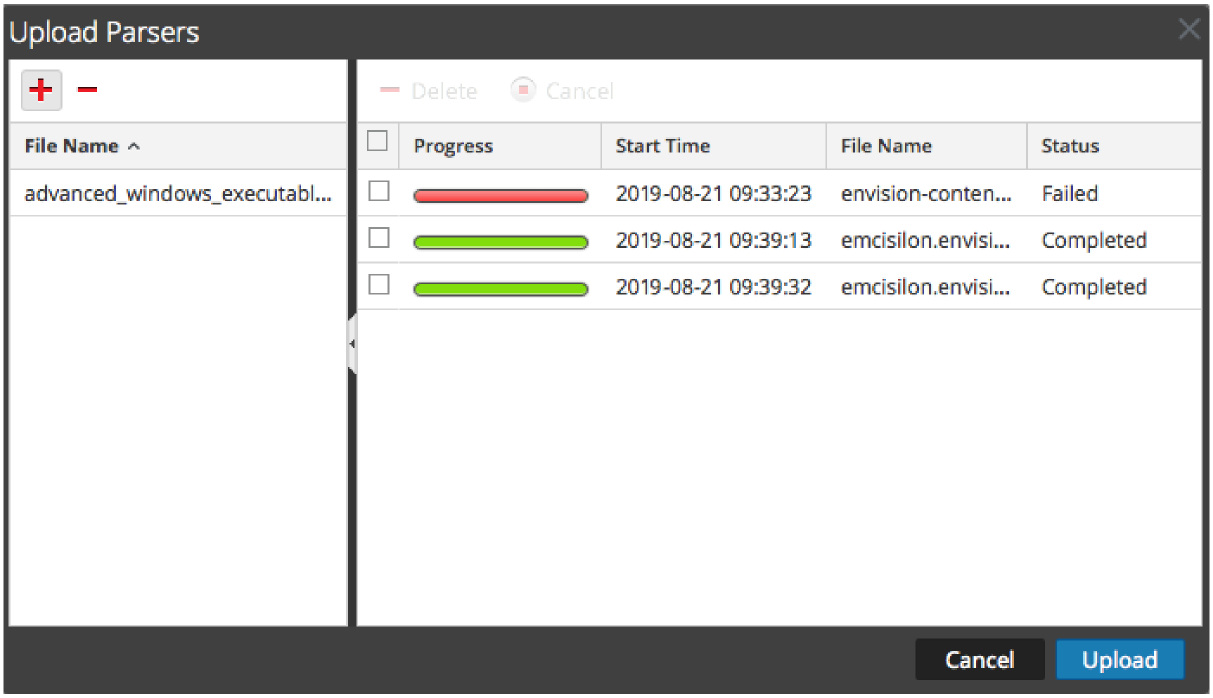
The Upload option in the Service Config view > Parsers tab displays the Upload Parsers dialog, in which you can manage the uploading of parsers to a Decoder or Log Decoder. In the File list, you prepare a list of parsers for uploading. You can add files from a directory structure, and delete files from the list if you decide that you don't want to upload a particular file. When the list is ready, clicking Upload starts the upload process.

1. Go to **ADMIN > Services**, select a service, and > **View > Config.**
   The Config view for the selected service is displayed.
2. Click the **Parsers** tab.
3. Click Upload.
   The Upload Parsers dialog is displayed.

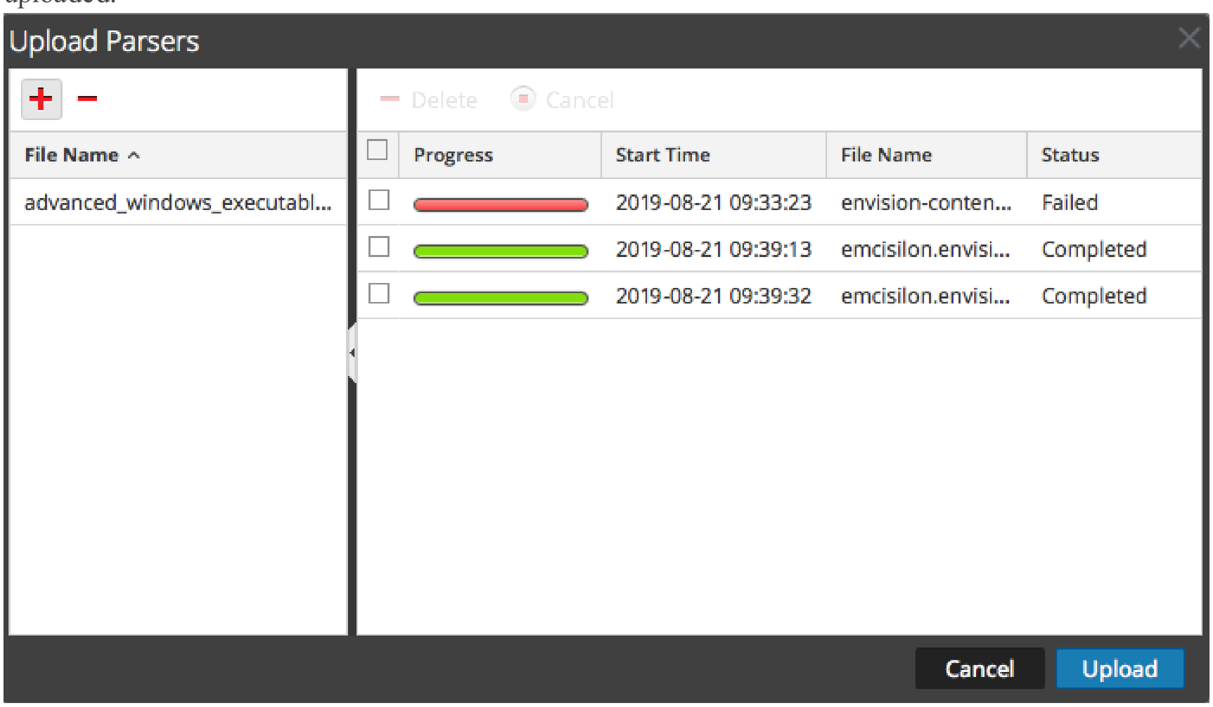


4. Click .
   A file selection dialog is displayed.

5. Select the **.flex**, **.parser**, and **.lua** files to be updated, and click **Open**.
   The dialog closes, and the selected files are displayed in the File list.



6. Click **Upload**.
   The Upload Job grid shows the progress of the upload jobs with each job representing a file being uploaded.



7. Use any of the Upload grid tools to manage the upload of selected jobs: pause and resume, cancel, and delete.

Once a job is complete, it is deployed on the Decoder and listed with the deployed parsers in Parsers tab.

Manage Upload Jobs

You can use any of the Upload grid tools to manage the upload of selected jobs: pause, resume, cancel, and delete.

- To cancel uploading a set of parsers while the upload is in queue or progress, click Cancel.
- To pause uploading a set of parsers, if the upload is not yet complete, click Pause.
- To resume uploading a set of parsers after a pause, click Resume.
- To delete an upload job, click .

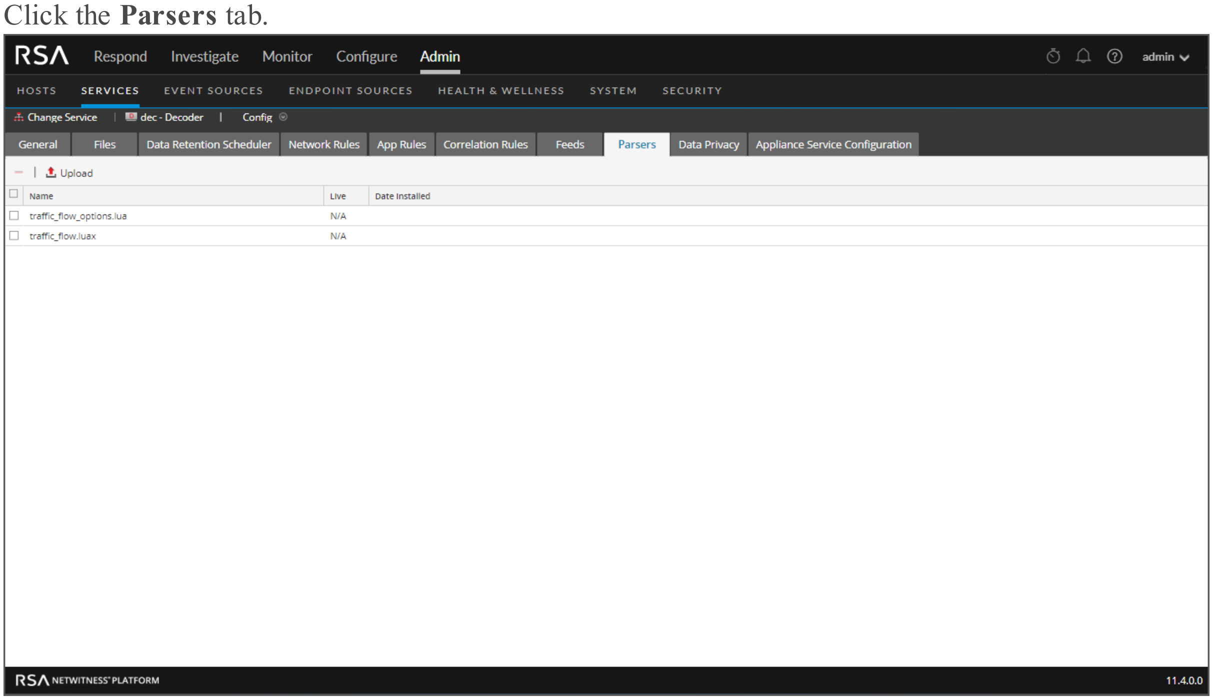
Delete Deployed Parsers

The Delete option in the Service Config view > Parsers tab provides a way to delete deployed parsers from a Decoder or Log Decoder. Parsers can be added and removed while a Decoder is running without affecting capture.

Note: Unless otherwise stated, any reference to Decoders applies to Log Decoders as well.

To delete a parser from a Decoder:

1. Go to **ADMIN** > **Services**, select a Decoder, and > **View** > **Config.**
   The Services Config view for the selected service is displayed.
2. Click the **Parsers** tab.



3. In the **Parsers** tab, select one or more parsers to delete.

4. Click **-**.
   A dialog requests confirmation that you want to delete the parsers.
5. If you want to delete the parsers, click **Yes**.
   The parsers are removed from the Decoder immediately.

Enable and Configure the Entropy Parser

Beginning with NetWitness Platform 11.0, the administrator can configure a Decoder to use a NetWitness native parser, known as the Entropy parser. When the Entropy parser is enabled, analysts have visibility into channels that are trying to blend in with other traffic, but do not follow normal protocol behavior. This helps to identify channels that do not conform to the normal environment traffic baseline, and may be worthy of investigation.

The parser creates meta keys, based on statistics collected by the native NetWitness Platform parser, that help to identify behavior of any channel that is getting lots of network traffic. When the parser is first enabled, the analyst needs to become familiar with overall behavior for the different channels seen in a captured session to understand the frequency of bytes and the normal client and server payload. Once the normal behavior is known, analysts can use the meta keys to find behavior that does not match the expected.

By default, the Entropy parser generates 10 additional meta keys that do not add significantly to the load on a Decoder, and are useful for this specialized case. The parser is disabled by default.

Enable indexing if you have interest in exploring interesting sessions based on payload byte analysis of the packets. By default, to make indexing easier, the normal `Float32` value for `entropy.req` and `entropy.res` is multiplied by 10k and stored in a `UInt16` (thus giving four digits of precision, 0 to 10,000).

However, if you define the `entropy.*` fields in the Decoder language to be `Float32`, the Decoder will store it as a float with a range of 0.0 to 1.0. Take care to change the language everywhere if you decide to keep it as a `Float32`.

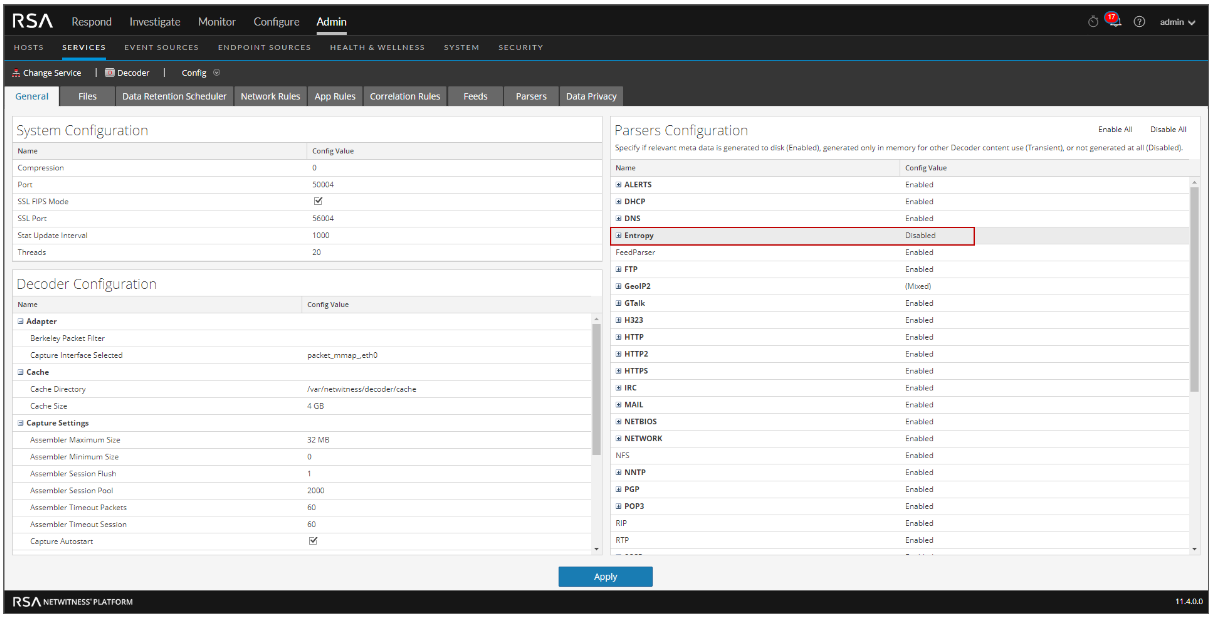
RSA does not recommend indexing as a `Float32` because of the high unique counts due to minute changes in precision.

These are the new meta keys generated by the Entropy parser by default:

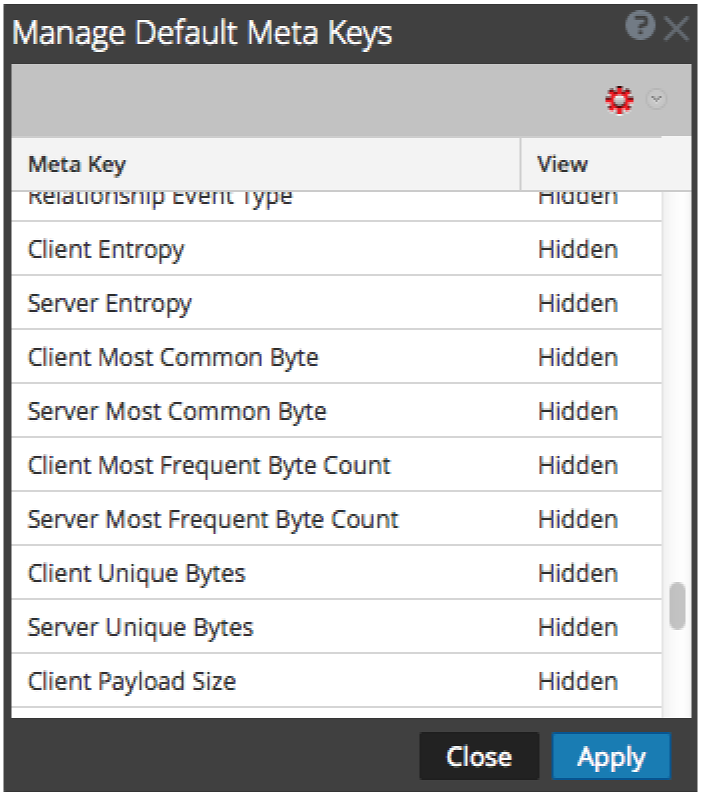
- `entropy.req` and `entropy.res`: These meta keys capture entropy using the Shannon entropy equation, which has a floating point value as a result. The floating point value of 0 to 1.000 is multiplied by 10000 and written in NetWitness Platform as `UInt 16`, an unsigned integer of 0 through 10000. .
- `mcb.req` and `mcb.res`: The most common byte is simply which byte for each side (0 thru 255) was seen the most.
- `mcbc.req` and `mcbc.res`: The most common byte count is the number of times the most common byte (above) was seen in the session streams.
- `ubc.req` and `ubc.res`: - Unique byte count is the number of unique bytes seen in each stream. 256 would mean all byte values of 0 thru 255 were seen at least once.

To enable and configure the Entropy parser on a Decoder:

1. Log in to RSA NetWitness, and select **ADMIN** > **Services** in the NetWitness Platform menu.
2. In the Services view, select the Decoder that you want to configure, and then  **View** > **Config**.
   The Services Config view for the selected Decoder is displayed.
3. The Entropy parser is disabled by default. Click the drop-down list for **Entropy** in the **Config Value** column, and select **Enabled**. If you want to disable some of the meta keys, click the drop-down list and select **Disabled** next to the meta key.



4. Click **Apply**.
   The Entropy parser is enabled and begins creating the new meta keys as configured in the Concentrator custom index file.
5. In the Service Config view select the Concentrator that is aggregating traffic from this Decoder. Select **View** > **Files** and open the Custom Index file for the Concentrator. Look for the Entropy parser meta keys to see if they are included and uncommented.
   By default the keys are commented out and therefore not enabled. To enable that part of the language the administrator needs to copy that part of index file into the `index-concentrator-custom.xml` and uncomment the `key description` line for each meta key. An example of the custom index file with the Entropy parser keys and instructions is shown below in Entropy Parser Configuration in the Concentrator Custom Index File.
6. With the Entropy meta keys enabled, they are available to analysts in Investigate, but hidden by default. To make the meta keys visible in the Investigate Values view, edit the default meta keys in the Default Meta Keys dialog so that they are open instead of hidden. You can manage these meta key the same way you manage other meta keys.



Entropy Parser Configuration in the Concentrator Custom Index File

The following is an excerpt of the Concentrator Index file lines that the administrator must copy to the custom index file. The comments provide guidance on configuring the parser.

```
<!-- This section is commented out because it's only used by the Entropy
parser which is disabled by default. To enable this part of the language, copy
to index-concentrator-custom.xml and uncomment the keys. HOWEVER, take note
that depending on how the Entropy parser is configured, the entropy.req and
entropy.res format might be a Float32 instead of a UInt16. So make sure to
change to the correct type if necessary.-->

<!-- Entropy parser meta - enable indexing if you have interest in exploring
this for interesting sessions based on payload byte analysis of the packets.
By default, to make indexing easier, the normal Float32 value for entropy.req
and entropy.res is multiplied by 10k and stored in a UInt16 (thus giving 4
digits of precision, 0 to 10,000). However, if you define the entropy.* fields
in the Decoder language to be Float32, it will store it as a float with a
range of 0.0 to 1.0. Take care to change the language everywhere if you decide
to keep it as a Float32. We do not recommend indexing as a Float32 because of
the high unique counts due to minute changes in precision. -->

<!--

<key description="Entropy Request (Client)" format="UInt16" level="IndexNone"
name="entropy.req" valueMax="10001"/>

<key description="Entropy Response (Server)" format="UInt16" level="IndexNone"
name="entropy.res" valueMax="10001"/>

-->
```

```
<!-- The most common byte is simply which byte for each side (0 thru 255) was
seen the most -->

<!--

<key description="Most Common Byte Request" format="UInt8" level="IndexNone"
name="mcb.req"/>

<key description="Most Common Byte Response" format="UInt8" level="IndexNone"
name="mcb.res"/>

-->

<!-- The most common byte count is the number of times the most common byte
(above) was seen in the session streams -->

<!--

<key description="Most Common Byte Count Request" format="UInt32"
level="IndexNone" name="mcbc.req" valueMax="500000"/>

<key description="Most Common Byte Count Response" format="UInt32"
level="IndexNone" name="mcbc.res" valueMax="500000"/>

-->

<!-- Unique byte count is the number of unique bytes seen in each stream. 256
would mean all byte values of 0 thru 255 were seen at least once -->

<!--

<key description="Unique Byte Count Request" format="UInt16" level="IndexNone"
name="ubc.req"/>

<key description="Unique Byte Count Response" format="UInt16"
level="IndexNone" name="ubc.res"/>

-->

<!-- The payload size metrics are the payload sizes of each session side at
the time of parsing. However, in order to keep indexing from having high
unique counts (bad for performance), the two payload size metas below are
indexed in buckets. -->

<!--

<key description="Payload Size Request" format="UInt32" level="IndexNone"
bucket="true" name="payload.req" valueMax="500000"/>

<key description="Payload Size Response" format="UInt32" level="IndexNone"
bucket="true" name="payload.res" valueMax="500000"/>

-->
```

Flex Parsers

One of the files available for editing in the Services Config view > Files tab is `NwFlex.xml`, the Flex parser.

NwFlex.xml

There are two kinds of Flex parsers:

- **Service identification based solely on port.** These are parsers that use only the source or destination ports to identify the session application type (service). These are the most basic and easiest to define.
- **Service identification based on a found token(s).** These parsers use tokens to identify the service type. This is also an easy way to expand which service types are identified. These are important when identifying non-internet standard applications. These parsers require that the protocol has a definable token that can uniquely identify the service type.

Five common parser operations are:

- Match Port and Identify Immediately
- Match Port and Delay Identification
- Match Token and Identify Immediately
- Match Multiple Tokens
- Match Token and Create Metadata

Detailed language information and samples are provided in this topic. This topic describes the XML schema used to define a FlexParse file. The SML node, attribute, and values referenced in descriptive text are **bold**. The root node of every file must be the **parsers** node. Under that node there can be any number of parser nodes. Each parser node defines a single parser. A parser node can have an optional **declaration** node and any number of **match** nodes.

Topics

- Arithmetic Functions
- Common Parser Operations
- General Functions
- Logging Functions
- Nodes
- Payload Functions
- Regex
- String Functions

Arithmetic Functions

This topic defines language for the flex parser arithmetic functions.

This topic defines language for the flex parser arithmetic functions. All numbers are 64-bit unsigned values and subject to both underflow and overflow, depending on the operation.

Language Definition

The following table provides language definitions.

Node Name	Attribute Name	Description
`and`		Performs bitwise AND between two numbers.
	`name`	Variable to AND result into.
	`value`	Number to AND into result.
`or`		Performs bitwise OR between two numbers.
	`name`	Variable to OR result into.
	`value`	Number to OR into result.
`increment`		Performs ADDITION of two numbers.
	`name`	Variable containing the initial value AND to receive ADDITION results.
	`value`	Number to ADD to initial value.
`decrement`		Performs SUBTRACTION of two numbers.
	`name`	Variable containing initial value AND to receive SUBTRACTION results.
	`value`	Number to SUBTRACT from initial value.
`divide`		Performs DIVISION of two numbers.
	`name`	Variable containing the initial value AND to receive DIVISION results.
	`value`	Number by which to divide the initial value. Division by zero generates an error and stops any further processing of the current session by this parser.
`modulo`		Performs MODULO of two numbers.
	`name`	Variable containing the initial value AND to receive MODULO results.
	`value`	Number by which to divide the initial value. Division by zero generates an error and stops any further processing of the current session by this parser.
`multiply`		Performs MULTIPLICATION of two numbers.

Node Name	Attribute Name	Description
	name	Variable containing the initial value AND to receive MULTIPLICATION results.
	value	Number by which to MULTIPLY the initial value.
shiftleft		Performs a binary shift left.
	name	Variable containing the initial value AND to receive shift results.
	value	Number of bits to shift by.
shiftright		Performs a binary shift right.
	name	Variable containing the inital value AND to receive shift results.
	value	Number of bits to shift by.

Common Parser Operations

This topic provides some examples of common parser operations.

This topic includes five common parser operations.

Match Port and Identify Immediately

```
<?xml version="1.0" encoding="utf-8"?>
<parsers
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="parsers.xsd">
          <parser name="CustApp" desc="Acme Custom App" service="45324">
               <declaration>
                    <port name="port" value="45324" />
               <declaration>
               </match name="port">
                    <identify />
               </match>
          </parser>
</parsers>
```

Match Port and Delay Identification

```
<?xml version="1.0" encoding="utf-8"?>
<parsers
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="parsers.xsd">
     <parser name="MSRPC" desc="Microsoft RPC protocol" service=135">
          <declaration>
               <port name="port" value="135" />
               <number name="state" scope="session" />
               <session name="end" value="end" />
          </declaration>
          <match name="port">
               <assign name="state" value="1" />
          </match>
          <match name="end">
               <if name="state" equal="1" />
                    <identify />
               </if>
          </match>
     </parser>
```

```
</parsers>
```

Match Token and Identify Immediately

```
<?xml version="1.0" encoding="utf-8?>
<parsers
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="parsers.xsd">
     <parser name="RDP" desc="Remote Desktop Protocol" service="3389">
          <declaration>
               <token name="signature" value="Cookie: mstshash=" />
          </declaration>
          <match name="signature">
               <identify />
          </match>
     </parser>
</parsers>
```

Match Multiple Tokens

```
<?xml version="1.0" encoding="utf-8"?>
<parsers
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="parsers.xsd">
     <parser name="MyServiceMultiToken" desc="Multiple Tokens" service="333">
          <declaration>
               <number name="state" scope="stream" />
               <token name="user" value="USER " />
               <token name="pass" value="PASS " />
               <session name="session" value="end" />
          </declaration>
          <match name="user">
               <or name="state" value="1" />
          </match>
          <match name="pass">
               <or name="state" value="2" />
          </match>
          <match name="session">
               <if name="state" equal="3">
                    <identify />
               </if>
          </match>
```

```
    </parser>
</parsers>
```

Match Token and Create Metadata

```
<?xml version="1.0" encoding="utf-8"?>
<parsers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="parsers.xsd">
    <parser name="SHELL" desc="Command Shell Identification">
        <declaration>
            <token name="cmd.exe" value=" (C) Copyright 1985-2001 Microsoft
            Corp" options="linestart" />
            <meta name="client" key="client" format="Text" />
        </declaration>
        <match name="cmd.exe"
            <register  name="client" value="MS Command Shell" />
        </match>
    </parser>
</parsers>
```

General Functions

This topic defines language for the flex parser general functions.

General Functions Language Definition

Node Name	Attribute Name	Description
apptype		Gets the currently defined service type for the current session.
	name	A number variable to receive the current service type.
identify		Marks the session with the parser's service type if the service type has not already been identified.
assign		Assigns a value to a variable.
	name	The unique identifier assigned to the item in the declaration section.
	value	Optional. If specified, the action defined in the match is only applied when the declaration matches the given value.
getmeta		Retrieves the value of meta that generated a callback. This function will return empty results (0, zero length string) if called when there was no meta callback.
	name	The variable to receive the value of the meta key that generated the callback.
gettoken		Returns the current matched token.
	name	A string variable to receive the current matched token. If there is no current token, the variable is assigned an empty string.
end		This terminates the execution of the current **match** section.
if		Compares two values. If the comparison is true, executes any sub-actions. Comparisons can be **number** or **string** types, as long as both values are the same type.
	name	The unique variable identifier assigned to the item in the **declaration** section.
	equal notequal less lessequal greater greaterequal and or	The operation value to compare. If true, any sub-actions are executed.

Node Name	Attribute Name	Description
`register`		Adds metadata to the session.
	`name`	The unique identifier of a meta variable to be created, as defined in the **declaration** section.
	`value`	The value of the metadata to be created.
`while`		Compares two values and executes any sub-actions if the comparison is true. Comparisons can be **number** or **string** types, as long as both values are the same type.
	`name`	The unique variable identifier assigned to the item in the declaration section.

	`equal`
	`notequal`
	`less`
	`lessequal`
	`greater`
	`greaterequal`
	`and`
	`or`
		Specifies the operation value to compare. If true, any sub-action is executed. The **and** and **or** attributes signify bitwise operations and can only be applied to **number** variables.

`call`		Executes the specified **match** element. This can be any match element defined in the same flex parser regardless of how it was declared.
	`value`	The name of the match element, or a string variable containing the name of a match element. • If the match element name is specified, the parser will not load if the named matched element doesn't exist. • If a string variable is specified, the **call** element will execute any child elements that it may have if the string value resolves to a match element after executing the named match element. • If no **match** element can be found matching the string value, no action is taken.

Logging Functions

This topic defines language for the flex parser logging functions.

Logging functions provide a means for a flex parser to write to the system log. Logging functions can be extremely useful when creating a new flex parser, but should be kept to an absolute minimum when a flex parser is deployed to a production system.

Language Definition

Node Name	Attribute Name	Description
failure		Logs a message to the system log with the log level **Failure.**
	value	A string to include as the log message.
warning		Logs a message to the system log with the log level **Warning.**
	value	A string to include as the log message.
info		Logs a message to the system log with the log level **Info**.
	value	A string to include as the log message.
debug		Logs a message to the system log with the log level **Debug.**
	value	A string to include as the log message.

Nodes

This topic defines language for the flex parser nodes.

Nodes Language Definition

Node Name	Attribute Name	Description
parsers		The root node in each definition file.
	xmins:xsi	Defines the namespace to use for the schema inclusion. This attribute is not required; however, language definition is not possible without it. This node must have the following value: http://www.w3.org/2001/XMLSchema-instance
	xsi:noNamespaceSchemaLocation	Defines the XSD schema validation file used to validate the language definition. This attribute is not required; however, language definition is not possible without it. This node must have the following value: parsers.xsd
parser		The node that defines a single parser definition. This node must be directly under the parsers node. There can be more than one per file.
	name	The name that uniquely identifies the parser. This name should be short and succinct. This is used by the system to allow enabling and disabling. It should contain only the letters [a-z] and [A-Z].
	desc	Provides a friendly description of what the parser does.
	service	The unique number assigned to the session when identified.
declaration		Delineates the definition. Each of these definitions can have an associated match entry.

Node Name	Attribute Name	Description
token		Specifies a definition for identifying a token somewhere in the session protocol. This defines a `match` callback when the specified tokens are encountered in a session payload. The `read` position is set to the byte immediately following the matched token.
	name	This is a unique identifier for the declaration.
	value	This is the exact token value to be identified.
	options	Options specify that the token should start on a new line or at an end of a line (`linestart` or `linestop`).
meta-callback		Registers a callback for the flex parser whenever meta of a specific format is created. This can be further qualified to generate callbacks only for sessions that have been identified as a specific apptype (for example, 80 for HTTP).
	name	Name of the match element to be executed when a callback occurs. (String)
	key	Name of the meta key that generates callbacks. (String)
	format	The data type of the meta key that will generate the meta.
	apptype	The meta callback is only generated if the session being parsed has been identified with the specified apptype. (Unsigned Integer, Optional)
number		Defines a numeric variable that can be referenced elsewhere within the parser definition. All numeric values are 64-bit unsigned values.
	name	This is a unique identifier for the declaration.

Node Name	Attribute Name	Description
	`scope` (optional)	Specifies when to reset the variable. This can either be for each side of a two-sided session or only after a new session is detected. The possible values are **`global`**, **`constant`**, **`stream`**, and `session` (default).
`string`		Defines a numeric variable that can be referenced elsewhere within the parser definition.
	`name`	This is a unique identifier for the declaration.
	`scope` (optional)	Specifies when to reset the variable. This can either be for each side of a two-sided session or only after a new session is detected. The possible values are **`global`**, **`constant`**, **`stream`**, and `session` (default).
port		Defines a **match** callback when a session is encountered using the specified port. The **read** position is set to the first byte of the first stream (client) in the session.
	`name`	This is a unique identifier for the declaration.
	`value`	This is the port number to identify.
`session`		Defines a `match` callback for session begin/end events. These events only occur if a token for the parser is encountered in the session.
	`name`	This is a unique identifier for the declaration.
	`value`	Specifies that processing takes place at the beginning of a new session or at the end of a session (`begin` or `end`).
`stream`		Defines a `match` callback for stream begin/end events. These events only occur if a token for the parser is encountered in the stream.
	`name`	This is a unique identifier for the declaration

Node Name	Attribute Name	Description
	value	Specifies that processing takes place at the beginning or at the end of a stream (begin or end).
function		Defines a match section that can be used as a generic function. No callbacks are associated with this declaration.
	name	This is a unique identifier for the declaration.
meta		Defines the type of data that the parser will create.
	key	Specifies the key name. The key needs to be 1-16 bytes in size.
	format	Specifies the variant type (for example, **Text, IPv4, UInt32**). Refer to the SDK documentation for a full list.
pattern		Defines a regular expression variable for use by the regex function
	name	This is a unique identifier for the declaration.
	scope (optional)	Specifies when to reset the variable. This can be for each side of a two-sided session or only after a new session is detected. Possible values are **global, constant, stream,** and session (default).
	value (optional)	Specifies a regular expression to assign to the pattern variable. This attribute is only valid when the **scope attribute** is set to constant.
match		The possible entries for taking an action once a match criterion has been found for a declaration. These nodes can be nested to provide deeper logic. There are several categories of execution elements (functions) that can appear as children of a match element: • General • Arithmetic • String • Payload

Payload Functions

This topic defines language for the flex parser payload functions.

These functions operate on a `read` position, set at the beginning of a `match` element.

Language Definition

Node Name	Attribute Name	Description
`find`		Searches the stream payload starting at the read position for a provided string value. If the value is found, the offset from the read position is returned. Any child elements will then execute. If not found, any child elements will not execute.
	`name`	A `number` variable to receive the offset from the `read` position where the match begins.
	`value`	A string to find.
	`length` (optional)	A limit to the length of the payload to be searched. If a limit is not provided, the remainder of the payload is searched. It is recommended to always use the smallest value possible here in order to reduce the effect on performance.
`install-decoder`		To enable tokens to match on payload data that may be fragmented or otherwise encoded. A scan decoder can be installed to preprocess a section of the payload before it is scanned for tokens. An example would be an HTTP response that uses the chunked transfer encoding with gzip content encoding. By parsing the HTTP header, the necessary type, offset, and length parameters can all be set, after which the HTTP response payload would appear to the token scanning as if neither encoding had been applied. However, this incurs significant overhead.
	`type`	The type of decoder to install. Valid options are: gzip, deflate, chunked, chunked-gzip, chunked-deflate.
	`offset`	Offset from the current read position to begin decoding.
	`length`	The maximum payload length to decode.
`isdecoding`		Tests whether an installed decoder is currently active. If so, any children of this function will execute. This function has no parameters.
`move`		Moves the `read` position forward in the current stream by a specified number of bytes. If there is sufficient data in the stream, the `read` position is updated and any child elements will then execute. If not found, the `read` position remains unchanged and any child elements will not execute.

Node Name	Attribute Name	Description
	value	The number of bytes to move the read position.
	direction (optional)	The direction to move the current read position. Can be forward (default) or **reverse.**
packetid		Returns the id of the packet for the current read position. It is possible for the result to be 0, which indicates that the packet id could not be determined.
	name	A number variable to receive the current packet id.
payload-position		Returns the current read position. This is a zero based index into the stream payload.
	name	A number variable to receive the current read position.
read		Reads a specified number of bytes starting at the read position into a variable. If there is sufficient data in the stream, the read position is updated, the data read assigned, and any child elements will then execute. If not found, the read position remains unchanged and any child elements will not execute.
	name	The name of a string or number variable to receive stream data. If a number variable is provided, the bytes read are interpreted as a single unsigned numeric value.
	length	The number of bytes to read from a stream.
	endianess (optional)	The byte ordering to use when reading into a number variable. Can be big (default) or little. The attribute is invalid when reading into a string variable.

Regex

This topic defines language for the flex parser regex node.

Regex searches the stream payload starting at the `read` position for matches to a provided regular expression. If matches are found, the offset from the `read` position and, optionally the matched string, is returned. Any child elements execute. If no matches are found, child elements do not execute.

Language Definition

Attribute Name	Description
`name`	A `number` variable to receive the offset from the `read` position where the match begins.
`value`	A regular expression to find.
`length` (optional)	A limit to the length of the payload to be searched. If a limit is not provided, the remainder of the payload is searched. It is recommended to always use the smallest value possible here in order to reduce the effect on performance.
`found` (optional)	The name of a `string` variable to receive a matched string.

String Functions

This topic provides language definitions for the flex parser string functions.

String Functions Language Definition

Node Name	Attribute Name	Description
append		Attaches a number or string to the end of a `string` variable.
	name	The unique identifier of a string variable to which the specified value is to be attached.
	value	A number or string to attach.
find		Searches a string for a provided string value. If it is found, the position is returned and any child elements will execute. Otherwise, child elements will not execute.
	name	A `number` variable to receive the zero-based position, where the provided value string was found in the **in** string.
	value	A string to find.
	in	A string to search.
	length (optional)	A limit to the length of the **in** string to be searched. If a limit is not provided, all of **in** will be searched.
length		Assigns the length of a string to a `number` variable.
	name	A `number` variable to receive the length of the specified string.
	value	A string value whose length is to be determined.
regex		Searches a string for matches to the provided regular expression. If a match is found, the position and, optionally, the matching string is returned. Any child elements will then execute. If not found, any child elements will not execute. Regular expression operations can adversely affect system performance.
	name	A number variable to receive the zero-based position, where the provided regular expression matched in the in string.
	value	A regular expression to be searched for.
	in	A string to search.

Node Name	Attribute Name	Description
	length (optional)	A limit to the length of the in string to be searched. If a limit is not provided, all of **in** will be searched.
	found (optional)	The name of a string variable to receive the matched string.
substring		At least one of the optional attributes from and length must be specified.
	name	The unique identifier of a string variable to receive the extracted value.
	value	A string value from which to extract a substring.
	from (optional)	The zero-based position from which to begin the substring. If not specified, it defaults to zero.
	length (optional)	The number of characters to extract. If not specified, it defaults to the remaining length of the string.
tolower		Converts a string to all lowercase letters.
	name	The name of a string variable to process.
toupper		Converts a string to all uppercase letters.
	name	The name of a string variable to process.
urldecode		Decodes a string containing url-encoded characters.
	name	A string variable to receive the decoded string.
	value	A url-encoded string to decode.
base64decode		Decodes a base-64 encoded string.
	name	A string variable to receive the decoded string.
	value	A url-encoded string to decode.
uudecode		Decode a uuencoded string.
	name	A string variable to receive the decoded string.
	value	A uuencoded string. The header and trailing lines should not be included.
quotedprintabledecode		Decode a Quoted-printable encoded string.
	name	A string variable to receive the decoded string.
	value	A quoted-printable encoded string.
convert-ebcdic		Convert an EBCDIC string to its ASCII equivalent.

Node Name	Attribute Name	Description
	`name`	A string variable to receive the decoded string.
	`value`	A url-encoded string to decode.

GeoIP2 Parsers

This topic describes the GeoIP2 parser for Decoders. This parser converts IP addresses into geographic locations, such as the country name and city where the IP address is typically found.

> **Note:** In version 11.3 and later, the native GeoIP2 parser replaces the GeoIP parser (which has been permanently removed). The GeoIP2 parser provides the same basic functionality as the GeoIP parser as well as many enhancements. For example, it converts IP addresses into geographic locations, provides the latest Maxmind GeoIP package, and supports IPv6 addresses as well as IPv4.

Available in NetWitness Platform version 11.2 or later, the GeoIP2 Parser is enabled by default for upgrades and new installations. The GeoIP2 parser provides the latest Maxmind GeoIP package and supports IPv6 addresses as well as IPv4.

To edit the GeoIP2 parser configuration:

1. Go to **ADMIN** > **Services**.
2. In the **Administration services** view, select a Log Decoder or a Decoder.
3. Click the settings icon (⚙) and select **View** > **Config**. In the Parsers Configuration panel, select **GeoIP2** to view and update configuration options.
4. Define the IP addresses to lookup. The GeoIP2 parser enables the following IP addresses by default: `ip.src`, `ip.dst`, `ipv6.src`, and `ipv6.dst`. You can update options by using `parsers.options` to remove or add new IP addresses. For example, you can edit `parsers.options` and pass a comma-separated list of IP addresses to use as follows:
   `GeoIP2="ipaddr=ip.src,ip.dst,ipv6.src,ipv6.dst,alias.ip"`
   This adds a new IP address to lookup called `ip.addr`. However, since `alias.ip` does not end in `.src` or `.dst`, the parser will elect to place the GeoIP2 metadata generated in meta keys without a `.src` or `.dst` suffix. So, you would see country, city, and so on, after the `alias.ip` metadata.

> **Note:** The list you pass for `alias.ip` replaces the default list. So, if you pass `alias.ip=ip.src`, it generates only GeoIP2 metadata for `ip.src`, and generates no metadata for other IP addresses.

> **Note:** `parsers.options` is used for passing options to multiple parsers. So if you add GeoIP2 to it, you should not delete any other options being passed to other parsers (like Entropy).

The following table provides the full list of metadata that the GeoIP2 parser can potentially generate and indicates which metadata is or is not enabled by default:

Enabled by Default	Not Enabled
`country`, `country.src`, `country.dst`	`latdec`, `latdec.src`, `latdec.dst`
	`longdec`, `longdec.src`, `longdec.dst`
`domain`, `domain.src`, `domain.dst`	`isp`, `isp.src`, `isp.dst`
`org`, `org.src`, `org.dst`	`city`, `city.src`, `city.dst`

You can enable the other metadata using the standard parser configurations.

Note: By disabling some metadata by default, the GeoIP2 parser does not work the same as the GeoIP parser (which did not, by default, disable any metadata it generated). If you need any of the disabled metadata, you need to enable them (once only) for each Decoder, after upgrading to 11.2 or later. Keep in mind that the `isp` and `org` meta keys usually produce an equivalent value to `domain`.

Lua Parsers

One of the files available for editing in the Services Config view > Files tab is **`NwLua.xml`**, the Lua parser.

There are a number of Lua parsers available from Live. See RSA Content for:

- A complete list of these parsers
- Their interdependencies
- The Flex parsers that are subsumed by each Lua parser

Five common parser operations are:

- Match Port and Identify Immediately
- Match Port and Delay Identification
- Match Token and Identify Immediately
- Match Multiple Tokens
- Match Token and Create Metadata

HTTP Parsers

The HTTP parser is a native parser that is used for Decoders to parse both requests and responses in HTTP messages. In version 11.4 and later, the HTTP parser provides a decompression option.

The decompression option mimics the decompression option in the Lua HTTP parser and is controlled by the `'decompression'` option for the HTTP parser. Parser options are set in the `/decoder/parsers/config/parsers.options` configuration node. To set an option on the HTTP parser, you append an `HTTP=""` clause to the `parsers.options` field so that HTTP can understand the `'decompression'` option.

For example, you could add `HTTP="decompression=true"` to the parser option list to enable decompression of all HTTP compressed bodies.

You can use the following values in the `'decompression'` field.

Value	Description	Example Entry in parsers.options
`true`	Decompress all bodies	`HTTP="decompress=true"`
`false`	Do not decompress. This is the default.	`HTTP="decompress=false"`
`1`	Decompress `application/*` content	`HTTP="decompress=1"`
`2`	Decompress `audio/*` content	`HTTP="decompress=2"`
`4`	Decompress `font/*` content	`HTTP="decompress=4"`
`8`	Decompress `image/*` content	`HTTP="decompress=8"`
`16`	Decompress `message/*` content	`HTTP="decompress=16"`
`32`	Decompress `model/*` content	`HTTP="decompress=32"`
`64`	Decompress `text/*` content	`HTTP="decompress=64"`
`128`	Decompress `video/*` content	`HTTP="decompress=128"`

The numeric values can be combined by addition to search for multiple types of content. For example, if you want to decompress application and text content, use `1 + 64 = 65`, which becomes `HTTP="decompress=65"`.

To set the decompression option:

1. Go to **ADMIN** > **Services** and select a Decoder, and in the actions menu (), select **View** > **Explore**.
2. Expland **decoder** > **parser**s and select **config**.
3. In **parsers.options**, append `HTTP="decompress=<`*`option from table`*`>"`.

Visibility into HTTP/2 Sessions

You can search for metadata items derived from headers in the HTTP/2 stream to gain visibility into HTTP/2 sessions.

To turn on header parsing for HTTP/2 sessions:

1. Go to **ADMIN** > **Services** and select a Decoder, and in the actions menu (), select **View** > **Explore**.
2. Expland **decoder** > **parser**s and select **config**.
3. In **parsers.options**, append `HTTP2="headers=true"`.

Snort Parsers

Snort rules and configuration are added to the `parsers/snort` directory for NetWitness Investigate and the Decoder. The Decoder supports the payload detection capabilities of Snort rules. The rules files must have the extension `.rules` and the configuration files must have the extension `.conf`. The Decoder implementation of Snort rules is centered on using the content strings defined in a Snort rule as a token. Once a token is matched, the rule header and additional rule options can be evaluated. Currently, rules that do not define any content (using `content` or `uricontent` rule options) are not supported.

Configuration

The configuration files are loaded prior to loading rules.

Configuration Options	Description
Variable Definitions	**Description**
`ipvar`	The full language for defining IP address variables is supported, including lists, CIDR, and negation.
`portvar`	The full language for defining IP address variables is supported, including lists, ranges, and negation.
`var`	Not supported; use `ipvar` or `portvar`.
Action Definitions	**Description**
`ruletype`	The definition of additional `ruletypes` is supported. However, only rules that have a base rule type of `alert` are supported.
General Configuration	**Description**
`nopcre`	This configuration option disables all rules with `pcre`'s.

Meta Key Usage

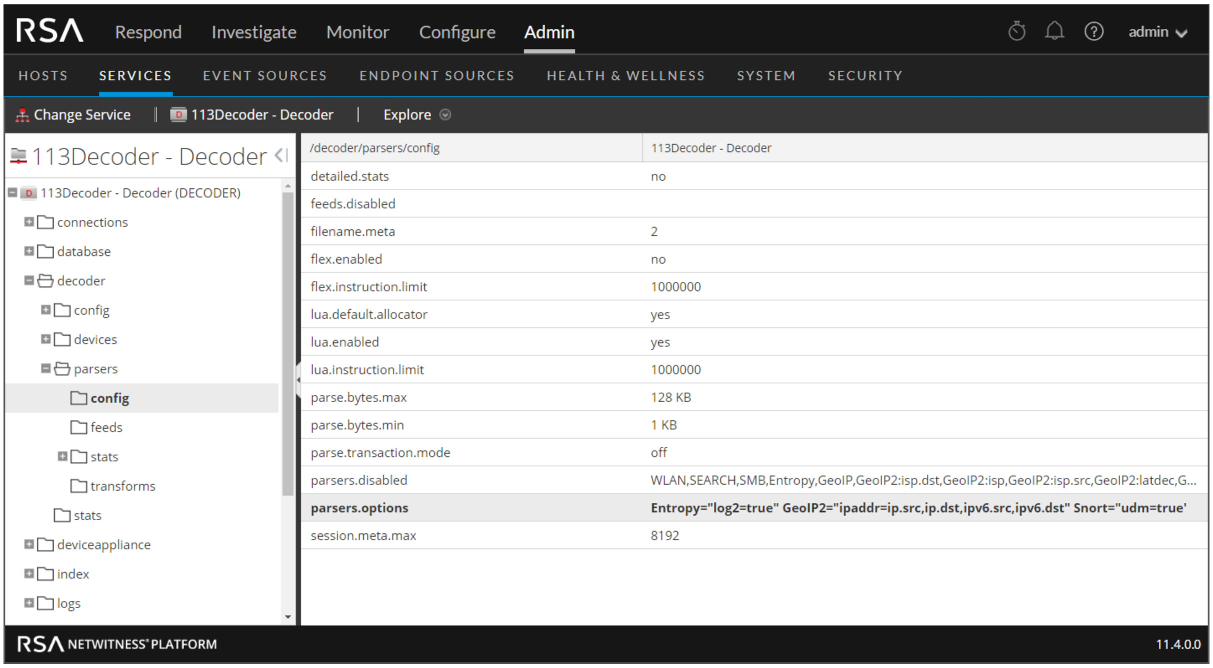
In version 11.3 or later, Snort parser meta key usage has been updated with a new option for the Snort parser. The new option, `Snort="udm=true"`, uses the aligned Unified Data Model (UDM) key set. For information about UDM, see https://community.rsa.com/community/products/netwitness/rsa-content/udm.

By default the legacy key set, which contains keys that are consistent with previous releases, is used. Refer to General Options for a description of how the two key sets differ.

To use the aligned UDM key set for Snort parser meta keys:

1. In the NetWitness Platform User Interface, go to **ADMIN** > **Services**.
2. Select a Decoder and then click > **View** > **Explore**.
3. In the left panel, select **decoder** > **parsers** > **config**.

4. In the right panel, in parser.options, add Snort="udm=true".
   The following image shows an example of adding Snort="udm=true".



> **Note:** To pass options to parsers, you must first give the name of the parser and then the options to be passed in this format:
> <ParserName>="<ParserOptions>"<Whitespace><ParserName2>="<Parser2Options>"
> Each ParserName=Value option must be separated by whitespace. Normally, the Value must have double quotes around it. The Value itself can sometimes list multiple Option=Value pairs, each separated by whitespace, and if those values have whitespace, they must be in escaped double quotes. To escape a quote, place a backslash before it: \".
> This is an example of defining options for Parser1, Parser2, and Parser3:
> Parser1="Option1=\"Option1 Value With Space\" Option2=Option2ValueNoSpace"
> Parser2="Option1=Value" Parser3="op1=val1 op2=val2 op3=\"another value\""

Rules

Snort rules are parsed and loaded when PCS is loaded (any import or capture in Investigate, initial capture start and parser reload in a Decoder).

- Any rule that does not properly parse is ignored.
- Any valid Snort rule should successfully parse; however, there are rule options that are not supported by Decoders which are not fully parsed.

Section	Description
Header	The header conditions are evaluated when a rule receives the first token callback for a stream. The header is evaluated once per stream, and prevents any further consideration of a rule against a specific stream if the conditions are not met.
Actions	The specified action or a rule must be defined (either one of the native Snort actions, or defined in the configuration using the `ruletype` statement) for the rule to be considered valid. The Decoder only uses rules with alert actions.
Protocols	The Decoder supports the current Snort protocol keywords (`tcp`, `udp`, `icmp`, `ip`).
IP Addresses	The full language for defining IP addresses is supported, including lists, CIDR, and negation.
Port Numbers	The full language for defining port numbers is supported, including lists, ranges and negation.
Direction Operator	The directional operator supports the from-to (‘->’) and bidirectional (‘<>’) values. The to-from (‘<-’) value is invalid and causes the rule to fail to load.

General Options

General options for Snort rules can result in different meta keys being written, depending on whether the Snort parser is in Aligned Key mode or Legacy Key mode.

Aligned Key Mode

Option	Description
`msg`	If the rule matches, the `msg` value is added as `sig.name` meta.
`sid`	If the rule matches, the `sig.id` value is added as meta.
`classtype`	If the rule matches, the `classtype` name is added as `threat.cat` meta.
`priority`	If the rule matches and it has a `priority` option, it is used to determine the type of the `risk.num` meta.

Legacy Key Mode

Option	Description
msg	If the rule matches, the msg value is added as risk.info, risk.warning, or risk.suspicious meta, depending on rule priority.
sid	If the rule matches, the sid value is added as meta.
classtype	If the rule matches, the classtype name is added as threat.cat meta.
priority	If the rule matches and it has a priority option, it is used to determine the type of risk meta associated with the msg value.

Payload Options

The Decoder supports the following payload rule options.

Option	Description
content	The content option creates a token for the Decoder to match. Only tokens of three or more bytes are accepted. It is also important to note that the Decoder differs from Snort in that rules are evaluated across the payload of the reconstructed stream and not just a single packet. This can result in differences in rules matches between Snort and a Decoder, especially when considering positional options.
nocase	Currently not supported. This option is ignored and case-sensitive matching is used.
depth	This option is applied to the distance of the token from the current offset position, or the beginning of the stream if no offset is set. If the token position is greater than this value, it is not a match.
offset	This option is applied to the distance of the token from the beginning of the stream. If the token position is less than this value, it is not a match.
distance	This option is applied to the distance of the token from the end of the previous token match. If the relative token position is less than this value, it is not a match.
within	This option is applied to the distance of the token from the end of the previous token match. If the relative token position is greater than this value, it is not a match.
http_uri	Any token that matches is verified to fall within an http_uri as indicated by the HTTP parser. No URI normalization is applied.
uricontent	There is no URI normalization applied. Otherwise, this is equivalent to the content option with the http_uri modifier.

Option	Description
pcre	Currently, Perl Compatible Regular Expressions (PCREs) are only applied to URIs and must specify the U option.

Non-payload Options

Option	Description
flow	Verifies that the rule is only applied to the client or server stream.
to_client	Limits the rule to only matching on a stream that a Decoder has defined as Server.
from_server	Synonym for to_client.
from_client	Limits the rule to only matching on a stream that a Decoder has defined as Client.
flowbits	Maintains state per session and is reset at the end of each session.
set	When the rule matches, the specified flowbit is set.
unset	When the rule matches, the specified flowbit is cleared.
toggle	When the rule matches, the specified flowbit is flipped.
isset	When the rule is evaluated, the specified flowbit state must be set for the rule to match.
isnotset	When the rule is evaluated, the specified flowbit state must not be set for the rule to match.
noalert	Prevents the rule from generating metadata if it matches.

Search Parser

The Search Parser is a custom parser used to generate metadata by scanning for predefined keywords and regular expressions. The parser searches the payload of a reconstructed session for string matches and can execute a regular expression search. You can configure the parser by editing the **search.ini** file.

> **Caution:** The search parser can have a significant impact on system performance. It is important that both the search mechanism and the data to which it is applied to be well understood before creating new search definitions and enabling the search parser.

The search definition is used across all protocols. There are three basic search methods:

- Keyword: Search a stream for a specific set of words
- Pattern: Search a stream for a regular expression match
- Keyword + Pattern: Search a stream for a regular expression if it contains any of a given set of keywords.

Search Methods

The Search parser uses three basic search methods:

- Keyword: Search a stream for a specific set of words.
- Pattern: Search a stream for a regular expression match.
- Keyword+Pattern: Search a stream for a regular expression if it contains any of a given set of key words.

Syntax

```
Maxrecon=<max_size>Maxsearch=<max_ssearch_length>MatchLimit=<max_matches_per_
stream
Search Name
Services=<service_id_list>Keywords=<keyword_list>|Pattern=<expression>Case=0|1
Proximity=<number_of_bytes>Recon=0|1
Raw=0|1
```

Parameters

Parameters used in this command:

Parameter	Description
`autocheck`	Automatically fixes all problems without prompting
`headerOnly`	Check/display the header of each file

Parameter	Description
`chatty`	Displays a hex dump of every object in the file (huge amount of data)
`dump#-#`	Indicates a zero-based object or range of objects in the file to output in hex to the console

Example

Following is an example of the command:

To check all NetWitness database files located in the Collection named Default. If any problems are found, the command will describe the problem and ask if you would like to fix it.

```
dbcheck C:\Documents and Settings\User\My Documents\NetWitness\
Investigations\Default\*.nw*
```

Wireless LAN Configuration

One of the files available for editing in the Services Config view > Files tab is **`wlan-config.xml`**, the wireless LAN configuration file.

It controls the 802.11 parsers. Its chief purpose is to control decryption of raw 802.11 frames captured by the Decoder. This file is optional. If decryption of 802.11 traffic is not desired, there is no need to create the file.

There are five link-level parsers related to wireless LAN packet capture:

- IEEE 802.11 parser (data frames and beacons only)
- Radiotap w/ 802.11 header
- Absolute Value Systems (AVS) w/ 802.11 header
- Prism II w/ 802.11 header
- CACE′s ″Per Packet Information″ (PPI) w/ 802.11 header

The 802.11 wireless parsers introduced in 9.8 all share a single configuration file. This wlan-config.xml file is used to define any wireless access points the user may have in the network, and its primary purpose is to control decryption. The BSSID of the access point and the SSID that it′s authoritative for is added to this file as well as all of the active default keys used by the access point.

Troubleshooting Parsers

This topic provides guidance for troubleshooting issues related to parsers.

Lua Parser Errors

Lua parsers occasionally generate errors. If a parser enters a state where it generates multiple errors, these errors can hinder performance. Beginning with 11.4, a new option is available for Lua parsers that instructs the Decoder to automatically disable the parser after a configurable number of errors.

The value is set on the `/decoder/parsers/config/parsers.options` configuration node as shown in the following example:

```
Entropy="log2=true" GeoIP2="ipaddr=ip.src,ip.dst,ipv6.src,ipv6.dst" error_
parser="errorMax=5" addy_parser="errorMax=10"
```

This configuration node enables you to set options for different parsers. In this example, the Lua parser `error_parser` is configured with a maximum error count of 5, and the `addy_parser` to 10. The `errorMax` setting has a valid range of values from 0 (meaning the feature is disabled) to 65,535, and takes effect when parsers are loaded or reloaded.

To disable a Lua parser after a defined number of errors:

1. Go to **ADMIN** > **Services**, select a Decoder and then select > **View** > **Explore**.
2. In the left panel, expand **decoder** > **parsers**, and select **config**.
3. In **parsers.options**, add the following command, where *<any_parser>* is the Lua parser for which you want to limit errors, and *<n>* is the number of errors to which the parser is limited:
```
<any_parser>="errorMax=<n>"
```

Results of Automatically Disabling a Parser

When a parser is automatically disabled, a log message is generated (per parser thread) and states that the parser has been disabled, as shown in the following example:

```
(W) 2019-Apr-25 16:25:33 [Parse] Lua parser 'error_parser' has been disabled
because it exceeded the configured error threshold (5)
```

If detailed statistics for parsers are enabled, the text of the last error is populated in the parser's detailed statistics under `/decoder/parsers/definitions/<parser-name>`. Also, a new attribute has been added to the XML returned by the `/decoder/parsers?msg=schema` call. When the parser is in an error state, the attribute `error` is set to `1` and the `enabled` attribute is set to `0`. When the parser is reset, the values are reset to `0` and `1` respectively.

Resetting the Parser

If a parser is disabled because it has exceeded the error threshold, it can be reset by reloading the parser. This causes the statistics to reset so that if the faulty parser is still in place, it will function again until the error threshold is met.

Configure Feeds

NetWitness Platform uses feeds to create metadata based on externally defined metadata values. A feed is a list of data that is compared to sessions as they are captured or processed. For each match, additional metadata is created. This data could identify and classify malicious IPs or incorporate additional information such as department and location based on internal network assignments. Some examples of feeds include threat feeds to identify BOTNets, DHCP mappings, or even Active Directory (AD) information such as physical location or logical department.

You can use the Live module in NetWitness Platform to obtain feeds from outside sources. "Live Content in NetWitness Platform" in the *Live Services Management Guide* provides an overview of the Live content management tool.

Within the NetWitness Platform user interface, you can view the list of currently deployed feeds, along with an indicator if a feed that originated from Live was installed through NetWitness Platform or manually. Feeds can be added, removed, and updated while a Decoder is running without affecting capture.

Custom Feed Definition File Structure

The NetWitness Platform Custom Feed wizard allows creation and deployment of custom Decoder feeds based on deterministic logic that offers the meta keys specific to the selected Decoders and Log Decoders. Although the wizard guides users through the process to create both on-demand and recurring feeds, it is helpful to understand the form and content of a feed file when you create a feed.

Feed filenames in RSA NetWitness Platform are in the form `<filename>.feed`. To create a feed, NetWitness Platform requires a feed data file in `.csv` or `.xml` format and a feed definition file in `.xml` format, which describes the structure of a feed data file. The Custom Feed wizard can create the feed definition file based on a feed data file, or based on a feed data file and the corresponding feed definition file.

The files that you use to create an on-demand feed must be stored on your local file system. The files used to create a recurring feed must be stored at an accessible URL, whence NetWitness Platform can fetch the most current version of the file for each recurrence. After a NetWitness Platform feed is created, you can download the feed to your local file system, edit the feed files, and then edit the NetWitness Platform feed to use the updated feed files.

Sample Feed Definition File

This is an example of a feed definition file named `dynamic_dns.xml`, which NetWitness Platform creates based on your entries in the Custom Feed wizard. It defines the structure of the feed data file named `dynamic_dns.csv`.

Note: The feed file path should be `.csv` regardless of the Feed Type (Default or STIX).

```
<?xml version="1.0" encoding="utf-8"?>
 <FDF xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="feed-definitions.xsd">

   <FlatFileFeed name="Dynamic DNS Domain Feed"
   path="dynamic_dns.csv"
   separator=","
   comment="#"
   version="1">

   <MetaCallback
   name="alias.host"
   valuetype="Text"
   apptype="0"
   truncdomain="true"/>

   <LanguageKeys>
            <LanguageKey name="threat.source" valuetype="Text" />
            <LanguageKey name="threat.category" valuetype="Text" />
            <LanguageKey name="threat.desc" valuetype="Text" />
   </LanguageKeys>
```

```
<Fields>
<Field index="1" type="index" key="alias.host" />
<Field index="4" type="value" key="threat.desc" />
<Field index="2" type="value" key="threat.source" />
<Field index="3" type="value" key="threat.category" />
</Fields>
</FlatFileFeed>

</FDF>
```

Define Multiple Values in a Single Field

The information in this section applies to RSA NetWitness® Platform version 11.3.1 and later.

Custom feeds support multiple values in a single field. This allows feeds to generate multiple values of the same meta.

For example:

```
<Fields>

<Field index="1" type="index" key="lc.cid" />

<Field index="2" type="value" key="user" separator=";" openchar="("
closechar=")"/>

</Fields>
```

The second field is defined to accept multiple values in the corresponding CSV file. The following is an example of the defined values:

```
clc1,(bob;tom;sam) clc12,susan clc123,doris
```

This definition means that when we see the `lc.id` with a value of `cic1`, three `username` meta values are generated with the values of `bob`, `tom` and `sam`.

The separator in the field definition for a multi-value field **must** be different than the separator for the feed file itself. Escaping the separator character in field values is **not** supported.

Feed Definition Equivalents for Custom Feed Wizard Parameters

The NetWitness Platform Custom Feed wizard provides options to define the structure of the data feed file. These correspond directly to attributes in the feed definition (`.xml`) file.

NetWitness Platform Parameter	Feed Definition File Equivalent
(Define Feed Tab) Feed Type	Select: **Default** - to define a feed based on a `.csv` formatted feed data file. **STIX** - to define a feed based on STIX formatted `.xml` file.

NetWitness Platform Parameter	Feed Definition File Equivalent
(Define Feed Tab) Feed Task Type	Select: **Adhoc** - to create an on-demand feed. **Recurring** - to update the `.csv` or `.xml` file persistently and store it in a location accessible by NetWitness Platform, so NetWitness Platform downloads a file at regular intervals and pushes it to the downstream devices.
(Define Feed tab) **Name**	The custom feed name in the feed data file. It corresponds to the `flatfeedfile name` attribute in the feed definition file. For example, Dynamic DNS Test Feed. **Note:** You can use special characters to define the name of the custom feed.
(Define Feed tab) **File/ Browse**	This is the name of the feed data file. It corresponds to the `flatfeedfile path` attribute in the feed definition file. For example, `dynamic_dns.csv`.
(Advanced Options tab) **XML Feed File**	The name of the feed definition file. For example, `dynamic_dns.xml`.
(Advanced Options tab) **Separator**	The separator character used to separate attributes in the feed data file. It corresponds to the `flatfeedfile separator` in the feed definition file. For example, a comma.
(Advanced Options tab) **Comment**	The character used to identify a comment in the feed data file. It corresponds to the `flatfeedfile comment` attribute in the feed definition file. For example, `#`.
(Define Columns tab, Define Index) **Type**	The type of lookup value in the index position of the feed data file. **IP** means that each row in the feed data file contains an IP address in the lookup value position. The IP value is in dotted-decimal format (for example, 10.5.187.42). **IP Range** means that each row in the feed data file contains a range of IP addresses in the lookup value position. The IP range is in CIDR format (for example, 192.168.2.0/24). **Non IP** means that the each row in the feed data file contains a metadata value other than IP address in the lookup value position. The Service Type and Truncate Domain, and Callback Keys fields become active for a Non IP index.
(Define Columns tab, Define Index) **CIDR**	Specifies that the IP value in the lookup position is in CIDR format. The **CIDR** attribute sets the IP address format in the field to Classless Inter-Domain Routing (CIDR) notation.
(Define Columns tab, Define Index) **Service Type**	For a Non IP index, the integer service type to filter meta lookups. It corresponds to the `MetaCallback apptype` attribute in the feed definition file. A value of 0 indicates no filtering by service type.

NetWitness Platform Parameter	Feed Definition File Equivalent
(Define Columns tab, Define Index) **Truncate Domain**	For a Non IP index, for meta values that contain domain names (for example, hostnames), the system can strip off the host specific element in the data. Truncate Domain corresponds to the `MetaCallback truncdomain` attribute. If the value is `www.example.com`, it is truncated to `example.com`. A value of **False** selects no truncation, and **True** selects truncation.
(Define Columns tab, Define Index) **Callback Keys**	For a Non IP index, the available meta keys to match on instead of ip.src/ip.dst (the defaults for IP index type) are selectable from the drop-down list. The Callback Key corresponds to the `MetaCallback name` attribute, and the index column of the csv file must contain data that can match the chosen meta key. For example, if the `user` meta key is chosen, the index column of the `.csv` file needs to be populated with users to be matched.
(Define Columns tab, Define Index) **Index Column**	Identifies the column in the feed data file that provides the lookup value for the row. Each position in each row of the feed data file is identified by a **Field index** attribute in the feed definition file. A field with an index of **1** is the first entry in a row, the second field has an index of **2**, the third field has an index of **3**, and so on.
(DEFINE VALUES) **Key**	The name of the `LanguageKey`, as defined in the feed definition file, for which meta is created from this row of the feed data file. It corresponds to the `Field key` attribute in the feed definition file. A key applies only to a field whose type is set to `value`. In the feed definition file, there is a list of `LanguageKeys` from `index.xml`, or a summary name if Source Name and Destination Name are used. For example, `reputation` is a summary name for `reputation.src` and `reputation.dst`. This value is referenced by the Field key attribute.

Sample Files for a MetaCallback Feed Using CIDR Index Range for IPv4 and IPv6

These sample files demonstrate how to use CIDR index ranges for IPv4 and IPv6 in custom MetaCallback feeds. As with other custom feeds, you must create feed data file in .csv format, and a feed definition file in .xml format.

> **Note:** Using MetaCallback feeds with CIDR index ranges is supported only through the Advanced Configuration wizard or the REST interface.

The following example shows the contents of both a `.csv` file and an `.xml` file for a MetaCallback feed using CIDR index ranges for IPv4 or IPv6.

.csv file:

```
192.168.0.0/24, Sydney
192.168.1.0/24, Melbourne
```

.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>

<FDF xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="feed-definitions.xsd">

<FlatFileFeed name="ip_test" path="ip_test.csv" separator="," comment="#">
```

```
	<MetaCallback name="DstIP" valuetype="IPv4" apptype="0"
truncdomain="false">
		<Meta name="ip.dst"/>
	</MetaCallback>
	<LanguageKeys>
		<LanguageKey name="alert" valuetype="Text" />
	</LanguageKeys>
	<Fields>
		<Field index="1" type="index" range="cidr"/>
		<Field index="2" type="value" key="alert" />
	</Fields>
</FlatFileFeed>
</FDF>
```

Note: To configure a CIDR index range for feeds with single or multiple MetaCallbacks of value type IPv4 or IPv6, the field of type index MUST contain a range attribute with `range="cidr"`. Also, configuring "cidr" index ranges for feeds with MetaCallbacks of multiple different value types is not supported.

Feed Definitions File

This topic introduces the feed definitions file, which is available for editing in the Services Config view > Files tab. One of the files available for editing in the Services Config view > Files tab is **`feed-definitions.xml`**, the feed definitions file.

You can define feeds in the `feed-definitions.xml` file. The Decoder uses an XML schema to define feed messages when it creates a binary .feed file from the feeds defined here.

For details on the feed definition language, refer to Custom Feed Definition File Structure

Create a Custom Feed

You can create a custom feed using the Custom Feed wizard. To complete this procedure, you need a feed data file in `.csv` or `.xml` format. If you also have an associated feed definition file in `.xml` format, which describes the structure of the feed data file, you can use the feed definition file to create a feed. The Custom Feed wizard can create the feed based on a feed data file, or based on a feed data file and corresponding feed definition file.

> **Note:** For information about STIX and creating a STIX custom feed, see "Create a STIX Custom Feed" in the *Decoder and Log Decoder Configuration Guide*.

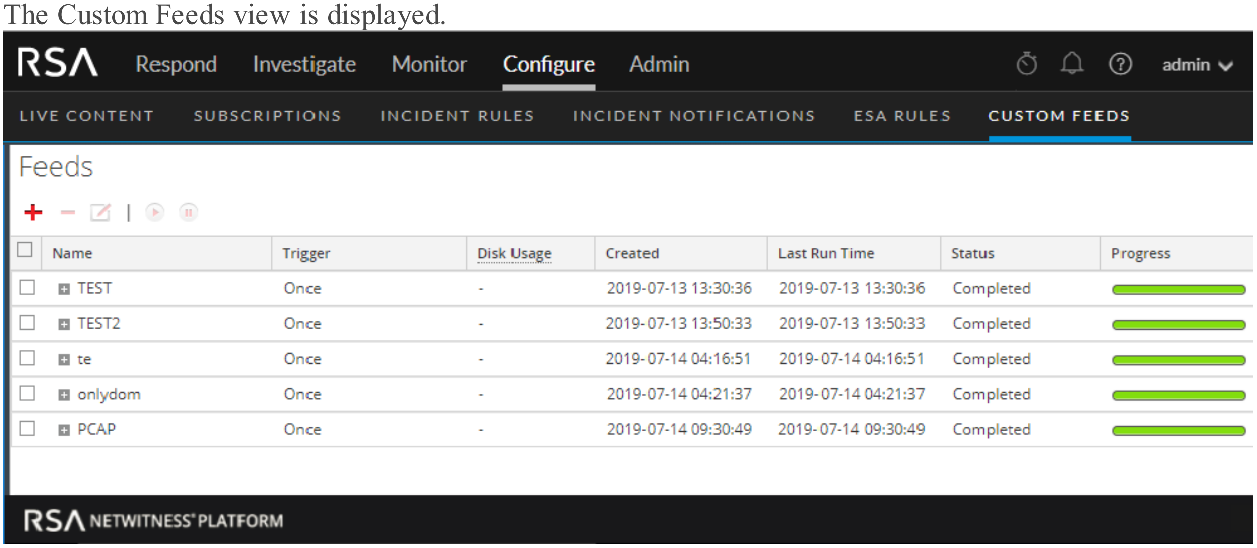
The feed data file and optionally the feed definition file (`.xml`) must be available on the local file system for an on-demand custom feed. For a recurring custom feed, the files must be available at a URL that is accessible to the NetWitness Platform server.

> **Note:** When you create a source and destination-based feed on a Log Decoder, it only populates the source meta key. You cannot use a range-based or CIDR feed. You must list every single IP address. To resolve this issue, create two different feeds using IP addresses and you can use CIDR in these feeds.

To create a custom feed:

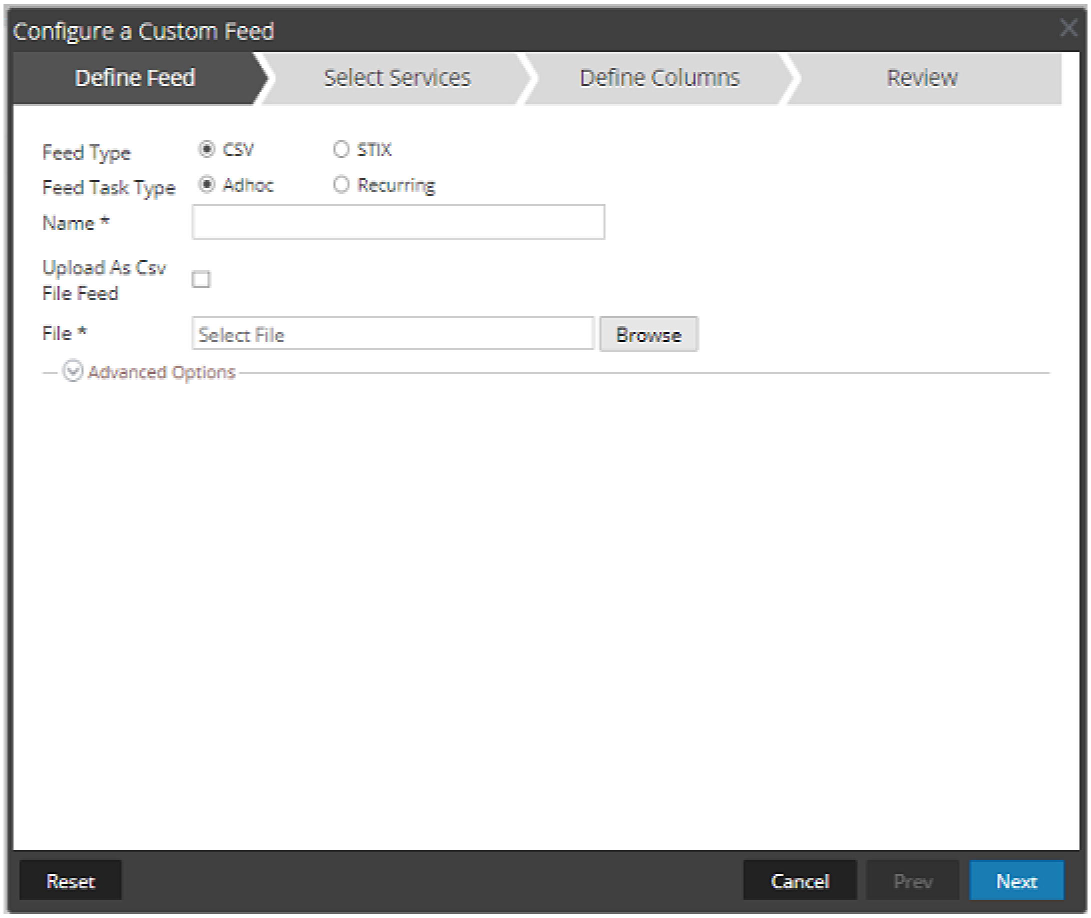
1. Go to **Configure > Custom Feeds.**

   The Custom Feeds view is displayed.



2. In the **Feeds** panel, click  > **Custom Feed** > **Next**.

   The Configure a Custom Feed wizard is displayed, with the Define Feed form open.



3. Select the Feed Type: **CSV** or **STIX**.
4. To define a feed based on a `.csv` formatted feed data file, select **CSV** (which is the default) in the **Feed Type** field.
5. To define an on-demand feed task that executes once, select **Adhoc** in the **Feed Task Type** field and do one of the following:

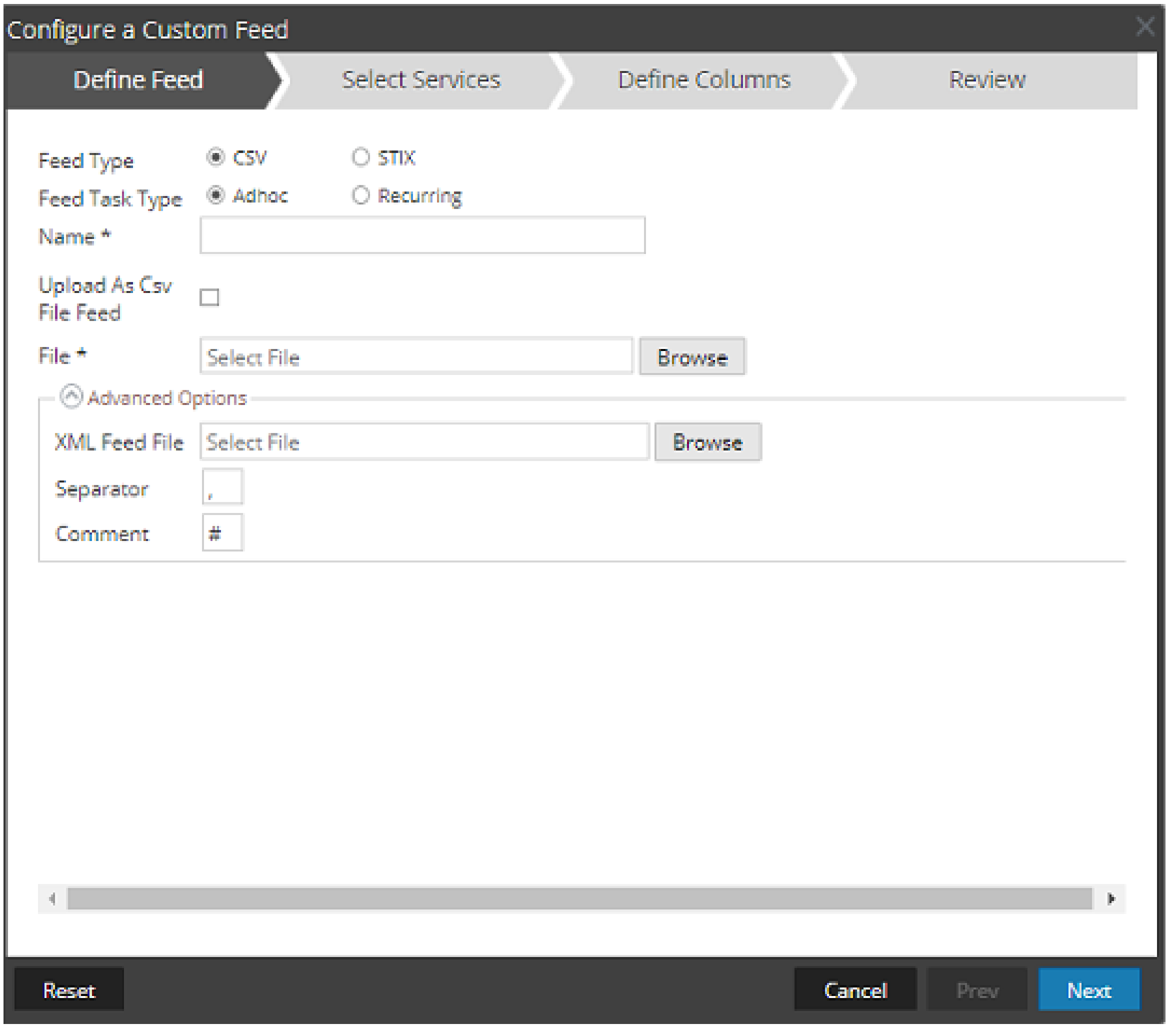
   a. (Conditional) To define a feed based on a CsvFileFeed file, select the **Upload as Csv File Feed** checkbox, type the feed **Name**, select a .csv content file from the local file system, and click **Next**. If you do not select the checkbox, the .csv file will be a FlatFileFeed file.

   > **Note:** When you select the Upload as Csv File Feed checkbox, the XML feed options under Advanced are unavailable.

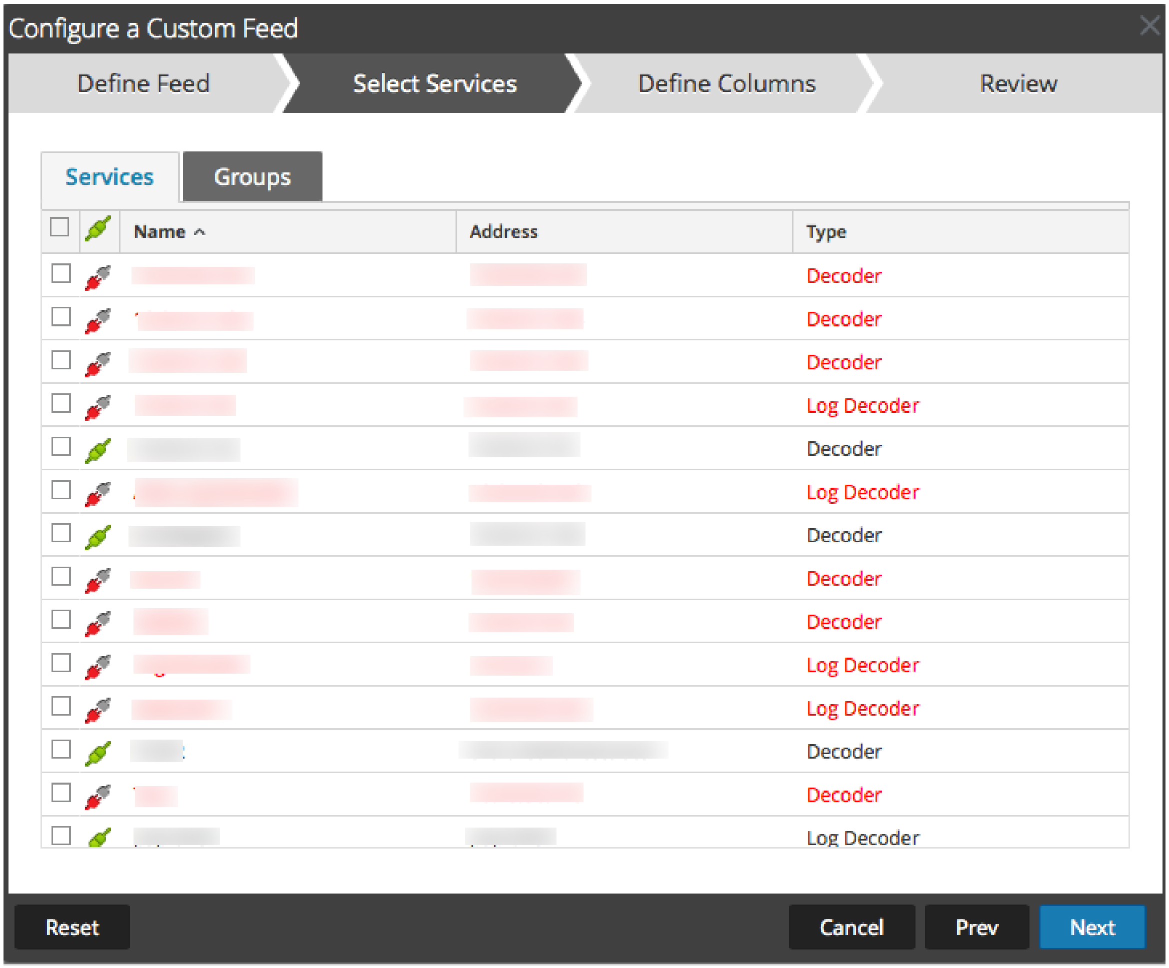
   b. (Conditional) To define a feed based on an XML feed file, select **Advanced Options**.

   > **Note:** Ensure that the Upload as Csv File Feed checkbox is deselected.

   The Advanced Options are displayed:



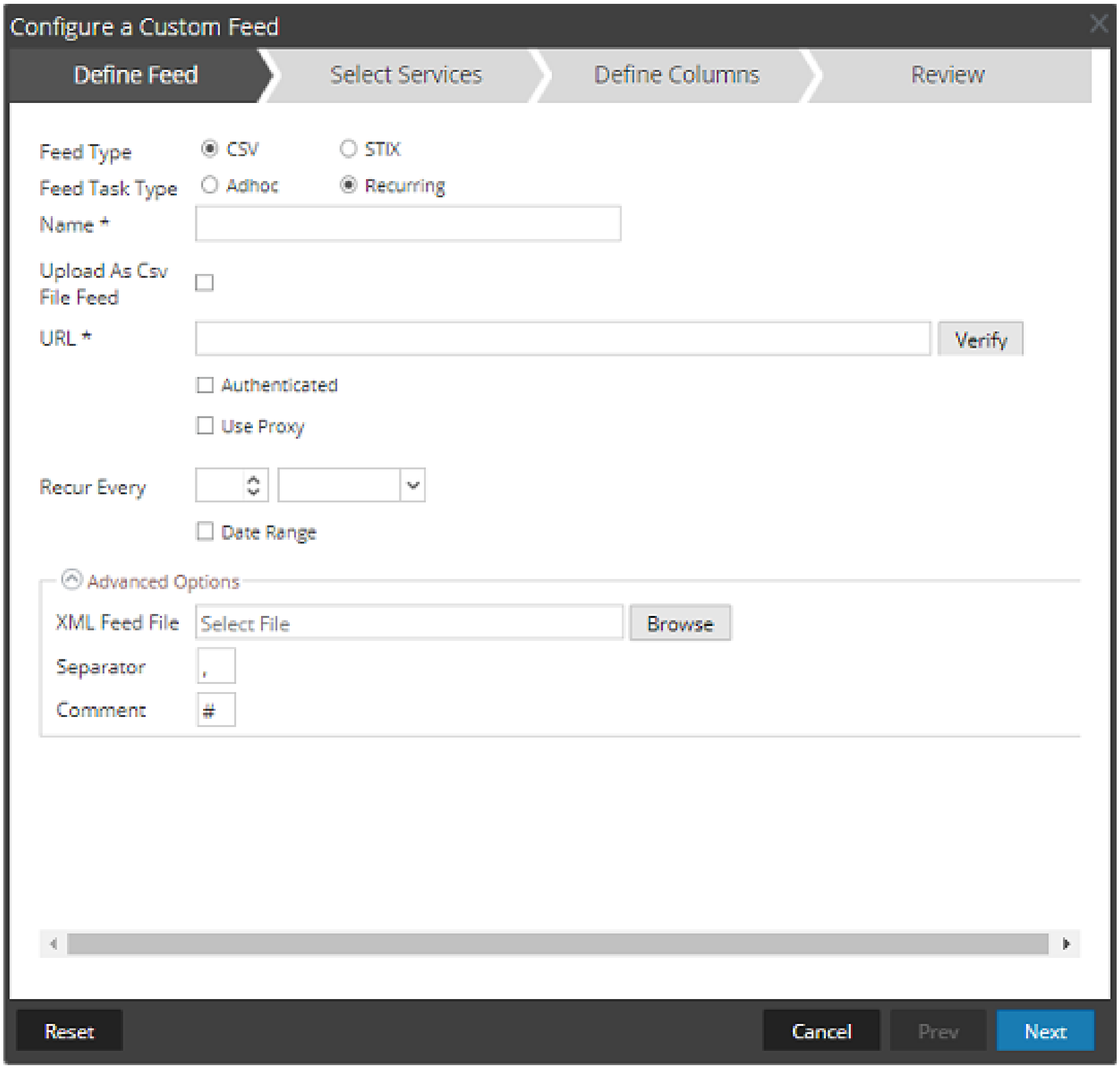
c. Select an XML feed file from the local file system, choose the separator (default is comma), specify the comment characters used in the feed data file (default is #), and click **Next**.
The Select Services form is displayed. This is an example of the form for a feed based on a feed data file with no feed definition file. If you are defining a feed based on a feed definition file, the Define Columns tab is not needed.



6. To define a recurring feed task that executes repeatedly at specified intervals, during a specified date range:

   a. In the Define Feed form, select **Recurring** in the **Feed Task Type** field.

      The Define Feed form includes the fields for a recurring feed.



b. In the **URL** field, enter the URL where the feed data file is located, for example, `http://<hostname>/<feeddatafile>.csv`, and click **Verify**. NetWitness Platform verifies the location where the file is stored in order to enable checking for the latest file automatically before each recurrence.

c. (Optional) If the URL has restricted access and requires authentication using your username and password, select **Authenticated**.
NetWitness Platform provides your user name and password for authentication to the URL.

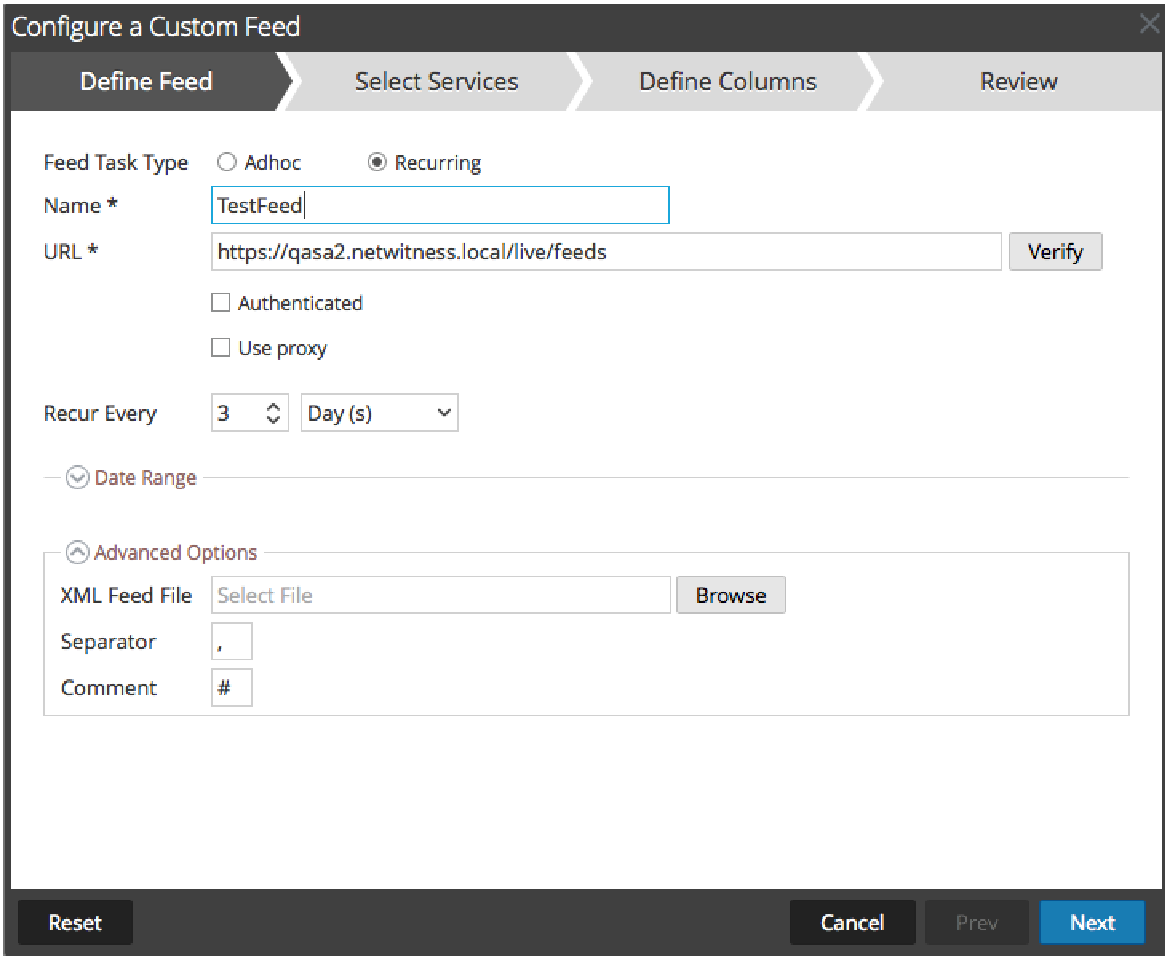
d. If you want the NetWitness server to access the Feed URL through a proxy, select **Use Proxy**. For more information on configuring a proxy, see "Configure Proxy for NetWitness Platform" in the *System Configuration Guide*. By default, the **Use Proxy** checkbox is not set.

e. To define the interval for recurrence, do one of the following:

- Specify the number of minutes, hours, or days between recurrences of the feed.

- Specify recurrence every week, and select the days of the week.

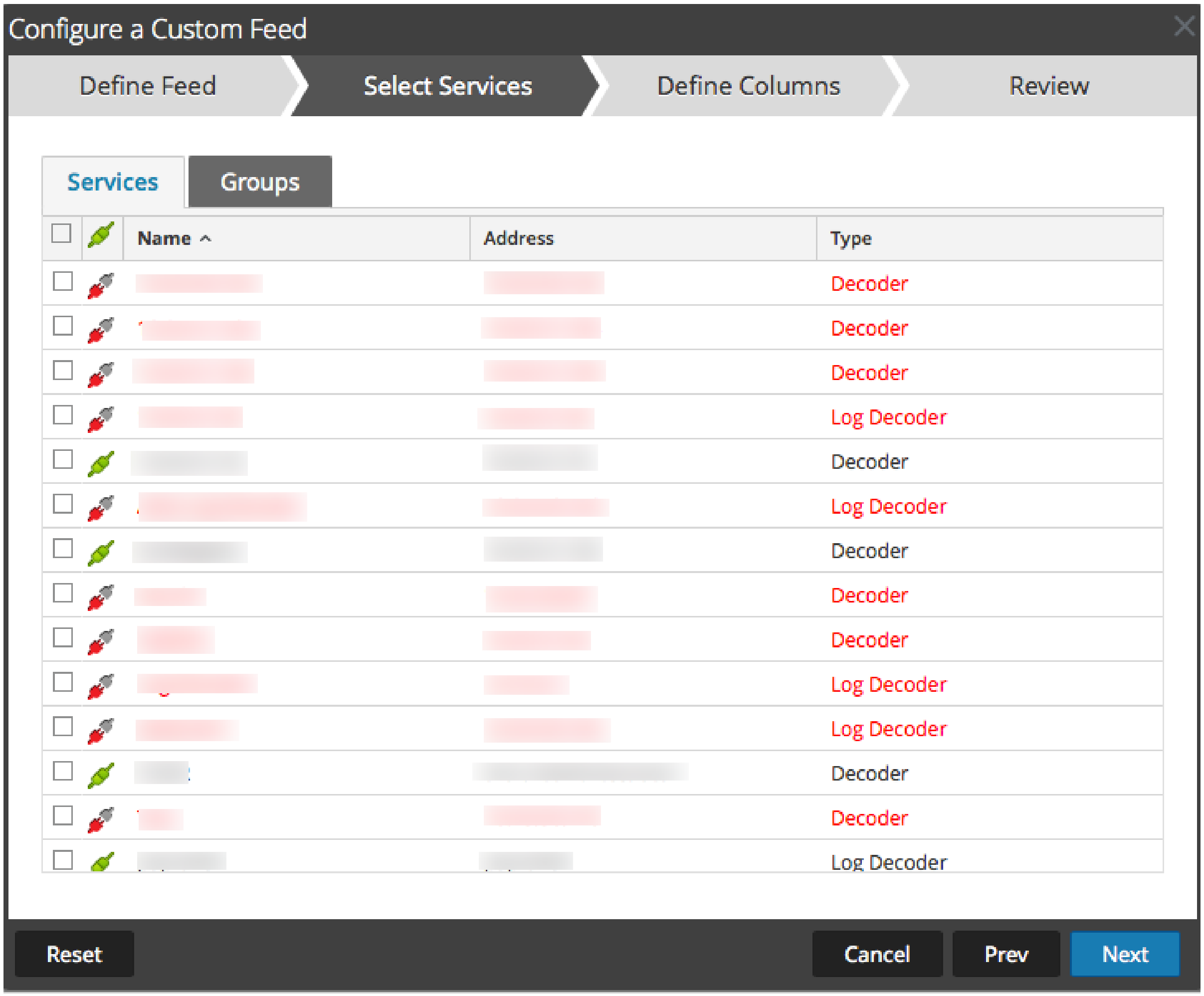
f. To define the date range for the execution of the feed to recur, specify the **Start Date** and time and the **End Date** and time.



7. (Conditional) If you want to define a feed based on an XML feed file:

a. Type the feed **Name**, select **Advanced Options**. The Advanced Options fields are displayed.

b. Select an XML feed file from the local file system, choose the **Separator** (default is comma), specify the **Comment** characters used in the feed data file (default is #) and click **Next**. The Select Services form is displayed.



8. To identify services on which to deploy the feed, do one of the following:

   a. Select one or more Decoders and Log Decoders, and click **Next**

   b. Click the **Groups** tab and select a group. Click **Next**.

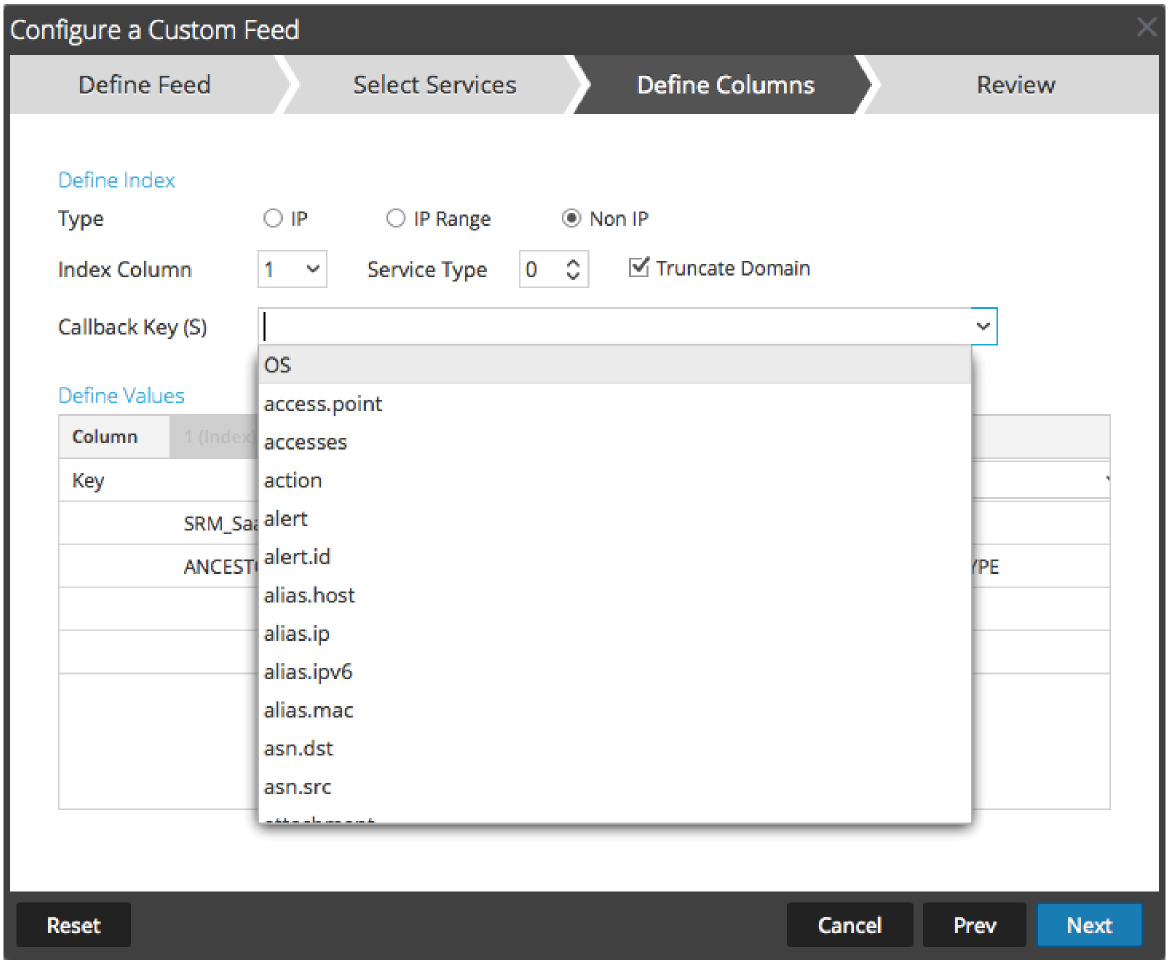
   The Define Columns form is displayed.

9. To map columns in the Define Columns form:

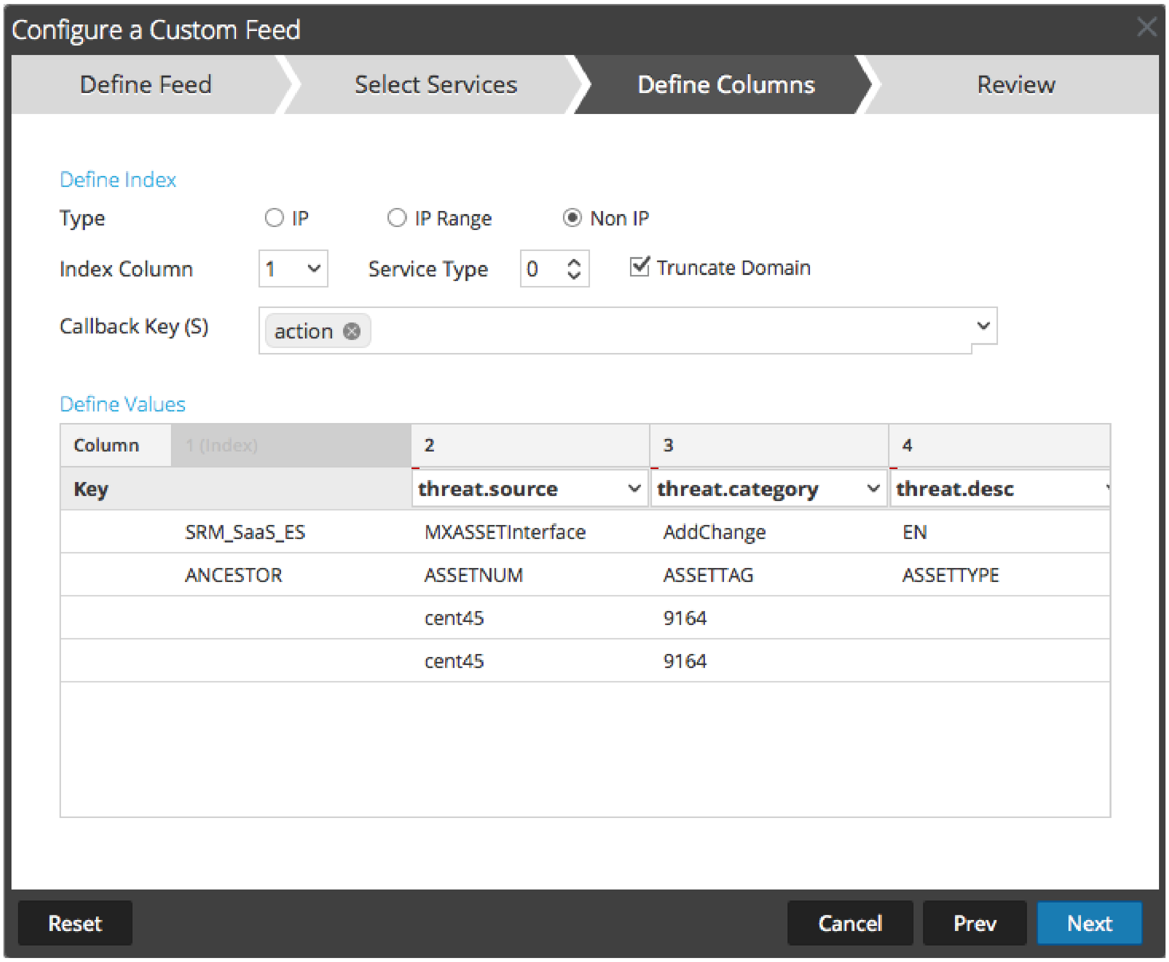
   a. Define the Index type: **IP**, **IP Range**, or **Non IP**, and select the index column.

   b. (Conditional) If the index type is **IP** or **IP Range** and the IP address is in CIDR notation, select **CIDR**.

   c. Conditional) If the index type is **Non IP**, additional settings are displayed. Select the service type and **Callback Keys,** and optionally select the **Truncate Domain** option.

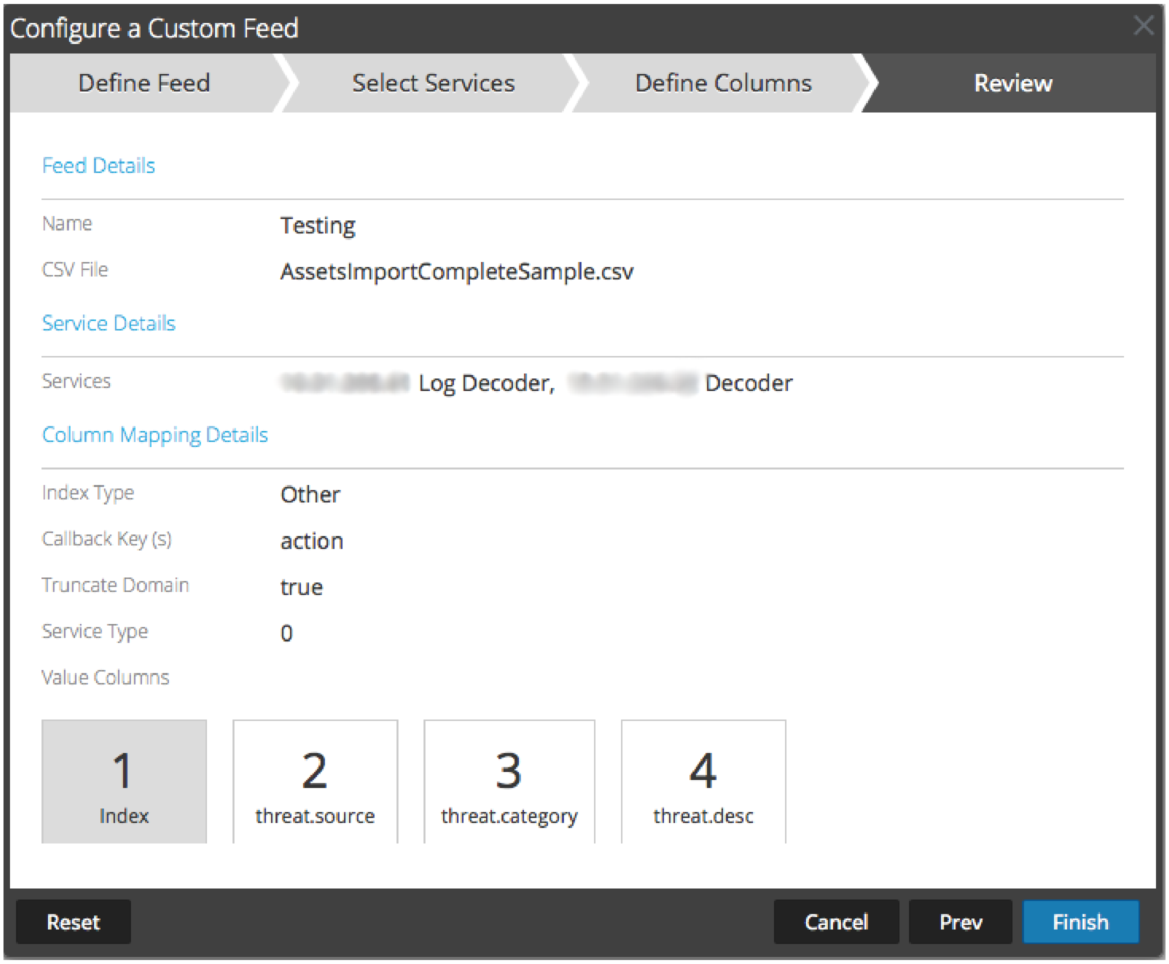


d. Select the language key to apply to the data in each column from the drop-down list. The meta keys displayed in the drop-down list is based on the meta keys available for the service define values. You can also add other meta keys based on advanced expertise.

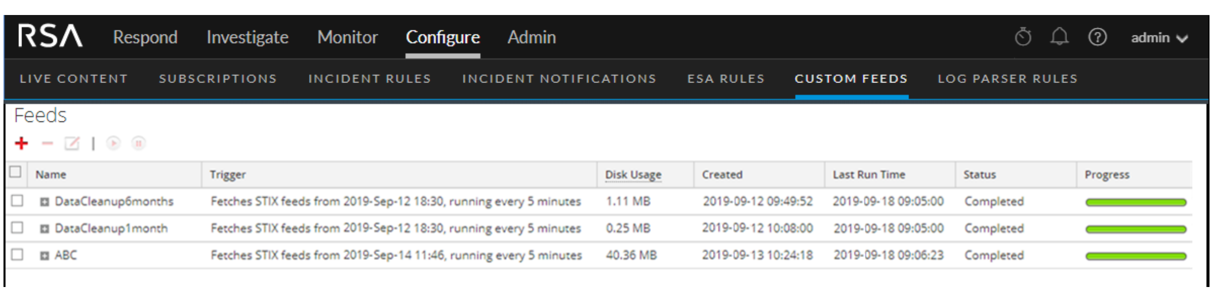


e. Click **Next**.

The Review form is displayed.



10. Anytime before you click **Finish**, you can:
    - Click **Cancel** to close the wizard without saving your feed definition.
    - Click **Reset** to clear the data in the wizard.
    - Click **Next** to display the next form (if not viewing the last form).
    - Click **Prev** to display the previous form (if not viewing the first form)
11. Review the feed information, and if correct, click **Finish**.
12. Upon successful creation of the feed definition file, the Create Feed wizard closes, and the feed and corresponding token file are listed in the Feed grid and progress bar tracks completion. You can expand or collapse the entry to see how many services are included, and which services were successful.



Create a STIX Custom Feed

Structured Threat Information Expression (STIX™) is a structured language for describing cyber threat information so it can be shared, stored, and analyzed in a consistent manner. For more information about STIX, see https://stixproject.github.io/.

You can create a custom feed using a STIX-formatted feed data file (`.xml`) in RSA NetWitness Platform. NetWitness Platform supports Structured Threat Information Expression (STIX) 1.0, 1.1 and 1.2 versions only.

Caution: If a STIX recurring feed is configured and you update Security Analytics from 10.6.x to NetWitness Platform 11.x, you must re-configure the STIX recurring feed.

In NetWitness Platform, STIX feeds of type Indicator or Observable that contain properties such as the IP addresses, File hashes, Domain names, URIs and Email addresses are supported. The property values in the Equals operator are supported. Attributes such as Type and Title are also read from the STIX. A STIX file with a single STIX_Package is supported.

TAXII (Trusted Automated eXchange of Indicator Information) is the main transport mechanism for cyber threat information represented in STIX. Using the TAXII services, organizations can share cyber threat information in a secure and automated manner.

The STIX and TAXII communities work closely together to ensure that they continue to provide a full stack for sharing threat intelligence.

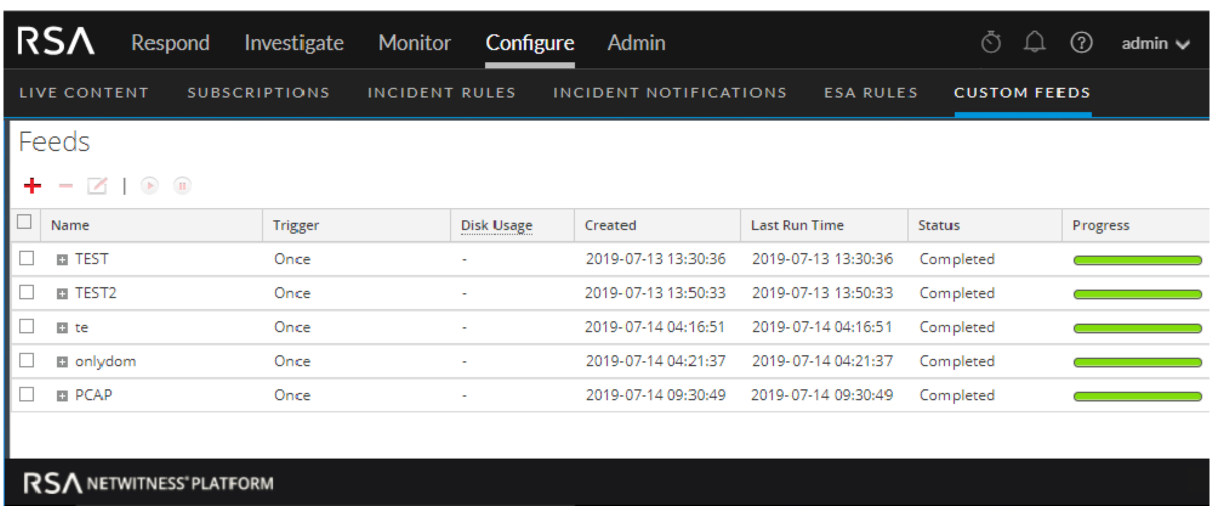
Apart from the TAXII server, STIX data can also reside on a REST server and you can fetch the STIX file from the REST server by providing the URL of the REST server. For example, `http://stixrestserver.internal.com`.

The STIX feed data file and optionally the feed definition file, both in `.xml` format must be available on the local file system for an on-demand custom feed. For a recurring custom feed, the files must be available at a URL that is accessible to the NetWitness Platform server.

To create a STIX custom feed:

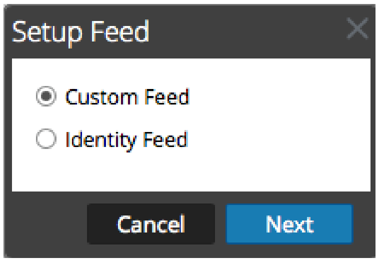
1. Go to **Configure > Custom Feeds**.

   The Custom Feeds view is displayed.



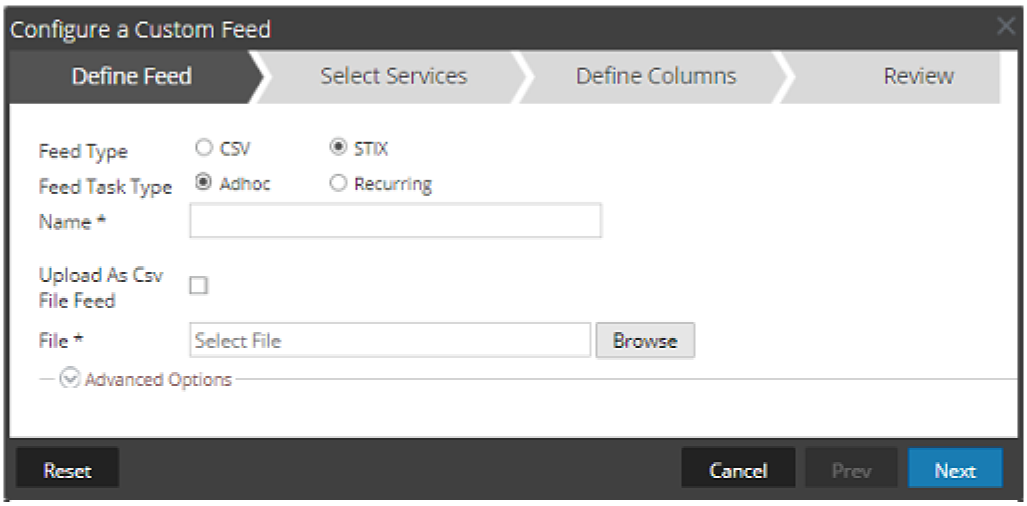
2. In the toolbar, click .

   The Setup Feed dialog is displayed.

3. To select the feed type, click **Custom Feed** and **Next**.

   The Configure a Custom Feed wizard is displayed, with the Define Feed form open.

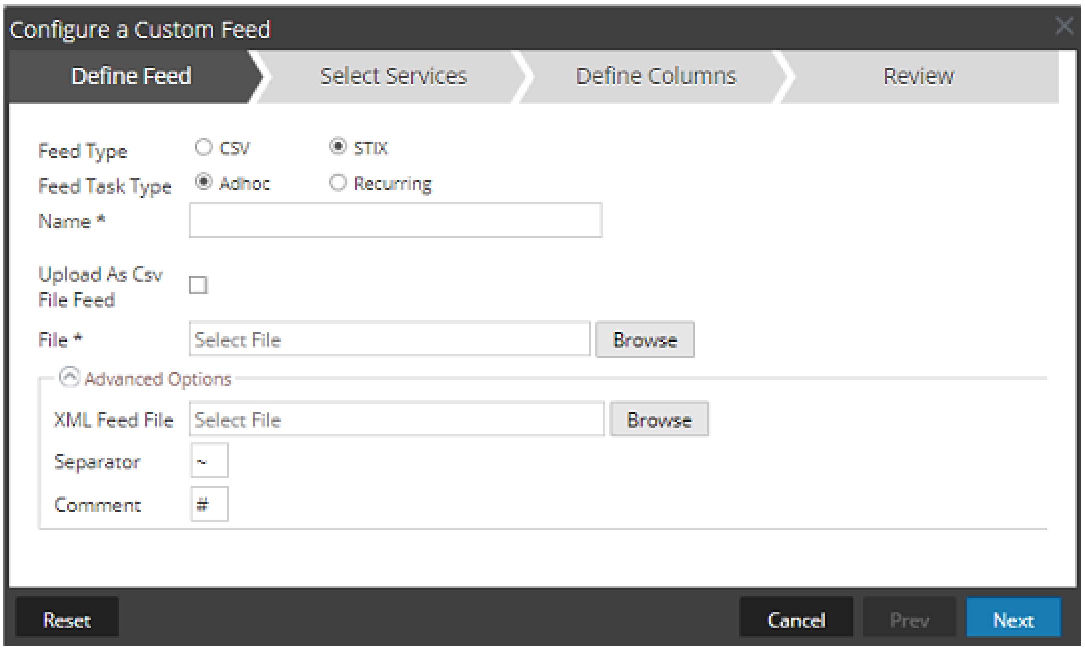
4. To define a feed based on a STIX formatted `.xml` file, select **STIX** in the **Feed Type** field.

5. To define an on-demand feed task that executes once, select **Adhoc** in the **Feed Task Type** field and do one of the following:

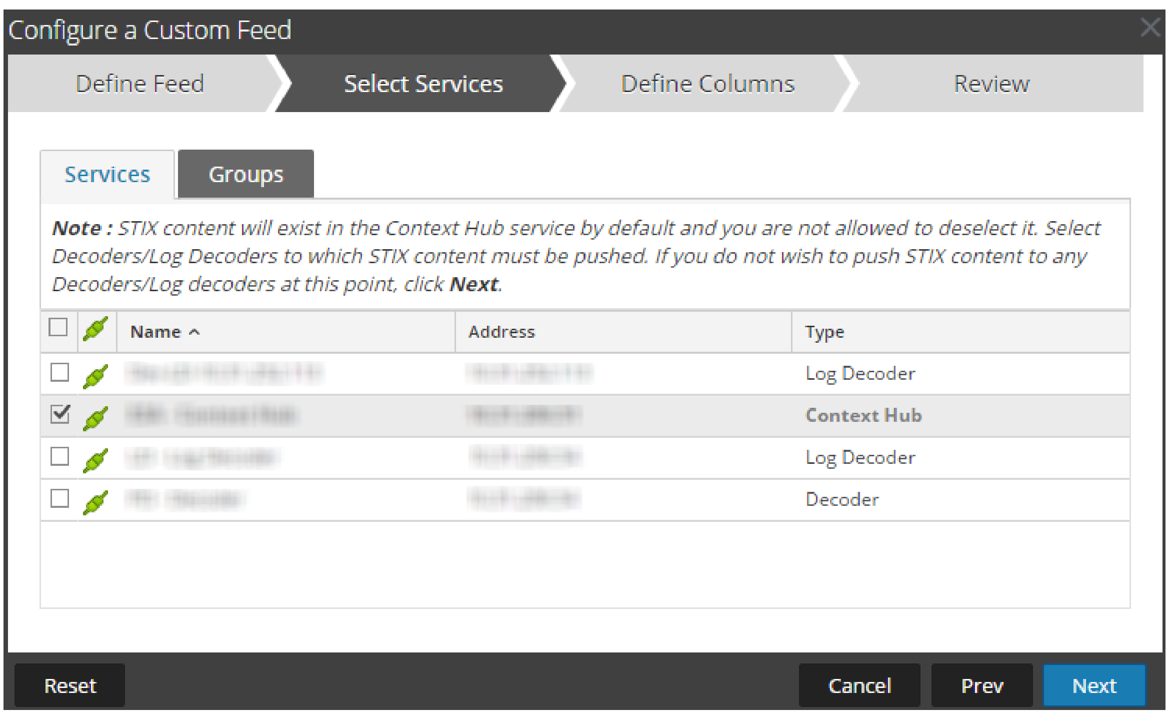
   a. (Conditional) To define a feed based on STIX-formatted `.xml` file, type the feed **Name**, select a STIX formatted `.xml` content **File** from the local file system, and click **Next**.

   b. (Conditional) To define a feed based on an XML feed file, select **Advanced Options**.

      The Advanced Options are displayed.



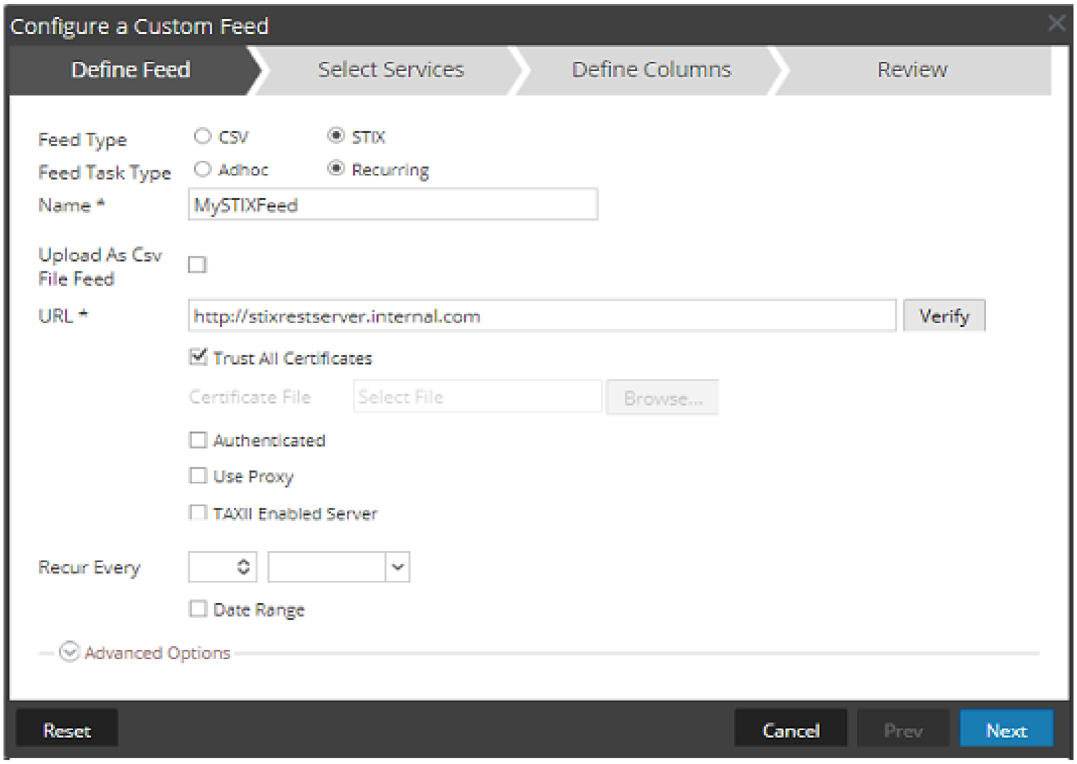
c. Select an XML feed file from the local file system, choose the Separator (default is comma), specify the Comment characters used in the feed data file (default is #), and click **Next**.
The Select Services form is displayed. This is an example of the form for a feed based on a feed data file with no feed definition file. If you are defining a feed based on a feed definition file, the Define Columns tab is not needed.



6. To define a recurring feed task that executes repeatedly at specified intervals, during a specified date range.

a. Select **Recurring** in the **Feed Task Type** field.

The Define Feed form includes the fields for a recurring feed.

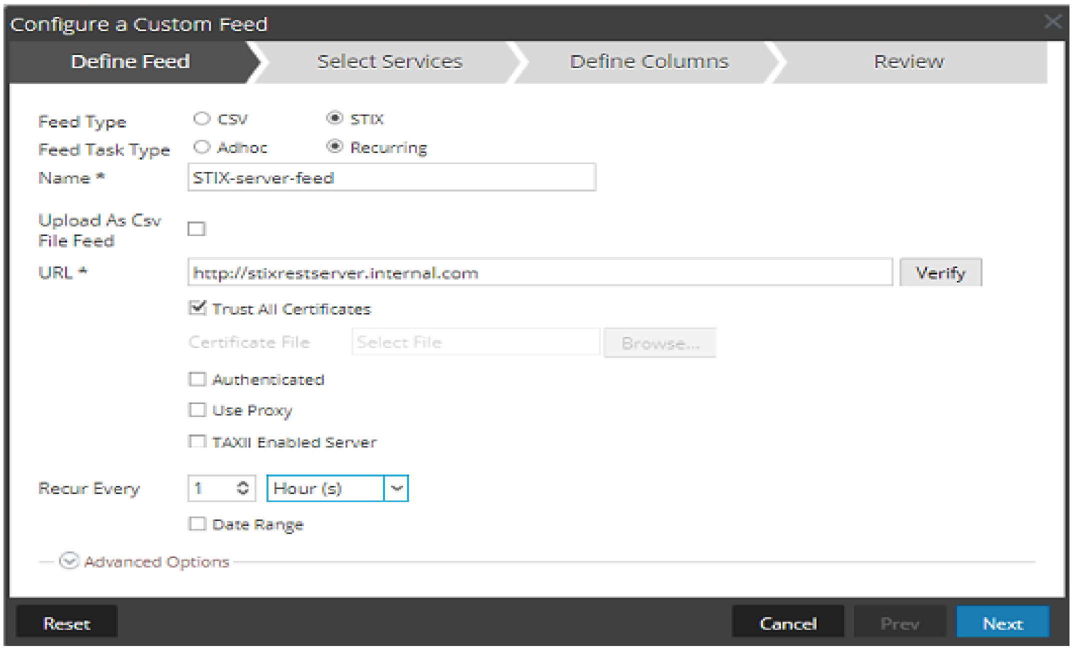


b. In the **URL** field, do one of the following:

- To define a recurring feed based on STIX which pulls STIX packages from a TAXII Server, enter the TAXII server's discovery service URL, for example, `http://hailataxii.com/taxii-discovery-service`.

  Note: A Context Hub service installed on Event Stream Analysis host must be reachable for the specified TAXII server.

- To define a recurring feed based on a STIX-formatted `.xml` file using the REST Server, enter the URL of the REST server where the STIX data file is located, for example, `http://stixrestserver.internal.com.`



NetWitness Platform verifies the connection to the server, so that NetWitness Platform can check for the latest file automatically before each recurrence.

c. If you do not want NetWitness Platform to verify the REST server's SSL certificate, Select **Trust All Certificate**. This option is enabled by default (checked).

d. For client authentication with the REST URL, in the **Certificate** field, click **Browse** and select the self signed certificate. The supported certificate formats are `.cer`, `.crt` with `Base64` and `DER` encoded files.

e. (Optional) If the URL has restricted access and requires authentication using your username and password, select **Authenticated**.

   NetWitness Platform provides your user name and password for authentication to the URL.

f. Select **TAXII Enabled Server**, if you want to select a TAXII collection from the list.
   For a valid URL, one or more TAXII collections that contains the STIX data file is displayed based on your credentials. Select the required TAXII collection from the list. Only one collection can be added from a TAXII server for a feed.

   Note: Though multiple feeds from multiple TAXII servers are supported, only one account (username and password) is supported per TAXII server.

g. If you want the NetWitness Platform server to access the feed URL through a proxy, select **Use Proxy**. For more information on configuring a proxy, see "Configure Proxy for NetWitness Platform" in the *System Configuration Guide*. (Go to the Master Table of Contents to find all RSA NetWitness Platform 11.x documents.) By default, the **Use Proxy** checkbox is not selected.

h. (Optional) Click **Verify** to test the settings.

Note: Make sure all the required connection parameters such as Authentication, Proxy, Certificate trust, TAXII Enabled Server, and others, are configured before you click Verify.

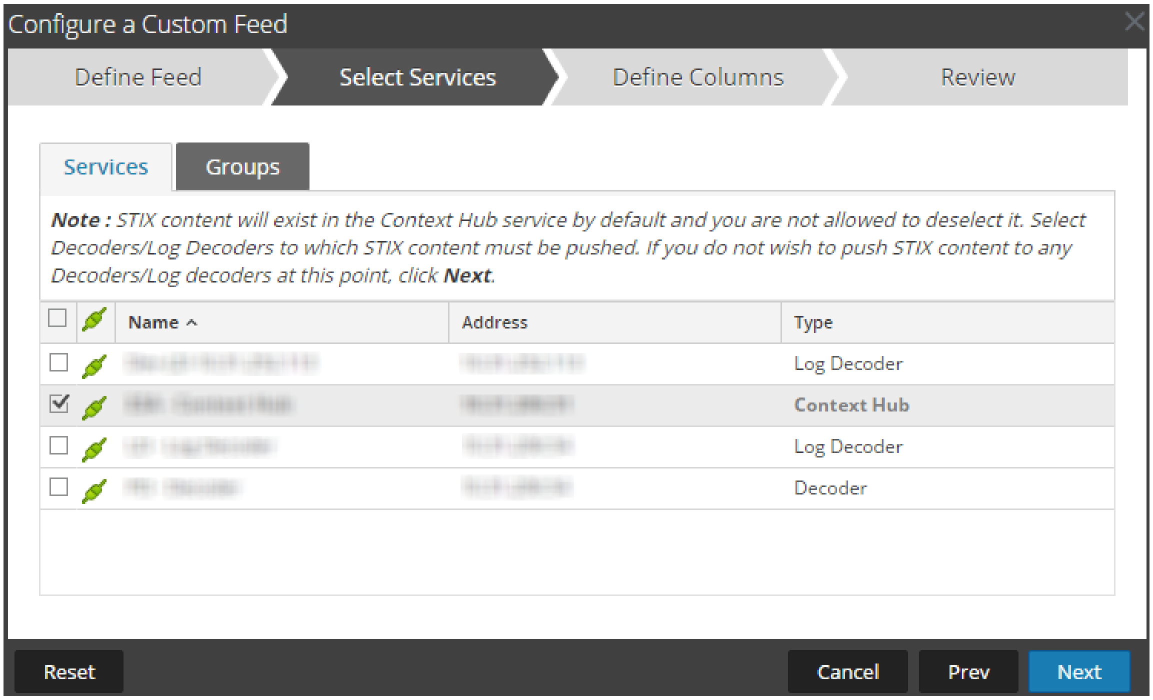
i. To define the interval of recurrence for pushing to the Decoder or Log Decoder, do one of the following:
   - Specify the number of minutes, hours, or days between recurrences of the feed.
   - Specify recurrence every week, and select the days of the week.

j. To define the date range for the execution of the feed to recur, specify the **Start Date** and time and the **End Date** and time. The Start Date defines from when you want to fetch the data.

7. (Conditional) If you want to define a feed based on an XML feed file:
   - Type the feed **Name**, select **Advanced Options**.

     The Advanced Options fields are displayed.
   - Select an XML feed file from the local file system, choose the **Separator** (default is comma), specify the **Comment** characters used in the feed data file (default is #).
   - In the **Remove STIX data older than** field, specify the number of days for which STIX packages pulled from TAXII server is to be stored. The STIX packages older than the specified number of days is deleted automatically.
   - Click **Next**.

     The Select Services form is displayed.
8. To identify services on which to deploy the feed, do one of the following:
   a. Select one or more Decoders and Log Decoders, and click **Next**.
   b. In case of STIX feed, Context Hub is selected by default and you cannot deselect it. In addition, you can select one or more Decoders and Log Decoders and click **Next** or click the **Groups** tab and select a group. Click **Next**.

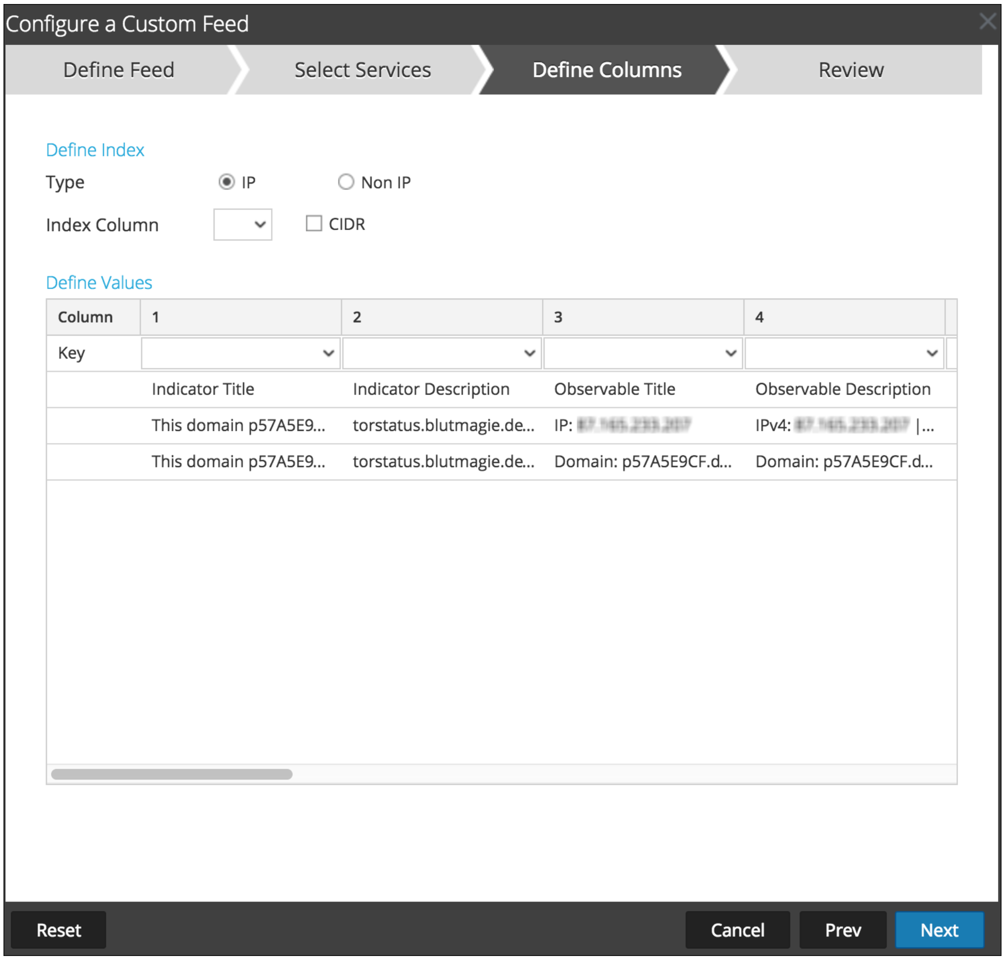


If the data from the STIX server is large, the following message is displayed: "Fetching sample date is taking longer than expected. Choose one of the following options." You have two options: continue to wait or map without sample data.

- If you click **Continue to Wait**, the Feed Wizard continues to wait till the sample data is fetched or a timeout (10 minutes) occurs, whichever is sooner. If there is a timeout, no sample data is retrieved.
- If you click **Map without Sample data**, the mapping column is displayed without any sample data.

The Define Columns form is displayed.

9. To map columns in the Define Columns form:
   a. Define the Index type: **IP**, **IP Range**, or **Non IP**, and select the index column.
   b. (Conditional) If the index type is **IP** or **IP Range** and the IP address is in CIDR notation, select **CIDR**.
   c. (Conditional) If the index type is **Non IP**, additional settings are displayed. Select the service type and **Callback Keys,** and optionally select the **Truncate Domain** option.

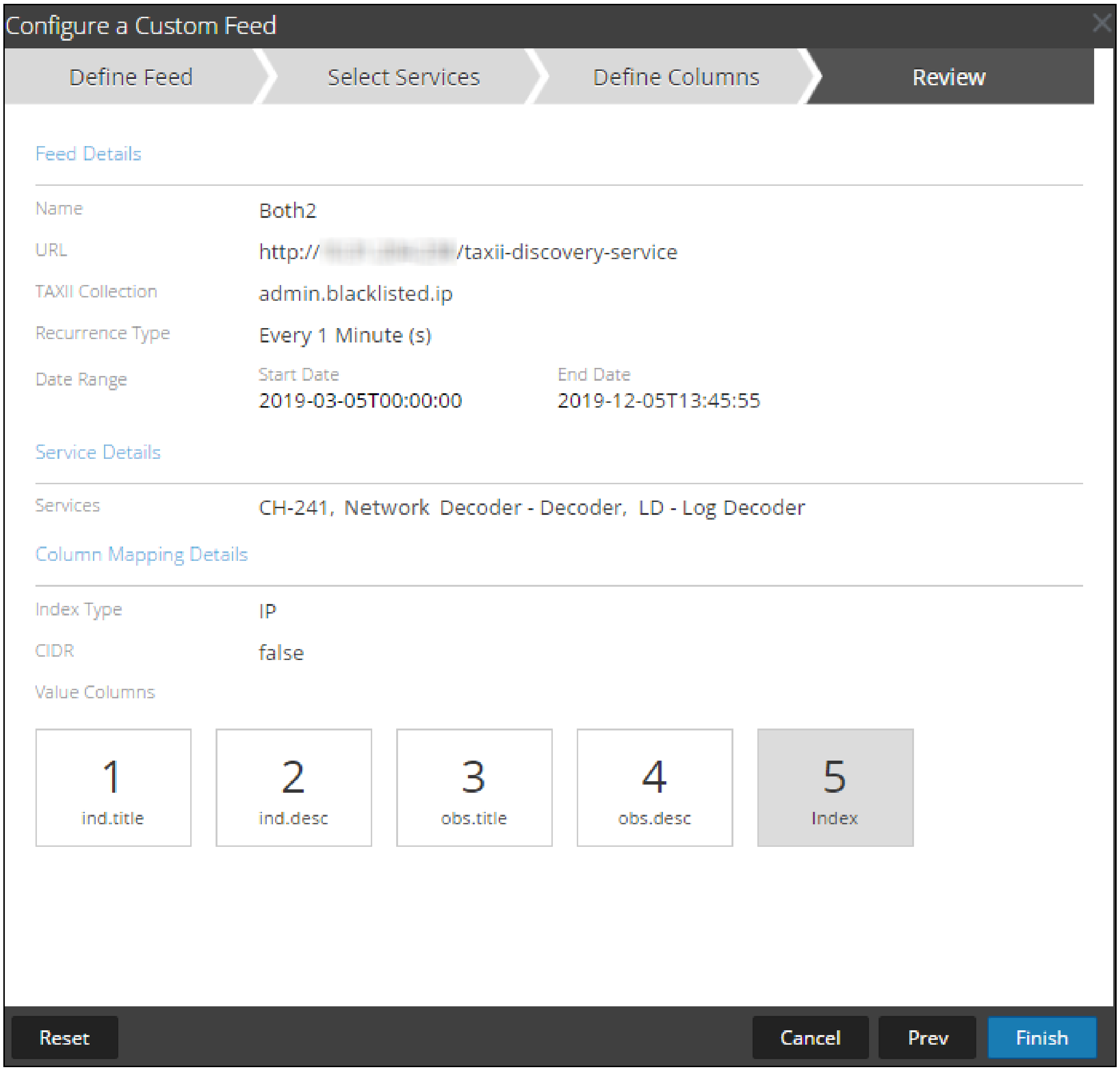


- If the Index Type is Non IP, you can select multiple index columns in the Index Columns. The values from all the selected columns are merged in the first index column that you selected and the merged values are pushed to the Log Decoder for parsing. For example, in the Index Columns if you select 2,4,7 as index columns the values from the 2,4, and 7 columns are merged in the column 2 and the values are pushed to Log Decoder for parsing.
- Indexing cannot be done for columns such as Indicator Title, Indicator Description, Observable Title, and Observable Description, as the look up cannot be performed for those columns.

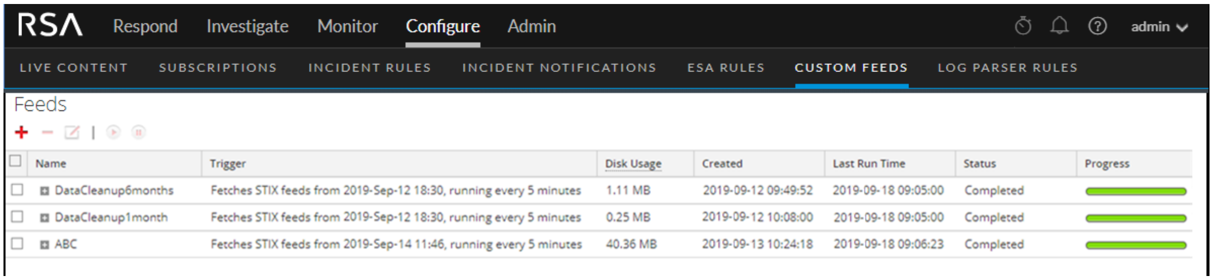
d. Select the language key to apply to the data in each column from the drop-down list. The meta displayed in the drop-down list is based on the meta available for the service define values. You can also add other meta based on advanced expertise.

e. Click **Next**.

The Review form is displayed.



10. Anytime before you click **Finish**, you can:
    - Click **Cancel** to close the wizard without saving your feed definition.
    - Click **Reset** to clear the data in the wizard.
    - Click **Next** to display the next form (if not viewing the last form).
    - Click **Prev** to display the previous form (if not viewing the first form).
11. Review the feed information, and if correct, click **Finish**.
    Upon successful creation of the feed definition file, the Create Feed wizard closes, the feed and corresponding token file are listed in the Feed grid, and progress bar tracks completion. You can expand or collapse the entry to see how many services are included, and which services were successful.



Note: Health and Wellness raises alerts when the available heap memory of Context Hub server is critically low. If the status of Context Hub server is Unhealthy due to low memory. For more information on how to troubleshoot `OutOfMemoryError` on `Contexthub` Server, refer to "Troubleshooting" in the *Live Services Management Guide*.

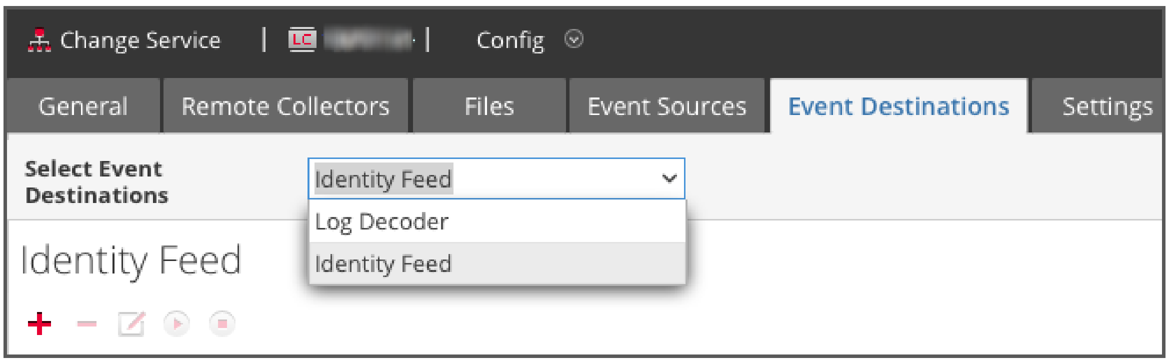
Create an Identity Feed

You can create an Identity feed and populate it to selected Decoders and Log Decoders. In order to create an identity feed, you need to have:

- A Log Collector service with an Identity Feed Event Processor
- A Log Collector service with Windows Collection configured and enabled

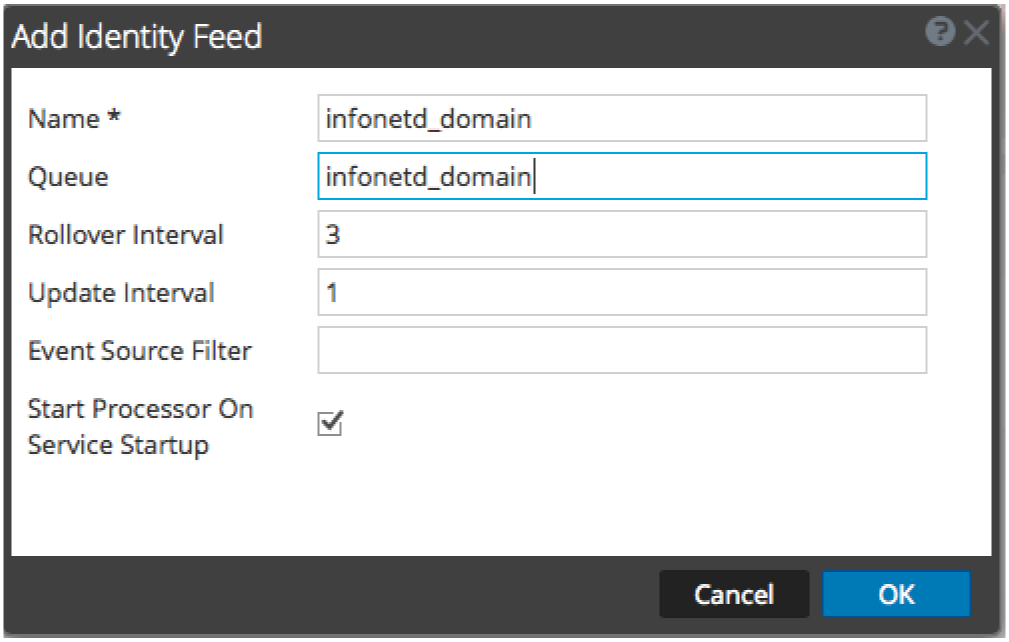
To create an identity feed:

1. Add a destination for the feed.
   a. Go to **ADMIN** > **Services** and in the **Services** list
   b. Select a **Log Collector** service, and select **View** > **Config**.
   c. Select the **Event Destinations** tab.
   d. In the Select **Event Destinations** field, select **Identity Feed**.



   e. Click and enter a unique name for the feed.

   The Queue name identifies the feed within the Log Collector. Use the name of the feed for the Queue.

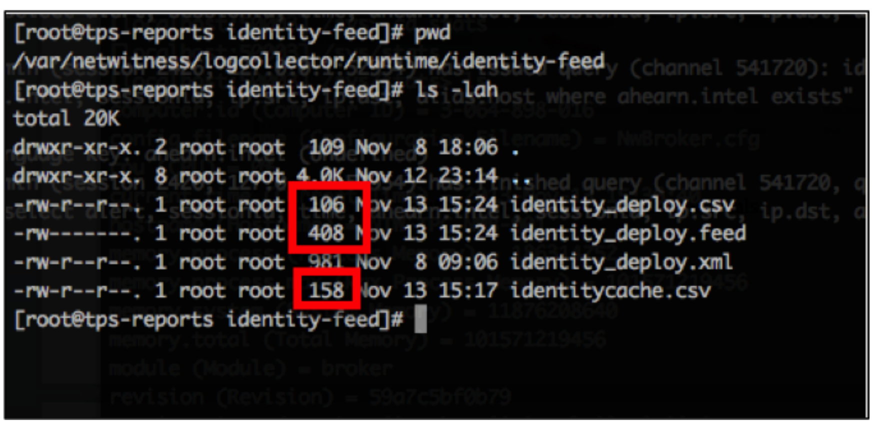


   f. Click **OK**.

2. Test generation of messages.

   a. Have users log into Windows boxes on the domain to generate the appropriate log messages on the domain controllers for testing.

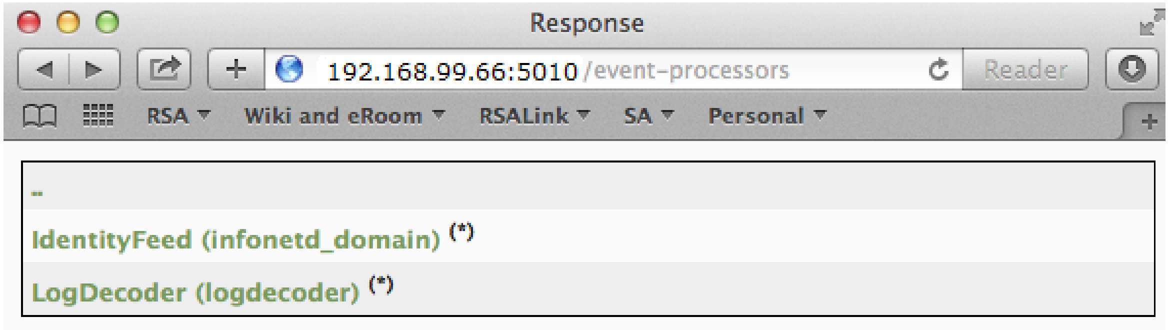
   b. Verify that data is written to the feed files. SSH to the Log Decoder/Collector or Virtual Log Collector being configured. Navigate to `/var/netwitness/logcollector/runtime/identity-feed` and verify that the `Identity_deploy` files are getting populated with data.



   c. Open up a web browser (Non-Internet Explorer browsers preferred) and log in to the REST interface of the Log Collector. Use administrative credentials when logging in. For example, if the IP address of your Log Collector is 192.168.99.66, the URL would be:

   - SSL not enabled: **http://192.168.99.66:50101/event-processors**
   - SSL enabled: **https://192.168.99.66:50101/event-processors**

   The browser screen should look like this:

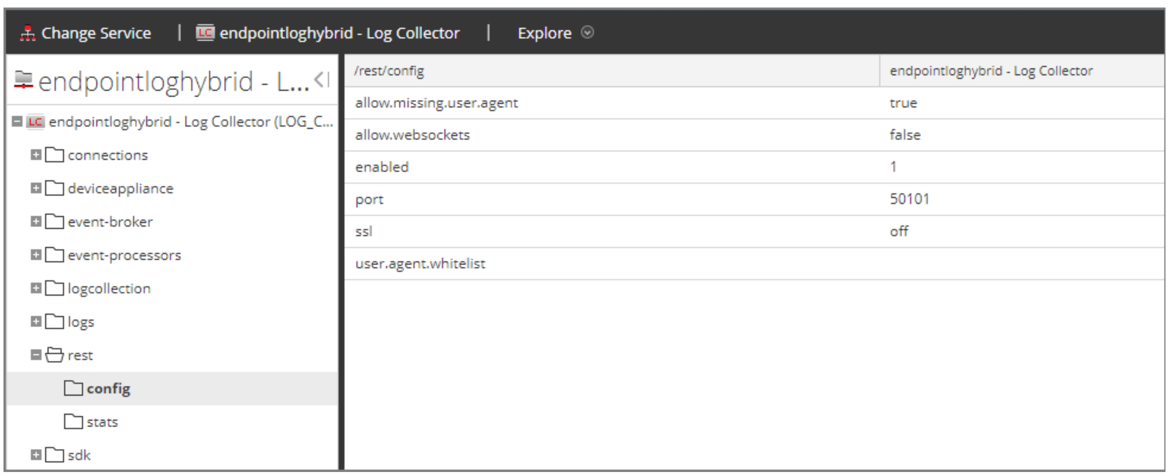


The screen contains the name of the identity feed you created earlier (`infonetd_domain`, in this example).

For the identity feed to function correctly, port 50101 must be active on the Log Collector, and you must determine whether SSL encryption is active.

d. Go to **ADMIN** > **Services** > ***<Log Collector being setup>***  > **View** > **Explore**.

e. In the left pane, expand **rest** > **config**.



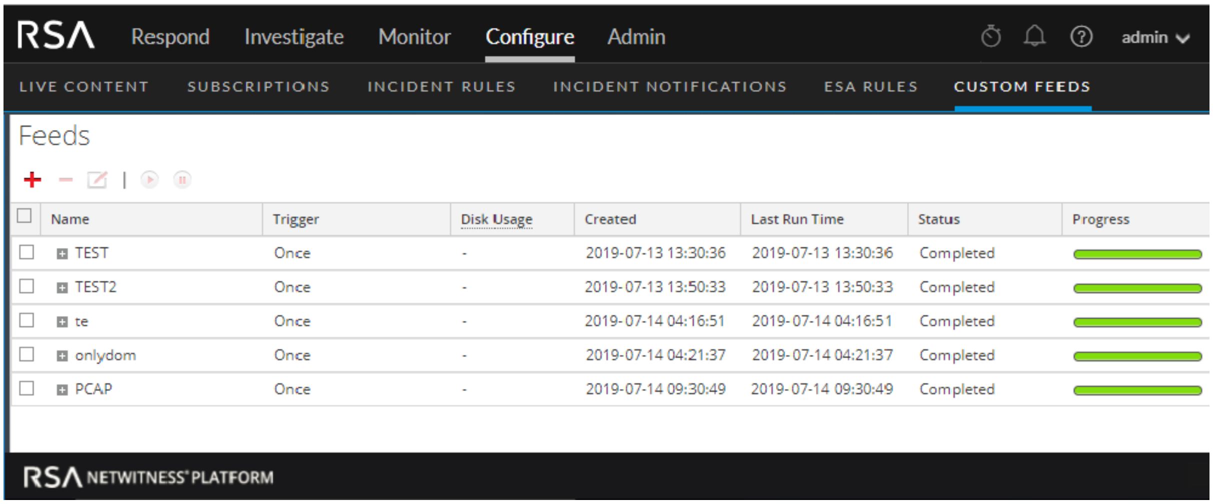
For REST to be active, **enabled** must be set to **1**.

f. Note the value for **ssl**. If SSL should be enabled for your environment, this must be set to **on**.

> **Note:** If you changed the setting for either the **enabled** or **ssl** option you must restart the Log Collector service before moving forward.

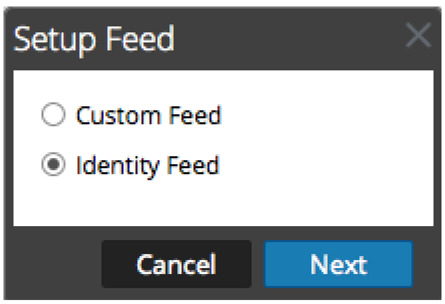
3. Go to **Configure** > **Custom Feeds**.

The Feeds dialog is displayed.



4. In the toolbar, click + .

   The Setup Feed dialog is displayed.



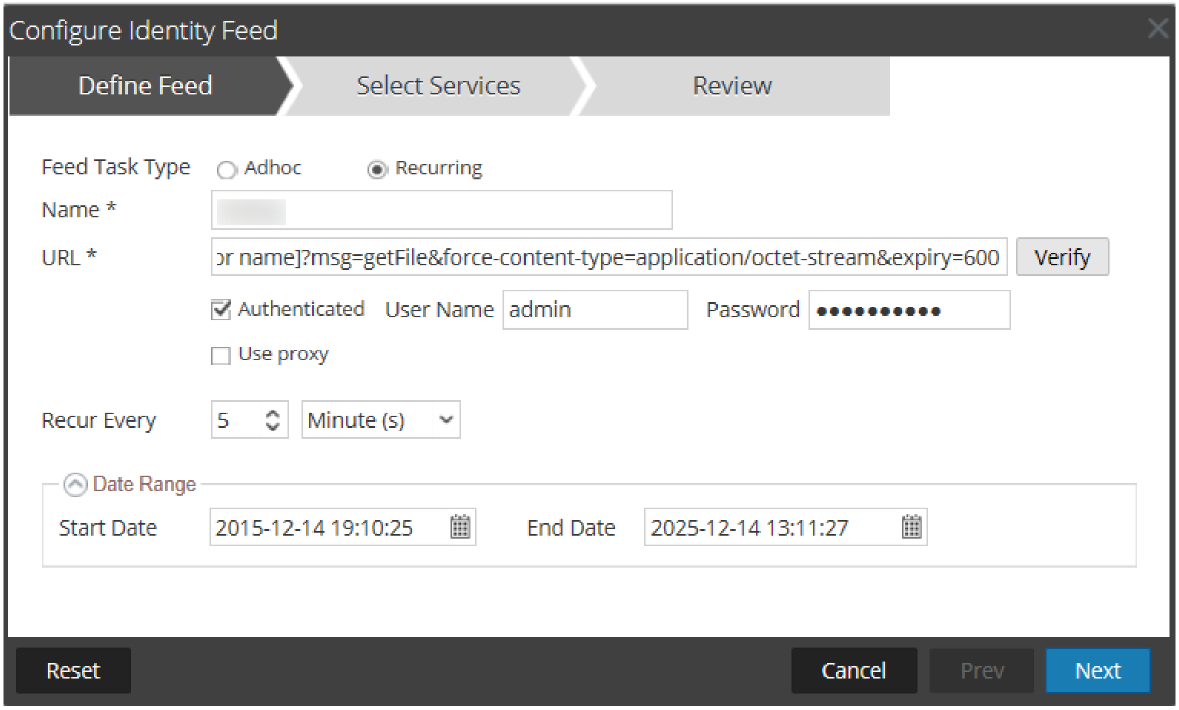
5. Make sure **Identity Feed** is selected and click **Next**.

   The Configure Identity Feed panel opens with the **Define Feed** tab displayed.

6. (Conditional) You can create an on-demand or recurring feed.

   - To define an on-demand Identity feed task that executes once, select **Adhoc** in the **Feed Task Type** field, type the feed **Name**, and browse for and open the feed.
   - To define a recurring Identity Feed task that executes on a recurring basis, select **Recurring** in the **Feed Task Type** field.

     The **Define Feed** dialog includes the fields for a recurring feed.



> **Note:** RSA NetWitness Platform verifies the location where the file is stored, so that NetWitness Platform can check for the latest file automatically before each recurrence.

7. Enter a value and verify the URL field.

   a. In the **URL** field, enter the URL where the feed data file is located. This is the REST API interface that was setup earlier. Make sure you have the following information to construct the URL:

   - The IP address of the Log Collector being used to construct the Identity Feed file.
   - The identity queue name, as set in step 2c.
   - Whether or not SSL is enabled on the Log Collector REST port, as set in step 2f.

   You can construct this value as follows:

   - SSL enabled: https://***<LogCollector>***:50101/event-processors/***<ID Event processor name>***?msg=getFile&force-content-type=application/octet-stream&expiry=600
   - SSL not enabled: http://***<LogCollector>***:50101/event-processors/***<ID Event processor name>***?msg=getFile&force-content-type=application/octet-stream&expiry=600

   So, using the example from earlier, the complete value that you would enter into this field is as follows:

   ```
   http://192.168.99.66:50101/event-processors/infonetd_
   domain?msg=getFile&force-content-type=application/octet-
   stream&expiry=600?msg=getFile&force-content-type=application/octet-
   stream&expiry=600
   ```

   b. For the URL verification to work correctly, it is important that the NetWitness Platform UI server can access the Log Collector’s REST API port (50101). This can be tested by going to the NetWitness Platform UI server via SSH. Once there, run the following command:

- SSL enabled: `curl -vk https://`***`<ip of log collector>`***`:50101`
- SSL not enabled: `curl -v http://`***`<ip of log collector>`***`:50101`

If the `curl` command does not connect then there may be a network firewall or routing issue between the NetWitness Platform UI server and the Log Collector.

Example of a bad connection:

```
* About to connect() to 192.168.99.66 port 50105 (#0)
* Trying 192.168.99.66... No route to host
* couldn't connect to host
* Closing connection #0
curl: (7) couldn't connect to host
```

Example of a good connection:

```
* About to connect() to 192.168.99.66 port 50105 (#0)
* Trying 192.168.99.66... connected
* Connected to 192.168.99.66 (192.168.99.66) port 50105 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7
NSS/3.19.1 Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 192.168.99.66:50105
> Accept: */*
>
< HTTP/1.1 401 Unauthorized
< Content-Length: 71
< Connection: Keep-Alive
< Pragma: no-cache
< Expires: -1
< Cache-Control: no-cache, no-store, must-revalidate
< WWW-Authenticate: Basic realm="NetWitness"
< Content-Type: text/xml; charset=utf-8
<
<?xml version="1.0" encoding="utf-8"?>
<error>401 Unauthorized</error>
* Connection #0 to host 192.168.99.66 left intact
* Closing connection #0
```

8. The REST API requires a username and password when attempting to pull the `identity_deploy.csv` file from the Log Collector. This can be any username and password that is available on the service itself. For more information, see the "Services Security View" topic in the *Hosts and Services Guide*.

   To see which accounts are available, go to **ADMIN > Services > <log collector being setup> > Actions > View > Security**.

   Under the Users table, you see all the users that can be used in this step. It is suggested that a separate user account is created specifically for this setup, and is used nowhere else in the environment, for added security. For details, see "Add a User and Assign a Role" in the *System Security and User Management Guide*. (Go to the Master Table of Contents to find all RSA NetWitness Platform 11.x documents.)

9. To define the recurrence interval, do one of the following:

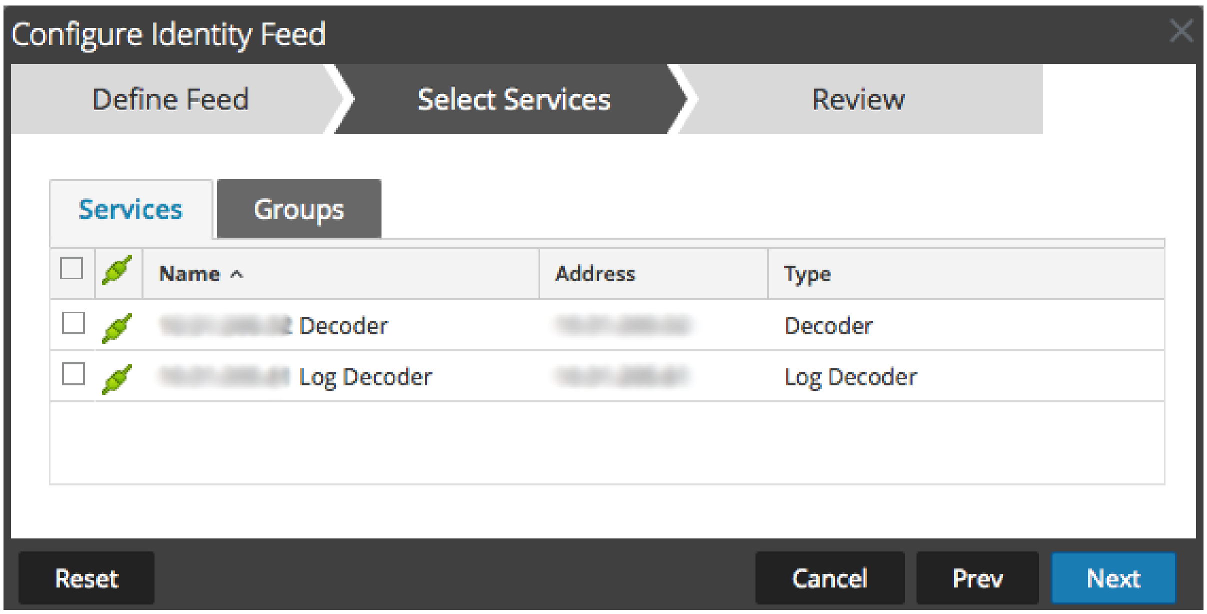
- Specify the number of minutes, hours, or days between recurrences of the feed.
- Enter the date range for the execution of the feed to recur, specify the **Start Date** and time and the **End Date** and time.

10. If using SSL encryption, you need to install the REST API SSL certificate for the Log Collector into the NetWitness Platform UI server. For more information, see Import the SSL Certificate.

    If, after importing the SSL certificate, the verification of the URL still fails, see Cannot Verify Identity Feed URL.

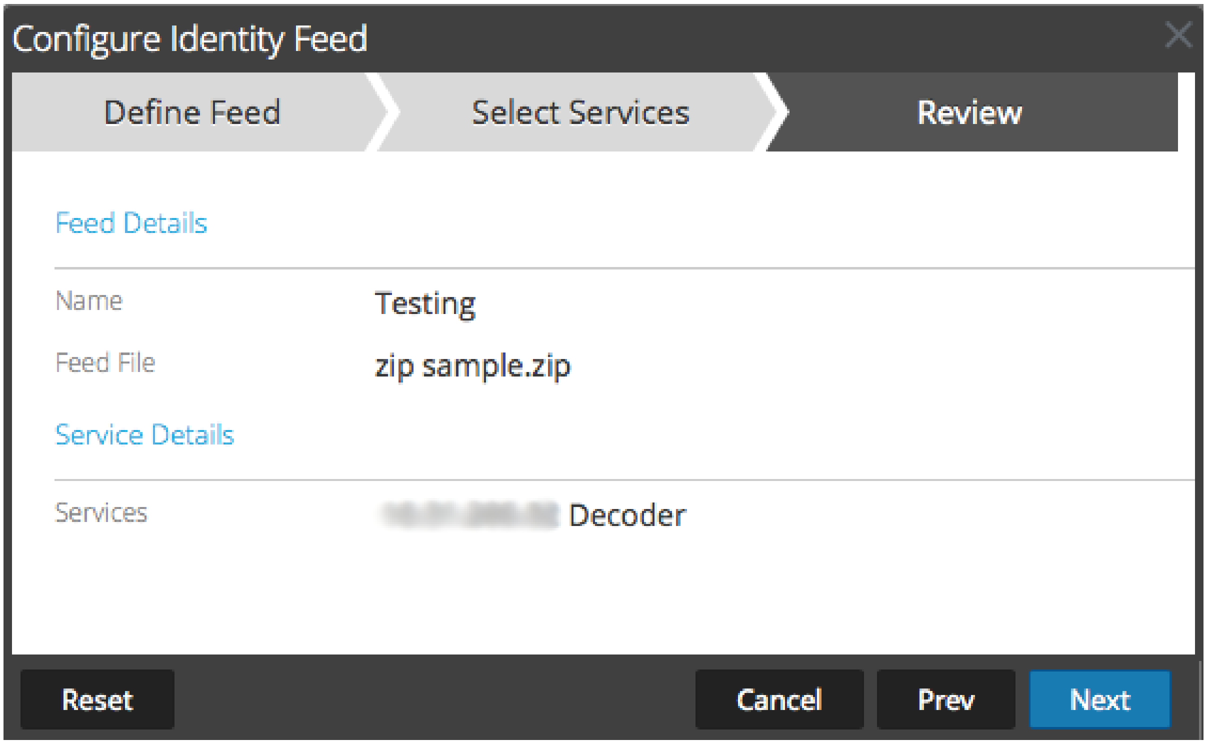
11. Click **Verify** to verify your identity feed configuration before you proceed to the Select Services dialog.
12. Click **Next**.

    The Select Services dialog is displayed.



13. To identify services on which to deploy the feed, select one or more Decoders and Log Decoders and click **Next**.
14. Click the **Groups** tab, select a group, and click **Next**.

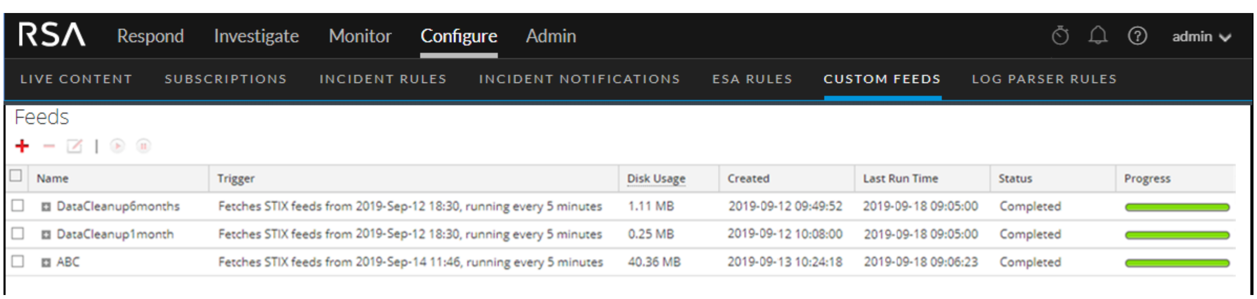
    The Review dialog is displayed.



> **Note:** If a group of devices with Decoders and Log Decoders is used to create recurring or custom feeds and this group is deleted, you can edit the feed and add a new group to the feed.

15. Anytime before you click **Finish**, you can:
    - Click **Cancel** to close the wizard without saving your feed definition.
    - Click **Reset** to clear the data in the wizard.
    - Click **Next** to display the next form (if not viewing the last form).
    - Click **Prev** to display the previous form (if not viewing the first form).
16. Review the feed information, and if correct, click **Finish**.

Upon successful creation of the feed definition file, the Create Feed wizard closes, and the feed and corresponding token file are listed in the Feed grid and progress bar tracks completion. You can expand or collapse the entry to see how many services are included, and which services were successful.



Import the SSL Certificate

If SSL is configured on the Identity feed's Log Collector, follow these steps to import the Log Collector's SSL certificate into the NetWitness Platform UI server key store. If this certificate is not imported, the NetWitness Platform UI server will be unable to pull the Identify feed file from the Log Collector.

1. To pull the SSL certificate off the Log Collector, SSH into the NetWitness Platform UI server and run the following command:

   ```
   echo -n | openssl s_client -connect <HOST>:<PORT> | sed -ne '/-BEGIN
   CERTIFICATE-/,/-END CERTIFICATE-/p' > /tmp/<SERVERNAME>.cert
   ```

   This command saves the SSL certificate to `/tmp/<SERVERNAME>.cert`. For example:

   ```
   echo -n | openssl s_client -connect 192.168.99.66:50101 | sed -ne '/-BEGIN
   CERTIFICATE-/,/-END CERTIFICATE-/p' > /tmp/logcollector.cert
   ```

2. To import the SSL certificate into the NetWitness Platform UI server, SSH into the UI server and run the following command:

   ```
   keytool -importcert -alias <name an alias for the cert> -file <the cert
   file pathname> -keystore /etc/pki/java/cacerts
   ```

   For example:

   ```
   keytool -importcert -alias logcollector01 -file /tmp/logcollector.cert -
   keystore /etc/pki/java/cacerts
   ```

3. The system requests a password. Enter the password for the keystore on the NetWitness Platform UI server, not for the jetty keystore. The default password is **changeit**.
4. Restart **jettysrv** to allow jetty to read the new certificate in the store.

Cannot Verify Identity Feed URL

If the Identity feed URL cannot be verified, and you are using SSL, make sure you followed the steps in Import the SSL Certificate.

If there are issues, it is possible that the internal name of the certificate does not match the hostname of the Log Collector. The following procedure checks this.

1. SSH to the NetWitness Platform UI server.
2. Run the following command to output the CN name of the SSL cert:

   ```
   echo -n | openssl s_client -connect <log decoder>:50101 | sed -ne '/BEGIN
   CERTIFICATE-/,/-END CERTIFICATE-/p'
   ```

   For example:

   ```
   echo -n | openssl s_client -connect salogdecoder01:50101 | sed -ne '/BEGIN
   CERTIFICATE-/,/-END CERTIFICATE-/p'
   ```

3. Retrieve the CN name of the SSL certificate.

```
depth=0 C = US, CN = NetWitness-SAlogdecoder01
verify error:num=18:self signed certificate
verify return:1
depth=0 C = US, CN = NetWitness-SAlogdecoder01
verify return:1
-----BEGIN CERTIFICATE-----
MIIC2zCCAcOgAwIBAgIBADANBgkqhkiG9w0BAQQFADAxMQswCQYDVQQGEwJVUzEi
MCAGA1UEAxMZTmV0V2l0bmVzcy1TQWxvZ2RlY29kZXIwMTAeFw0xNDAxMTEwMDM1
```

4. Edit the `/etc/hosts` file and add the IP address and CN name to the file.

```
# Created by NetWitness Installer on Fri Jan 10 21:42:10 UTC 2014
127.0.0.1 SAserver01 localhost.localdom localhost
::1 SAserver01 localhost.localdom localhost ip6-localhost ip6-loopback

192.168.10.23 NetWitness-SAlogdecoder01
```

5. Restart the network service on the appliance.
6. Confirm that the name placed in the **/etc/hosts** file is used instead of the FQDN or IP address in the Identity feed URL.
7. Re-verify the Identity feed URL.

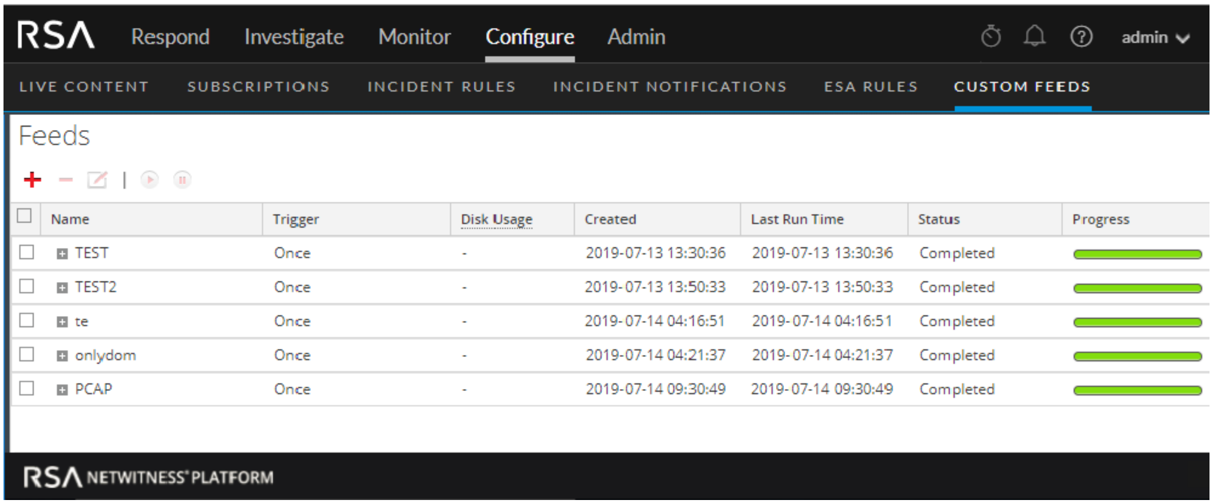
Edit, Upload, or Remove a Feed

You can upload a feed, edit an existing feed, or remove a feed.

To edit an existing feed:

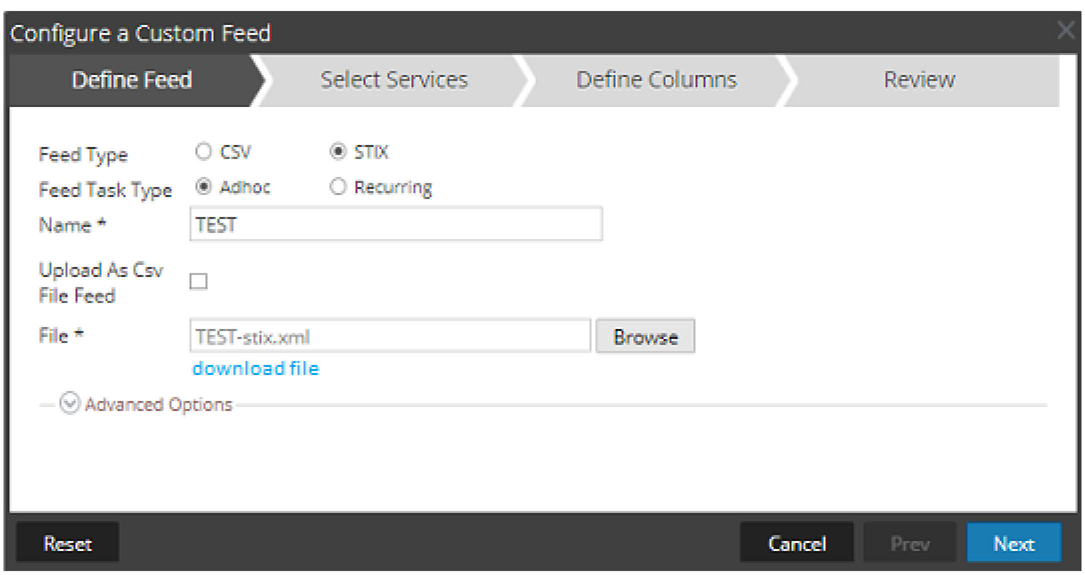
1. Go to **CONFIGURE > Custom Feeds**.

   The Feeds view is displayed.



2. In the toolbar, select a feed and click .

   The Configure Custom Feed or Configure Identity Feed panel opens in the Custom Feed wizard.



3. If you want to edit the feed file:

a. Click **download file**.

   For an Identity feed, the .zip file is downloaded. For a custom feed, the .csv or .xml file is downloaded to your local file system. For a STIX feed, the .xml file is downloaded to your local file system.

b. Edit and save the file.

c. In the **Define Feed** tab, browse for and open the edited file.

4. Edit any other parameters in the **Define Feed** tab, **Select Services** tab, and **Define Columns** tab that apply to the type of feed.
5. Anytime before you click **Finish**, you can:
   - Click **Cancel** to close the wizard without saving your changes.
   - Click **Reset** to clear the data in the wizard.
   - Click **Next** to display the next form (if not viewing the last form).
   - Click **Prev** to display the previous form (if not viewing the first form).
6. In the **Review** tab, review the feed information, and if correct, click **Finish**.

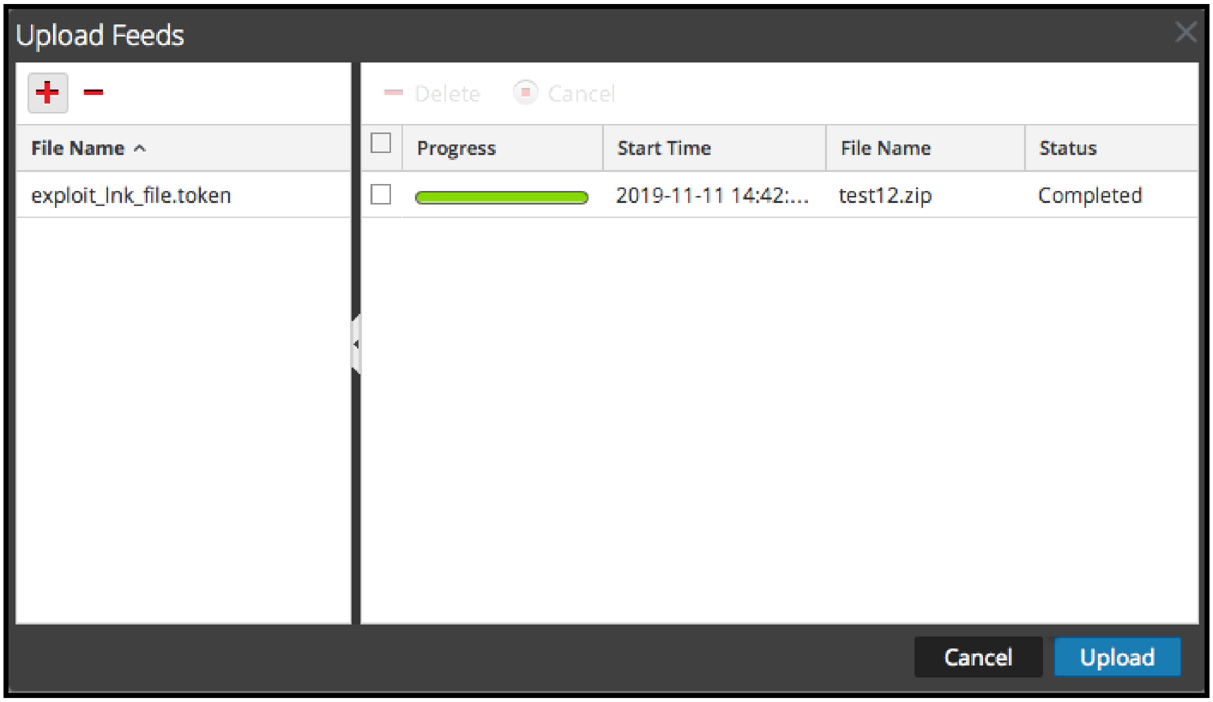
The feed is recreated with the updated file and new feed specifications. The feed is added to the Feeds list and progress bar tracks completion. Upon successful creation of the feed definition file, the Create Feed wizard closes, and the feed and corresponding token file is listed in the Feeds list. You can expand or collapse the entry to see how many services are included, and which services are successful.

To upload a feed to a Decoder or Log Decoder:

1. Go to **Admin** > **Services**.
2. Select a Decoder or Log Decoder service and click > **View** > **Config**.

   The Services Config view is displayed with the General tab open.
3. Select the **Feeds** tab.
4. In the Feeds tab toolbar, click Upload.

   The Upload Feeds dialog is displayed.



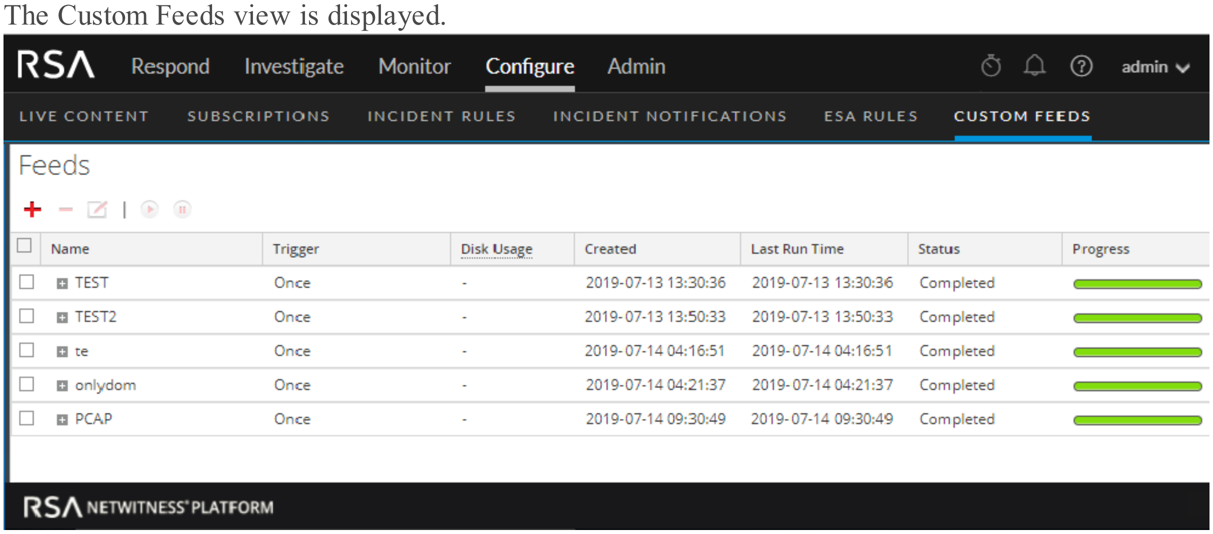
5. In the **File** grid, click and select a feed file. Supported files are *.feed, *.token, and *.filter.
6. Select the feed file from the **File** list and click **Upload**.

   The Upload Job list is updated to show the progress and status of the uploaded feed.

To remove a feed:

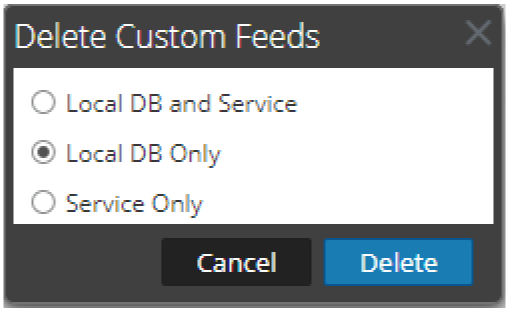
1. Go to **CONFIGURE > Custom Feeds**.

   The Custom Feeds view is displayed.



2. In the toolbar, select a feed and click .

   The Delete Custom Feeds dialog is displayed.



You can select one of the following options to delete the feed:

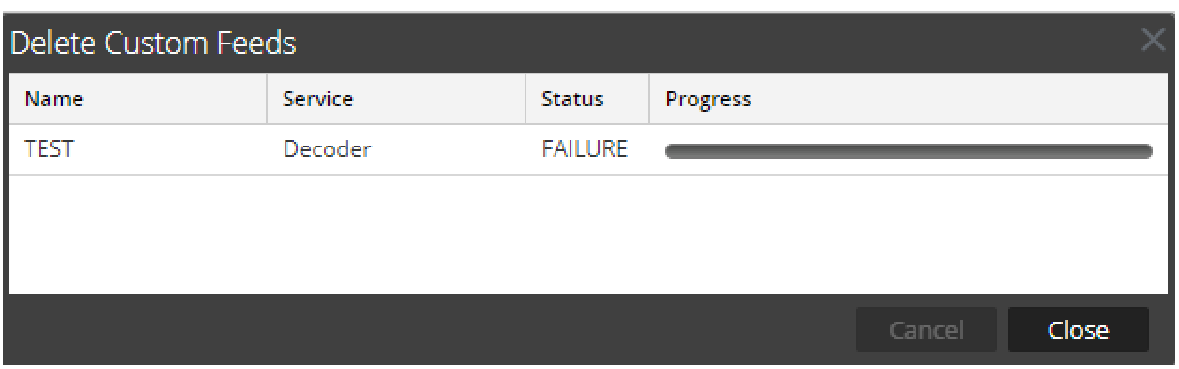
- If you choose to delete the feed from **Local DB and Service**, the feed is deleted from both the service and the local NetWitness Platform box. The deleted feed will no longer seen on the NetWitness Platform user interface.
- If you choose to delete the feed from **Local DB Only**, the feed is deleted from the local NetWitness Platform box. The deleted feed will not be seen on the NetWitness Platform user interface; however, the last deployed version of the feeds will be present on the service. The undeployed feeds will be deleted forever.
- If you choose to delete the feed from **Service Only**, the feed is deleted from the service. The deleted feed will appear on the NetWitness Platform user interface and can be deployed again

3. Select where you want to delete the feed and click **Delete**.

   A warning dialog is displayed.

4. Click **yes** to confirm that you want to delete the feed from the select areas.

   - If you chose to delete the feed from the **Local DB Only**, the feed is deleted.
   - If you chose to delete the feed from the **Local DB and Service** or **Service Only**, the Delete Custom Feeds view is displayed showing the progress of the deletion on the service.



Create Custom Meta Keys Using a Custom Feed

This topic provides information on how to add custom meta keys using a custom feed in the Log Decoder, and highlights the configuration changes to reflect the custom meta keys in the Concentrator, ESA, Archiver, Warehouse Connector, and Reporting Engine schema. You can create custom meta keys to retrieve data, to investigate and analyze the logs and packets. Custom meta keys enable you to add an enrichment context for the log and packet data.

Here is an example of creating a custom meta key in the Log Decoder. In this scenario, an organization wants to track the location of an asset such as a printer. So, a custom meta key **source location** is introduced, which indicates the location of the asset, for example Printer1, which is located in the 'Fifth Floor A wing'.

> **Note:** Custom meta keys can be created in the Decoder as well. Select the `index-decoder-custom.xml` file when you create a custom meta key in the Decoder.

Add a Custom Meta Key in the Log Decoder

To add custom meta keys using custom feed:

1. Go to **ADMIN** > **Services** .
2. Select a Log Decoder service and click > **View** > **Config** > **Files** tab > **index-logdecoder-custom.xml.**

```
<Language>
  <?xml version="1.0" encoding="utf-8"?>
  <Language level="IndexNone" defaultAction="Auto">
  <!-- Reserved Meta key for Feed -->
  <Key description="Source Location" level="IndexNone"
name="location.src" format="Text"/>
 </Language>
 name = Name of the key (max is 16 chars)
```

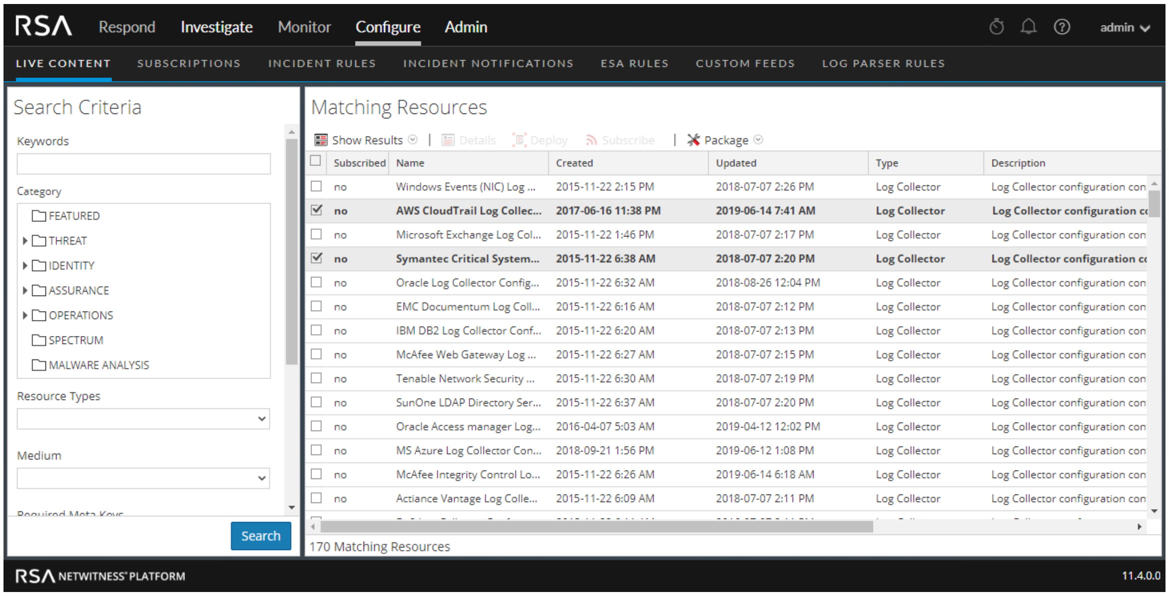
3. Restart the Log Decoder service. In the Services view, click > **Restart**.

Deploy a Log Decoder Feed in Live

To deploy the feed in the live environment:

1. Go to **CONFIGURE > Live Content.**
2. Select a group of resources, or a previously created resource package. To select a resource or group of resources:
   a. In the **Live Search View,** browse Live resources (for example, search for the **Log Collector** resource Type).
   b. In the **Matching Resources** panel, select **Show Results > Grid**.

c. Select the checkbox to the left of the resources that you want to deploy.

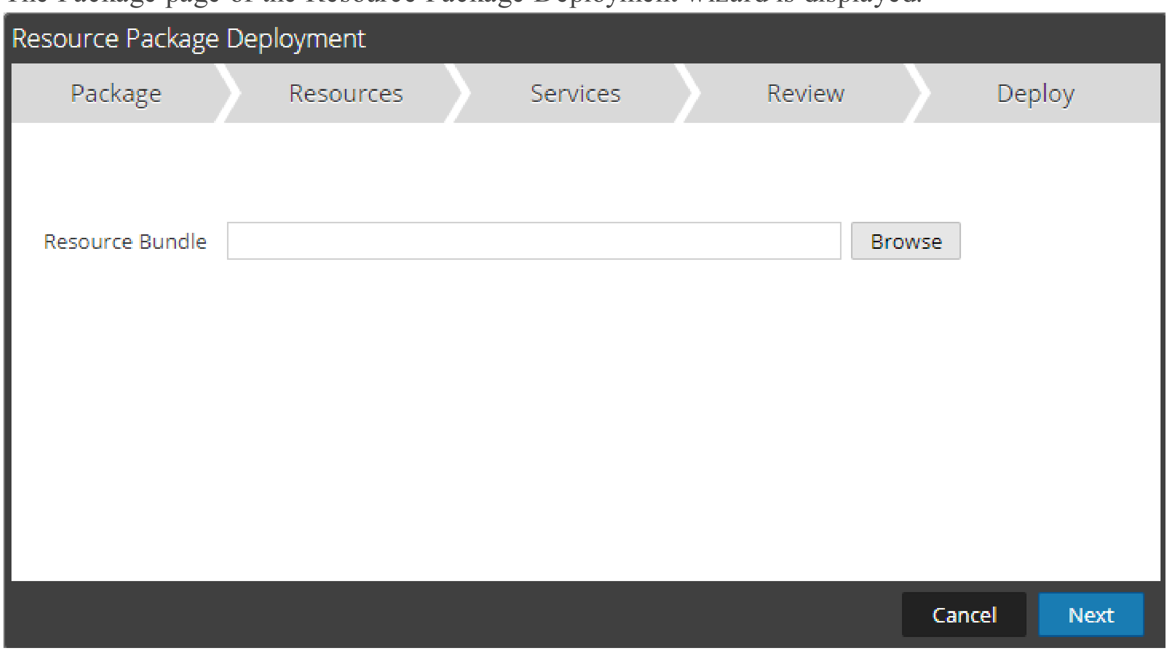


d. In the Matching Resources toolbar, click 

.

3. To select a resource package to deploy:

a. In the **Live Search** view - **Matching Resources** toolbar, select **Package** > **Deploy** :
The Package page of the Resource Package Deployment wizard is displayed.



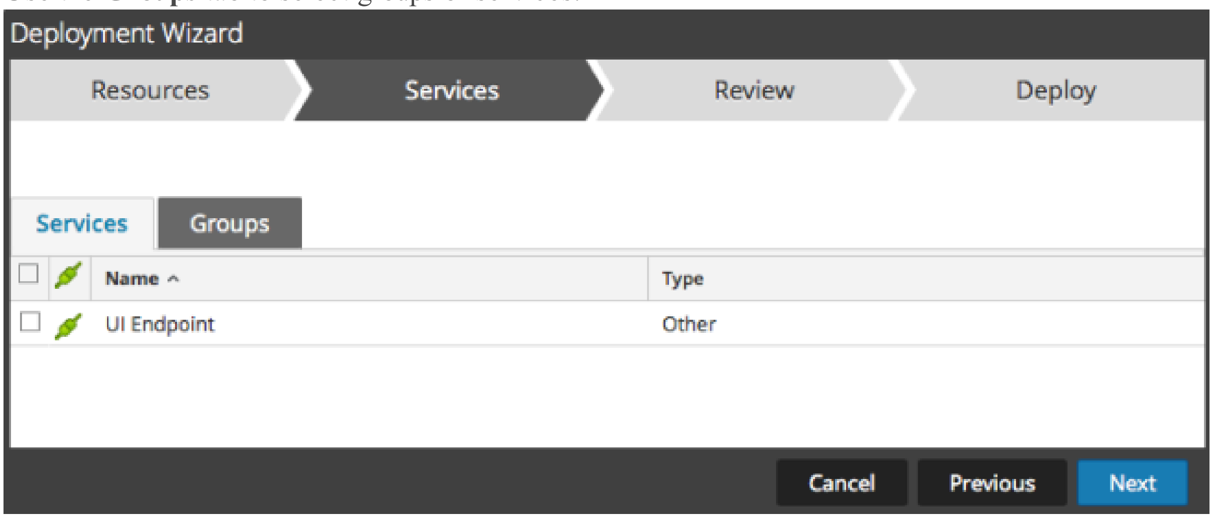
b. Click **Browse** and select a package from your network.

c. Click **Open.**
At this point, whether you are deploying a package or a group of resources, the Deployment Wizard opens, and the Resources page is displayed.

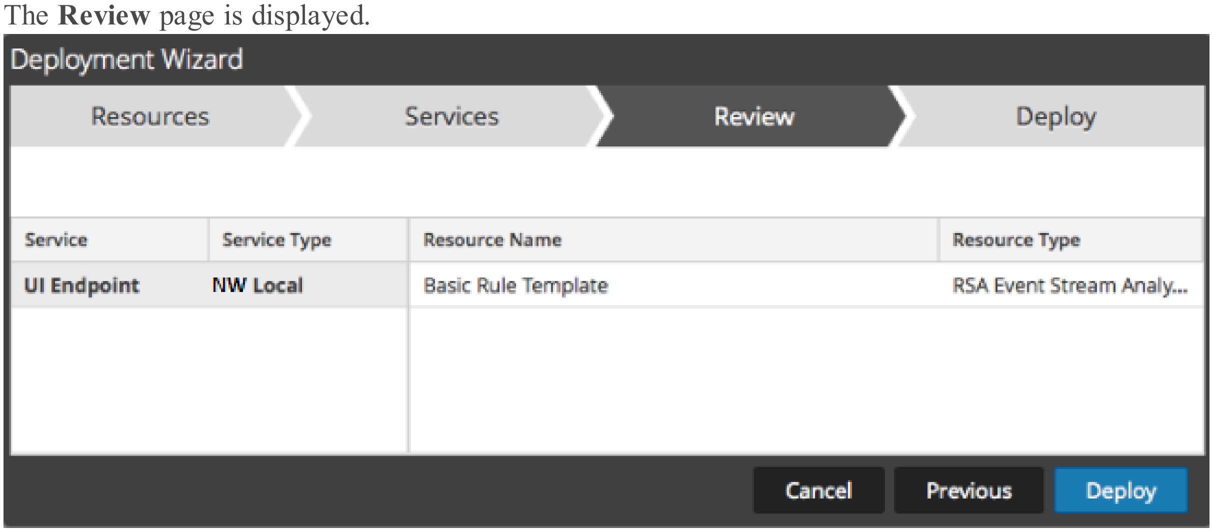
3. Click **Next**.
   The **Services** page is displayed. It has two tabs, **Services** and **Groups**, which provide a list of services and service groups that are configured in the Admin > Services view. The columns are a subset of the columns available in the Services view.

   Note: The Live server is "smart" about deploying resources to Services. For example, it does not deploy resources that have a Medium of packets to any Log Decoders. This means that only applicable content resources are deployed to each Service.

4. Select the services to which you want to deploy the content. You can select any combination of services and service groups.
   Use the **Services** tab to select individual services, list of services and service groups that are configured in the Admin Services view.
   Use the **Groups** tab to select groups of services.

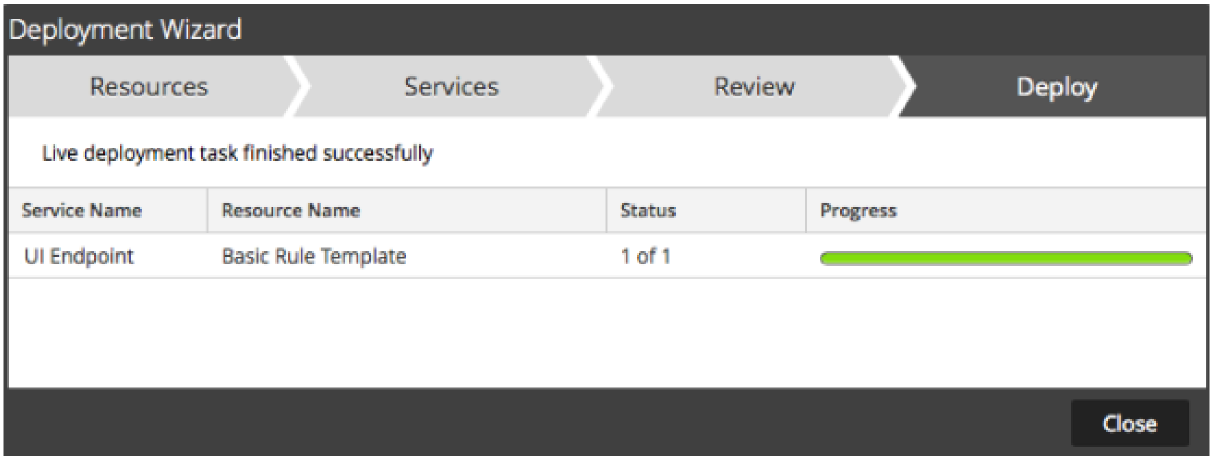


5. Click **Next**.
   The **Review** page is displayed.

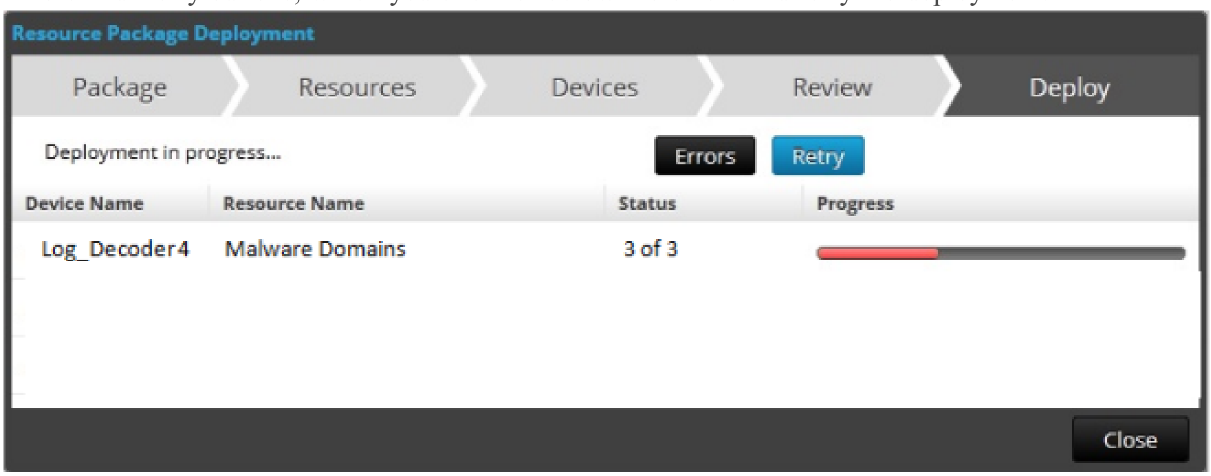


   Make sure that you have selected correct resources and the services to which you want to deploy them.

6. Click **Deploy**.
   The **Deploy** page is displayed. The Progress bar turns green when you have successfully deployed the resources to the selected services.



   If you try to deploy resources and services that are not compatible, NetWitness Platform displays the Errors and Retry buttons, which you click to review the errors and retry the deployment.



7. Click **Close**.

> **Note:** The Source IP should be indexed by selecting the type as 'IP' as the ip.src. and ip.dst are in IPv4 format.

In this scenario, a custom meta key location.src (location source) is added by indexing the hostname (alias.host). In this example, the printer hostname are populated in meta key 'alias.host'. Select **alias.host** as callback key, and index type as 'Non IP' in the Feed Wizard as shown below. In the Define Values section, select the custom meta key from the drop down menu.

Add the Custom Meta Key Entry in the Concentrator Custom Index file

To add the custom meta key entry in the Concentrator custom index file:

1. Go to **ADMIN** > **Services** > **Concentrator**.
2. Click > **View** > **Config** > **Files** tab > **index-concentrator-custom.xml.**
3. Add the custom meta key entry in the Concentrator index file.

```
 <Language>
  <?xml version="1.0" encoding="utf-8"?>
  <Language level="IndexNone" defaultAction="Auto">
   <!-- Reserved Meta key for Feed -->
   <Key description="Source Location"  level="IndexValues"
name="location.src" format="Text"  valueMax="10000"
defaultAction="Open"/>
  </Language>
```

4. To restart the Concentrator service, in the Services view, click > **Restart**.

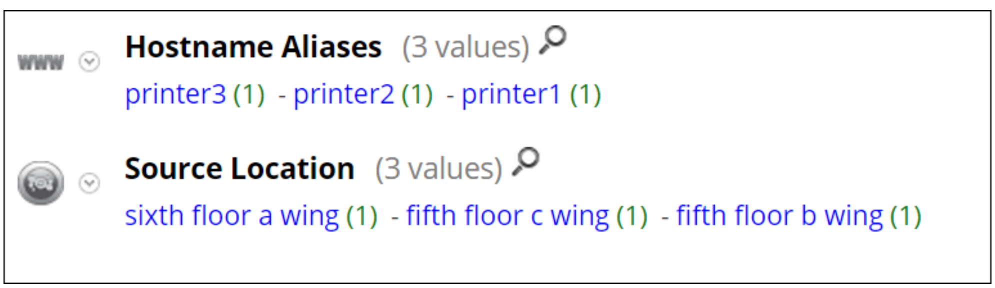
Note: In case of the Broker, the Broker derives its index from the Concentrator from which it aggregates. So you do not need to create custom meta in the Broker. If you have not indexed the meta key in the Concentrator, the Broker does not display the meta key in Investigate.

Investigate a Custom Meta Key

Note: You have to log out and log back in to the NetWitness Platform User Interface in order to view the custom meta key in Investigate.

To investigate a custom meta key:

1. Go to **Investigate**.
   The Investigate dialog, which provides services, is displayed.
2. Select a Concentrator service, and click **Navigate**.



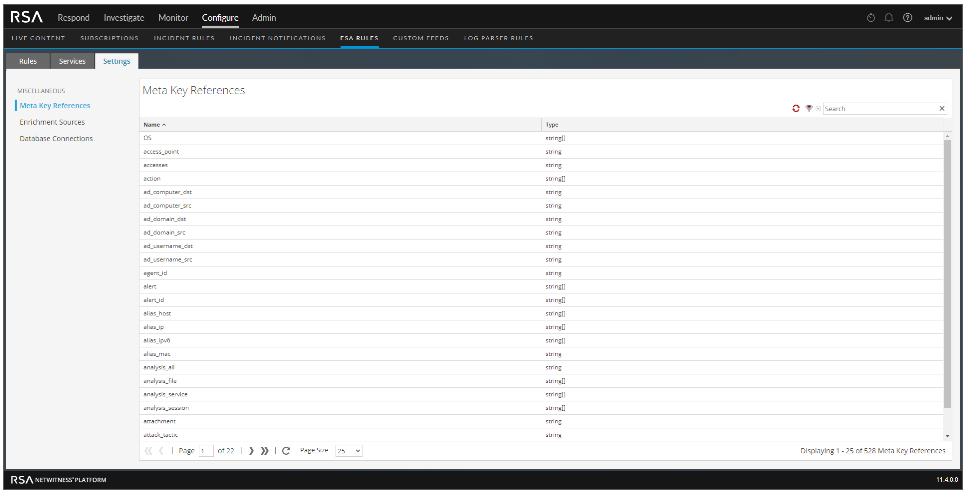
Additional Procedures

The following procedures must be executed if you have Warehouse Connector, Archiver, Reporting Engine, and ESA configured.

Verify the Custom Meta Keys on ESA

After you add custom meta keys on the Concentrator, you can verify that your meta keys are updated on ESA.

1. Go to **Configure** > **ESA RULES** > **Settings** tab.



2. In the Meta Key References, click the Meta Re-Sync (Refresh) icon ( ).

3. Verify that the custom meta keys appear on ESA. If you do not see the meta keys, you may need to restart the Concentrator.

Update the Schema in Archiver

If you want to configure the Archiver, using the new custom meta keys, you need to update the Archiver schema in the Reporting Engine. To update the Archiver schema in Reporting Engine:

1. Go to **ADMIN** > **Services** > **Archiver**.

2. Select > **View** > **Config** > **Files** > **index-archiver-custom.xml**.

3. Add the custom meta key entry in the Archiver index file.

```
<Language>
  <?xml version="1.0" encoding="utf-8"?>
  <Language level="IndexNone" defaultAction="Auto">
  <!-- Reserved Meta key for Feed -->
  <Key description="Source Location" level="IndexValues"
name="location.src" format="Text"
  valueMax="10000" defaultAction="Open"/>
 </Language>
```

4. To restart the Archiver service, click > **Restart.**
   The Archiver schema is updated with the custom meta key.

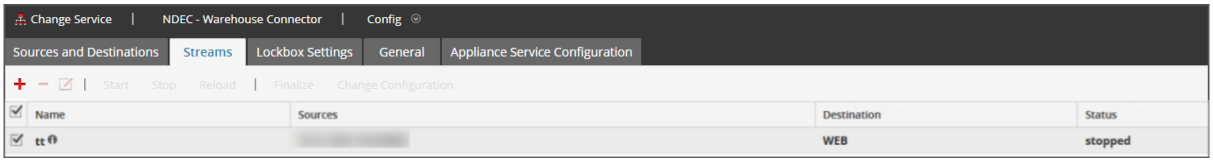
Update the Schema in Warehouse Connector

If you want to configure the Warehouse Connector with custom metadata and use it in a Warehouse Connector report then you need to update the Warehouse Connector schema in the Reporting Engine.

If the Log Decoder or Decoder, where the custom meta key is added, is one of the sources in the Warehouse Connector stream, you need to update the schema in the Warehouse Connector.

To update the Warehouse Connector schema in the Reporting Engine:

1. Go to **ADMIN** > **Services** > **Warehouse Connector**.
2. Click > **View** > **Config**.
   The Services Config view of Warehouse Connector is displayed.
3. Click the **Streams** tab.
4. Select the stream and then click **Reload**.
   The Warehouse Connector pulls the schema from the downstream devices (Log Decoder/Decoder).



For more information on streams, see "Configure Streams" in the *Warehouse Connector Configuration Guide*.

Update the Schema in Reporting Engine

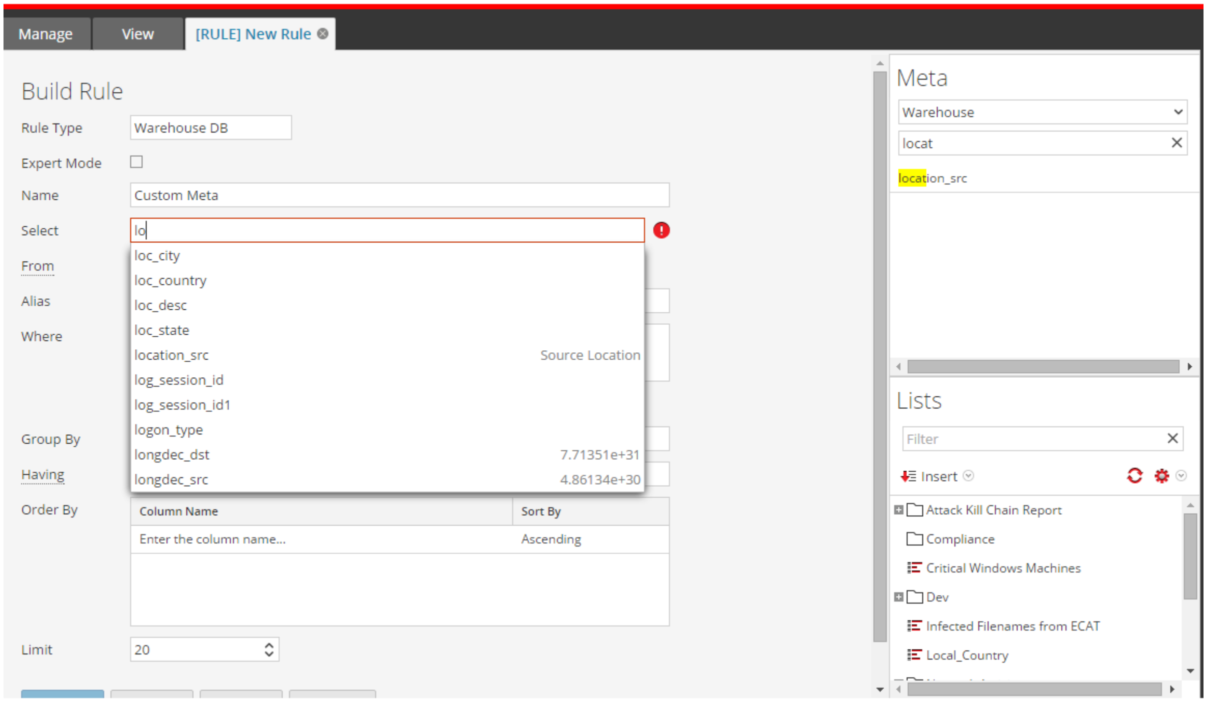
To update the schema in Reporting Engine:

1. Go to **ADMIN** > **Services** > **Reporting Engine.**
2. Click > **Restart.**

Note: Restart the Reporting Engine or wait for thirty minutes for the schema to be updated.

To view the custom meta key:

1. Navigate to **Monitor** > **Reports** > **Rules**.
2. In the toolbar, click .
3. Select **Warehouse DB**.
4. In the **Build Rule** tab, search for the custom meta from the right panel.
   The custom meta key is displayed.


Manage
View
[RULE] New Rule
Build Rule
Rule Type
Warehouse DB
Expert Mode
Name
Custom Meta
Select
lo
loc_city
loc_country
loc_desc
loc_state
location_src
Source Location
log_session_id
log_session_id1
logon_type
longdec_dst
7.71351e+31
longdec_src
4.86134e+30
From
Alias
Where
Group By
Having
Order By
Column Name
Sort By
Enter the column name...
Ascending
Limit
20
Meta
Warehouse
locat
location_src
Lists
Filter
Insert
Attack Kill Chain Report
Compliance
Critical Windows Machines
Dev
Infected Filenames from ECAT
Local_Country

Decoder and Log Decoder Additional Procedures

This topic explains the additional procedures an administrator could choose to follow which are not essential for the configuration of the Decoder or Log Decoder.

Topics

Configure 10G Capability

This topic guides administrators in how to tune a Network Decoder specifically for high speed packet capture using NetWitness Platform 11.x. This applies when capturing packets on a 10G interface card.

Caution: Packet capture at high speeds requires careful configuration and pushes the Decoder hardware to its limits, so please read this entire topic when implementing a 10G capture solution.

Note: Enabling intra-session for HTTP pipelining is not supported when capturing at 10G rates on a single Decoder, as the HTTP_Lua parser required to function can cause dropped packets at that rate of ingest. In this case, the load should be spread across multiple Concentrator-Decoder pairs.

RSA NetWitness Platform provides support for high-speed collection on the Decoder. You can capture network packet data from higher speed networks and optimize your Network Decoder to capture network traffic up to 8Gb/sec sustained and 10Gb/sec burst, depending on which parsers and feeds you have enabled.

Enhancements that facilitate capture in these environments include the following:

- Utilization of the `pf_ring` capture driver capability to leverage the commodity 10G Intel NIC card for high-speed capture.
- Introduction of `assembler.parse.valve` configuration, which automatically disables application parsers when certain thresholds are exceeded, to limit risk of packet loss. When the application parsers are disabled, network layer parsers are still active. When stats fall below exceeded thresholds, application parsers are automatically re-enabled.
- Utilization of Berkeley Packet Filters (BPF) in 10G environments. For information about using BPF, see "(Optional) Configure System-Level (BPF) Packet Filtering" in Configure Capture Settings.

Hardware Prerequisites

- A Series 4S, Series 5 or Series 6 Decoder
- An Intel 82599-based ethernet card, such as the Intel x520. All RSA-provided 10G cards meet this requirement. Two examples are:
  - All SMC-10GE cards provided by RSA.
  - A Dell Network Daughter Card using an Intel controller to provide 10G network interfaces. This is included in all Series 5 hardware.
- For the Series 4S / Dell R620 only: 96 GB of DD3-1600 memory in **dual-rank** DIMMs. Single-rank DIMMs may decrease performance by as much as 10%. To determine the speed and rank of the installed DIMMs, run this command:
  `dmidecode -t 17.`
- Sufficiently large and fast storage to meet the capture requirement. Storage considerations are covered later in this topic.
- Each Network Decoder configured with a minimum of 2 DACs or SAN connectivity.

Software Prerequisites

- Dell R620-based systems, such as the Series 4S, must have their BIOS updated to v1.2.6 or later.
- The 10G Decoder capability is only supported on RSA-provided Decoder Installation images. All required software is installed by default.
- If upgrading from a previous release, perform the upgrade first before proceeding with configuration

Install the 10G Decoder

Note: You can skip to "Configure the 10G Decoder" if you are starting with new Series 5 or Series 6 hardware.

Perform the following steps to install the NetWitness 10G Decoder:

Download and Update the BIOS

Note: BIOS revisions earlier than v1.2.6 have issues properly identifying the location of the 10G capture card within the system. It is recommended that customers update to the latest v2.2.3 BIOS, but is not required for 10G if they are running v1.2.6 or later.

1. Download BIOS v2.2.3 from the following location:

   http://www.dell.com/support/home/us/en/19/Drivers/DriversDetails?driverId=V7P04

2. Download the Update Package for the Red Hat Linux file.
3. Copy the file to the NetWitness server.
4. Login as `root`.
5. Change the permissions on the file to execute.
6. Run the following file:

   ```
   ./BIOS_V7P04_LN_2.2.3.BIN
   ```

7. Reboot the system when execution is complete and a reboot is requested.

Note: The BIOS installation procedure takes approximately 10 minutes.

Locate the 10G Decoder Packages

The packages required to configure the 10G Decoder should already be present on the Decoder installation image. You should not have to install any additional packages.

Verify 10G Decoder Packages Are Installed

Installation of the 10G Decoder packages is handled automatically. Therefore, there should be no action to enable the 10G functionality.

- If you upgraded the kernel packages as part of an upgrade, a reboot is required. The operating system will recompile and install the drivers for the upgraded kernel.
- You can verify that the installation was successful if you see additional `PFRINGZC` interfaces available when selecting the Capture Port Adaper as described below.

Configure the 10G Decoder

Perform the following steps to configure the 10G Decoder:

1. From the **Decoder Explore** view, right-click **Decoder** and select **Properties**.
2. In the properties drop-down menu, select **reconfig** and enter the following parameters:
   `update=1 op=10g`
   This adjusts the Decoder packet processing pipeline to allow for higher raw data throughput, but less parsing ability.
3. From the **Decoder Explore** view, right-click **database** and select **Properties**.
4. In the **Properties** drop-down menu, select **reconfig** and enter the following parameters:
   `update=1 op=10g`
   These parameters adjust the packet database to use very large file sizes and Direct I/O.
5. To capture traffic on a single port,
   a. Go to **ADMIN**>**Services**, select the target Network Decoder type and click  >**View**>**Config**.
   b. In Decoder Configuration section, in **Capture Interface Selected** field, select the interface. For example, `PFRINGZC,p1p1` or `PFRINGZC,p1p2`.
6. To capture traffic on two ports simultaneously on the same 10G card,
   a. Go to **ADMIN**>**Services**, select the target Network Decoder type and click  >**View**>**Config**.
   b. In the Decoder Configuration section, select the first interface in **Capture Interface Selected** field. For example, `PFRINGZC,p1p1` or `PFRINGZC,p1p2`.
   c. Select **Explore** from the drop-down menu and click `decoder> config` node. In the **capture.device.params** field, enter the second interface variable that used on this 10G card for simultaneous capture. For example, `device=zc:p1p2,zc:p1p1`
7. If the write thread is having trouble sustaining the capture speed, you can try the following:

   Change `/datebase/config/packet.integrity.flush` to `normal`.

   Note: You can adjust the `packet.file.size` to a higher value, but keep the file size under 10 GB, as the whole file is buffered in memory.
8. (Optional) Application parsing is extremely CPU intensive and can cause the Decoder to drop packets. To mitigate application parsing-induced drops, you can set `/decoder/config/assembler.parse.valve` to `true`. These are the results:

- When session parsing becomes a bottleneck, application parsers (HTTP, SMTP, FTP, and others) are temporarily disabled.
- Sessions are not dropped when the application parsers are disabled, just the fidelity of the parsing performed on those sessions.
- Sessions parsed when the application parsers are disabled still have associated network meta (from the network parser).
- The statistic `/decoder/parsers/stats/blowoff.count` displays the count of all sessions that bypassed application parsers (network parsing is still performed).
- When session parsing is no longer a potential bottleneck, the application parsers are automatically re-enabled.
- The assembler session pool should be large enough that it is not forcing sessions.
- You can determine if sessions are being forced by the statistic `/decoder/stats/assembler.sessions.forced` (it will be increasing). Also `/decoder/stats/assembler.sessions` will be within several hundred of `/decoder/config/assembler.session.pool`.

9. (Optional) If you need to adjust the MTU for capture, add the `snaplen` parameter to `capture.device.params`. Unlike previous releases, the `snaplen` does not need to be rounded up to any specific boundary. The Decoder automatically adjusts the MTU set on the capture interfaces.
10. The following configuration parameters are deprecated and no longer necessary
    - The `core= parameter` in `capture.device.params`
    - Any configuration files under `/etc/pf_ring` directory

> **Note:** An Ethernet device installed post imaging does not require any configuration for use as a capture device. It does require configuration if it is used as a network interface, or for system tools to access it without manual configuration.

> **Note:** For version 11.4.0.1, all the ports configured as mentioned in Step 6 (device=zc:p1p2,zc:p1p1) must be up and running - p1p2, p1p1. If one of the port is down, then perform one of the following operations, depending on your environment.
> **Work Around 1:** (Recommended)
> This work-around must be done after every restart on the port which is down. If the Decoder doe not require a frequent restart, this workaround can be performed.
> If p1p1 is up and p1p2 is down, you need to manually make the p1p2 port up and running.
> 1. SSH to the Decoder.
> 2. Run the following command.
> `ifup p1p2`
> #`p1p2` is the port name. If the port name is em4, then the command must be `ifup em4`.

> **Note:**
> **Work Around 2:**
> This requires editing of configuration file. Make sure you do not edit anything other the specified configuration. Backup the file before editing. This backup file can be used to restore, if the system shows any unusual behavior.
> If p1p1 is up and p1p2 is down and not coming up automatically on every restart,
> 1. SSH to the Decoder.
> 2. In the configuration file `/etc/sysconfig/network-scripts/ifconfig-<p1p2>`, change the value of `ONBOOT=no` to `ONBOOT=yes`.
> 3. Run the following command.
> `service network restart`
> If you are upgrading to any version, you must revert the configuration file to its original configuration before upgrading.

Typical Configuration Parameters

Typical configuration parameters are listed below. Actual parameters may vary depending on the amount of memory and CPU resources available.

1. Session and packet pool settings (under `/decoder/config`):
   - `pool.packet.pages` = `1000000`
   - `pool.session.pages` = `300000`
2. Packet write block size under (`/database/config/packet.write.block` size) set to `filesize`.

   > **Note:** This configures the Decoder to buffer the file with huge pages and write using direct I/O for maximum performance.

3. Parse Thread Count (under `/decoder/config`).

   `parse.threads =12`

Storage Considerations

When capturing at 10G line rates, the storage system holding the packet and meta databases must be capable of sustained write throughput of 1400 MBytes/s.

Using the Series 4S Hardware (With Two or More DAC Units)

The Series 4S is equipped with a hardware RAID SAS controller capable of an aggregate 48Gbit/s of I/O throughput. It is equipped with eight external 6 Gbit ports, organized into two 4-lane SAS cables. The recommended configuration for 10G is to balance at least two DAC units across these two external connectors. For example, connect one DAC to one port on SAS card, and then connect another DAC to the other port on the SAS card.

For environments with more than two DACs, chain them off each port in a balanced manner. This may require re-cabling of DACs in an existing deployment, but should not affect data that has already been captured on the Decoder.

If adding new capacity, use the currently available `NwArrayConfig` script to provision the DAC units. The script automatically adds one DAC per execution (that means, if adding three DACs, then the script must be run three times), adding the DACs to the `NwDecoder10G` configuration as separate mount points. The independent mount points are important, as this configuration allows the `NwDecoder10G` to segregate write I/O from capture from the read I/O needed to satisfy packet content requests.

Using SAN and Other Storage Configurations

The Decoder allows any storage configuration that can meet the sustained throughput requirement. The standard 8-Gbit FC link to a SAN is not sufficient to store packet data at 10G; in order to use a SAN it may be required to perform aggregation across multiple targets using a software-RAID Scheme. Thus environments using SAN are required to configure connectivity to the SAN using multiple FCs.

Capturing 10G with 40G Network Fabric

The Network Decoder is capable of physically connecting to a 40Gbps network infrastructure if you use the applicable hardware described in the *Series 6 Hardware Setup Guide*, located here: https://community.rsa.com/community/products/netwitness/hardware-setup-guides.

To support this feature without causing significant packet drops, the following conditions must be met:

- The capture rate must be throttled to 10Gbps by the network.
- The guidelines described later in this topic in Tested Live Content still apply.

Parsing and Content Considerations

Parsing raw packets at high speeds presents unique challenges. Given the high session and packet rates, parsing efficiency is paramount. A single parser that is inefficient (spends too long examining packets) can slow the whole system down to the point where packets are dropped at the card.

For initial 10G testing, start with only native parsers (except SMB/WebMail). Use the native parsers to establish baseline performance and with little to no packet drops. Do not download any Live content until this has been done and the system is proven to capture without issue at high speeds.

After the system has been operational and running smoothly, Live content should be added very slowly - especially parsers.

Best Practices

Whether you are updating a currently deployed system or deploying a new system, it is recommended you use the following best practices to minimize risk for packet loss. One caveat is if you are updating a current 10G deployment but not adding any additional traffic. For example, a current Decoder capturing off a 10G card at 2G sustained should see no difference in performance, unless part of the update also entails adding additional traffic for capture.

- Incorporate baseline parsers (except SMB/Webmail, both of which generally have high CPU utilization) and monitor to ensure little to no packet loss.
- When adding additional parsers, add only one or two parsers at a time.
- Measure performance impact of newly added content, especially during peak traffic periods.

- If drops start occurring when they did not happen before, disable all newly-added parsers and enable just one at a time and measure the impact. This helps pinpoint individual parsers causing detrimental effects on performance. It may be possible to refactor it to perform better or reduce its feature set to just what is necessary for the customer use case.
- Although lesser performance impacts, feeds should also be reviewed and added in a phased approach to help measure performance impacts.
- Application Rules also tend to have little observable impact, though again, it is best not to add a large number of rules at once without measuring the performance impact.
- If you continually get a timeout message in the Investigate > Events view, such as `The query on channel 192577 was auto-canceled by the system for exceeding time usage limits. Check timeout values. Query running time was 00:05:00 (HH:MM:SS)`, first check the query console to determine if there are issues around time it takes for a service to respond, index error messages, or other warnings that may need to be addressed to increase query response time. If there are no messages indicating any specific warnings then try increasing the Core Query Timeout from the default 5 minutes to 10 minutes as described in "View Query and Session Attributes per Role" section of the *System Security and User Management Guide*.

Finally, making the recommended configuration changes outlined in the Configuration section will help minimize potential issues.

Tested Live Content

All (not each) of the following parsers can run at 10G on the test data set used:

- MA content (7 Lua parsers, 1 feed, 1 application rule)
- 4 feeds (alert ids info, nwmalwaredomains, warning, and suspicious)
- 41 application rules
- DNS_verbose_lua (disable DNS)
- fingerprint_javascript_lua
- fingerprint_pdf_lua
- fingerprint_rar_lua
- fingerprint_rtf_lua
- MAIL_lua (disable MAIL)
- SNMP_lua (disable SNMP)
- spectrum_lua
- SSH_lua (disable SSH)
- TLS_lua
- windows_command_shell
- windows_executable

NOT TESTED:

- SMB_lua, native SMB disabled by default
- html_threat

OTHER:

- HTTP_lua reduces the capture rate from >9G to <7G. At just under 5G this parser can be used in place of the native without dropping (in addition to the list above).
- xor_executable pushes parse CPU to 100% and the system can drop significantly due to parse backup.

Aggregation Adjustments Based on Tested Live Content

A 10G Decoder can serve aggregation to a single Concentrator while running at 10G speeds. Deployments using Malware Analysis, Event Stream Analysis, Warehouse Connector, and Reporting Engine are expected to impact performance and can lead to packet loss.

For the tested scenario, the Concentrator aggregates between 45 and 70k sessions/sec. The 10G Decoder captures between 40-and 50k sessions/sec. With the content identified above, this is about 1.5 to 2 million meta/sec. Due to the high volume of session rates, the following configuration changes are recommended:

- Nice aggregation on the Concentrator limits the performance impact on the 10G Decoder. The following command turns on nice aggregation.
  `/concentrator/config/aggregate.nice = true`
- Due to the high volume of sessions on the Concentrator, you may consider activating parallel values mode on the Concentrator by setting `/sdk/config/parallel.values` to `16`. This improves Investigation performance when the number of sessions per second is greater than 30,000.
- If multiple aggregation streams are necessary, aggregating from the Concentrator instead has less impact on the Decoder.
- Further review for content and parsing is required for deployments where you want to use other NetWitness Platform components (Warehouse Connector, Malware Analysis, ESA, and Reporting Engine).

Optimize Read/Write Operations When Adding New Storage

A 10G Decoder is optimized to stagger read and write operations across multiple volumes so that the current file being written is on a different volume from the next file that will be written. This allows maximum throughput on the raid volume when reading data from the last file being written while writing the current file on a different volume. However, if volumes are added after a Decoder has been in use, the ability to stagger is limited because one or more volumes are already full so the new volume is the only place new files can be written.

To remedy this situation, an administrator can run a `stagger` command on an existing NetWitness Platform database (packet, log, meta, or session), that has at least two volumes, to stagger the files across all volumes in the most optimal read/write pattern. The major use case is when new storage is added to an existing Decoder and you want to stagger the volumes BEFORE restarting capture.

The configuration nodes for this command are the session, meta, and packet databases. Each of these lives under `/database/config`, which is usually a root node. The config nodes for a Decoder are:

- `/database/config/packet.dir`
- `/database/config/meta.dir`
- `/database/config/session.dir`

The *NetWitness Platform Core Database Tuning Guide* has information on how those configurations are formatted.

The `stagger` command is typically only useful for a 10G Decoder and usually just for the packet database. Maximum performance is achieved for storing and retrieving packets when multiple volumes are present. In this scenario, the Decoder always fills the volume with the most free space. When the volumes are roughly the same size, this results in a staggered write pattern, which allows maximum throughput for reading and writing across all volumes. However, this only naturally occurs when multiple packet storage volumes are present at the time the Decoder is first deployed.

A typical use case is adding more storage to an existing Decoder to increase retention. However, when adding storage to an deployment that has already filled the existing volumes with stored packets, the Decoder will naturally fill the new storage with packets before rolling out any packets on the existing storage. This results in a suboptimal read/write pattern because most reads will occur on the same volume that is currently being written to. In a 10G deployment, reads are blocked from the volume when writes are occurring. This does not stop ALL reads on that volume, because the file is buffered in memory before being written, but it does result in suboptimal read performance.

With the `stagger` command, you can add more storage and then have the service naturally stagger the files across ALL volumes (existing and new) so that read performance is optimized.

Caution: This command should only be performed AFTER the storage is mounted and the Decoder configured to use it (for example, after adding the mount point(s) to `packet.dir`).

The downside to this command is it can take some time to stagger and the Decoder should not be capturing during the stagger operation.

Recommended workflow:

1. Add all storage and configure mount points.
2. Add new storage mount points to `packet.dir` (or `session.dir/meta.dir`) and restart service (very important!).
3. Ensure capture is stopped.
4. Run stagger operation but make sure the connection that initiated the stagger operation is never terminated until the operation is complete. If the connection is terminated, then the stagger operation will be canceled. If the operation is canceled, the files that were already staggered will remain in place. The operation can be resumed by rerunning the same command (the work already done will not need to be done again). If running stagger from NwConsole, run the `timeout 0` command before sending the `stagger` command. This will prevent the normal 30-second command timeout.
5. Start capture after `stagger` command finishes.

The following are the parameters for the command:

- `type` - The database that will be staggered (session, meta, or packet). Typically only the packet database is useful for staggering, but it is possible to do the session or meta database when multiple volumes are present for those databases. Since the session and meta databases write far less data than the packet database, typically staggering those databases results in less noticeable performance gains.

- `dryRun` - If `true` (the default), will only return a description of the operations that would be performed. If `false`, then the files will actually be moved to an optimal read/write pattern. You MUST pass false to actually stagger the files.

Example usage from NwConsole:

```
login <decoder>:50004 <username> <password>
timeout 0
send /database stagger type=packet dryRun=false
```

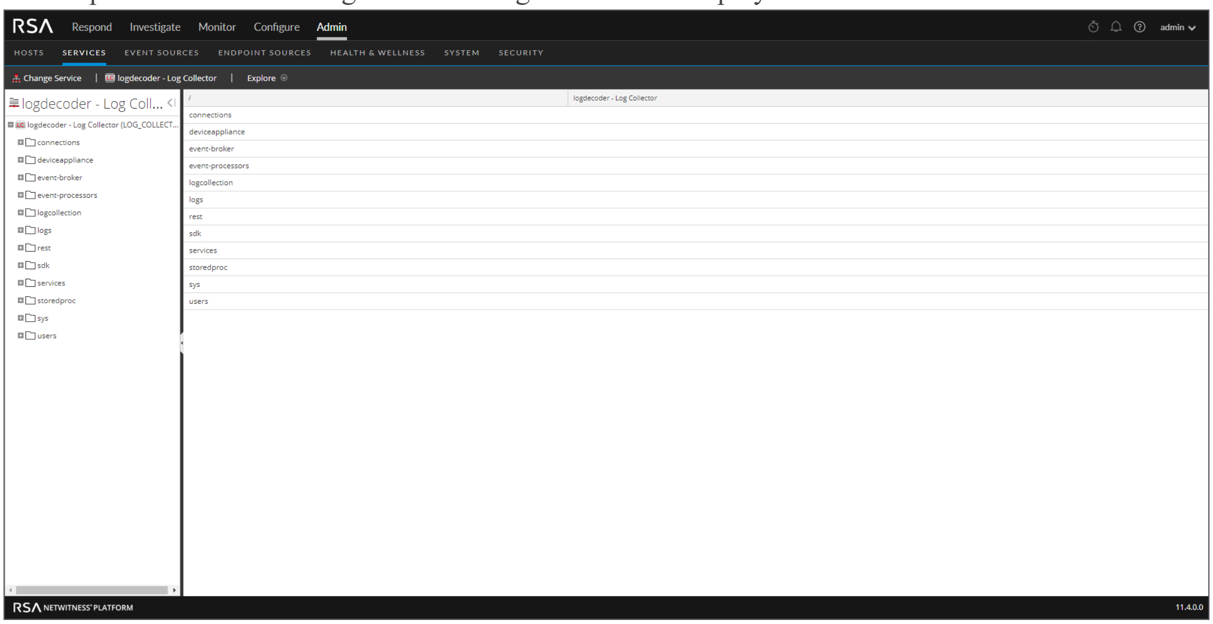
If you run this command via the RESTful API, please pass the additional parameter `expiry=0` to prevent a timeout from the service. You will also need to ensure the HTTP client does not disconnect before the operation completes.

Configure a Log Decoder to Accept Protobuf

There are occasions when you want to analyze log files that are in protobuf (Protocol Buffer) format. You can configure a Log Decoder with a Log Collector service to accept logs in protobuf (Protocol Buffer) format.

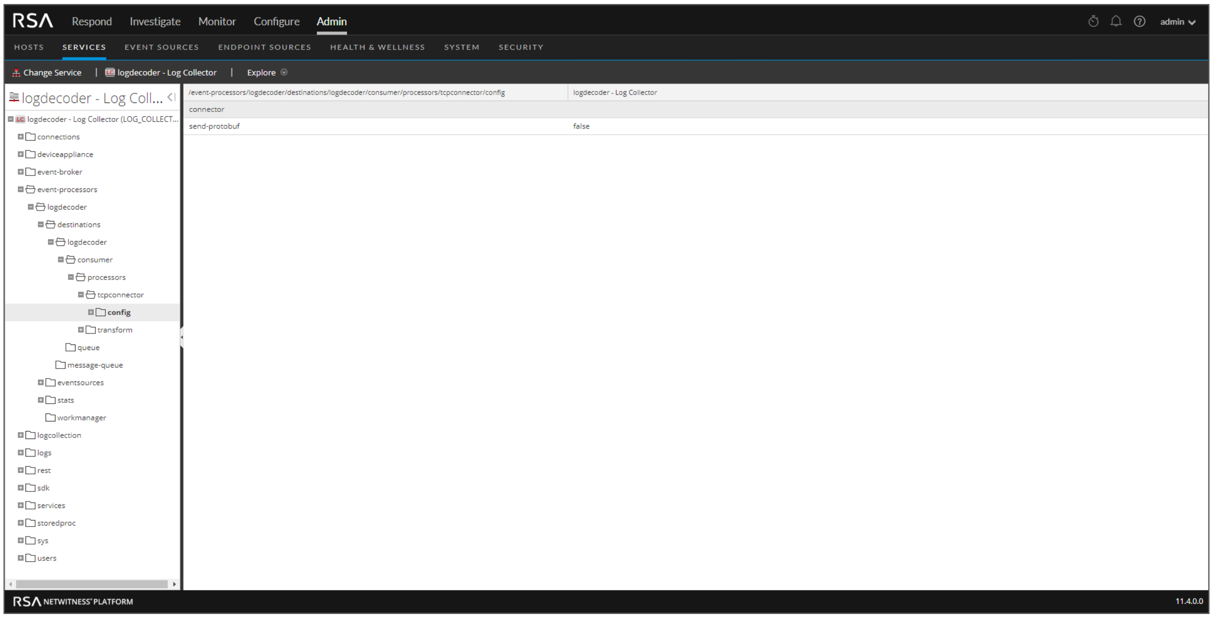
To import a log file to a Log Decoder:

1. Go to **ADMIN** > **Services**.
2. Select a Log Decoder with a Log Collector service in the **Service** list, and select > **View** > **Explore**.
   The Explore view for the Log Decoder -Log Collector is displayed.



3. Navigate to **`event-processors/logdecoder/destinations/logdecoder/consumer/processors/tcpconnector/config`**

Your screen should look similar to the following.



4. For the **send-protobuf** field, select **false**, and change the value to **true.**
5. Navigate to `event-processors/logdecoder/destinations/logdecoder/consumer/processors/tcpconnector/config/connector/channel/tcp` and change the **port** value to **50202**.
6. Navigate to `event-processors/logdecoder/destinations/logdecoder/consumer/processors/tcpconnector/config/connector/event` and change the following parameters:
   - Clear the **delimiter** field
   - Change **format** to **%text%**

Configure Session Split Timeouts

The default behavior of the Decoder is to automatically end sessions that exceed a configured size or have been inactive for a period of time. When the session is ended due to timeout, any subsequent packets received in that session appear to be stored in a new session. You can mitigate the effect of session splitting due to long periods of inactivity between packets using this procedure.

When a Decoder session exceeds a configured size (32MB by default, the `/decoder/config/assembler.max.size`) or has been inactive for a period of time, the session is split. NetWitness Platform has the previous packet and the next packet and can propagate session state from the initial session fragment to the subsequent session fragment.

Each session fragment is annotated (`session.split` meta) such that it can be identified and associated with other fragments from the actual network session. Directionality as determined by the initial session reduces the occurrence of fragments having reversed directionality.

If there is a gap in time between packets large enough that there are no longer any packets for the session in memory, the session is removed from the Decoder. If a subsequent packet shows up after this occurs, a new session is created with no context to the preceding session. The issue is the inability to continue a session when we encounter a gap between packets of a session that is larger than the packets we are buffering (based upon available memory and timeout configurations). Once the last packet of a session is removed from memory, the session is also removed, and with it the necessary context for ensuring consistent directionality.

There are two timeout settings in a Network Decoder, `/decoder/config/assember.timeout.session` and `assembler.timeout.packet`. Both default to 60 seconds. The setting `assembler.timeout.session` controls how long a session lives in Assembler without receiving another packet. The setting `assembler.timeout.packet` controls how long a session waits before getting parsed. If the session is kicked out of Assembler before this timeout, then it automatically goes to parsing.

The session timeout is the number of seconds since the last packet was added to that session. Therefore, this timeout resets on every packet added to that session. The packet timeout is the number of seconds since the very first packet for that session was added (in other words, the packet that created the session). This is never reset and once the timeout expires, the session is parsed.

The important point is a session can be parsed but still remain in Assembler. A session in Assembler can still have packets added to it, even if it has already been parsed. Packets added after the session is parsed will never be seen by parsers, but they will be attached to the session and can be viewed by a subsequent `/sdk content` or `/sdk packets` call.

After a session is parsed, the session AND its metadata are written to disk. At this point, they can be aggregated and "seen" by `sdk` commands. Packets are written in order of capture and are not reordered by what session they belong to. Nor are they necessarily written when the session and meta data are written.

You can disable both timeout nodes, `/decoder/config/assembler.timeout.session` and `assembler.timeout.packet`, by setting them to zero in the Services Explore view.

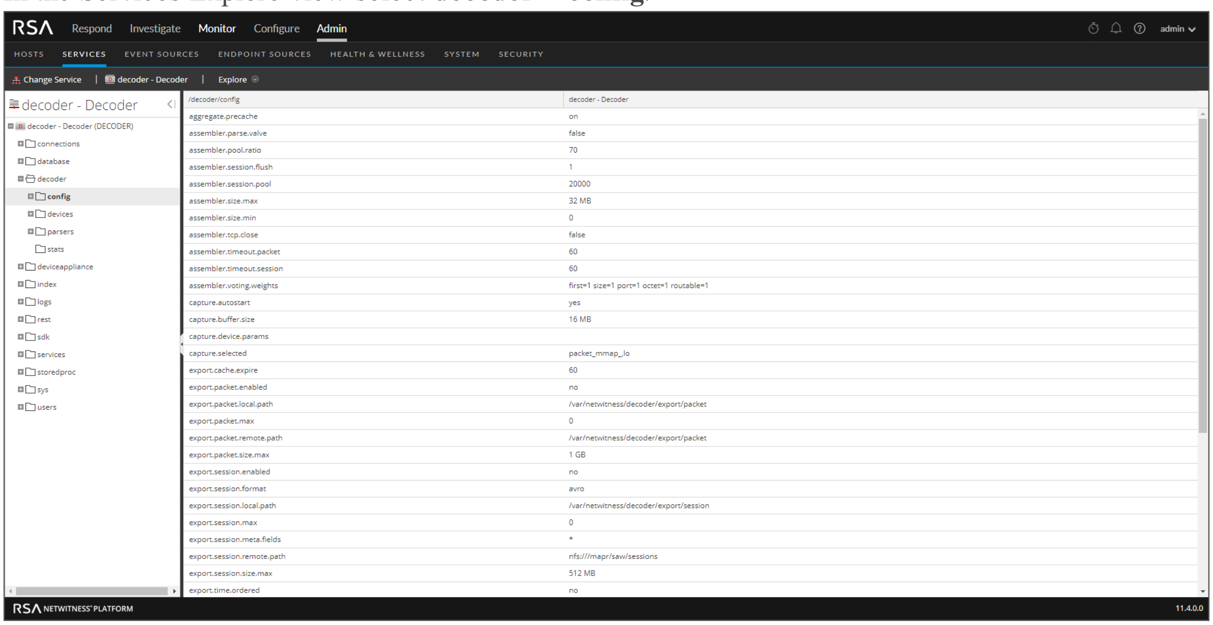
If both timeouts are disabled, the sessions are still split due to time or size expiration. However, the Decoder keeps track of the network stream for as long as it has sufficient memory. Thus, when more packets arrive on the same network stream, the Decoder adds `split` meta items to the subsequent sessions. Using a combination of the `split` metadata and the stream key, it is possible to reconstruct the network stream from the multiple sessions.

The length of time for which sessions are tracked is limited by the number of session pool entries available on the Decoder, and therefore the actual time window varies according to the rate at which new sessions are added. If new sessions are added at a high rate, the size of the time window decreases. The size of the pool is set using the configuration entry `/decoder/config/assembler.session.pool`, which sets the maximum number of sessions that will be tracked at a time.

The `/decoder/stats/assembler.timespan` statistic allows you to see when the Decoder is no longer tracking session splits because the ingest rate is too high and the Decoder does not have enough memory to track. This statistic shows the number of seconds tracked within the session table, which is the effective time window in which the Decoder can link together sessions. Under normal operation this statistic matches the value of `/decoder/config/assembler.timeout.session`, but when running in Time Split mode, the `/decoder/stats/assembler.timespan` statistic grows or shrinks depending on the ingest rate.

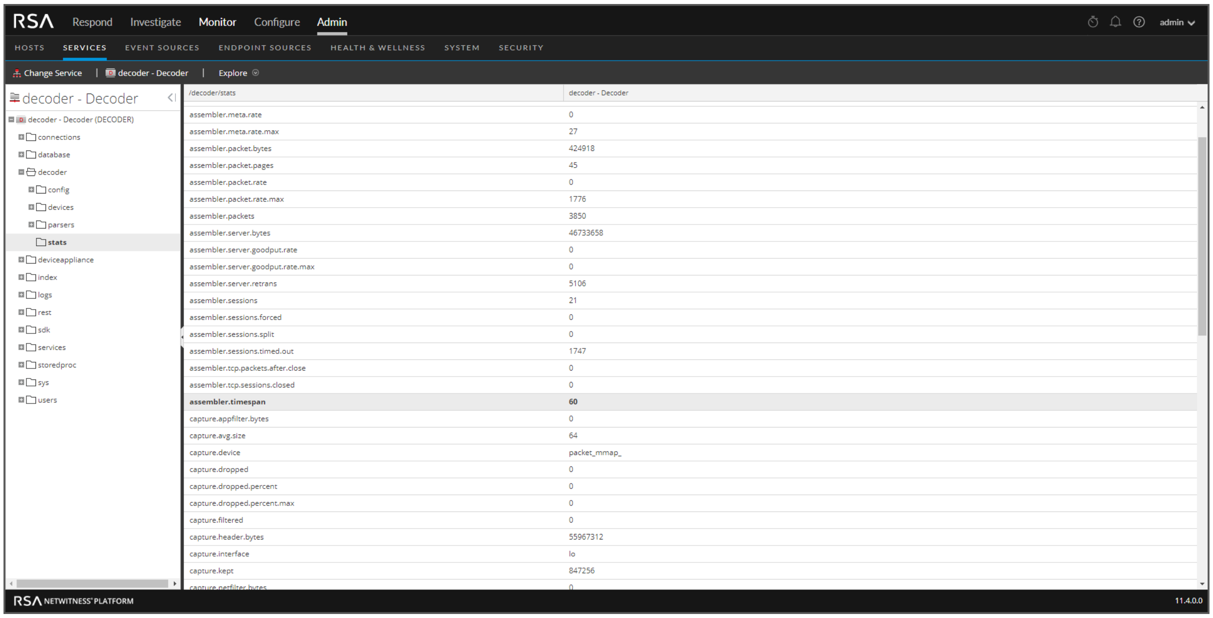
To configure Time Split mode, set the following configuration parameters and restart the Decoder:

1. In the ADMIN > Services view, select the Decoder service and  **> View > Explore.**
2. In the Services Explore view select **decoder** > **config**.



3. Click in the **Value** column next to the parameter and set these two parameters :
   `/decoder/config/assembler.session.flush = 0`
   `/decoder/config/assembler.timeout.session = 0`
4. To see when the Decoder is no longer tracking session splits because the ingest rate is too high and the Decoder does not have enough memory to track, view the `/decoder/stats/assembler.timespan` statistic, in the Services Explore view select **decoder** >

stats.



Configure Syslog Forwarding to Destination

In addition to collecting Syslog messages, you can configure the Log Decoder to forward Syslog messages to another Syslog receiver.

> **Note:** No configuration is necessary to collect Syslog messages on the Log Decoder.

NetWitness Platform forwards Syslog messages after it has parsed the messages and before it writes the messages to the Log Decoder.

> **Note:** You must configure Syslog Forwarding using the steps defined in this topic using the **Explore** view.

The Log Decoder must be in the **Started** state before you can configure Syslog Forwarding. To configure Syslog Forwarding:

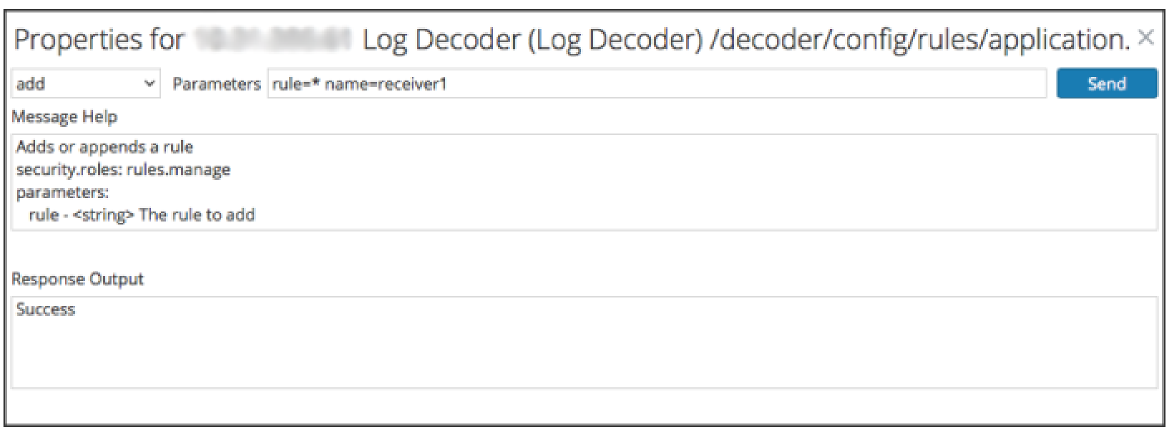
1. Configure Log Decoder application layer rules (Application rules) to tag Syslog messages with metadata that instructs NetWitness Platform to forward the messages:

   a. In the **Services** view, select a Log Decoder, and in the Actions column, select > **View** > **Explore**.

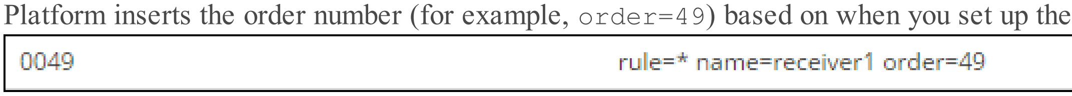
   b. Go to the **/decoder/config/rules/application** node, right-click **application,** and click **Properties**.

   c. In the **Properties** view, specify the **add** command with the following parameters:
   `rule=<query> name=<name>`
   Example 1: `rule=*name=receiver1`
   Example 2: `rule="device.type='winevent_nic'" name=receiver`)

   d. Click **Send**.



   NetWitness Platform creates the `name=receiver1 rule=* order=<n>` rule. NetWitness Platform inserts the order number (for example, `order=49`) based on when you set up the rule.



   e. Go to the `/decoder/config/rules/application` node and click the `name=receiver1 rule=* order=49` rule.

f. Add **alert forward** parameters to the rule parameters.

```
rule=* name=receiver1 order=49 alert forward
```

or

```
rule=* name=receiver1 forward
```

All other rule parameters have the same meaning as they do in other application rules.

The following Application rule example selects all logs with the * rule. It creates a transient alert meta with the value "receiver1" and tags the entire log for forwarding it to the syslog forwarding destination. You can define as many different forwarding rules as you need with the same name or unique names.

2. Define Syslog forwarding destinations and enable forwarding.

   a. In the **Services** view, select a Log Decoder, and **> View > Explore**.

   b. Syslog forwarding destinations are defined in the configuration node `/decoder/config/logs.forwarding.destination.`
   This configuration node contains one or more name/value pairs. The name corresponds to the name parameter in the application rule that you used to tag logs with forwarding meta. The value is a colon-separated triple of transport, host, and port followed by an optional formatting parameter.

   ```
   name=(udp|tcp|tls):host:port[:(retainsource|rfc3164)]
   ```

   The first parameter indicates the transport protocol and must be one of udp, tcp, or tls. Specifying udp will forward logs via RFC 3164 / RFC 5426 UDP syslog protocol. Specifying tcp will forward logs via a TCP connection with RFC 6587 framing. Specifying tls will forward logs in accordance with RFC 5425.
   The host is an IPv4 address, IPv6 address, or host name.

   The port is the port to which the logs are sent. This is typically port 514 for UDP syslog, and 6514 for TLS connections. There is no standard port assignment for syslog over TCP.

   Optionally, `retainsource` or `rfc3164` can be specified at the and of the destination string to indicate that additional formatting and information should be included with each log forwarded. Specifying `retainsource` will include z-connector headers at the beginning of the log based and will be populated by the `time`, `device`.(`ip|ipv6|host`), and `lc.cid` meta and is best used for forwarding to other log decoders. The `rfc3164` option will prepend a valid RFC3164 header to all events forwarded constructed of the `syslog.pri`, `time`, and `device`.(`ip|ipv6|host`) meta. In both cases, the original log text is unmodified.

   Example forwarding destination:

   ```
   gears=tls:gears.netwitness.local:6514
   ```

   Example forwarding over tcp to blackout on port 514 with z-connector headers:

   ```
   fwdrule=tcp:blackout.netwitness.local:514:retainsour
   ```

In the `/decoder/config/logs.forwarding.destination` parameter, specify the destination. For example:

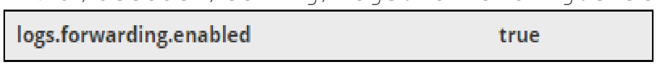
TLS Connections: `receiver1=tls:receiver1.netwitness.local:6514`

UDP Connections: `receiver1=udp:receiver1.netwitness.local:514`

TCP Connections: **`receiver1=tcp:receiver1.netwitness.local:514`**

> **Note:**
> You can configure:
> - Multiple rules to forward logs to the same destination.
> - Multiple rules to forward logs to multiple destination.
>
> For TLS connections, the certificate of the forwarding destination must be validated. The certificate authority that signed the destination's certificate must be present in the Log Decoder's CA trust store and the certificate must reside on the destination or Syslog receiver. Refer to "Configure Certificates" in the *Log Collection Configuration Guide* for information about manipulating the Log Decoder's CA trust store. (Go to the Master Table of Contents to find all RSA NetWitness Platform 11.x documents.)

c. In the `/decoder/config/logs.forwarding.enabled` parameter, specify **true**.



Configure Transaction Handling on a Decoder

Administrators can configure a Decoder to subdivide incoming sessions into smaller transaction sessions when using Lua parsers designed to create transactions. The feature allows analysts to perform analytics on the split sessions in downstream services such as Investigate.

> **Caution:** Use caution when enabling this feature when you are above the standard ingest rate supported by the Network Decoder as it may start to cause backups with aggregation and dropped packets.

Transaction Handling

The Decoder service configuration node has a new parameter for configuration of transaction handling: `/decoder/parsers/config/parser.transaction.mode`. This node controls the behavior of the Decoder when a parser defines a transaction within a network session.

The values for `parser.transaction.mode` correspond to the operating modes:

- `off` (transactions off)
- `meta` (transactions represented as meta items)
- `split` (transactions split sessions)

Transactions Off

When transactions mode is off, any application-level transactions created by parsers are ignored, and nothing is stored in the collection to represent the transaction.

Transactions Represented as Meta Items

In this mode of operation, when a parser generates an application-level transaction, a new meta item of type `{{trans}}` is added to the session in which the transaction occurred. The `{{trans}}` meta item contains a list of other meta items that constitute the transaction.

Transactions Split Sessions

In this mode of operation, when a parser generates an application-level transaction, the session is split. The session splitting is accomplished by:

1. A new session item is created.
2. Network meta items are copied from the parsed session into the new session.
3. Meta items marked in the transaction are moved from the original session to the new session.

The following meta items are duplicated into the split session from the session that was parsed:

- `time`
- `medium`
- `eth.src`
- `eth.dst`
- `eth.type`
- `ip.proto`
- `ip.src`
- `ip.dst`
- `ipv6.src`
- `ipv6.dst`
- `ip.proto`
- `port.src`
- `port.dst`
- `tcp.flags`
- `udp.srcport`
- `udp.dstport`
- `service`
- `udp.srcport`

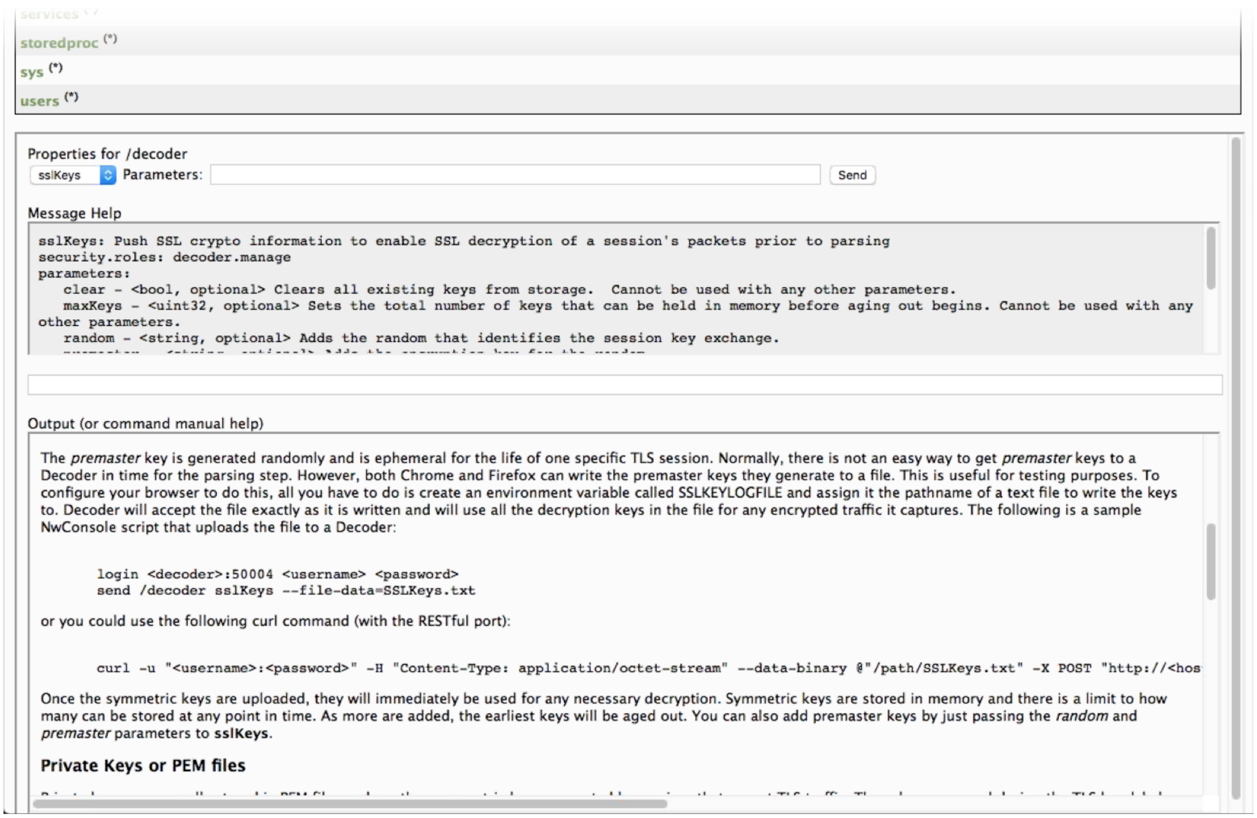
- `udp.srcport`
- `tls.premaster`

Decrypt Incoming Packets

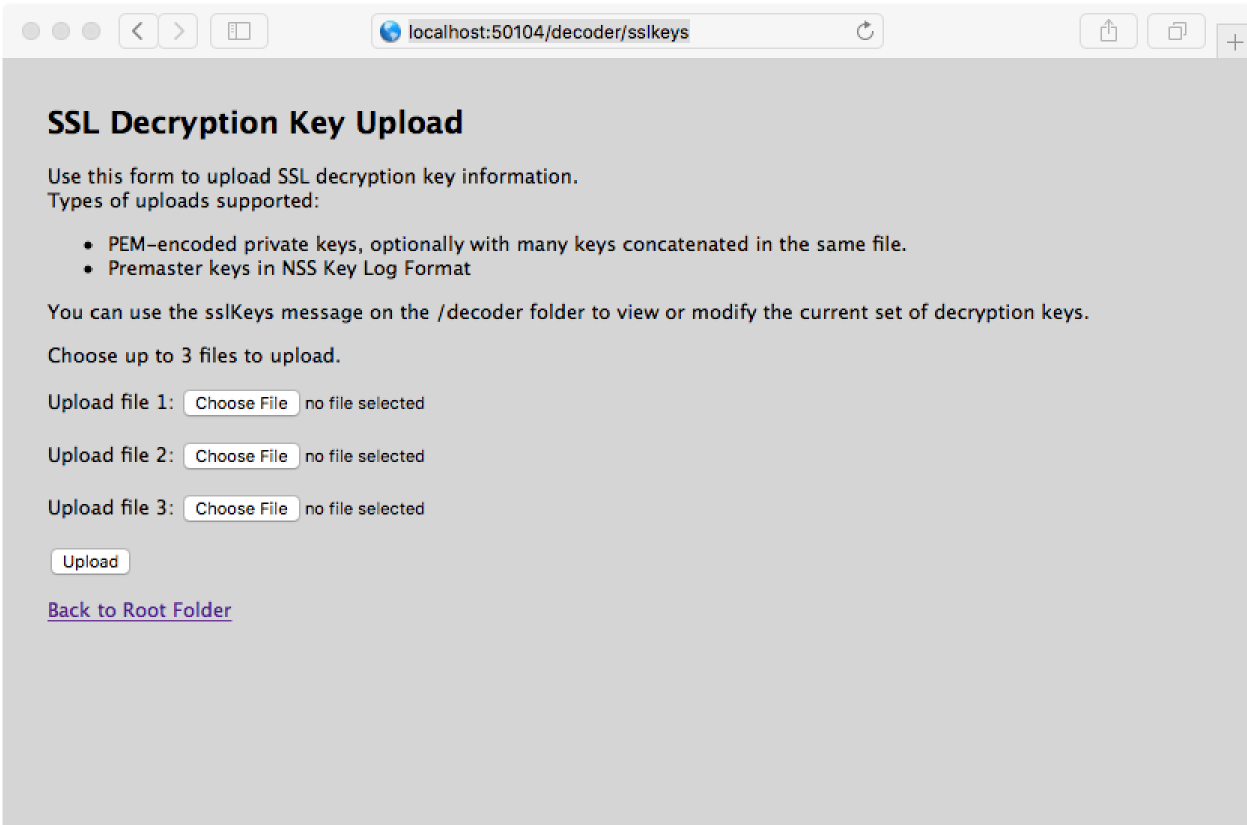
Beginning with NetWitness Platform 11.0, administrators can configure a Network Decoder to decrypt incoming packets using the `sslKeys` command. Enabled parsers see the unencrypted packet payload and create metadata accordingly. If the Decoder is not configured to decrypt incoming packets, most enabled parsers see only encrypted garbage and fail to create meaningful metadata.

Note: If FIPS is enabled, the list of ciphers for decryption is restricted to only those that are FIPS approved.

The `sslKeys` command provides a way to upload premaster or private keys to the Decoder, so that captured encrypted packets that match the keys can be decrypted before parsing. Administrators configure the Decoder by entering the `sslKeys` command using the NwConsole command line interface or the Decoder RESTful interface.



The RESTful interface form at the path: `/decoder/sslkeys` allows uploading a single PEM-encoded private key, a single file containing multiple private keys concatenated together, or a single file of multiple premaster keys.



Although the packets are decrypted during the parse stage, only the encrypted packets are written to disk. The matching premaster key used for decrypting is written to the `tls.premaster` meta key, which analysts can use to subsequently view unencrypted packets on demand.

Details for administrators to configure decryption of incoming packets, and for analysts to view unencrypted packets on demand are provided below.

Decryption of Secure SMTP

In version 11.3, NetWitness Platform supports opportunistic SSL/TLS decryption, which addresses RFC 3207 (https://tools.ietf.org/html/rfc3207). You can add an HTTPS parser option that provides a .csv list of destination ports of the session where the STARTTLS command will be searched, with at least one encryption key that has been uploaded. This enables the STARTTLS functionality.

Note: The .csv list must include all the destination ports of a session. If a destination port is not in the list, the search for STARTTLS will not be started.

The STARTTLS search is limited to the first 1024 bytes of the session's payload, to prevent performance degradation. If STARTTLS is found, the parser uses the next client packet with payload as the start of a TLS negotiation.

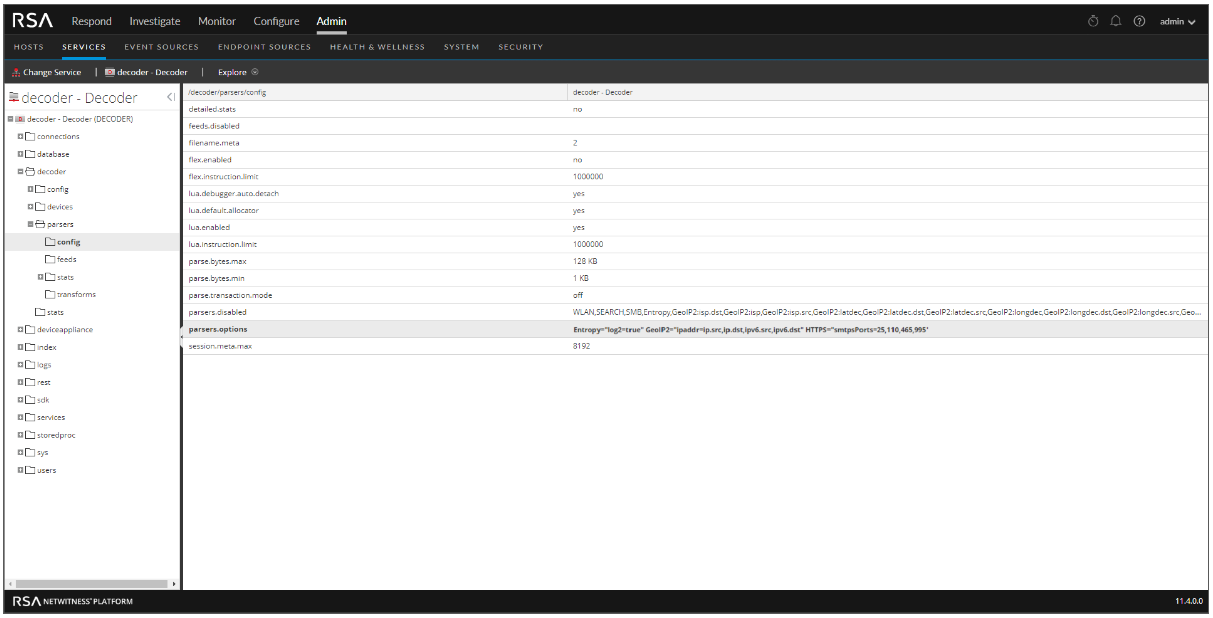
To add an HTTPS parser option for secure SMTP decryption:

1. In the NetWitness Platform User Interface, go to **ADMIN** > **Services**.
2. Select a Log Decoder service and click > **View** > **Explore**.
3. In the left panel, select **decoder** > **parsers** > **config**.

4. In **parsers.options**, enter an HTTPS parser option in the following format that provides a .csv list of the destination ports of the session where the STARTTLS command is searched:

   `HTTPS="smtpsPorts=<port#1,port#2,port#n>"`

   The following image shows an example of an HTTPS parser option for secure SMTP decryption.



Performance Considerations

Decrypting packets in real time requires extra work in the parsing stage. Before implementing this feature, plan carefully to ensure the incoming traffic bandwidth does not overwhelm the available compute power. You may need more Decoders to decrypt traffic than you would need if not decrypting.

Packets captured on a Decoder normally have a timeout of ~60 seconds in the assembly stage before they are sent to the parsing step. If the Decoder is under memory pressure due to very high bandwidth, the lifetime of the packets in Assembler may be shortened. To alleviate this situation, you can configure a longer timeout value and increase the amount of memory available to hold packets in Assembly. Also, in order to perform decryption of the packets, the Decoder must receive the decryption key before the parsing stage.

> **Note:** Currently, only TLS 1.2 and earlier protocols can be decrypted

With no feeds loaded, the following parsers enabled, and 50% of the sessions being decrypted,a Decoder can process traffic at 3 Gbps .

Parser Name	Description
SYSTEM	Session Details
NETWORK	Network Layer
ALERTS	Alerts
GeoIP	Geographic data based on ip.src and ip.dst
GeoIP2	Geographic data by default based on IPv4 (`ip.src`, `ip.dst`) and IPv6 (`ipv6.src`, `ipv6.dst`) meta keys
HTTP	Hyper Text Transport Protocol (HTTP)
HTTP_Lua	Hyper Text Transport Protocol (HTTP) Lua
FTP	File Transfer Protocol (FTP)
TELNET	TELNET Protocol
SMTP	Simple Mail Transport Protocol (SMTP)
POP3	Post Office Protocol (POP3)
NNTP	Network News Transport Protocol (NNTP)
DNS	Domain Name Service (DNS)
HTTPS	Secure Socket Layer (SSL) Protocol
MAIL	Standard E-Mail Format (RFC822)
VCARD	Extracts VCARD fullname and email information
PGP	Identifies PGP blocks within network traffic
SMIME	Identifies SMIME blocks within network traffic
SSH	Secure Shell (SSH)
TFTP	Trivial File Transfer Protocol (TFTP)
DHCP	Dynamic Host Configuration Protocol (DHCP and BOOTP)
NETBIOS	Extracts NETBIOS computer name information.
SNMP	Simple Network Management Protocol (SNMP)
NFS	Network File System (NFS) protocol
RIP	Routing Information Protocol (RIP).

Parser Name	Description
TDS	MSSQL and Sybase database protocol (TDS)
TNS	Oracle database protocol (TNS)
IRC	Internet Relay Chat (IRC) protocol
RTP	Real Time Protocol (RTP) for audio/video
SIP	Session Initiation Protocol (SIP)
H323	H.323 Teleconferencing protocol
SCCP	Cisco Skinny Client Control Protocol
GTalk	Google Talk (GTalk)
VlanGre	Vlan ID and GRE/EtherIP tunnel addresses
BITTORRENT	BitTorrent File Sharing Protocol
FIX	Financial Information eXchange Protocol
GNUTELLA	Gnutella file sharing protocol
IMAP	Internet Message Access Protocol
MSRPC	Microsoft Remote Procedure Call protocol
RDP	Remote Desktop Protocol
SHELL	Command Shell Identification
TLSv1	TLSv1
SearchEngines	A parser that extracts search terms
FeedParser	External Feed Parser

Encryption Keys

The `sslKeys` command accepts two types of encryption keys:

- Premaster key: the symmetric key used in the TLS payload stream for encryption and decryption.
- Private key: the asymmetric private key used during the TLS handshake that encrypts the premaster.

Premaster Key

The premaster key is generated randomly and is ephemeral for the life of one specific TLS session. Normally, there is not a good way to get premaster keys to a Decoder in time for the parsing step. However, both Chrome and Firefox can write the premaster keys they generate to a file. This is useful for testing purposes. To configure your browser to do this, create an environment variable called `SSLKEYLOGFILE` and assign it the pathname of a file to which the keys will be written. The Decoder will accept the file exactly as written and will use all the decryption keys in the file for any encrypted traffic it captures.

This is a sample NwConsole script that uploads the file to a Decoder:

```
login <decoder>:50004 <username> <password>

send /decoder sslKeys --file-data=SSLKeys.txt
```

This is an example using a curl command (with the RESTful port) to upload the file to a Decoder:

```
curl -u "<username>:<password>" -H "Content-Type: application/octet-stream" --
data-binary @"/path/SSLKeys.txt" -X POST
"http://<hostname>:50104/decoder?msg=sslKeys"
```

After the symmetric keys are uploaded, they will immediately be used for any necessary decryption. Symmetric keys are stored in memory and there is a limit to how many can be stored at any point in time. As more keys are added, the earliest keys will be aged out. You can also add premaster keys by just passing the `random` and `premaster` parameters to `sslKeys`.

Private Keys or PEM files

Private keys are normally stored in PEM files and are the asymmetric keys generated by services that accept TLS traffic. These keys are used during the TLS handshake to encrypt the premaster symmetric key that will be used for the rest of the payload encryption.

For example, if you have a web server whose traffic you want visibility into, you need to upload the private key it uses to encrypt traffic. You only need to do this once, as it is stored permanently (or until removed by a delete command). Private keys are automatically encrypted before storing to protect them. After upload, you must issue a parser reload command so that the newly installed key becomes visible to the HTTPS parser. Now, all TLS handshakes that use that private key will be able to be decrypted by the Decoder.

> **Note:** Not all ciphers suites use a "known" private key (for example, Ephemeral Diffie Hellman). Encrypted traffic with those ciphers cannot be decrypted unless the premaster key is uploaded to the Decoder before the session is parsed.

These are some sample commands that upload a PEM file to be used for decryption.

Using NwConsole:

```
send /decoder sslKeys pemFilename=MyKey.pem --file-data=/path/MyKey.pem
```

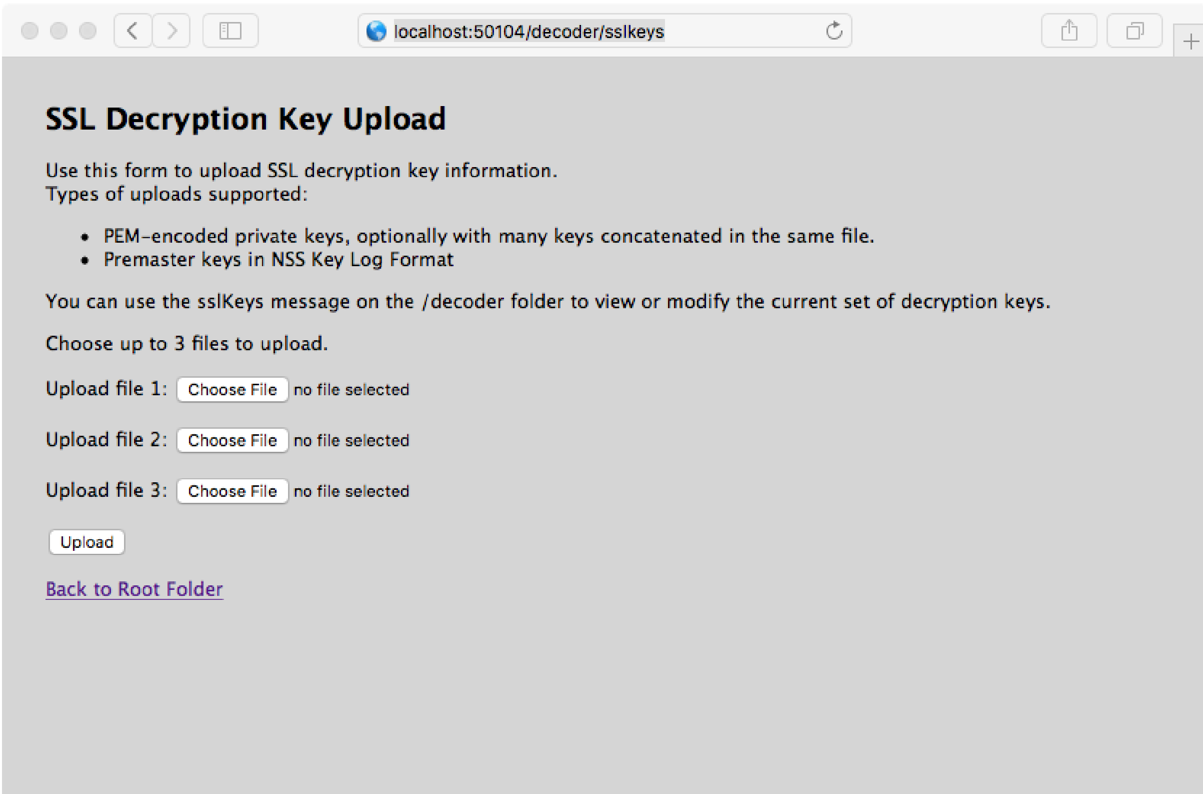
Using the RESTful interface (you must provide the `pemFilename` parameter in the URL):

```
curl -u "<username>:<password>" -H "Content-Type: application/octet-stream" --
data-binary @"/path/MyKey.pem" -X POST
"http://<hostname>:50104/decoder?msg=sslKeys&pemFilename=MyKey.pem"
```

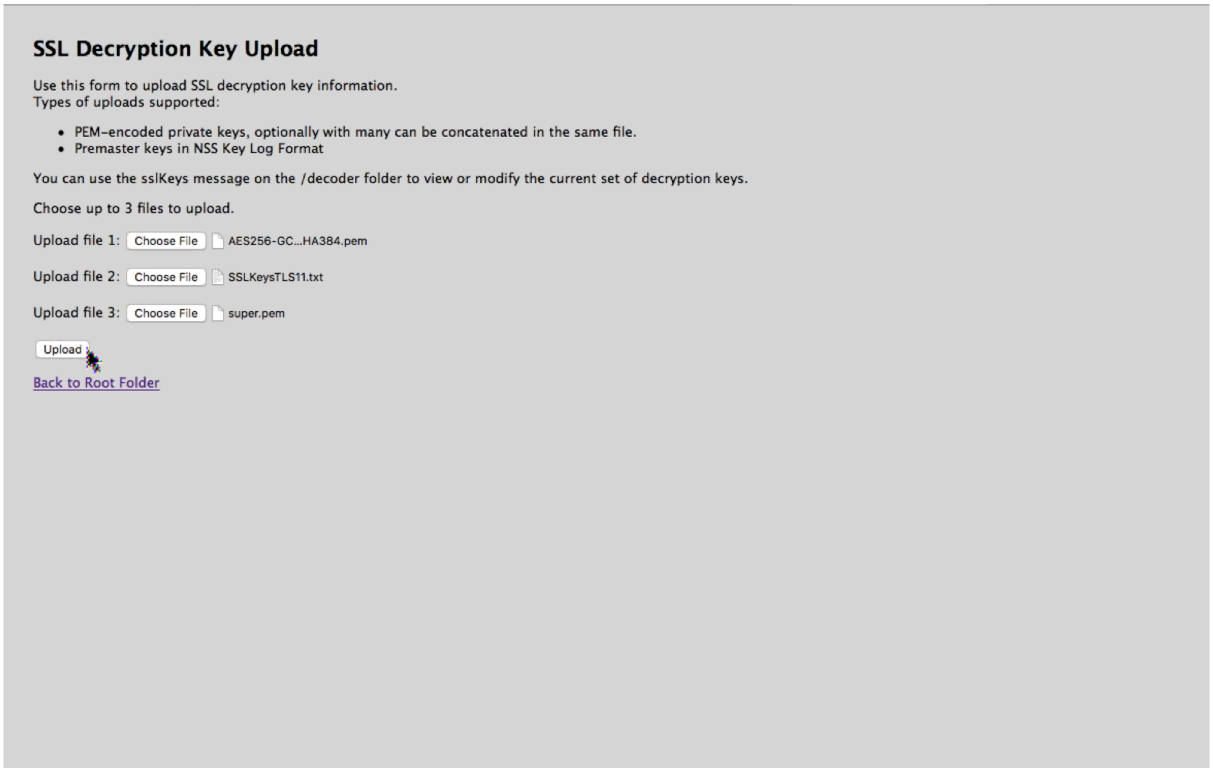
Upload Multiple Premaster and Private Keys

You can use the RESTful interface form to facilitate uploading of multiple keys, both premaster and private at the same time.

1. Open the RESTful API in your browser, and navigate to this path on the Decoder that you want to configure: `/decoder/sslkeys`.

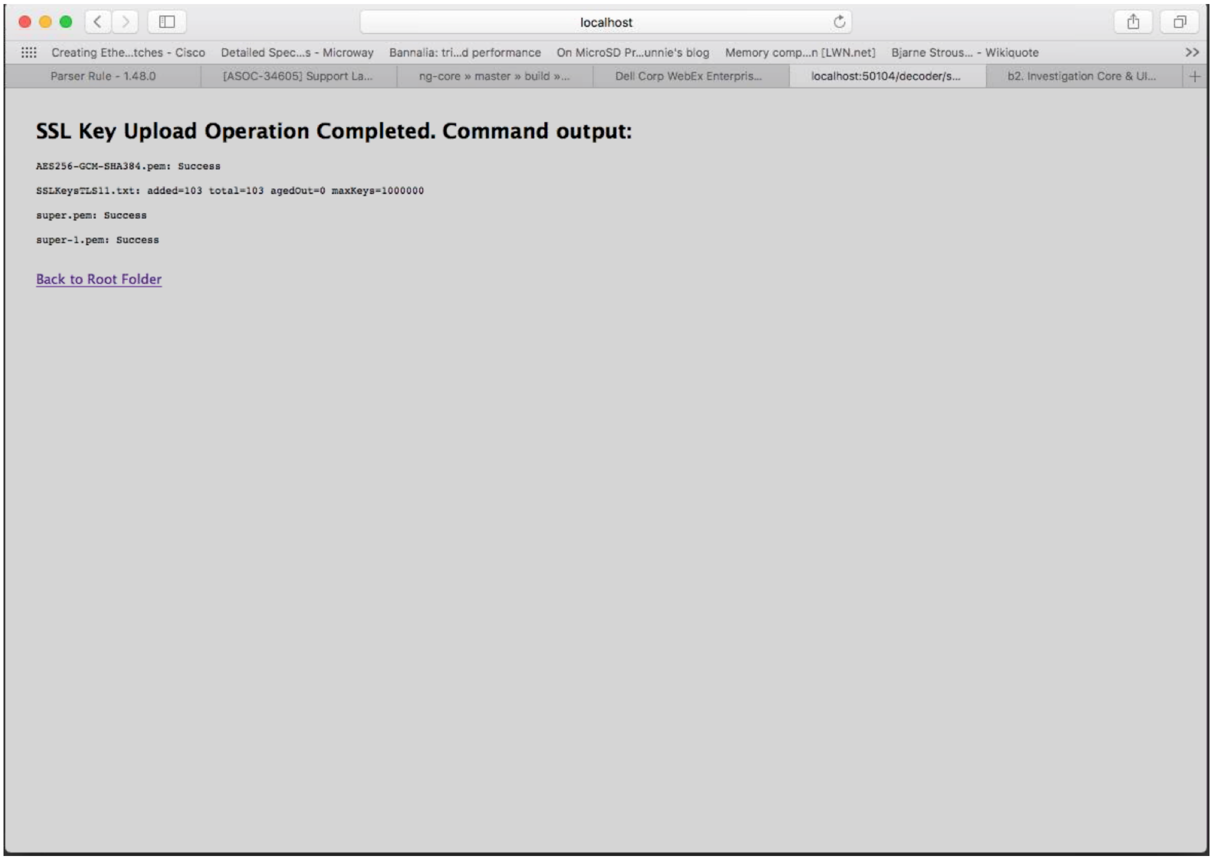


2. Next to **Upload File 1**, click **Choose File** and locate the premaster key file or PEM file that you want to upload on you local file system.
3. (Optional) Repeat for **Upload File 2** and **Upload File 3**.



4. Click **Upload**.

   The files are uploaded to the Decoder and results are displayed in the form.



Parameters for Managing Keys

The `sslKeys` command has several parameters for managing premaster and private keys. This is the full list of parameters:

Parameter	Description
`clear`	Removes all premaster keys from memory. Does not delete any PEM files installed on the system.
`maxKeys`	Changes the maximum number of premaster keys that are stored in memory.
`listPems`	Returns a list of all installed private key PEM files.
`deletePem`	Deletes the named PEM file from the file system. You can pass this parameter more than once to remove multiple files.
`random`	The random hash used to identify the premaster key.
`premaster`	The premaster key that will be installed for the previous `random` parameter. They must show up in pairs and `random` must be first.

Return Values

Most `sslKeys` commands return name/value pairs of statistics about the premaster keys in memory. The statistics are listed in the following table.

Name	Description
`added`	The number of premaster keys just added during this command.
`total`	The total number of premaster keys loaded in memory.
`agedOut`	The total number of premaster keys that were removed during this command; this is not a lifetime stat.
`maxKeys`	The current maximum allowed premaster keys

Viewing Unencrypted Traffic

If packets are decrypted during the parse stage, encrypted packets are written to disk, and the matching premaster key used for decrypting is written to the `tls.premaster` meta key, analysts can view the unencrypted packets using the `tls.premaster` meta key.

One Decoder API that you can use to see the unencrypted packets is the `/sdk/content` RESTful service. You need to know the Session ID of the encrypted packets and the `flags` parameter masked to the value 128 (or 0x80 in hex). Point your browser to the Decoder RESTful interface and type in the following command, substituting the actual Session ID for `<id>`:

`http://<decoder>:50104/sdk/content?session=<id>&flags=128&render=text`

The Decoder returns a simple web page showing the packets after they are decrypted.

If you want to see what the packets look like encrypted, type in one of the following commands, substituting the Session ID for <id>:

`http://<decoder>:50104/sdk/content&session=<id>&render=text`

`http://<decoder>:50104/sdk/content&session=<id>&flags&render=text`

For more information on the `/sdk/content` service, see the manual page for `/sdk content`.

Supported Cipher Suites

The following table lists which cipher-suites are supported using private keys, as well as those that are not supported.

Cipher Suite Name (RFC)	Name (OpenSSL)	Cipher Suite	TLS Version	KeyExch.	FIPS	Private Key
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE-RSA-AES256-GCM-SHA384	[0xc030]	TLSv1.2	Kx=ECDH	Compliant	Not Supported

Cipher Suite Name (RFC)	Name (OpenSSL)	Cipher Suite	TLS Version	KeyExch.	FIPS	Private Key
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDHE-ECDSA-AES256-GCM-SHA384	[0xc02c]	TLSv1.2	Kx=ECDH	Compliant	Not Supported
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDHE-RSA-AES256-SHA384	[0xc028]	TLSv1.2	Kx=ECDH	Compliant	Not Supported
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	ECDHE-ECDSA-AES256-SHA384	[0xc024]	TLSv1.2	Kx=ECDH	Compliant	Not Supported
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDHE-RSA-AES256-SHA	[0xc014]	SSLv3	Kx=ECDH	Compliant	Not Supported
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	ECDHE-ECDSA-AES256-SHA	[0xc00a]	SSLv3	Kx=ECDH	Compliant	Not Supported
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	DHE-DSS-AES256-GCM-SHA384	[0xa3]	TLSv1.2	Kx=DH	Compliant	Not Supported
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	DHE-RSA-AES256-GCM-SHA384	[0x9f]	TLSv1.2	Kx=DH	Compliant	Not Supported
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	DHE-RSA-AES256-SHA256	[0x6b]	TLSv1.2	Kx=DH	Compliant	Not Supported
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	DHE-DSS-AES256-SHA256	[0x6a]	TLSv1.2	Kx=DH	Compliant	Not Supported
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	DHE-RSA-AES256-SHA	[0x39]	SSLv3	Kx=DH	Compliant	Not Supported
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	DHE-DSS-AES256-SHA	[0x38]	SSLv3	Kx=DH	Compliant	Not Supported
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	DHE-RSA-CAMELLIA256-SHA	[0x88]	SSLv3	Kx=DH	Non-Compliant	Not Supported
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA	DHE-DSS-CAMELLIA256-SHA	[0x87]	SSLv3	Kx=DH	Non-Compliant	Not Supported
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	ECDH-RSA-AES256-GCM-SHA384	[0xc032]	SSLv3	Kx=ECDH/RSA	Compliant	Not Supported
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	ECDH-ECDSA-AES256-GCM-SHA384	[0xc02e]	TLSv1.2	Kx=ECDH/ECDSA	Compliant	Not Supported

Cipher Suite Name (RFC)	Name (OpenSSL)	Cipher Suite	TLS Version	KeyExch.	FIPS	Private Key
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	ECDH-RSA-AES256-SHA384	[0xc02a]	TLSv1.2	Kx=ECDH/RSA	Compliant	Not Supported
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	ECDH-ECDSA-AES256-SHA384	[0xc026]	TLSv1.2	Kx=ECDH/ECDSA	Compliant	Not Supported
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	ECDH-RSA-AES256-SHA	[0xc00f]	SSLv3	Kx=ECDH/RSA	Compliant	Not Supported
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	ECDH-ECDSA-AES256-SHA	[0xc005]	SSLv3	Kx=ECDH/ECDSA	Compliant	Not Supported
TLS_RSA_WITH_AES_256_GCM_SHA384	AES256-GCM-SHA384	[0x9d]	TLSv1.2	Kx=RSA	Compliant	Supported
TLS_RSA_WITH_AES_256_CBC_SHA256	AES256-SHA256	[0x3d]	TLSv1.2	Kx=RSA	Compliant	Supported
TLS_RSA_WITH_AES_256_CBC_SHA	AES256-SHA	[0x35]	SSLv3	Kx=RSA	Compliant	Supported
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	CAMELLIA256-SHA	[0x84]	SSLv3	Kx=RSA	Non-Compliant	Supported
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	ECDHE-RSA-DES-CBC3-SHA	[0xc012]	SSLv3	Kx=ECDH	Compliant	Not Supported
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	ECDHE-ECDSA-DES-CBC3-SHA	[0xc008]	SSLv3	Kx=ECDH	Compliant	Not Supported
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	EDH-RSA-DES-CBC3-SHA	[0x16]	SSLv3	Kx=DH	Compliant	Not Supported
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	EDH-DSS-DES-CBC3-SHA	[0x13]	SSLv3	Kx=DH	Compliant	Not Supported
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	ECDH-RSA-DES-CBC3-SHA	[0xc00d]	SSLv3	Kx=ECDH/RSA	Compliant	Not Supported
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	ECDH-ECDSA-DES-CBC3-SHA	[0xc003]	SSLv3	Kx=ECDH/ECDSA	Compliant	Not Supported
TLS_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA	[0x0a]	SSLv3	Kx=RSA	Compliant	Supported
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDHE-RSA-AES128-GCM-SHA256	[0xc02f]	TLSv1.2	Kx=ECDH	Compliant	Not Supported

Cipher Suite Name (RFC)	Name (OpenSSL)	Cipher Suite	TLS Version	KeyExch.	FIPS	Private Key
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ECDHE-ECDSA-AES128-GCM-SHA256	[0xc02b]	TLSv1.2	Kx=ECDH	Compliant	Not Supported
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDHE-RSA-AES128-SHA256	[0xc027]	TLSv1.2	Kx=ECDH	Compliant	Not Supported
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	ECDHE-ECDSA-AES128-SHA256	[0xc023]	TLSv1.2	Kx=ECDH	Compliant	Not Supported
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	ECDHE-RSA-AES128-SHA	[0xc013]	SSLv3	Kx=ECDH	Compliant	Not Supported
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	ECDHE-ECDSA-AES128-SHA	[0xc009]	SSLv3	Kx=ECDH	Compliant	Not Supported
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	DHE-DSS-AES128-GCM-SHA256	[0xa2]	TLSv1.2	Kx=DH	Compliant	Not Supported
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	DHE-RSA-AES128-GCM-SHA256	[0x9e]	TLSv1.2	Kx=DH	Compliant	Not Supported
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	DHE-RSA-AES128-SHA256	[0x67]	TLSv1.2	Kx=DH	Compliant	Not Supported
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	DHE-DSS-AES128-SHA256	[0x40]	TLSv1.2	Kx=DH	Compliant	Not Supported
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	DHE-RSA-AES128-SHA	[0x33]	SSLv3	Kx=DH	Compliant	Not Supported
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	DHE-DSS-AES128-SHA	[0x32]	SSLv3	Kx=DH	Compliant	Not Supported
TLS_DHE_RSA_WITH_SEED_CBC_SHA	DHE-RSA-SEED-SHA	[0x9a]	SSLv3	Kx=DH	Non-Compliant	Not Supported
TLS_DHE_DSS_WITH_SEED_CBC_SHA	DHE-DSS-SEED-SHA	[0x99]	SSLv3	Kx=DH	Non-Compliant	Not Supported
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	DHE-RSA-CAMELLIA128-SHA	[0x45]	SSLv3	Kx=DH	Non-Compliant	Not Supported
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA	DHE-DSS-CAMELLIA128-SHA	[0x44]	SSLv3	Kx=DH	Non-Compliant	Not Supported

Cipher Suite Name (RFC)	Name (OpenSSL)	Cipher Suite	TLS Version	KeyExch.	FIPS	Private Key
TLS_ECDH_RSA_ WITH_AES_128_GCM_ SHA256	ECDH-RSA-AES128-GCM-SHA256	[0xc031]	TLSv1.2	Kx=ECDH/RSA	Compliant	Not Supported
TLS_ECDH_ECDSA_ WITH_AES_128_GCM_ SHA256	ECDH-ECDSA-AES128-GCM-SHA256	[0xc02d]	TLSv1.2	Kx=ECDH/ECDSA	Compliant	Not Supported
TLS_ECDH_RSA_ WITH_AES_128_CBC_ SHA256	ECDH-RSA-AES128-SHA256	[0xc029]	TLSv1.2	Kx=ECDH/RSA	Compliant	Not Supported
TLS_ECDH_ECDSA_ WITH_AES_128_CBC_ SHA256	ECDH-ECDSA-AES128-SHA256	[0xc025]	TLSv1.2	Kx=ECDH/ECDSA	Compliant	Not Supported
TLS_ECDH_RSA_ WITH_AES_128_CBC_ SHA	ECDH-RSA-AES128-SHA	[0xc00e]	SSLv3	Kx=ECDH/RSA	Compliant	Not Supported
TLS_ECDH_ECDSA_ WITH_AES_128_CBC_ SHA	ECDH-ECDSA-AES128-SHA	[0xc004]	SSLv3	Kx=ECDH/ECDSA	Compliant	Not Supported
TLS_RSA_WITH_AES_ 128_GCM_SHA256	AES128-GCM-SHA256	[0x9c]	TLSv1.2	Kx=RSA	Compliant	Supported
TLS_RSA_WITH_AES_ 128_CBC_SHA256	AES128-SHA256	[0x3c]	TLSv1.2	Kx=RSA	Compliant	Supported
TLS_RSA_WITH_AES_ 128_CBC_SHA	AES128-SHA	[0x2f]	SSLv3	Kx=RSA	Compliant	Supported
TLS_RSA_WITH_ SEED_CBC_SHA	SEED-SHA	[0x96]	SSLv3	Kx=RSA	Non-Compliant	Supported
TLS_RSA_WITH_ CAMELLIA_128_CBC_ SHA	CAMELLIA128-SHA	[0x41]	SSLv3	Kx=RSA	Non-Compliant	Supported
TLS_RSA_WITH_ IDEA_CBC_SHA	IDEA-CBC-SHA	[0x07]	SSLv3	Kx=RSA	Non-Compliant	Supported
TLS_ECDHE_RSA_ WITH_RC4_128_SHA	ECDHE-RSA-RC4-SHA	[0xc011]	SSLv3	Kx=ECDH	Non-Compliant	Not Supported
TLS_ECDHE_ECDSA_ WITH_RC4_128_SHA	ECDHE-ECDSA-RC4-SHA	[0xc007]	SSLv3	Kx=ECDH	Non-Compliant	Not Supported
TLS_ECDH_RSA_ WITH_RC4_128_SHA	ECDH-RSA-RC4-SHA	[0xc00c]	SSLv3	Kx=ECDH/RSA	Non-Compliant	Not Supported
TLS_ECDH_ECDSA_ WITH_RC4_128_SHA	ECDH-ECDSA-RC4-SHA	[0xc002]	SSLv3	Kx=ECDH/ECDSA	Non-Compliant	Not Supported
TLS_RSA_WITH_RC4_ 128_SHA	RC4-SHA	[0x05]	SSLv3	Kx=RSA	Non-Compliant	Supported
TLS_DHE_RSA_ WITH_DES_CBC_SHA	EDH-RSA-DES-CBC-SHA	[0x15]	SSLv3	Kx=RSA	Non-Compliant	Not Supported

Cipher Suite Name (RFC)	Name (OpenSSL)	Cipher Suite	TLS Version	KeyExch.	FIPS	Private Key
TLS_DHE_DSS_WITH_DES_CBC_SHA	EDH-DSS-DES-CBC-SHA	[0x12]	SSLv3	Kx=DSS	Non-Compliant	Not Supported
TLS_RSA_WITH_DES_CBC_SHA	DES-CBC-SHA	[0x09]	SSLv3	Kx=RSA	Non-Compliant	Supported
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-RSA-DES-CBC-SHA	[0x14]	SSLv3	Kx=DSS	Non-Compliant	Not Supported
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-DSS-DES-CBC-SHA	[0x11]	SSLv3	Kx=DSS	Non-Compliant	Not Supported
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-DES-CBC-SHA	[0x08]	SSLv3	Kx=DES	Non-Compliant	Supported

TLS Certificate Hashing

The Network Decoder can produce hashes of certificates that are seen in the packet stream. These hashes are the SHA-1 value of any DER-encoded certificate encountered during a TLS handshake. The hashes produced can be used to compare network traffic with hashes from public SSL blacklists, such as the one from sslbl.abuse.ch.

The TLS Certificate hashing feature is disabled by default. To enable TLS Certificate hashing:

1. Go to **ADMIN** > **Services**, select a Network Decoder service and  > **View > Explore.**
2. Select **decoder** > **parsers** > **config**.
3. In the values column next to `parsers.options`, type `HTTPS="cert.sha1=true"`.
   When this option is enabled, the SHA-1 is stored as a text value in the `cert.thumbprint` meta key.

The SHA-256 hash of the certificate can be enabled by adding the parser option:

`HTTPS="cert.sha256=true"`

to a Network Decoder's `/decoder/parsers/config/parsers.options` configuration.

When this option is enabled, SHA-256 is stored as a text value in the meta keys:

```
cert.sha256
cert.thumbprint
```

JA3 and JA3S TLS Fingerprints

The Network Decoder can produce the JA3 value of TLS clients and the JA3S value of TLS servers that are observed in a Network session. The values that are produced conform to the values generated by the open source JA3 tools (https://github.com/salesforce/ja3). The JA3 features are disabled by default.

To enable JA3 features:

1. In the NetWitness Platform User Interface, go to **ADMIN** > **Services**.
2. Select a Decoder service and click  > **View** > **Explore**.
3. In the left panel, select **decoder** > **parsers** > **config**.
4. In **parsers.options**, add `HTTPS="ja3=true ja3s=true"`.
5. In the left panel, right click on **parsers** and select **properties**.
6. In the right panel, in the lower pane, set the value in the first drop down to **Reload**.
7. Click **Send**.

JA3 and JA3S can be enabled independently using these options. When enabled, the JA3 is stored as a text value in the meta key `ja3`. The JA3S is stored as a text value in the meta key `ja3s`.

Additional meta keys are activated using a mechanism similar to the JA3S and JA3S meta items. For example, `ja3.orig` is enabled by adding `ja3.orig=true` to the HTTPS parser options. This is also true for `ja3s.orig`, `tls.extensionlen`, and `tls.cipherlen`.

`ja3.orig`, `tls.extensionlen`, and `tls.cipherlen` are only created if JA3 is enabled.

`ja3s.orig` is only created if JA3S is enabled.

Community ID

The Network Decoder generates Community ID flow hash values that are compatible with the Community ID specification defined by `https://github.com/corelight/community-id-spec`.

The Community ID is written to a text-formatted meta key named `community`.

The Community ID is generated on sessions containing TCP over IPv4, TCP over IPv6, UDP over IPV4, and UDP over IPV6.

Edit Decoder System Configuration

When a service is first added to NetWitness Platform, default values for the system configuration parameters are in effect. In most cases, the default values for compression, statistics update interval, and number of threads in the thread pool are set at a good point for optimal system performance. You do not need to edit these setting unless an RSA Customer Support technician advises you to change them.

System Configuration

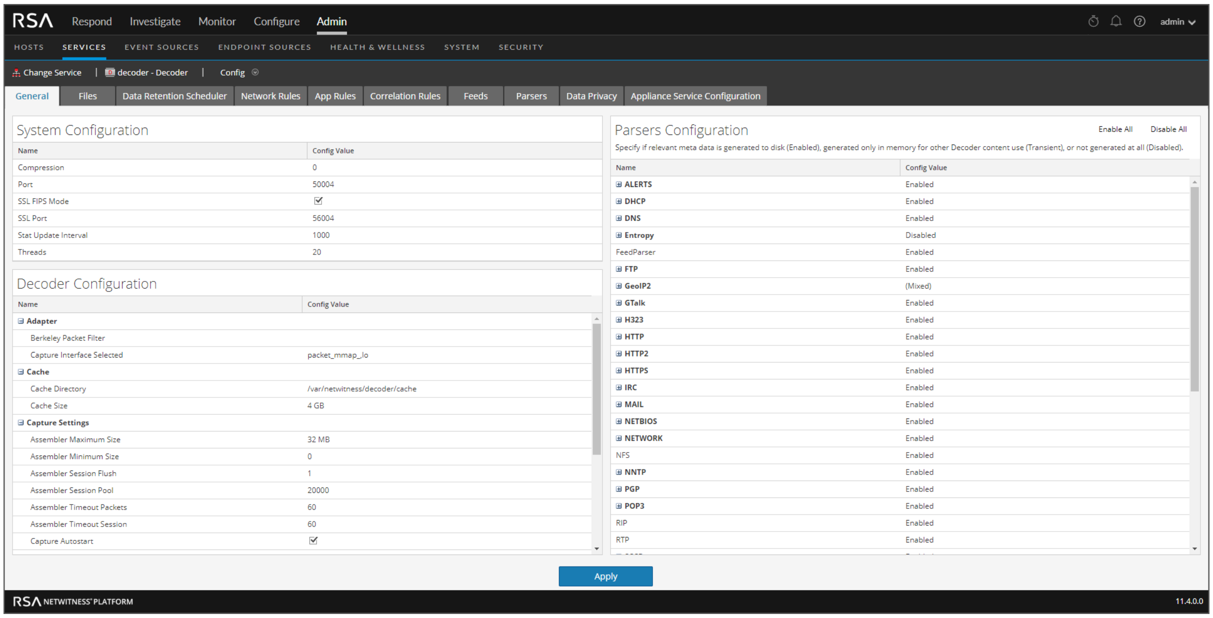
Name	Config Value
Compression	0
Port	50004
SSL FIPS Mode	☐
SSL Port	56004
Stat Update Interval	1000
Threads	20

One parameter that you may want to change for your environment is the SSL setting, which by default is not enabled. When enabled, the security of data transmission is managed by encrypting information and providing authentication with SSL certificates.

To edit system configuration parameters for a Decoder or Log Decoder:

1. Go to **Admin > Services**.
2. In the Admin > System view, select a Decoder or Log Decoder service, and select  **> View > Config**.

The Services Config view for the service is displayed with the General tab open.



3. Under **System Configuration**, click in a field that you want to edit **(Compression**, **Port**, **SSL FIPS Mode**, **SSL Port**, **Stat Update Intervals**, or **Threads**). Type a new value.

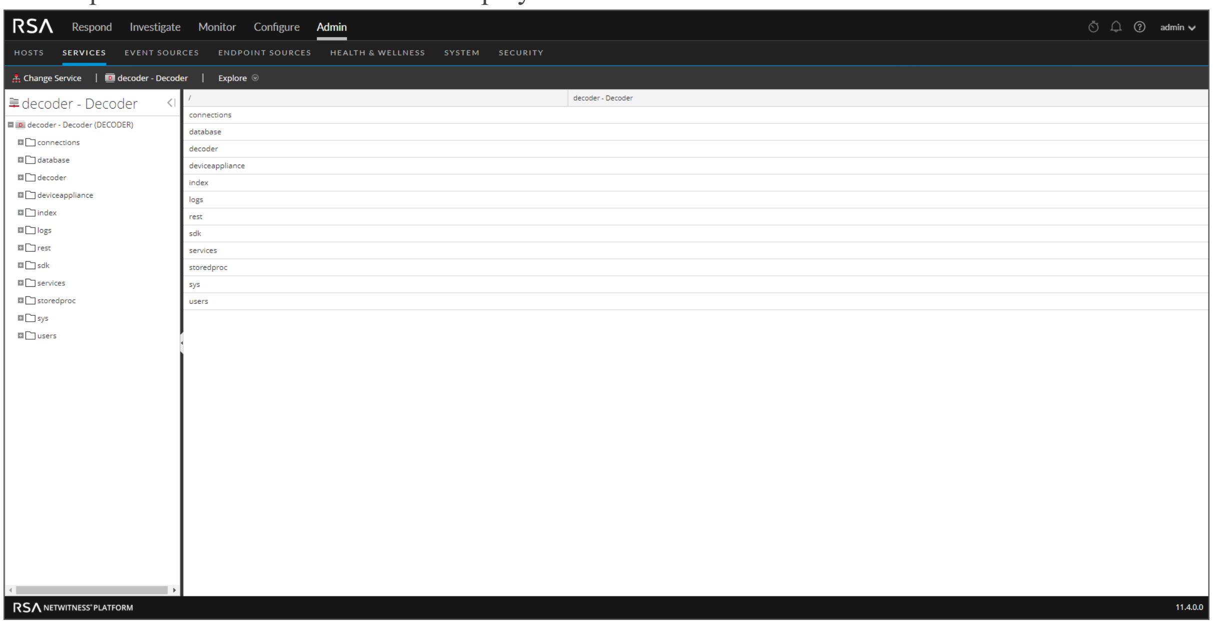
4. When finished editing, click **Apply.**
   The settings become effective immediately.

Enable CPU Usage Statistics for Installed Content

Beginning with RSA NetWitness® Platform 11.0, the Decoder provides CPU utilization statistics for all installed content, which you can use to reveal how much CPU time is used by parsers, feeds, application rules, and lexical scanning. The statistics are visible as Stat nodes in the service tree from the Explore view when `/decoder/parsers/config/detailed.stats` is enabled and the Decoder is capturing the stats.

Each piece of content is accounted as a single percentage value (0-100) regardless of the number of parse threads running. The percentage represents an average of the CPU utilization for the content across all threads.

To enable usage statistics monitoring:

1. In the NetWitness Platform User Interface, go to **ADMIN** > **Services**.
2. Select a Decoder service and click > **View** > **Explore**.
   The Explore view for the Decoder is displayed.
   
3. Open `/decoder/parsers/config/` and in the right pane, select the `detailed.stats` parameter.
4. Change the value to **enabled**. If the Decoder is not capturing data, start capture.
   When you open the Decoder Stats node in the Explore view, the new statistic is visible.

Enable Parser Mappings

This topic tells administrators how to enable event source mapping on a Log Decoder.

The Log Collector discovers the event source type on a per-message basis. If the correct parser is not identified for the event source, a small percentage of logs may be misidentified. The misclassified messages do not populate event source rules and alerts, and the reports do not have the correct data. If there are multiple event source types associated with an IP address, it makes it difficult for the parsers to identify the exact event source from which the logs are generated.

If you map an IP address to its event source type, the Log Decoder can identify the event source from which the log is generated. When messages are delivered to the Log Decoder from a mapped event source, only the assigned parsers are queried to find event matches.

You can assign event source types to IPV4, IPV6, or the hostname value of the event source. You can also assign multiple event source types to a single IP address. You can also use the Log Collector ID when different event source types with the same IP address are sent to different Log Collectors.

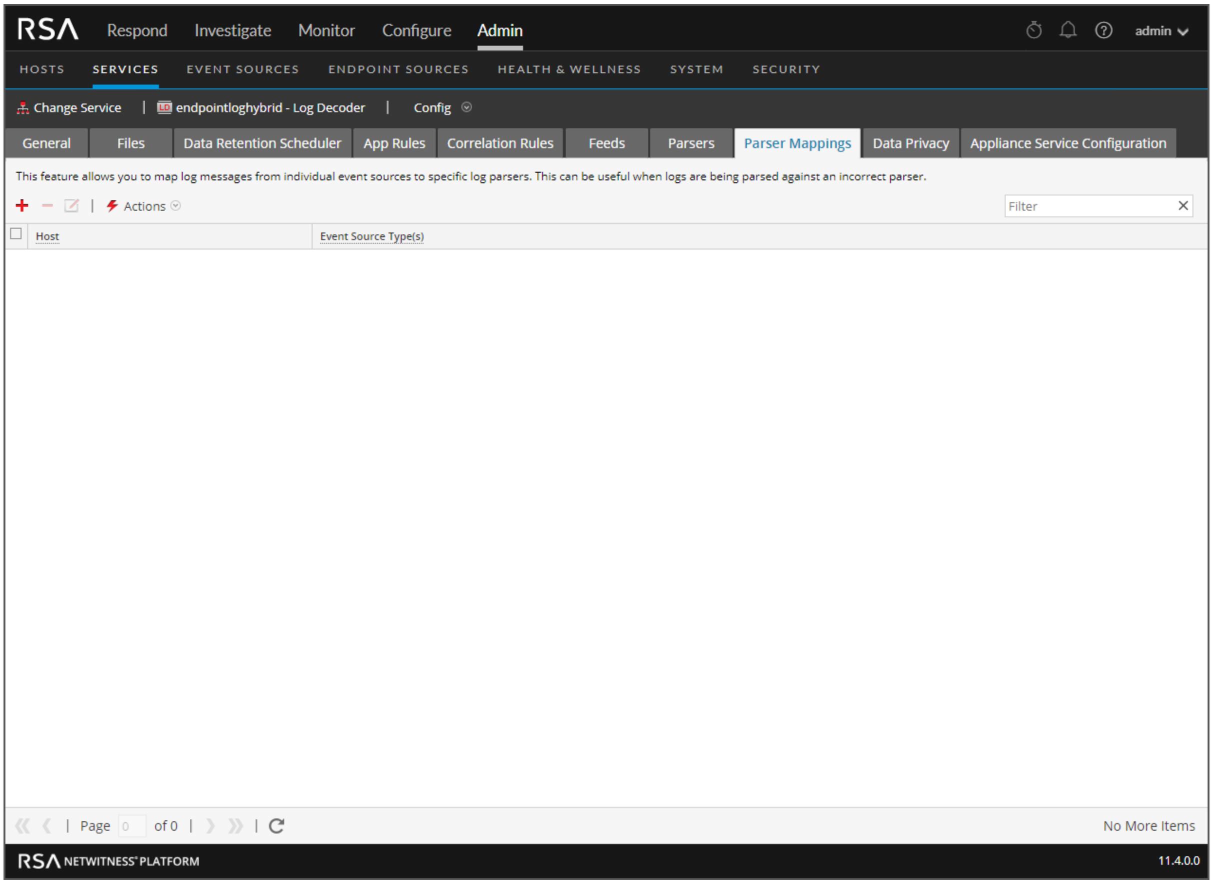
Note: You can also enable parser mapping functions by navigating to **Admin > Event Sources > Discovery.**

Enable IP Address to Event Source Mapping

To enable an IP address to event source mapping:

1. Go to **Admin > Services** and select a Log Decoder.
2. Select **> View > Config**.
3. In the Configuration page, select the **Parser Mappings** tab.

The Parser Mappings tab is displayed in the Services Config view.



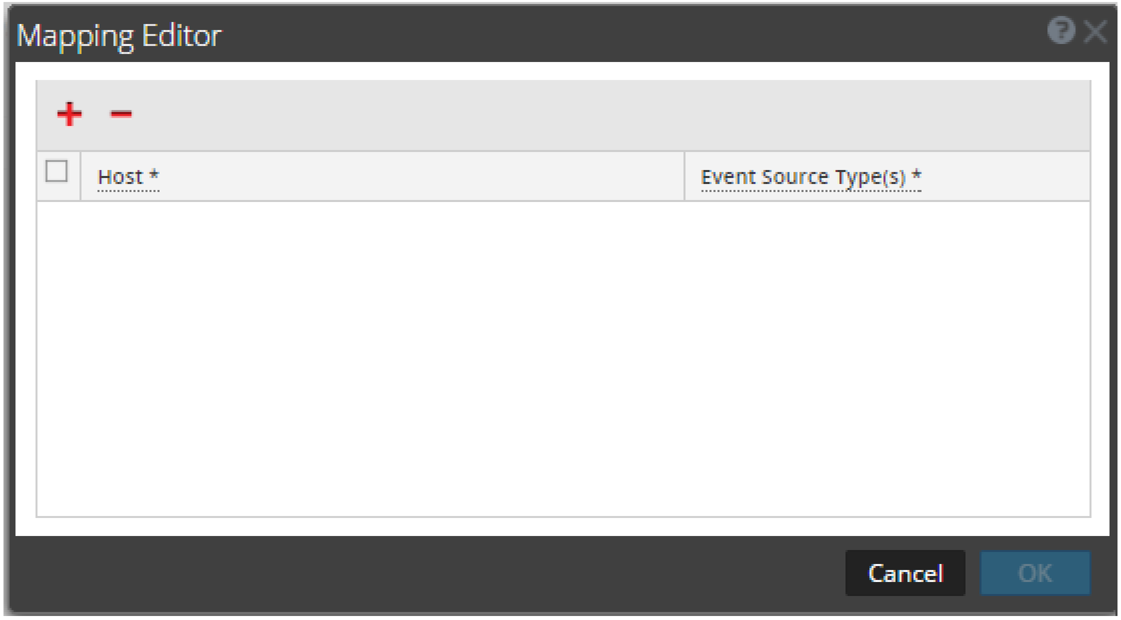
Update IP to Event Source Mapping

To update an IP to event source mapping:

1. Go to **Admin** > **Services**.
2. Select a **Log Decoder**, and in the **Actions** column, select > **View** > **Config**.

   The Services Config view is displayed.
3. Select the **Parsers Mapping** tab.
4. Click .

   The Mapping Editor is displayed.



5. Any of the following mappings can be defined:
   - **One Host and One Event Source Type**

     In the **Host** field, enter the hostname.

     For example: `10.0.0.1`
   - In the **Event Sources(**s) field, enter the event source type.

     For example: `apache`
   - **One Host and One or More Event Source Types**

     In the **Host** field, enter the hostname.

     For example: `10.0.0.1`
   - In the **Event Source(s)** field, enter the event source type.

     For example: `apache,sap,aix`
   - **One Host, One Log Collector, and One Event Source Type**

     In the **Host** field, enter the hostname and Log Collector ID.

     For example: `10.0.0.1,LC-1`
   - In the **Event Source(s)** field, enter the event source type.

     For example: `apache`
   - **One Host, One Log Collector ID, and One or More Event Source Types**

     In the **Host** field, enter the hostname and Log Collector ID.

     For example: `10.0.0.1,LC-1`
   - In the **Event Source(s)** field, enter the event source type.

     For example: `apache,sap,aix`

Note: The event source types are processed in the order you enter the parsers and if one or more parsers matches a log, the first parser in the list is queried. The Host/IP can be IPv4, IPv6, or Hostname.

6. Click **OK**.

   The Parser Mapping is added.

7. To cancel the parser mappings selection, click **Cancel**.

Read IP to Event Source Type Mappings

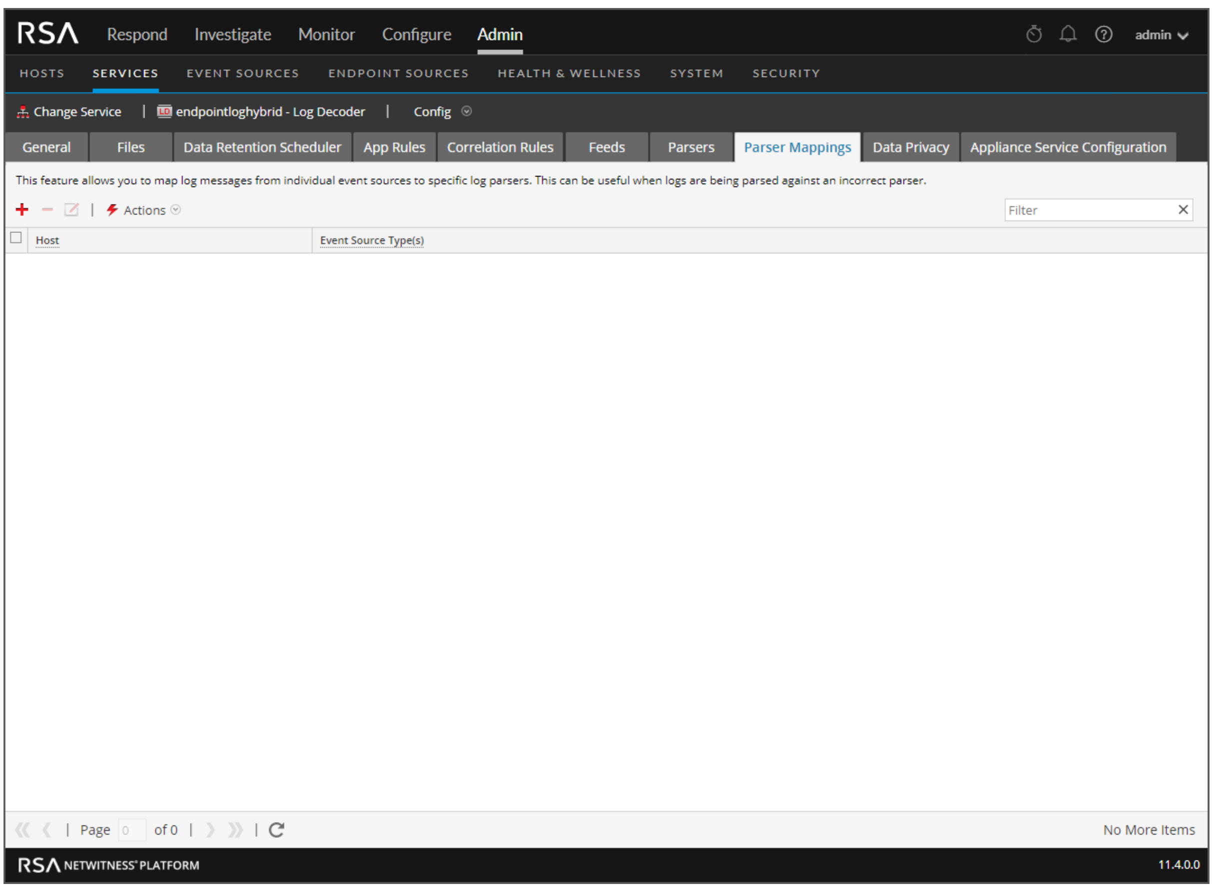
To read an IP to event source type mappings:

1. Go to **ADMIN > Services**, and select a Log Decoder service.
2. In the Actions column, select **> View > Config**.

   The Services Config view is displayed.

3. Select the **Parsers Mapping** tab.

   The mappings are displayed.



Edit IP to Event Source Type Mappings

To edit IP to event source type mappings:

1. Go to **ADMIN** > **Services**, and select a Log Decoder service.
2. In the Actions column, select > **View** > **Config**.

   The Service Config view is displayed.
3. Select the **Parser Mappings** tab.
4. Select the mapping you want to edit.

   Note: You can only edit one mapping at a time.
5. Click .
6. In the **Event Source(s)** field, modify the event source(s).

   Note: The host is not editable and the field is disabled.
7. Click **OK** to accept the edited Event Source.
8. To cancel the changes, click **Cancel**.

Delete IP to Event Source Type Mappings

To delete IP to event source type mappings:

1. Go to **ADMIN** > **Services**, and select a Log Decoder service.
2. In the Actions column, select > **View** > **Config**.

   The Service Config view is displayed.
3. Select the **Parser Mappings** tab.
4. Select the mapping you want to delete.
5. Click .

   The mapping is deleted and the grid is refreshed.
6. To cancel the changes, click **Cancel**.

Sort the Hostname or Event Source Type

To sort the hostname or event source type:

1. Go to **ADMIN** > **Services**, and select a Log Decoder service.
2. In the Actions column, select > **View** > **Config**.

   The Service Config view is displayed.

3. Select the **Parser Mappings** tab.
4. To sort a column, click in the column header.

Event Source Types are applied for your selected IP address. Logs are parsed against the parsers in the order they are listed.

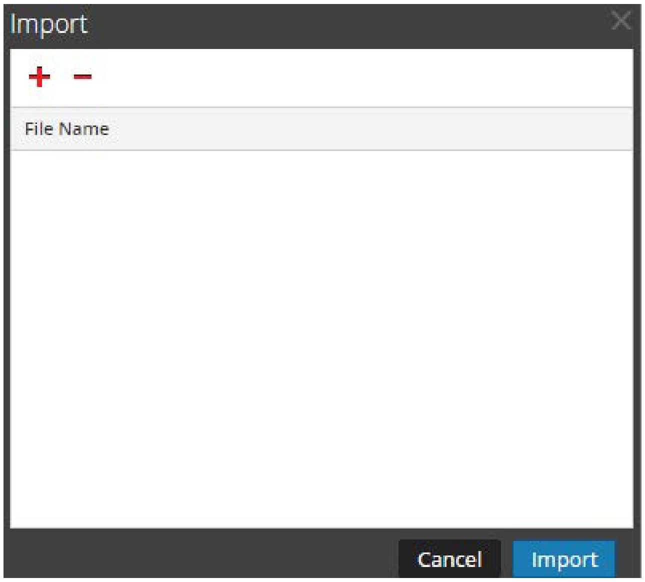
Import IP to Event Source Mapping Entries

To import IP to event source mapping entries:

1. Go to **ADMIN** > **Services**, and select a Log Decoder service.
2. In the Actions column, select > **View** > **Config**.

   The Service Config view is displayed.
3. Select the **Parser Mappings** tab.
4. Select **Actions** > **Import**.

   The Import dialog is displayed.

5. Click .
6. Select the file you want to import and click **OK**.
7. To load the parser, click **Import**.

Note: You can only import one .csv file at a time.

Export IP to Event Source Mapping Entries

To export IP to event source mapping entries:

1. Go to **ADMIN** > **Services**, and select a Log Decoder service.
2. In the Actions column, select > **View** > **Config**.

   The Service Config view is displayed.
3. Select the **Parser Mappings** tab.
4. Select the mappings you want to export.
5. Select **Actions** > **Export** > **Selection**.

   The Export Selection dialog is displayed.

6. Enter the file name and click **Export**.

Search IP to Event Source Mapping Entries

To search IP to event source mapping entries:

1. Go to **ADMIN** > **Services**, and select a Log Decoder service.
2. In the Actions column, select > **View** > **Config**.

   The Service Config view is displayed.
3. Select the **Parser Mappings** tab.
4. In the Parsers Mappings toolbar, enter the Host or Event Source in the **Filter** field.
5. Click **Enter**.

The Hosts or Event Sources that match the names entered in the **Filter** field are displayed.

Enable or Disable Lua and Flex Parsing Systems

This topic tells administrators how to enable or disable Lua and Flex parsing systems on a Decoder or Log Decoder. Flex parsers are deprecated and disabled by default.

The settings to enable or disable Lua and Flex parsing systems are configured correctly by default and you do not typically have to change them. However, you may need to adjust these settings at the request of RSA Customer Care or for troubleshooting purposes.

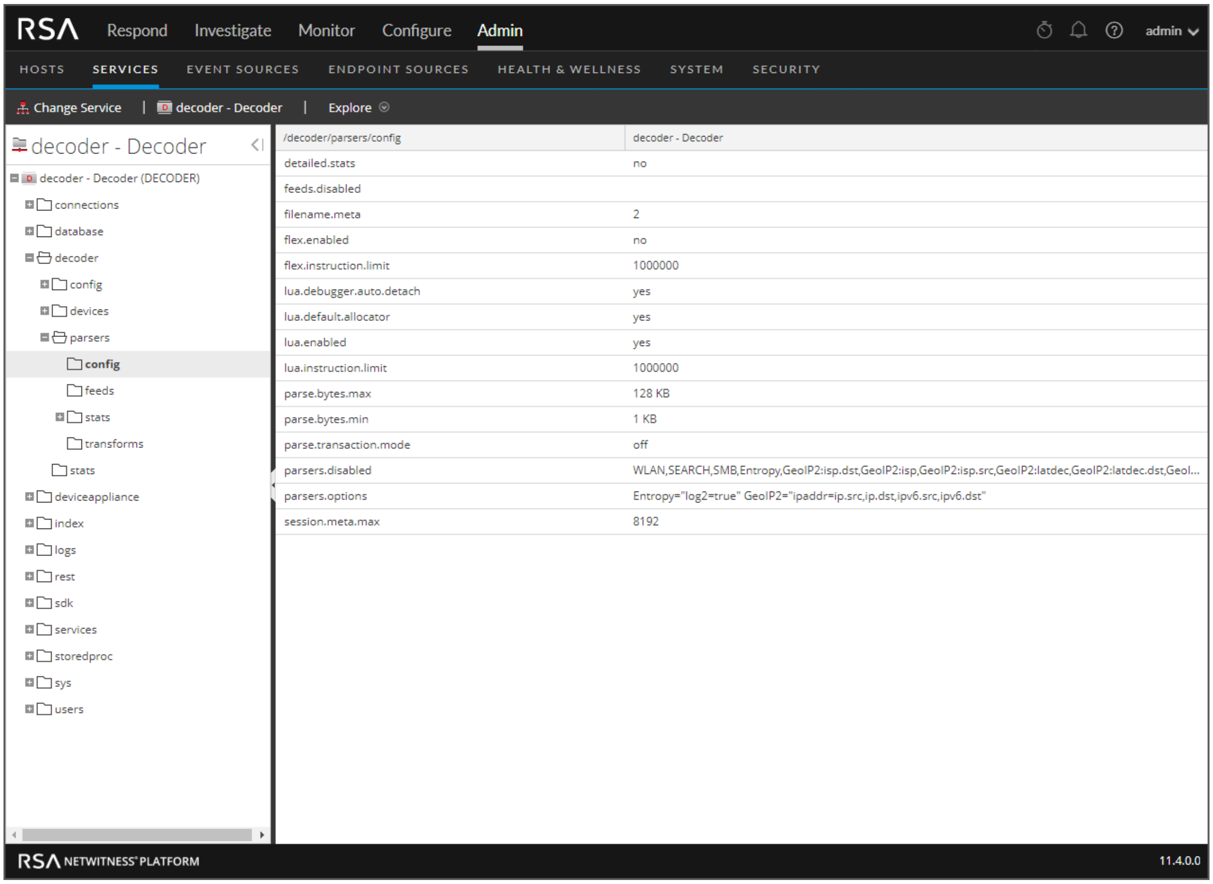
In addition to configuring individual parsers, you can enable and disable all Lua parsing as well as all Flex parsing in the Services Explore view. You enable and disable the Lua parsing and Flex parsing systems settings separately, but they work in the same way.

- If you **disable** the Lua or Flex parsing system, the corresponding parsing system is disabled and no parsers are loaded.
- If you **enable** the Lua or Flex parsing system, the corresponding parsing system is enabled and individual parsers are enabled and disabled following the current individual configurations.

To enable or disable Lua and Flex parsing systems on a Decoder or Log Decoder:

1. Go to **ADMIN > Services.**
2. Select a Decoder or Log Decoder and ⚙ **> View > Explore.**
   The Services Explore view for the selected service is displayed.

3. In the Node list, navigate to and select **`/decoder/parsers/config`**.



4. In the values panel:

- To enable the Lua parsing system, in the value field for `lua.enabled`, type **yes**.
- To disable the Lua parsing system, in the value field for `lua.enabled`, type **no**.
- To enable the Flex parsing system, in the value field for `flex.enabled`, type **yes**.
- To disable the Flex parsing system, in the value field for `flex.enabled`, type **no**.

Map IP Address to Service Type for Log Parsing

This topic describes the procedure to map an IP address to a service type for log parsing.

The Log Collector discovers event source type on a per-message basis. If the correct parser is not used for the specific event source, the messages that are common between event source types are misclassified. The misidentified messages will not populate service rules and alerts, and the reports will not have proper information. Also, if there are multiple services associated with an IP address, it can be difficult for the parsers to identify the exact service from which the log is generated.

If you map an IP address to its services, the log decoder can identify the service from which the log is generated. When messages come into the log decoder from a mapped service, the assigned parsers are loaded to find event matches.

You can assign service types to IPV4, IPV6 or hostname value of the event source. You can also assign multiple service types to a single IP address. You can also use the CollectorID when different service types with the same IP address are sent to different collectors.

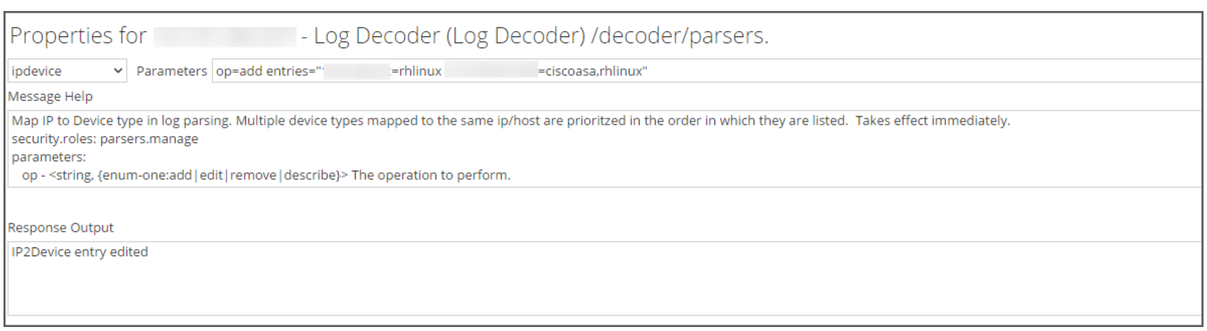
Map an IP Address to a Service Type

To map an IP address to a service type, do the following:

1. Go to **ADMIN** > **Services**.
2. In the **Services** view, select a Log Decoder, and in the **Actions** column, select > **View** > **Explore**.
3. Go to **/decoder/parsers** node, right-click **parsers**, and select **Properties**.
4. In the **Properties** view, specify the **ipdevice** command with the following parameters:

   `op=add/remove entries="`***`ipaddress`***`=`***`service`***`"`

   for example, `op=add entries="10.100.201.300=ciscoasa"`
5. Click **Send**.



IPdevice Command

In the `ipdevice` command, three operations are available:

- `add`: This operation adds or updates entries in the ipdevice map. Multiple space delimited address/type pairs may be specified.

  `op=add entries="`***address***`=`***service type***`"`

- `remove`: This operation removes entries from the ipdevice map. Multiple space delimited address/type pairs may be specified.
  `op=remove entries="`***address***`"`

- `describe`: This operation returns the values currently in the ipdevice map.

Map an IP Address to a Time Zone

This topic discusses RSA NetWitness® Platform support for Event Time.

Often times logs do not fully specify timestamps and may be missing time zone information. To properly normalize such timestamps to UTC, the Log Decoder provides the ability to associate devices from a specific address (IPv4 or IPv6) or hostname to a time zone or a fixed offset.

Timestamp consideration for log files:

- Some logs are in UTC format.
- Some logs that are not in UTC include a timezone offset value to adjust the time accordingly.
- For logs that use local time, with no offset provided, you can create a source mapping to manually adjust times for logs from that event source.

Three time zone formats are currently accepted and are shown in the following examples:

- Olson format: `America/Anguilla`
- POSIX format: `AST2:45ADT0:45,M4.1.6/1:45,M10.5.6/2:45`
- Offset by Minutes format: `= -500`

NetWitness Platform maps the device address (IPv4 or IPv6) or hostname to a specific time zone or offset. Event time meta that is parsed from a log that is from a mapped address and does not include an offset or time zone as part of the timestamp is adjusted to UTC according to the mapping.

To map an IP address to a time zone, do the following:

1. Go to **ADMIN > Services**.
2. In the **Services** view, select a Log Decoder, and in the **Actions** column, select **> View > Explore**.
3. Go to **/decoder/parsers** node, right click **Parsers**, and select **Properties**.
4. In the **Properties** view, specify the `iptmzone` command with the following parameters:

   `op=add entries="`***ipaddress***`=`***timezone***`"`

   for example: `op=add entries="10.10.10.10=Africa/Addis Ababa"`

5. Click **Send**.

tzinfo Command

You can view the strings that can be used to specify the time zone by using the `tzinfo` command.

To view the time zone strings, do the following:

1. Go to **ADMIN > Services**.
2. In the **Services** view, select a Log Decoder, and in the **Actions** column, select **> View > Explore**.
3. Go to **/decoder/parsers** node, right click **Parsers**, and select **Properties**.
4. In the **Properties** view, specify the `tzinfo` command with the following parameters:

   `op=tznames`

5. Click **Send**.

The list of time zone strings is returned in the **Response Output** text box.

iptmzone Command

In the `iptmzone` command, three operations are available:

- `add`: This operation adds or updates entries in the iptmzone map. Multiple space delimited address/type pairs may be specified.

  op=add entries="***address***=***time zone***"

- `remove`: This operation removes entries in the iptmzone map. Multiple space delimited address/type pairs may be specified.

  op=remove entries="***address***"

- `describe`: This operation returns the values currently in the iptmzone map.

Examples

The following examples provide instances for mapping IP addresses to time zones:

- If you want to map two different entries with different IPV4 values and time zone, enter the following parameter in the **iptmzone** command and click **Send**

  ```
  op=add entries=”10.10.10.10=America/Anguilla 10.10.10.11=Pacific/Rarotonga”
  ```

- If you want to remove an entry for a single IPV4 value and time zone, enter the following parameter in the **iptmzone** command and click **Send**.

  ```
  op=remove entries="10.5.245.9"
  ```

- If you want to create a single entry for an IPV6 value and time zone, enter the following parameter in the **iptmzone** command and click **Send**.

  ```
  op=add entries=”2001:DB8:85A3::8A2E:370:7334=America/Anguilla”
  ```

- If you want to create a single entry to map an IPV4, IPV6, or hostname with the Minute Offset, Olson, or POSIX format, enter the following string in the **iptmzone** command and click **Send**.

  ```
  op=add entries="10.168.0.2=America/Anguilla
  2001:DB8:85A3::8A2E:370:7334=0500 nwappliance21=EST5EDT,M3.2.0/2,M11.1.0"
  ```

Change the Date format

The Log Decoder parsing engine uses the `event.time` meta key to keep track of the time an event occurs for incoming log messages. You can use the `mdformat` for the **iptmzone** command to change the date format. The Log Decoder currently has the ability to change the dates in the logs to have the following format:

```
mdy or dmy (month/day/year or day/month/year)
```

Notes:

- Event time metadata in a parser does not account for `dmy` and `mdy`.
- Currently, there is no functionality to detect this scenario and adjust parsing automatically.

To change the date format, do the following:

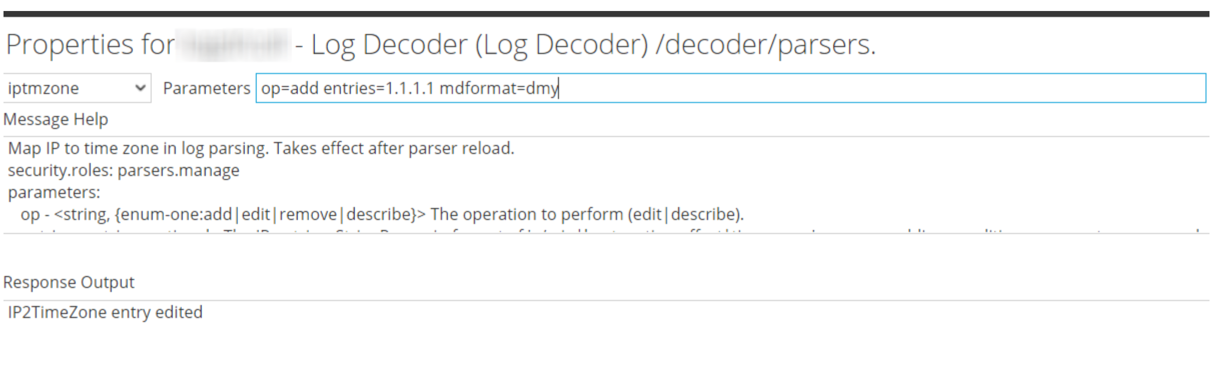
1. Go to **ADMIN > Services**.
2. In the **Services** view, select a Log Decoder, and in the **Actions** column, select; **> View > Explore**.
3. Go to **/decoder/parsers** node, right click **Parsers**, and select **Properties**.
4. In the **Properties** view, select **iptmzone** from the drop-down list, and specify the command with the following parameters:

   `op=add entries="<ipaddress>" mdformat=[dmy | mdy | none]`

   for example: `op=add entries="1.1.1.1" mdformat=dmy`

   Where:

   - `ipaddress` is the Device IP address, and
   - `mdformat` can be **dmy**, **mdy**, or **none**.

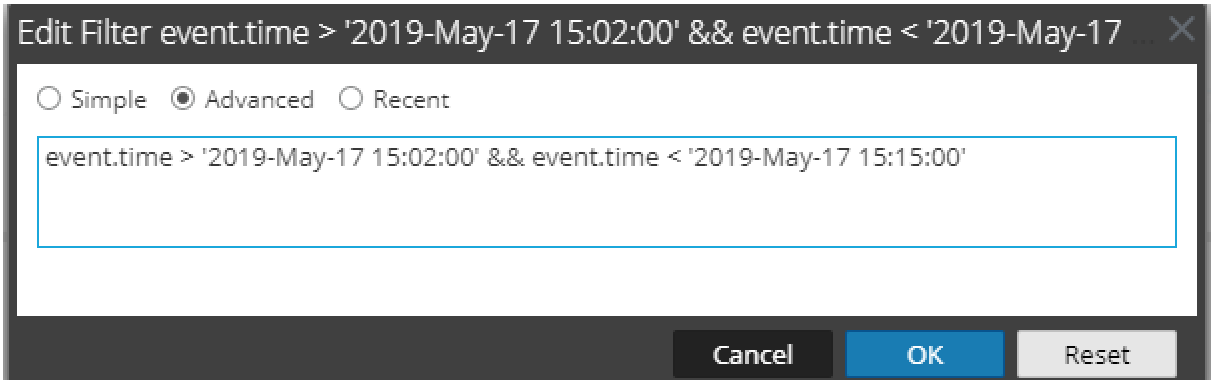


5. Click **Send**.

NetWitness Platform maps the IP address along with the date format in the Log Decoder. Event time meta items are updated according to their respective mappings.

Event Time Filter Example

If you want to see logs that have an event time between 3:02 PM and 3:15 PM on May 17, 2019, create the following filter:



> **Note:** The query builder may tell you "invalid expression," but that is a validation error in the User Interface. The query works because even though this is a text (string) search, the system actually compares the ASCII values. So, for example the expression **A < B** would be true.

Below is a sample of the event meta values returned from the query:

Missing Year Support

If there is no year associated with a format string, the parser attempts to populate the year heuristically. In most cases, the value is populated based on the current year. The current year is assigned as the message year (UTC), as per the clock on the Log Decoder.

The following are the various other scenarios:

1. Latent log during transition to new year.
2. Logs from forward time zones (or skewed clocks) just before new year transition.
3. Latent logs received where a leap day cannot be successfully assigned to an appropriate leap year.

Latent log during transition to new year

If the day is more than 31 days into the future, the year is decremented.

For example,

The message date is Dec-31. The current date is 2018-Jan-1. The temporary assigned date, 2018-Dec-31, will be more than 31 days into the future.

This will cause the year component to be decremented to 2017, resulting in a reasonable message time.

Logs from forward time zones (or skewed clocks) just before new year transition

If the day is more than 334 days into the past, the year is incremented.

For example,

The message date is Jan-1. The current date is 2017-Dec-31. The temporary assigned date, 2017-Jan-1, will be more than 334 days into the past.

This will cause the year component to be incremented to 2018, resulting in a reasonable message time.

Latent logs received where a leap day cannot be successfully assigned to an appropriate leap year

If the day (Feb 29) is invalid (say, the assigned year is NOT a leap year), the relative position to the current time cannot be calculated. To do this, the year is decremented in an attempt get a valid time stamp.

Because the position of the leap day relative to the transition to the new year, along with the expectation that no logs would be received this far into the future, we do not ever make an attempt to increment the year to produce a valid time stamp.

After the year is decremented, the same logic is then followed (refer to statement 1 and 2). If the result is still an invalid day. The parser throws and a valid message time cannot be parsed.

Limitations

Note: Currently, there is no mechanism to assign a year to the import of logs.

As this pertains to leap days, if you find the correct leap year, an inconsistent data would result. For example, if a set of messages is two years old and during a leap year, only a single day would remain which is accurate. Because of the heuristic described above, the remaining set of messages would have time stamps one year too.

For data consistency, usually you cannot find valid leap years beyond 1 month and less than 11 months.

Examples

The following examples provide instances for logs with year and logs without year:

- If the log is with year, event time is generated as below:

```
%emcavamar: 1704^^2017-04-07^^07:50:02^^1^^<event-source NodeID="avamar"
ProgramName="com.avamar"/>^^1270626^^SYSTEM^^ERROR^^OK^^MCS:DPN_
Proxy^^Internal server error^
```

```
<MESSAGE

     id1="1:01"

     id2="1"

     eventcategory="1605020000"

     functions="&lt;@msg:*PARMVAL($MSG)&gt;&lt;@event_time:*EVNTTIME
     ($MSG,'%W-%G-%F %H:%U:%O',fld2,fld3)&gt;"

     content="&lt;fld1&gt;^^&lt;fld2&gt;^^&lt;fld3&gt;^^&lt;fld4&gt;^^
     &lt;&lt;event-source NodeID=&quot;&lt;hostname&gt;&quot;
     ProgramName=&quot;&lt;fld8&gt;&quot;/&gt;^^&lt;fld9&gt;^^&lt;cate
     gory&gt;^^&lt;severity&gt;^^&lt;event_
     type&gt;^^&lt;agent&gt;^^&lt;event_description&gt;" />
```

- If the log is without year, event time is still generated, The current year is assigned as the message year (UTC), as per the clock on the Log Decoder.

```
%emcavamar: 1704^^04-07^^07:50:02^^1^^<event-source NodeID="avamar"
ProgramName="com.avamar"/>^^1270626^^SYSTEM^^ERROR^^OK^^MCS:DPN_
Proxy^^Internal server error^
```

```
<MESSAGE

     id1="1:01"

     id2="1"

     eventcategory="1605020000"

     functions="&lt;@msg:*PARMVAL($MSG)&gt;&lt;@event_time:*EVNTTIME
     ($MSG,'%G-%F %H:%U:%O',fld2,fld3)&gt;"

     content="&lt;fld1&gt;^^&lt;fld2&gt;^^&lt;fld3&gt;^^&lt;fld4&gt;^^
     &lt;&lt;event-source NodeID=&quot;&lt;hostname&gt;&quot;
     ProgramName=&quot;&lt;fld8&gt;&quot;/&gt;^^&lt;fld9&gt;^^&lt;cate
     gory&gt;^^&lt;severity&gt;^^&lt;event_
     type&gt;^^&lt;agent&gt;^^&lt;event_description&gt;" />
```

Obtain Log Files from a Pre-11.0 Log Decoder

NetWitness 11.0. added the capability to view a small sampling of recent logs for specific devices through detail tabs of the Discovery View. By default, Log Decoders prior to 11.0 do not have the necessary configuration to enable this feature, but a few minor changes can make it available.

To enable logs preview for a pre-11.0 Log Decoder, follow these steps on the Log Decoder:

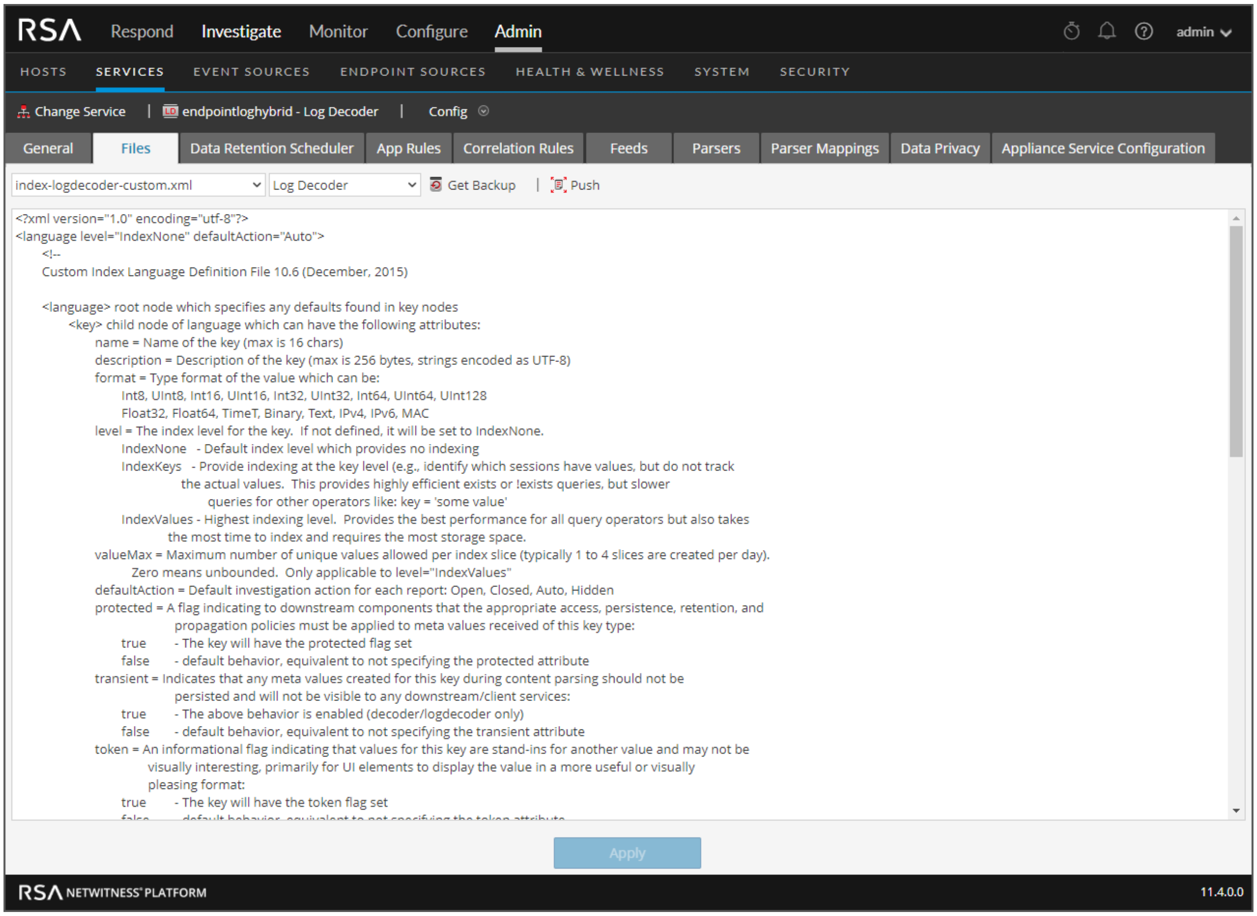
1. Go to **Admin** > **Services** > select a **Log Decoder**, then select > **View** > **Config**.
2. Click the **Files** tab and select **index-logdecoder-custom.xml** from the drop-down menu.
3. Add the following three lines at the end of the file (before the closing language tag):

```
<key description="Device IP" level="IndexValues" name="device.ip" format="IPv4" valueMax="100000"
defaultAction="Open"/>
<key description="Device IPv6" level="IndexValues" name="device.ipv6" format="IPv6"
valueMax="100000" defaultAction="Open"/>
```

```
<key description="Device Host" level="IndexValues" name="device.host" format="Text"
valueMax="100000" defaultAction="Open"/>
```

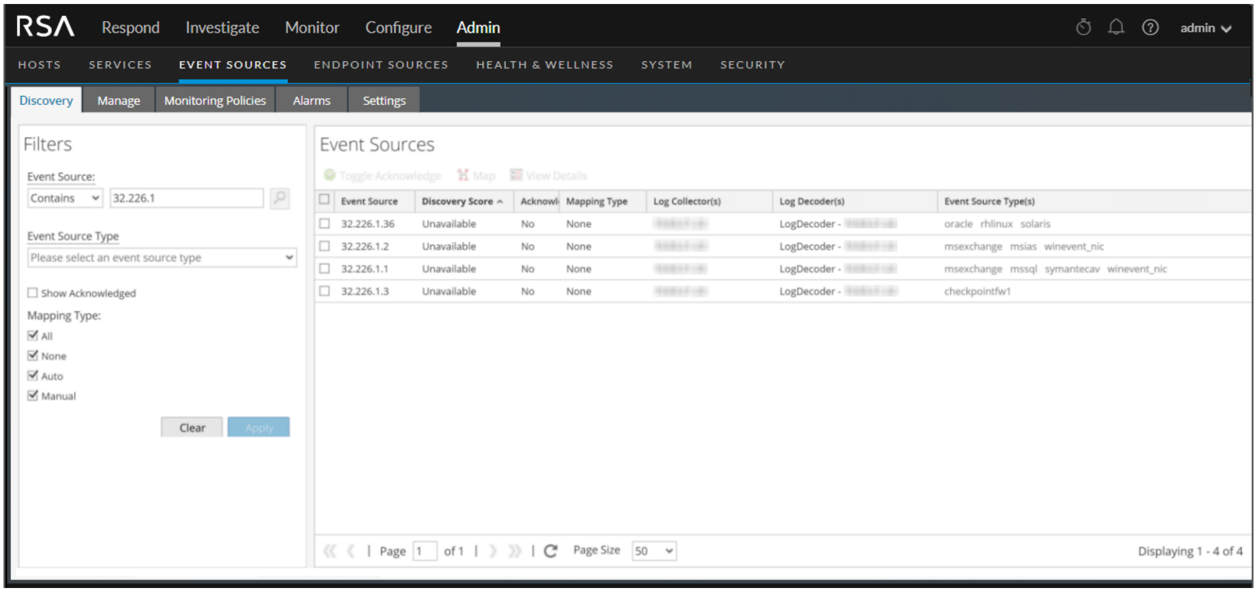
4. Click **Apply**.
5. Restart the Log Decoder service as follows.
   Select Log Decoder service > **Explore** > **decoder** > **Properties** > **reset**. You select **reset** from a drop down menu. Click **Send** after you select reset.

This is an example of the **index-logdecoder-custom.xml** file.



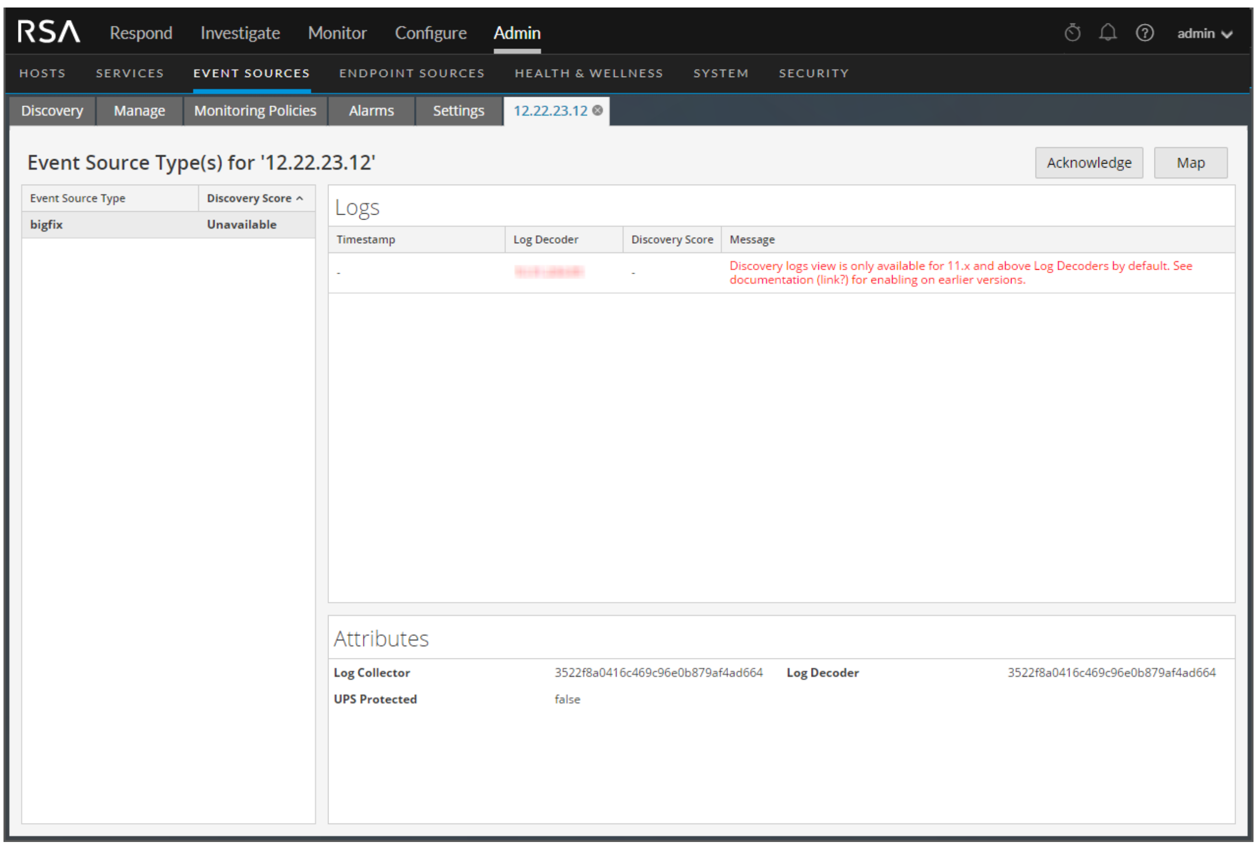
Note: Discovery Scores are only available for 11.x and above Log Decoders. Discovery Scores for pre-11.x Log Decoders are displayed as Unavailable.

The following example shows the Discovery Score as **Unavailable** in the **Details** view for a pre-11.0 Log Decoder.



Note: Device logs are only available for 11.x and above Log Decoders.

The following example shows the message that is displayed in the Logs panel for a pre-11.0 Log Decoder.



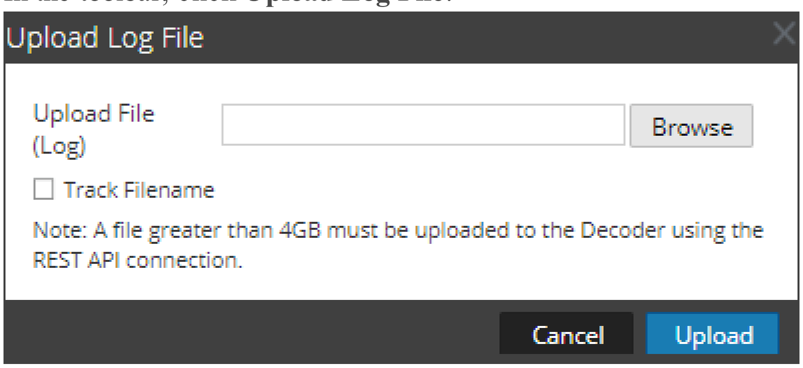
Upload a Log File to a Log Decoder

There are occasions when you want to analyze a log file that is not available on the service you are using. You can upload a log file captured on another service to NetWitness Platform. Log filenames are of the type **.log**.

When a log file is uploaded to a Log Decoder, the Log Decoder analyzes and generates meta for each log it contains. These logs are added to the already decoded logs on the Log Decoder and are available for analysis. NetWitness Platform includes a filename tracking option that makes searching for a particular set of logs easier. When the log file is uploaded with file tracking, the Log Decoder adds meta to each log based on the uploaded filename. You can then filter sessions for analysis using that meta.

The option to upload a log file is dimmed when other Log Decoder operations prevent an upload from occurring, for example, when the Log Decoder is capturing logs.

To import a log file to a Log Decoder:

1. Go to **ADMIN** > **Services**.
2. Select a Log Decoder in the **Service** grid, and select > **View** > **System**.
   The Services System view for the Log Decoder is displayed.
3. In the toolbar, click **Upload Log File**.
   

4. To choose a log file, click **Browse**.
   A directory view is displayed.
5. Select the log file that you want to upload.
   The filename is displayed in the **Upload File** field.
6. If you want the Log Decoder to add meta to the logs based on the filename, click the checkbox next to **Track Filename**.
7. To upload the file, click **Upload**.
   The selected file is uploaded and a status message indicates that the file is uploaded. The log file is available for analysis.

Upload a Packet Capture File

There are occasions when you want to analyze a packet capture file that is not available on the service you are using. You can upload a file captured on another service to NetWitness Platform. Supported packet capture file types are `pcap` and `pcap.gz`.

When a packet capture file is uploaded to a Decoder, the Decoder creates sessions from the packet capture file packets. These sessions are added to the already decoded sessions on the Decoder and are available for analysis. NetWitness Platform includes a filename tracking option that makes searching for a particular set of sessions easier. When the packet capture file is uploaded with file tracking, the Decoder adds meta to the sessions based on the uploaded filename. You can then filter sessions for analysis using that meta.

The option to upload a packet capture file is dimmed when other Decoder operations prevent an upload from occurring; for example, when the Decoder is capturing packets.

To select and upload a packet capture file:

1. Go to **ADMIN** > **Services**.

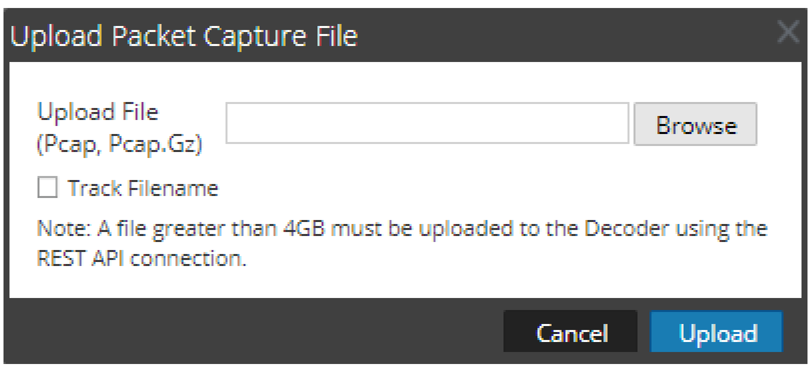
   The Administration Services view is displayed.

2. Select the Decoder name, and > **View** > **System**.

   The Services System view for the Decoder is displayed.

3. In the toolbar, click **Upload Packet Capture File**.

   The **Upload Packet Capture File dialog** is displayed.

4. To choose a capture file, click **Select**.

   A directory view is displayed.

5. Browse the directory and select the packet capture file that you want to upload.

   The filename is displayed in the **Upload File(pcap,pcap.gz)** field.

6. If you want the Decoder to add meta to the sessions based on the filename, click the checkbox next to **Track Filename**.

7. To upload the file, click **Upload**.

   A progress bar shows upload progress.

Upload time varies depending on the size of the file. When the file upload is complete, a status message is displayed. The file is now available for investigation.

Decoder and Log Decoder References

This is a collection of references, which provide information about the user interface for Decoders and Log Decoders in NetWitness Platform, with references to the procedures that describe the work you can do in that part of the user interface.

Topics

- Services Config View - Data Privacy Tab
- Services Config View - Data Retention Scheduler
- Services Config View - Feeds Tab
- Services Config View - Files Tab
- Services Config View - General Tab
- Services Config View - Parsers Tab
- Services Config View - Parser Mappings Tab
- Services Config View - Rules Tabs
- Services System View - Decoders

Services Config View - Data Privacy Tab

In the Data Privacy tab (**Admin > Services > Select a Decoder or Log Decoder >** ⚙ ⌄ **> Config > Data Privacy tab)**, administrators can configure data privacy parameters for certain Core services. For the Decoder and Log Decoder, you can set the default hash algorithm and salt.

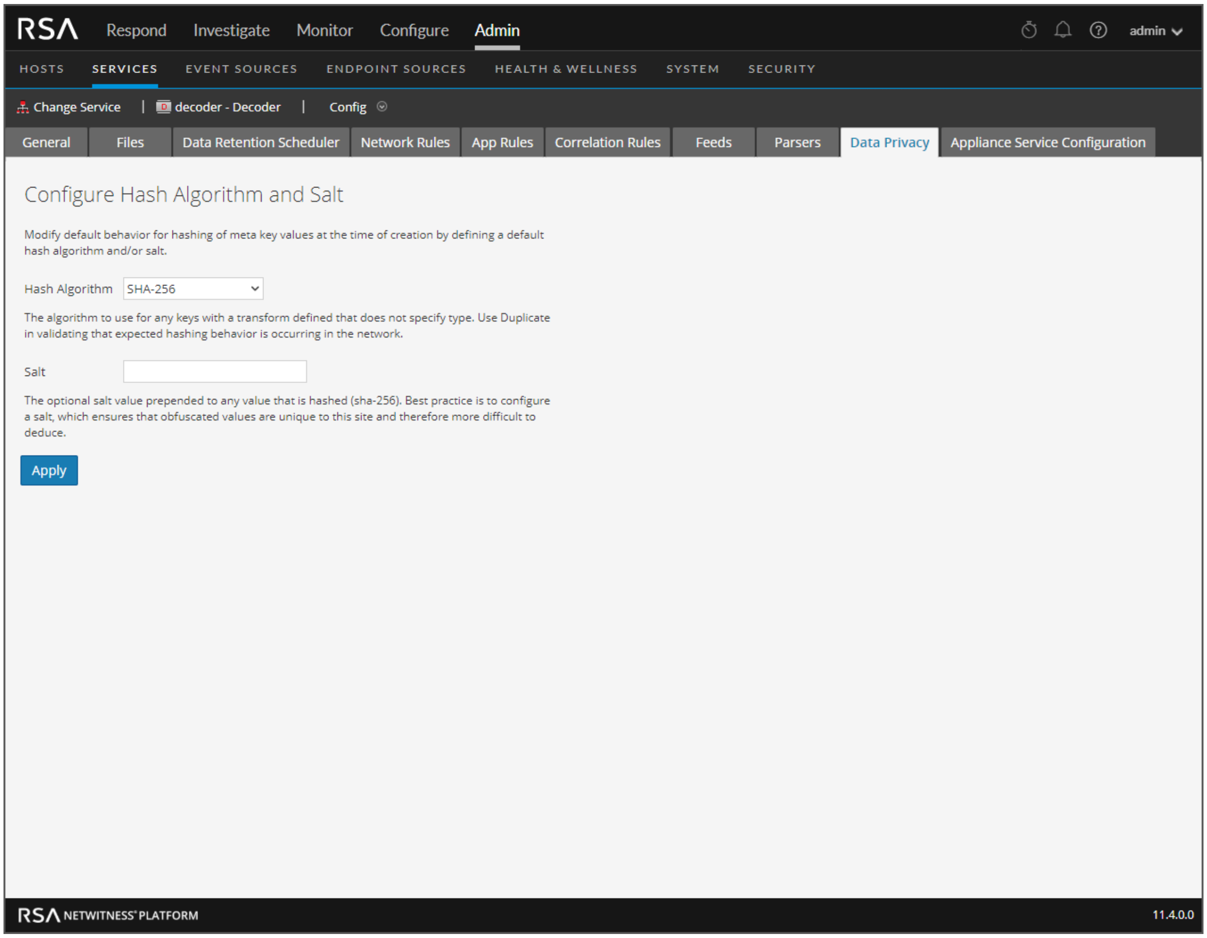
What do you want to do?

User Role	I want to ...	Documentation
Administrator	configure hash algorithm and salt	"Configure the Hash Algorithm and Salt" in the *Data Privacy Management Guide*. (Go to the Master Table of Contents to find all RSA NetWitness Platform 11.x documents.)

Related Topics

- Decoder and Log Decoder Quick Setup
- Configure Common Settings on a Decoder

Quick Look



The Data Privacy tab has the Configure Hash Algorithm and Salt configuration settings. The following table describes the parameters in this tab.

Parameter	Description
Hash Algorithm	Displays a drop-down list of hash algorithms to use for any keys with a transform that does not specify algorithm type. Possible values are SHA-256 and Duplicate. Duplicate is a special algorithm available for administrators to use when validating that expected hashing behavior is occurring in the network.
Salt	Indicates the optional salt value prepended to any value that is hashed. Best practices for security purposes dictate a salt value that is no less than 100 bits or 16 characters in length. Configuring a value ensures that obfuscated values are unique to this site and therefore more difficult to deduce. For more information on this field, see "Configure Data Obfuscation" in the *Data Privacy Management* guide.
Apply	Applies any changes.

Services Config View - Data Retention Scheduler

In the Data Retention Scheduler tab, you can set the rollover criteria for removing database records from primary storage using an age-based threshold. You can also schedule the timing to check whether the threshold is reached.

To access the Data Retention Scheduler tab, go to **Admin** > **Services** > select a **Decoder** or **Log Decoder** service and click > **View** > **Config** > **Data Retention** tab.

What do you want to do?

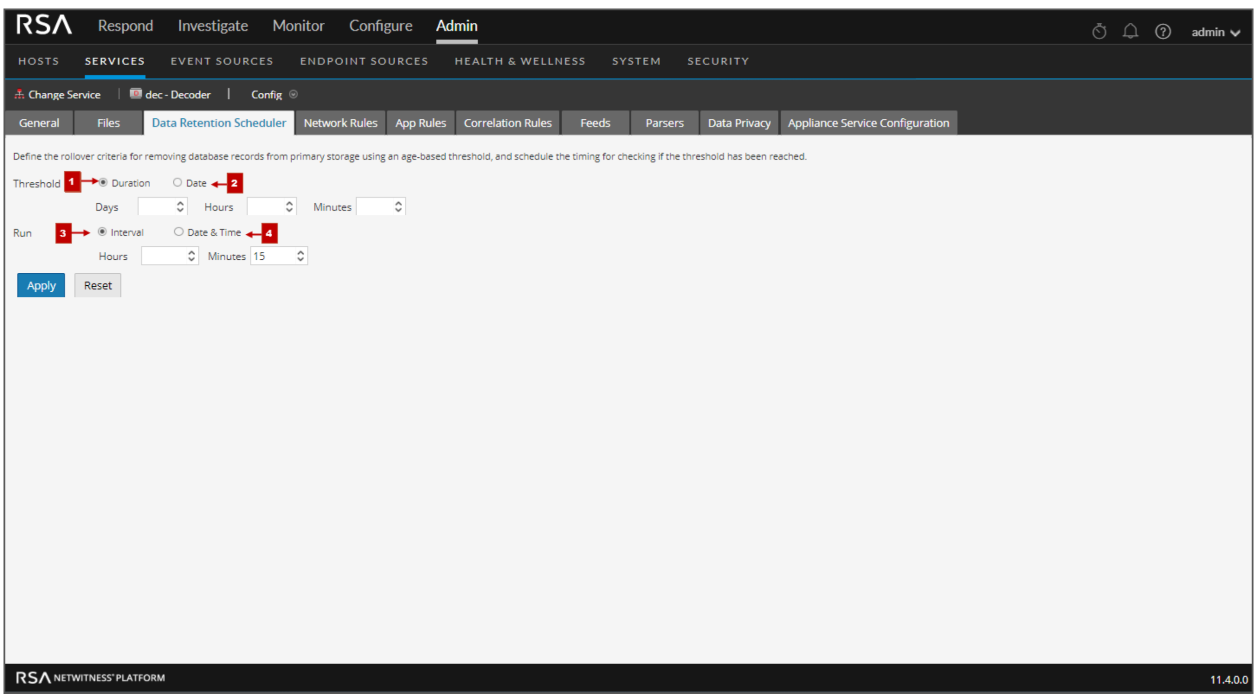
User Role	I want to...	Documentation
Administrator	Schedule the timing to see if the threshold is reached.	Configure Transaction Handling on a Decoder

Related Topics

- Configure Common Settings on a Decoder
- Decoder and Log Decoder Quick Setup

Quick Look

This is an example of the Data Retention Scheduler tab.



1	**Threshold Duration**: Removes database files older than the selected number of days, minutes, or hours.
2	**Threshold Date**: Removes database files older than the selected UTC date (YYYY-MM-DD-

	HH:MM:SS) that are not compatible with minutes, hours, or days parameters.
3	**Run Interval**: Indicates the number of hours between executions.
4	**Run Date and Time**: Defines which days of the week to execute the scheduler, as well as time of execution in HH:MM:SS format for the local time of the service.

Services Config View - Feeds Tab

Feeds and parsers are Lua programs loaded and compiled when either processing capture files in NetWitness Investigate or capturing data with Decoders. Most commonly, they are used for static meta extraction and service identification.

Note: Pre-11.0 versions of NetWitness used FLEXPARSE programs in addition to Lua programs; Flexparsers are deprecated in NetWitness Platform 11.0. Unless otherwise stated, any reference to Decoders applies to Log Decoders as well.

NetWitness Platform uses feeds to create metadata based on externally defined meta values. A feed is a list of data that is compared to sessions as they are captured or processed. For each match, additional metadata is created. This data can identify and classify malicious IPs or incorporate additional information such as department and location based on internal network assignments. Some examples of feeds include threat feeds to identify BOTNets, DHCP mappings, or even active directory information such as physical location or logical department.

Feeds can be added, removed, and updated while a Decoder is running without affecting capture. The Feeds tab (**Admin** > **Services** > select a Decoder or Log Decoder service and click > **View** > **Config** > **Feeds tab**) provides a user interface for managing feeds on Decoders.

What do you want to do?

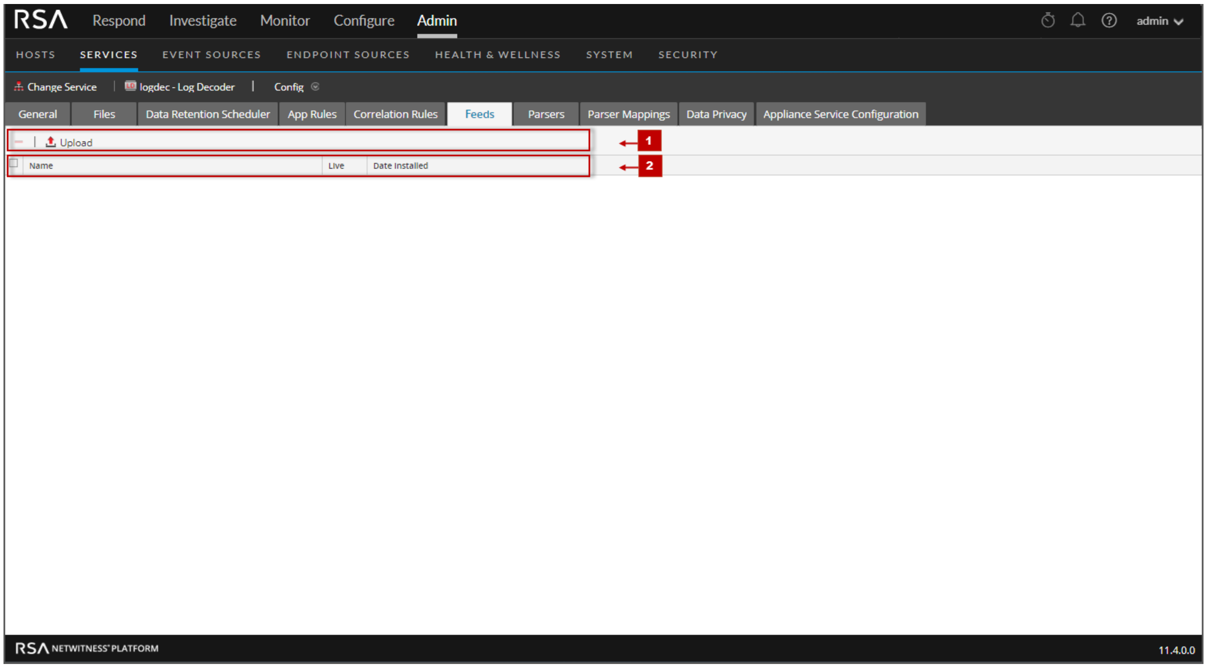
User Role	I want to...	Documentation
Administrator	configure feeds	Configure Parsers and Feeds
Administrator	enable and disable parsers	Enable and Disable Parsers and Log Parsers

Related Topics

- Configure Common Settings on a Decoder
- Decoder and Log Decoder Quick Setup
- Upload Feeds Dialog

Quick Look

This is an example of the Feeds tab.



1	Feeds Tab Toolbar - Provides options to work with feeds in the grid
2	Feed List - Lists all feeds that are currently deployed on the Decoder

Feeds Tab Toolbar

Feature	Description
Upload	Displays the Upload Feeds dialog.
−	Deletes the selected feeds.

Feeds List

The Feeds list provides a listing of all currently deployed feeds for the Decoder.

Column	Description
Name	The name of the feed or the feed file.
Live	Indicates if the feed originated from Live. Possible values are **Yes**, **No**, or **N/A**. • **Yes** = Installed through Live • **No** = Installed through NetWitness Platform • **N/A** = The feed has no attributes file created by NetWitness Platform to track the installation date. The feed may have been installed manually, not through NetWitness Platform or Live Services. Manually installed feeds still function properly.

Column	Description
Date Installed	The date the feed was pushed to the service.

Upload Feeds Dialog

This topic describes the features of the Upload Feeds dialog in the Services Config view > Feeds tab.

The **Upload** option in the Services Config view > Feeds tab displays the Upload Feeds Dialog, in which you can manage the uploading of feeds to a Decoder or Log Decoder.

What do you want to do?

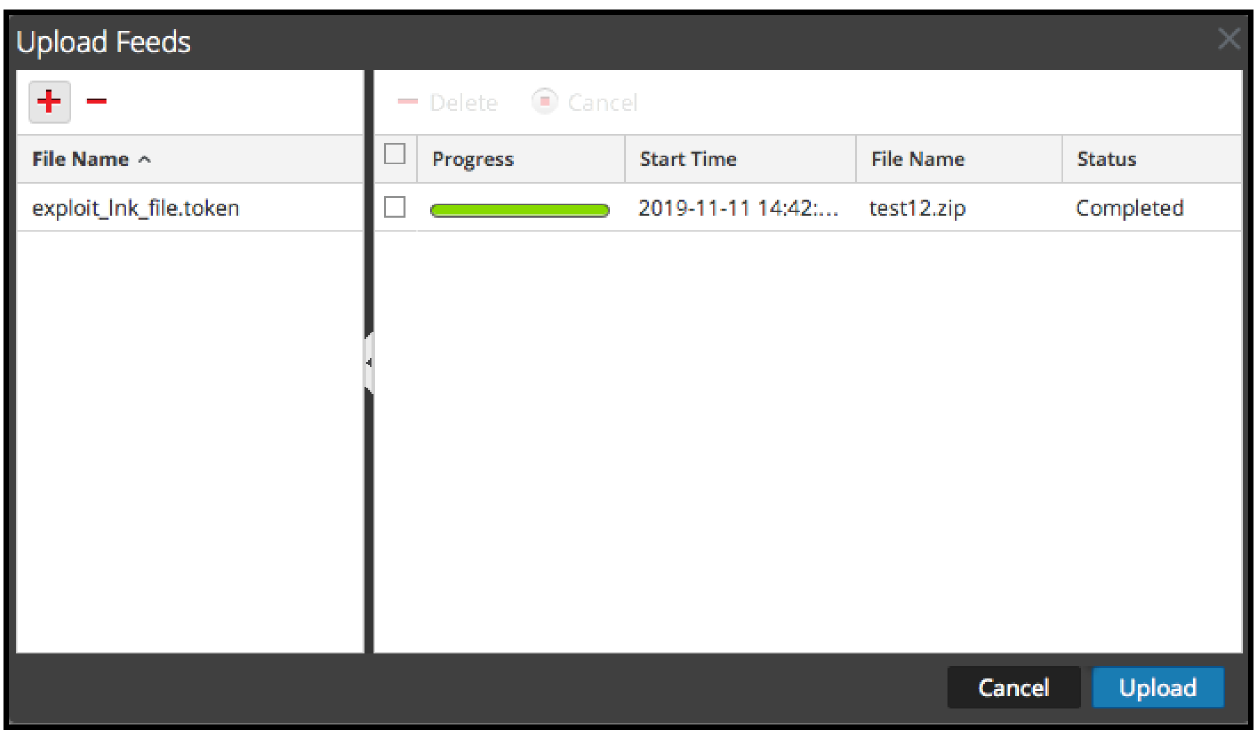
User Role	I want to...	Documentation
Administrator	prepare a list of feeds for upload	Edit, Upload, or Remove a Feed
Administrator	view and delete upload jobs	Edit, Upload, or Remove a Feed

Related Topics

- Decoder and Log Decoder Quick Setup
- Configure Common Settings on a Decoder
- Configure Parsers and Feeds

Quick Look

This is an example of the Upload Feeds dialog.



1 File List - Provides place to prepare a list of feeds for uploading
2 Upload Job List - Provides a view of upload jobs
3 Upload Feeds Dialog Buttons

File List

The File list is the place to prepare a list of feeds for uploading. You can add files from a directory structure, and delete files from the list if you decide that you don't want to upload a particular file. When the list is ready, clicking **Upload** starts the upload process.

Feature	Description
	Opens a view of the directory structure where you can select files to add to the File list.
	Deletes the selected files from the File list.
File Name	Lists the feed files you have added from a file system in preparation for uploading to a Decoder. When you click **Upload**, the files listed here are uploaded.

Upload Job List

The Upload Job list provides a view of upload jobs started by clicking **Upload**.

Feature/Column	Description
Delete	Deletes an upload job.
Progress	Displays progress of an upload job.
Start Time	Displays the start time of an upload job.
File Name	Lists filename of the feed being uploaded.
Status	Displays the status of upload job.

Upload Feeds Dialog Buttons

Feature	Description
Cancel	Closes the Upload Feed dialog.
Upload	Starts uploading the feed files listed in the File list. Each feed is listed in a separate row in the Upload job list.

Services Config View - Files Tab

The Decoder and Log Decoder configuration files are visible and editable in the Services Config view > Files tab. "Edit Core Services Configuration Files" in the *Hosts and Services Getting Started Guide* provides general instructions for editing files. (Go to the Master Table of Contents to find all RSA NetWitness Platform 11.x documents.)

Like other Core services, both the Decoder and Log Decoder have an index file, and may also have a crashreporter, netwitness, and scheduler. The Decoder and Log Decoder index files are named **`index-decoder-custom.xml`** and **`index-logdecoder-custom.xml`**.

Note: `Table-map.xml` and `table-map-custom.xml` are available only for Log Decoders with log content installed.

What do you want to do?

User Role	I want to...	Documentation
Administrator	obtain log files from pre-11.0 Log Decoder	Obtain Log Files from a Pre-11.0 Log Decoder
Administrator	edit files and parsers	Configure Parsers and Feeds

Related Topics

- Configure Common Settings on a Decoder
- Decoder and Log Decoder Quick Setup
- Create Custom Meta Keys Using a Custom Feed

Quick Look

Filename	Description
`GeoPrivate.ipl`	This fixed parser takes the IP addresses and converts them to geographical locations. The locations are displayed through the Google Earth display.
`feed-definitions.xml`	Used to create custom feeds, this is the XML schema used by the Decoder to define a **feed** message when it creates a **.feed** file.
`traffic_flow_options.lua`	Used to provide directionality information. Update this file with environment-specific internal and external subnets for the Lua parser to create proper directionality in metadata. The parser is described in RSA Content for RSA NetWitness Platform.
`search.ini`	This is the Search Parser configuration file. The Search Parser is a custom parser, used to generate metadata by scanning for pre-defined keywords and regular expressions.

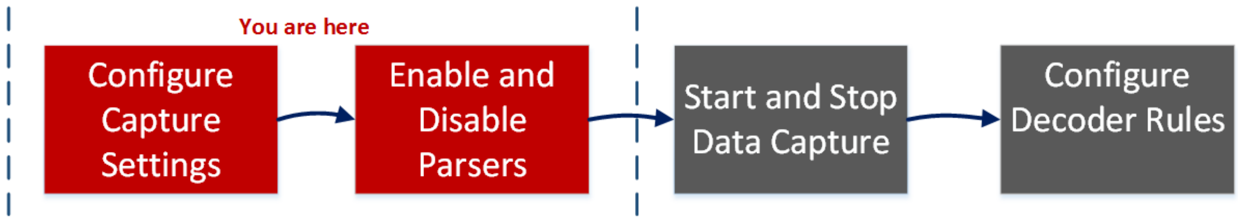
Filename	Description
`wlan-config.xml`	This is the wireless LAN configuration file (9/9/2009). This file controls the 802.11 parsers. Its chief purpose is to control decryption of raw 802.11 frames captured by the Decoder.

Services Config View - General Tab

The General tab for a Decoder in the Services Config view provides a way to manage basic service configuration, configure data capture, and select the parsers that are applied to the captured data. To access the General tab, go to **Admin** > **Services** > select a Decoder or Log Decoder and click > **View** > **Config** > **General tab**.

Workflow

The following figure depicts common Decoder configuration tasks with the steps you can complete in this view highlighted.



What do you want to do?

User Role	I want to...	Documentation
Administrator	configure capture settings*	Configure Capture Settings
Administrator	manage parsers and log parsers*	Enable and Disable Parsers and Log Parsers
Administrator	start and stop data capture	Start and Stop Data Capture
Administrator	configure rules	Configure Decoder Rules

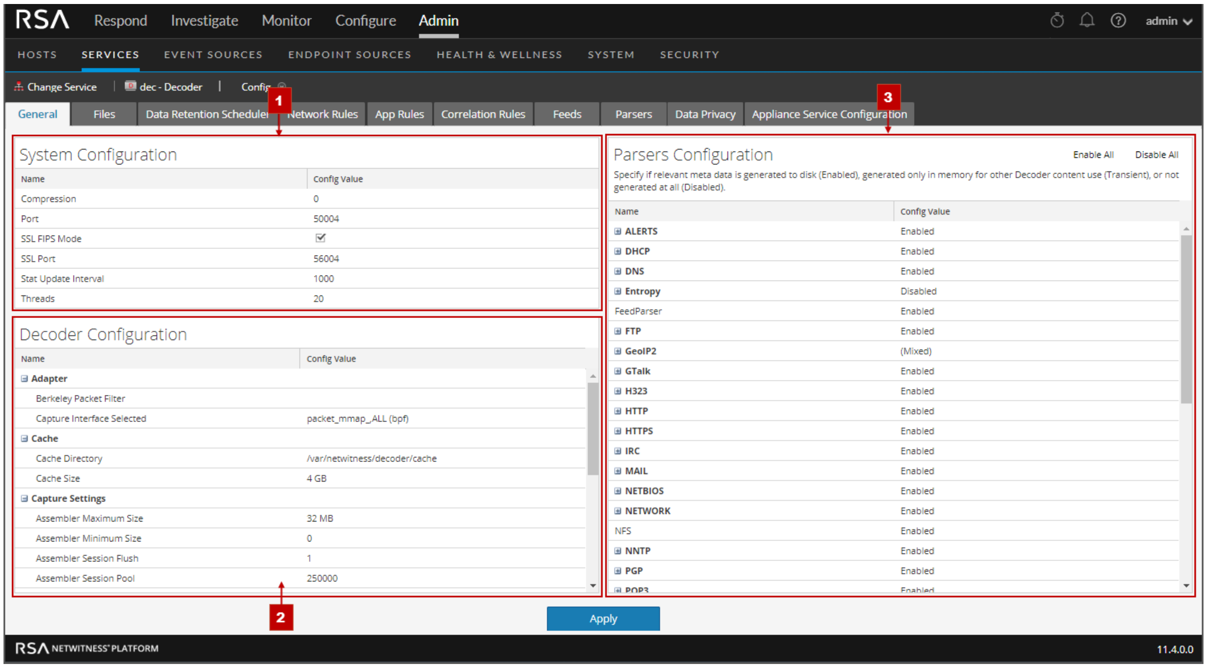
*You can complete these tasks here.

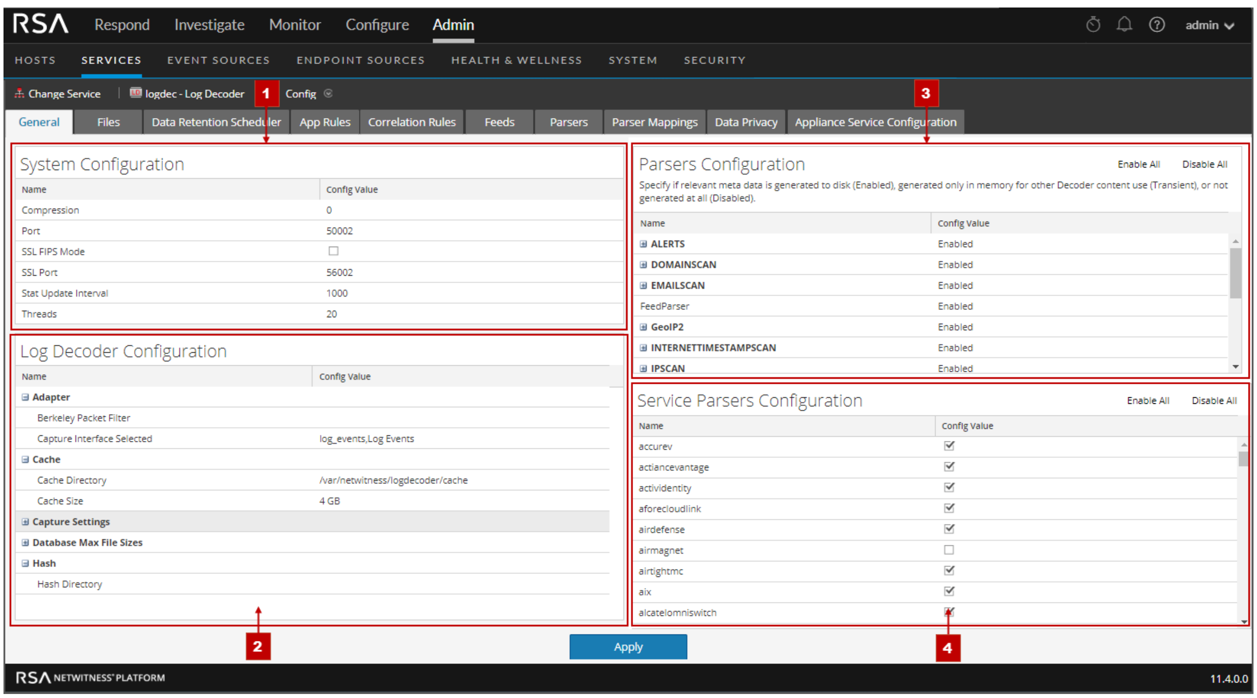
Related Topics

- Decoder and Log Decoder Quick Setup
- Configure Common Settings on a Decoder
- Configure Parsers and Feeds

Quick Look

The first figure is an example of the General tab for a Decoder. The second is the General tab for a Log Decoder.

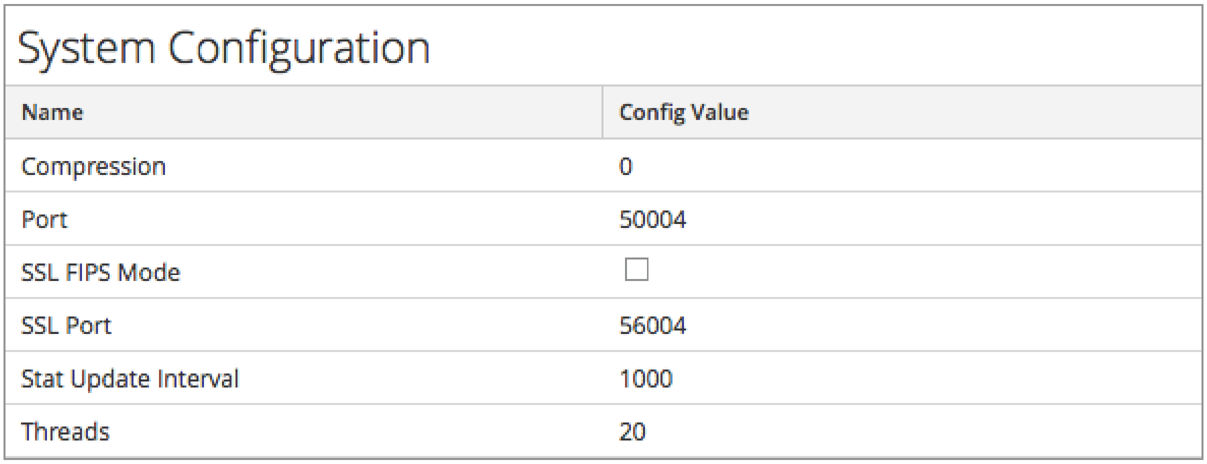




1	System Configuration - Manages service configuration for a Decoder.
2	Decoder Configuration or Log Decoder Configuration - Lets you view and edit service configuration parameters for a Decoder or Log Decoder.
3	Parsers Configuration - Lets you select parsers to use on the Decoder.
4	Service Parsers Configuration (Log Decoders only) - Lets you select service parsers to use on the Log Decoder.

System Configuration Section

The System Configuration section manages service configuration for a Decoder. When a service is first added, default values are in effect and should be changed only in special circumstances, for example, if Customer Support advises a change.



System Configuration

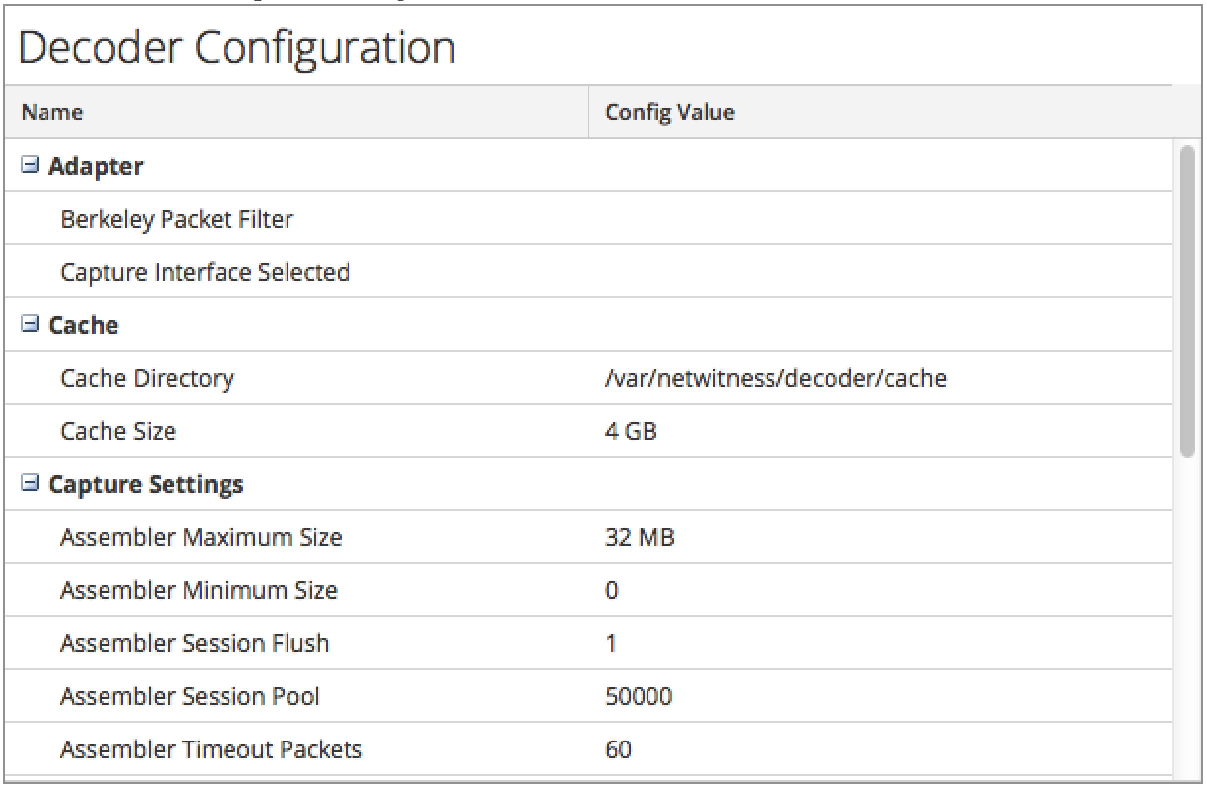
Name	Config Value
Compression	0
Port	50004
SSL FIPS Mode	☐
SSL Port	56004
Stat Update Interval	1000
Threads	20

The System Configuration section has these parameters.

Parameter	Description
Compression	The minimum number of bytes that must be transmitted per response before compression. A setting of 0 disables compression. The default value is **0**. A change in value is effective immediately for all subsequent connections.
Port	Determines the port used by the service. **Note:** If you change the port number, ensure that you restart the service.
SSL FIPS mode	If enabled, all the data transferred in the network will be encrypted using SSL.
SSL Port	Indicates the port used for encrypting using SSL.
Stat Update Interval	The number of milliseconds between statistic updates on the system. Lower numbers cause more frequent updates and can slow down other processes. The default value is **1000**. A change in value is effective immediately.
Threads	The number of threads in the thread pool to handle incoming requests. A setting of **0** lets the system decide. A change takes effect on service restart.

Decoder Configuration Section

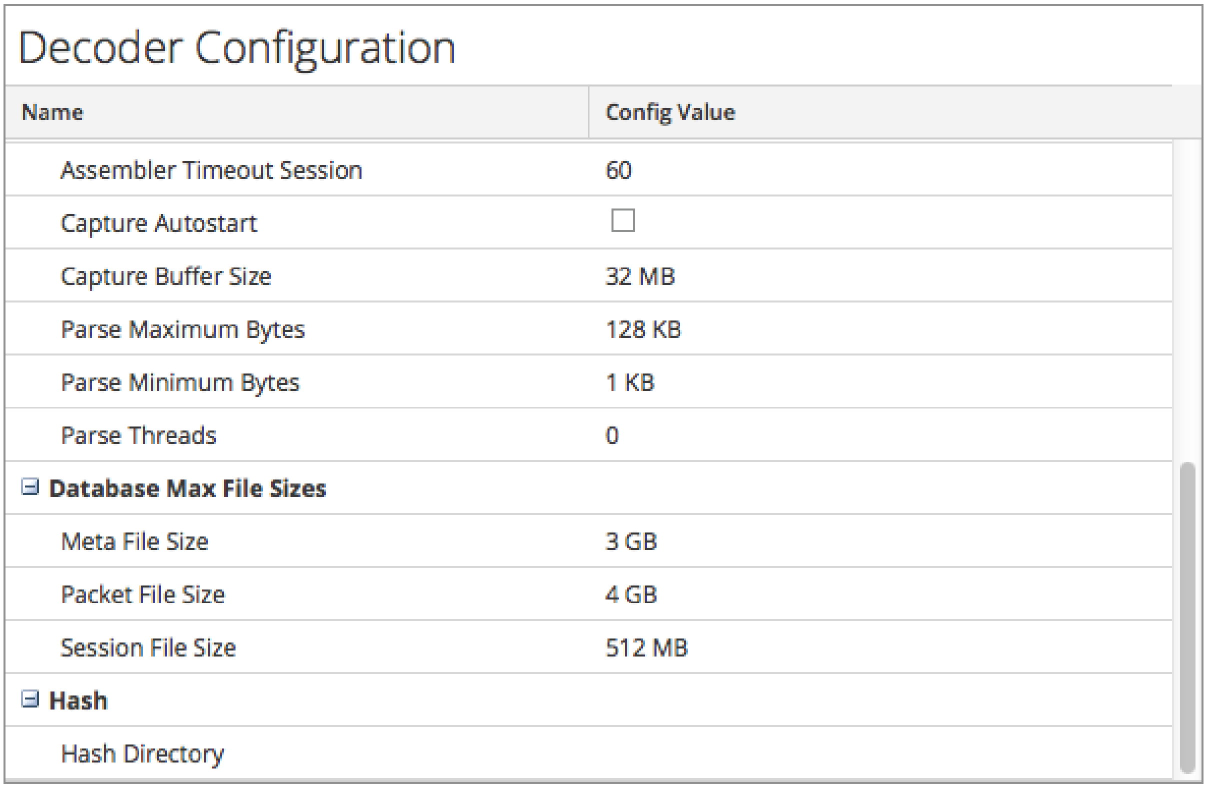
The Decoder Configuration section provides a way to view and edit service configuration parameters for a Decoder or Log Decoder. When a service is first added, default values are in effect. You can edit these values to manage traffic capture.



Decoder Configuration

Name	Config Value
⊟ Adapter	
Berkeley Packet Filter	
Capture Interface Selected	
⊟ Cache	
Cache Directory	/var/netwitness/decoder/cache
Cache Size	4 GB
⊟ Capture Settings	
Assembler Maximum Size	32 MB
Assembler Minimum Size	0
Assembler Session Flush	1
Assembler Session Pool	50000
Assembler Timeout Packets	60

Scrolling to the bottom of the section reveals these additional Decoder Configuration parameters.



Adapter Section

Adapter parameters configure the network interface for capture as described in Configure Capture Settings.

Cache Section

Cache parameters configure the cache directory and size for session cache files. The following table describes the cache settings. When a service is first added, default values are in effect and should be changed only in special circumstances, for example, if Customer Support advises a change.

Cache Parameter	Description
Cache Directory	The directory where session cache files are stored. The default value is `/var/netwitness/decoder/cache`. Change takes effect immediately.
Cache Size	The maximum size, in Megabytes (MB), that all files in the cache directory can attain before the oldest files are deleted. Once the threshold is reached, the cache size is reduced by 10%. The default value is **4 GB**. Change takes effect immediately.

Capture Settings Section

The Capture Settings section provides a way to configure operational capture settings. When a service is first added, default values are in effect and should be changed only in special circumstances, for example, if Customer Support advises a change.

Capture Settings Parameter	Description
Assembler Maximum Size	Specifies the maximum size in bytes that a session's packet data size can attain. The default value is **32 MB**. Change takes effect immediately.
Assembler Minimum Size	Specifies the minimum size in bytes that a session must have in order to generate metadata. A value of **0** means every session has metadata generated. The default value is **0.** Change takes effect immediately.
Assembler Session Flush	Specifies whether a session is removed from the assembler when the session's last chain is removed from the assembler. The default value is **1**. • **2** = if the first packet of a session times out of assembler, the session is removed from assembler after parsing is complete. Any subsequent packets for this session create a new session in assembler. • **1** = If the last chain of a session times out of assembler, the session is removed from assembler. Any subsequent packets for this session create a new session in assembler. • **0** = If the last chain of a session times out of assembler, the session is left in assembler until it times out. Any subsequent packets for this session are filtered Change takes effect on service restart.
Assembles Session Pool	Specifies the number of entries in the session pool. The default value is **350000**. Change takes effect on service restart.
Assembler Timeout Packets	Specifies the number of seconds before a packet or chain is timed out. T default value is **60**. Change takes effect immediately.
Assembler Timeout Session	Specifies the number of seconds before a session is timed out. Default value is **60**. Change takes effect immediately.
Capture Autostart	Specifies whether capture begins automatically each time Decoder is started. When checked, the value = yes. When unchecked, the value = no. The default value is **no**. Change takes effect immediately.
Capture Buffer Size	The capture memory buffer allocation in Megabytes. Default value is **64 MB**. Change takes effect on service restart.
Parse Maximum Bytes	The maximum number of bytes to scan a stream for additional tokens. When the first token is found, the stream is scanned up to the set number of bytes, but no further. A setting of **0** removes the early termination and the full stream is scanned regardless of size. The default value is **128 KB**. Change takes effect immediately.

Capture Settings Parameter	Description
Parse Minimum Bytes	The minimum number of bytes to scan a stream for the first token. If no token is found within the set number of bytes, scanning is terminated. A setting of **0** removes the early termination and the full stream is scanned regardless of size. The default value is **1 KB**. Change takes effect immediately.
Parse Threads	The number of parse threads to use for session parsing. A value of **0** means let the server decide. The default value is **0**. Change takes effect on service restart.

Database Max File Sizes Section

The Database Max File Sizes section controls the maximum file size for various databases. When a service is first added, default values are in effect and should be changed only in special circumstances, for example, if Customer Support advises a change.

File Size Parameter	Description
Meta File Size	The maximum size of meta database files in Megabytes. The default value is **10 MB**. Change takes effect on service restart.
Packet File Size	The maximum size of packet database files in Megabytes. The default value is **10 MB**. Change takes effect on service restart.
Session File Size	The maximum size of session database files in Megabytes. The default value is **100 MB**. Change takes effect on service restart.

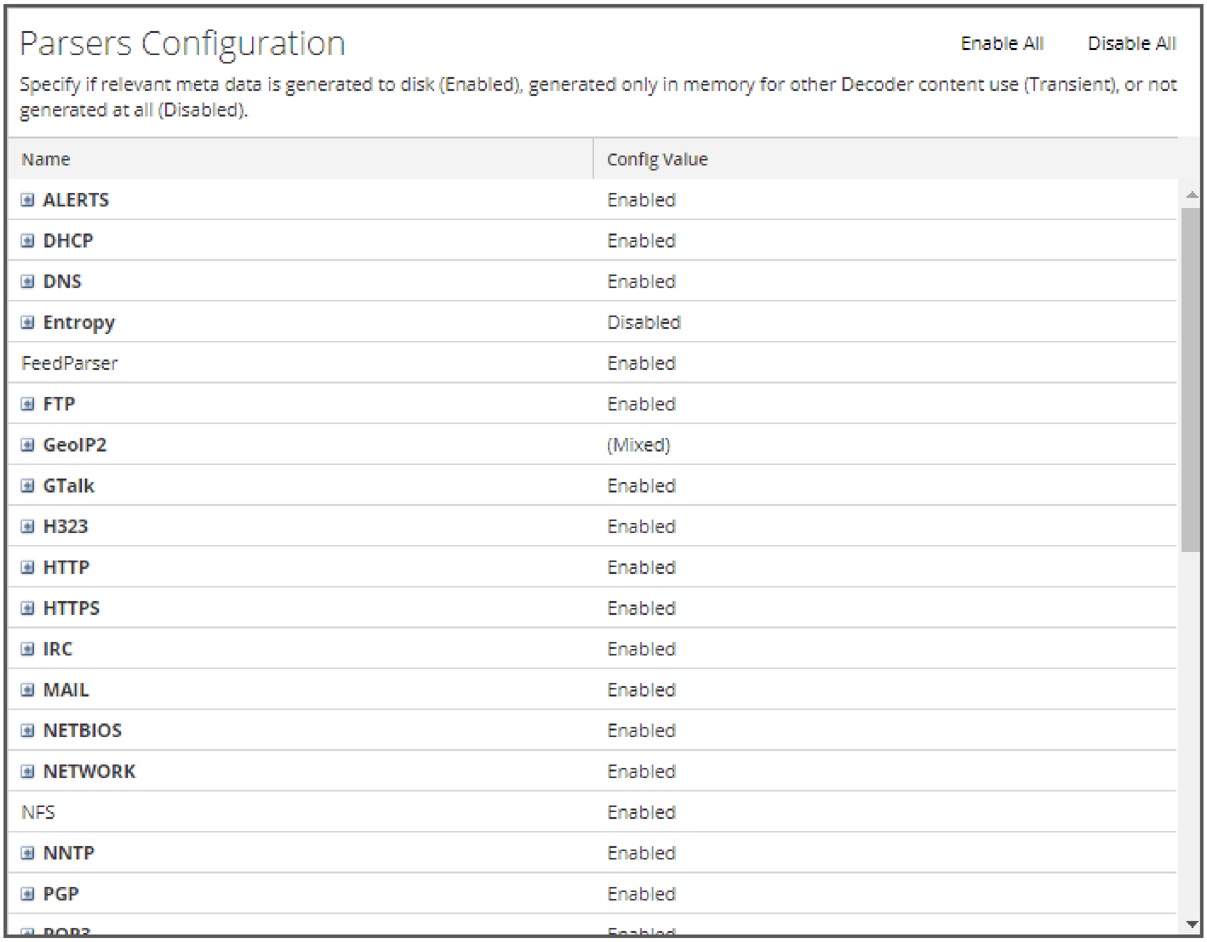
Hash Section

The Hash section settings control data base file hashing options. There is a small performance penalty when hashing.

Hash Parameter	Description
Hash Directory	The server directory where all hash files are written. If empty, each hash file is written to the same directory as the file being hashed. The default value is blank. Change takes effect on service restart.

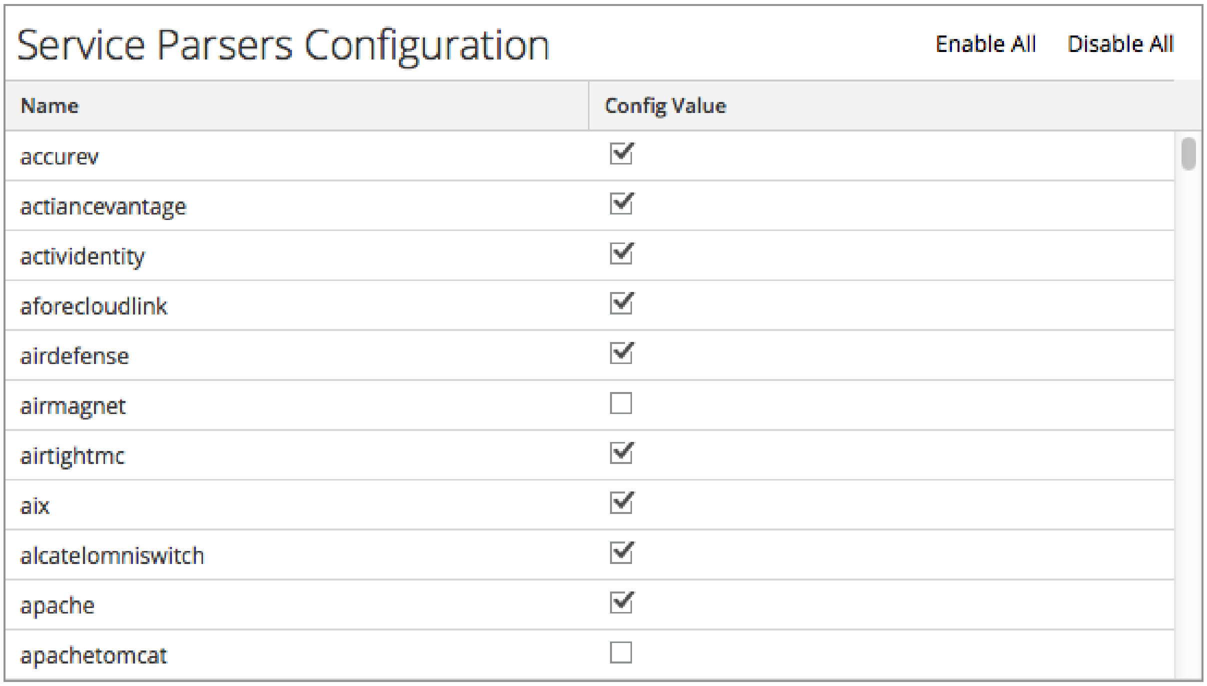
Parsers Configuration Panel

The Parsers Configuration panel provides a way to select parsers to use on the Decoder. Within some parsers, you can also configure the metadata that the parser creates. See Enable and Disable Parsers and Log Parsers for detailed information and procedures.



Service Parsers Configuration Section for Log Decoder

The Service Parsers Configuration section provides a way to select Service parsers to use on the Log Decoder.



Service Parsers Configuration
Enable All
Disable All
Name
Config Value
accurev
actiancevantage
actividentity
aforecloudlink
airdefense
airmagnet
airtightmc
aix
alcatelomniswitch
apache
apachetomcat

Services Config View - Parsers Tab

In the Services Config View > Parsers tab, you can view deployed parsers on a Decoder or Log Decoder, upload parsers, and delete deployed parsers. Parsers can be added and removed while a Decoder or Log Decoder is running without affecting capture.

To access the Parsers tab, go to **Admin** > **Services** > select a **Decoder** or **Log Decoder** service and click > **View** > **Config** > **Parsers** tab.

What do you want to do?

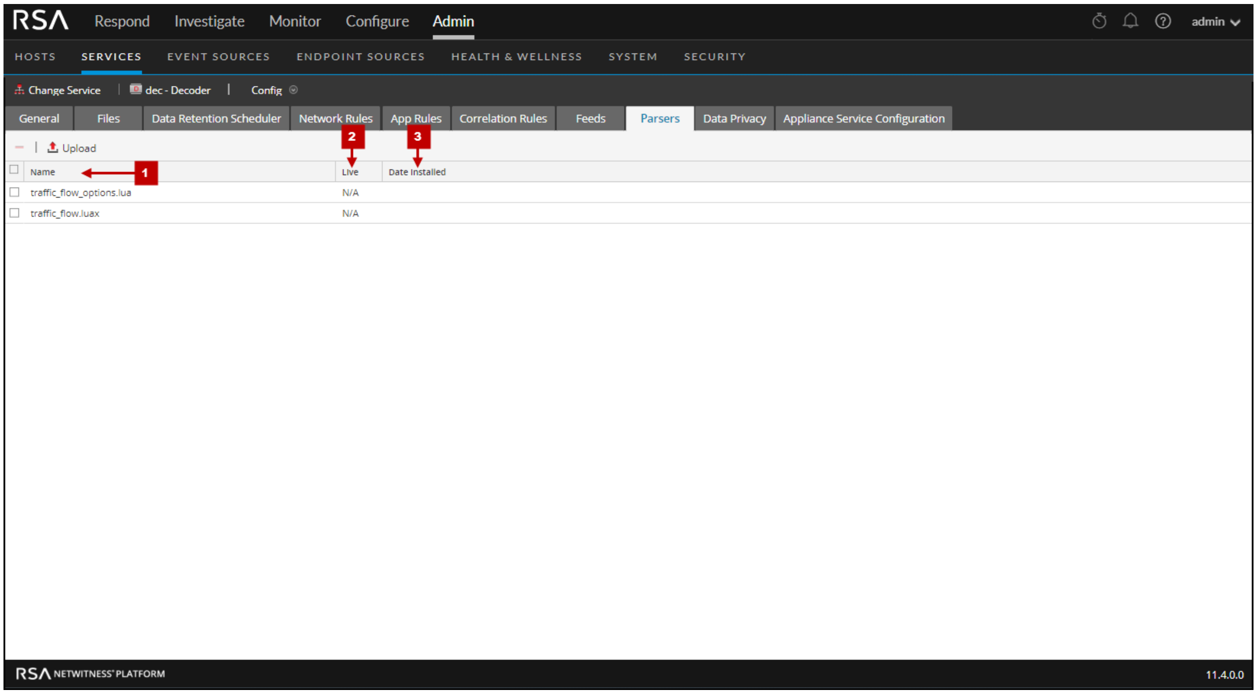
User Role	I want to...	Documentation
Administrator	View deployed parsers.	Enable and Disable Parsers and Log Parsers
Administrator	Upload parsers to a Decoder or Log Decoder.	Enable and Disable Parsers and Log Parsers

Related Topics

- Decoder and Log Decoder Quick Setup
- Configure Common Settings on a Decoder
- Upload and Delete Custom Parsers

Quick Look

This is an example of the Parsers tab. The Parsers grid lists all parsers that are currently deployed on the Decoder.



1 **Name**: The name of the parser or the parser file.

2 **Live**: Indicates if the parser originated from Live. Possible values are **Yes**, **No**, or **N/A**.

- **Yes** = Installed through Live Services.
- **No** = Installed through NetWitness.
- **N/A** = The parser has no attributes file created by NetWitness to track the installation date. The parser may have been installed manually, not through NetWitness or Live Services.

3 **Date Installed**: The date the parser was pushed to the service.

Parsers Tab Toolbar

The Parsers Tab toolbar has options to work with parsers in the grid.

Feature	Description
Upload	Enables you to upload parsers to a Decoder or Log Decoder.
	Requests confirmation that you want to delete the selected parsers. You can select **No** to cancel the deletion, or select **Yes** to delete the selected parsers.

Services Config View - Parser Mappings Tab

This topic provides a description of the configurable options for a Log Decoder in the Parser Mappings tab.

In the Parser Mappings Administrators can configure log parser mappings for Log Decoder services. To access the Parser Mappings tab, go to **Admin** > **Services** > select a Log Decoder service and click  > **View** > **Config** > **Parser Mappings** tab.

> **Note:** You can also configure log parser mappings for Log Decoder services by navigating to **Admin > Services > Event Sources > Discovery**.

This feature is intended to track a subset of event sources that is parsing against the wrong parser.

What do you want to do?

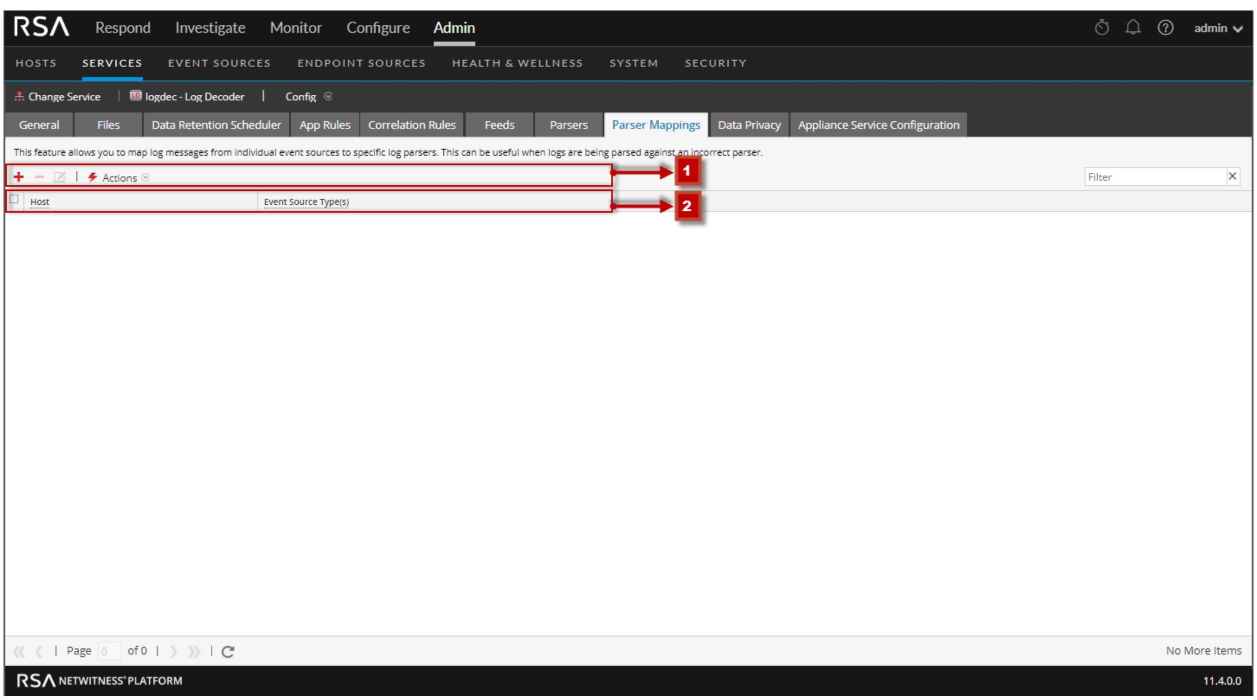
User Role	I want to...	Documentation
Administrator	Manage IPs for Event Source Mapping.	Enable Parser Mappings

Related Topics

- Decoder and Log Decoder Quick Setup
- Configure Common Settings on a Decoder

Quick Look

This is an example of the tab.



1	Parser Mappings Toolbar - Provides options to work with parser mappings in the list
2	Parser Mappings List - Lists all parsers that are currently mapped on the Log Decoder

Parser Mappings Toolbar

The Parser Mappings toolbar has options to work with parser mappings in the grid.

Feature	Description
	Add a parser mapping.
	Delete the selected parser mapping.
	Edit a parser mapping.
	Refresh the list of parser mappings.
	Display the Actions menu. • **Import** - Import a parser mapping to a file. • **Export** - Save a parser mapping to a file.

Parser Mappings List

The Parser Mappings list displays all parsers that are currently mapped on the Log Decoder.

Parameter	Description
Host	Displays the IP address of the host.
Event Source	Displays the event sources that are parsing incorrectly.

Parser Mappings Editor Dialog

The Parser Mappings Editor dialog allows you to update an IP to event source mapping.

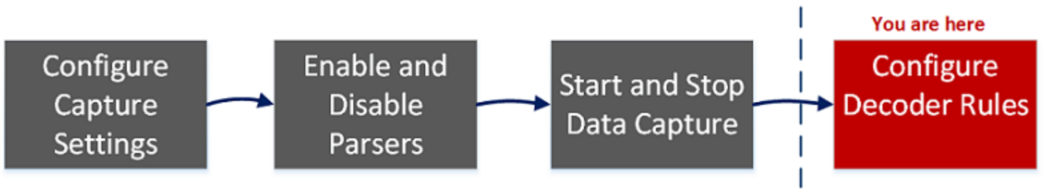
To access the Parser Mappings Editor dialog, in the Services Config view for a Log Decoder, select the Parser Mappings tab.

Services Config View - Rules Tabs

The Rules tabs in the Services Config view (**Admin** > **Services** > select a service and click > **View** > **Config**) enable you to define and manage capture rules. Each type of rule has a list with slightly different columns and different parameters in the Rule Editor dialog. Application and correlation rules apply to both Decoders and Log Decoders. Network rules apply only to Network Decoders.

Workflow

The following figure depicts the workflow for common Decoder configuration tasks with the steps you can complete in this view highlighted.



What do you want to do?

User Role	I want to...	Documentation
Administrator	configure capture settings	Configure Capture Settings
Administrator	manage parsers and log parsers	Enable and Disable Parsers and Log Parsers
Administrator	start and stop data capture	Start and Stop Data Capture
Administrator	configure rules*	Configure Decoder Rules
Administrator	import, export, or push a rule*	Configure Decoder Rules
Administrator	enable or disable a rule*	Configure Decoder Rules
Administrator	add, edit, or delete a rule*	Configure Decoder Rules

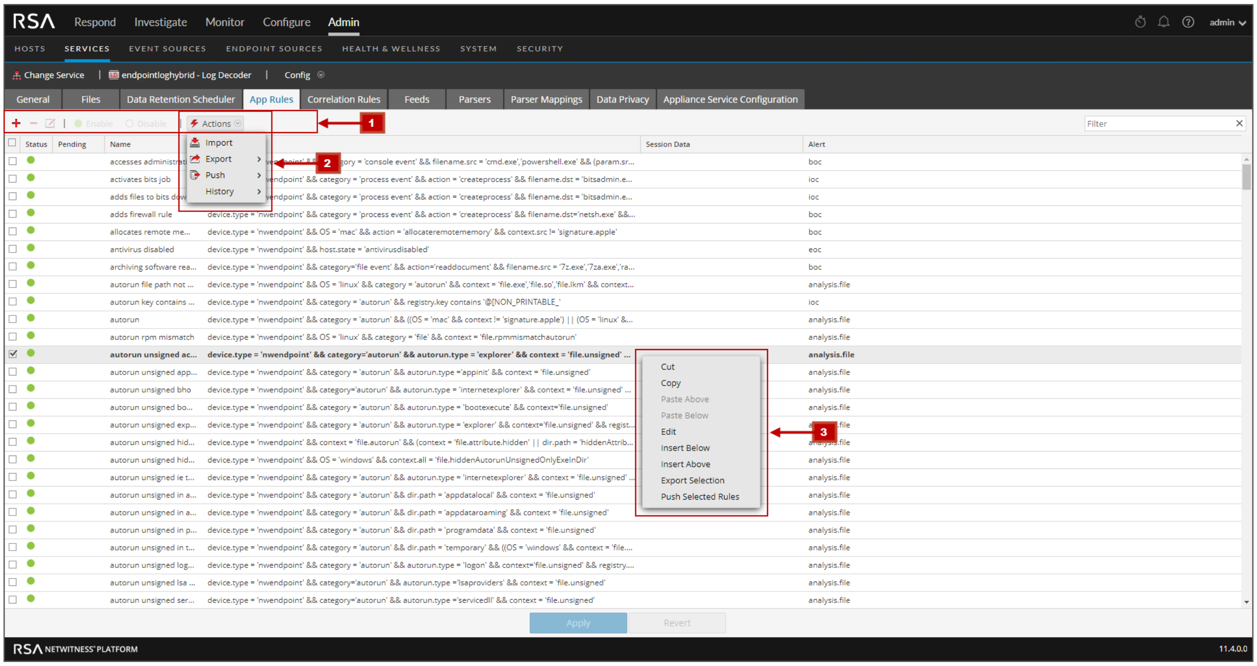
*You can complete these tasks here.

Related Topics

- Configure Common Settings on a Decoder
- Decoder and Log Decoder Quick Setup
- App Rules Tab
- Correlation Rules Tab
- Network Rules Tab

Quick Look

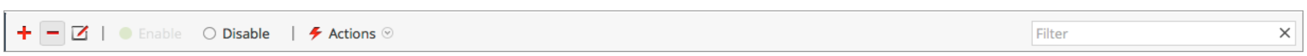
This is an example of the App Rules tab.

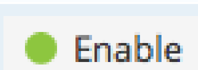


1	Rules Tab Toolbar - Provides options to work with rules in the list
2	Rules Actions Menu - Provide options to manage sets of rules
3	Rules List Context Actions - Displays the Rules List Context Menu

Rules Tab Toolbar

The toolbar is the same for all Config view > Rules tabs.

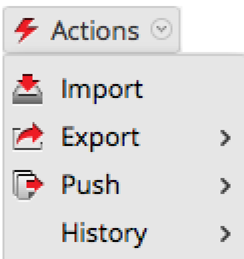


Feature	Description
Actions	Displays the **Actions** menu.
	Adds a new rule to a service.
	Deletes a rule from a service.
	Allows rule modification.
	Disables a rule (without deleting the rule).
	Enables (reactivates) a rule.
Filter	The input field for a search string. NetWitness Platform filters the rules dynamically as you type a search string. Clicking **x** clears the input field, restoring the unfiltered view.

Feature	Description
Apply	Saves the changes made to rules and applies the configured rules to a service. Until you apply changes, it is possible to reload the rules as they were before current modifications.
Revert	Discards unsaved changes to the list and reverts to the unedited rules.

Rules Actions Menu

The Actions menu has options that help to manage sets of rules.



Option	Description
Import	Imports a set of rules into the user interface so that it can be applied to a service. You can edit the rules before applying.
Export	Saves selected rules or all rules to an .nwr file on the client machine.

Push	Allows rules to be applied to other services (Decoders or Log Decoders) or Decoders belonging to a service group. When pushing, the rules can either be merged (update existing rules and append new ones) or replaced.
	• **Push > All.** Pushes all rules to other services. All rules on the target services are removed and replaced with all of the rules on the source service.
	• **Push > Selection.** Pushes selected rules to other services. You have two options:
	  • **Replace.** Deletes all rules on the target services and replaces them with the selected rules from the source service.
	  • **Merge.** Merges the selected rules with the existing rules on the target services

History	Displays the last ten snapshots of rules applied through NetWitness Platform. You can select and apply (restore) a snapshot to the Decoder at anytime.

Rules List Context Actions

Within a rules list, right-clicking a row displays the Rules list context menu.

Option	Description
Cut	Deletes the current rule.

Option	Description
Copy	Copies the current rule.
Paste Above	Pastes the copied rule above the current rule.
Paste Below	Pastes the copied rule below the current rule.
Edit	Edits the current rule.
Insert Below	Inserts imported rules below the current rule.
Insert Above	Inserts imported rules above the current rule.
Export Selection	Exports the selected rules.
Push Selected Rules	Pushes the selected rules to other services.

App Rules Tab

The App Rules tab (**Admin** > **Services** > select a Decoder or Log Decoder and click > **View** > **Config** > **App Rules tab**) enables you to manage application rules. NetWitness Platform applies application rules at the session level.

What do you want to do?

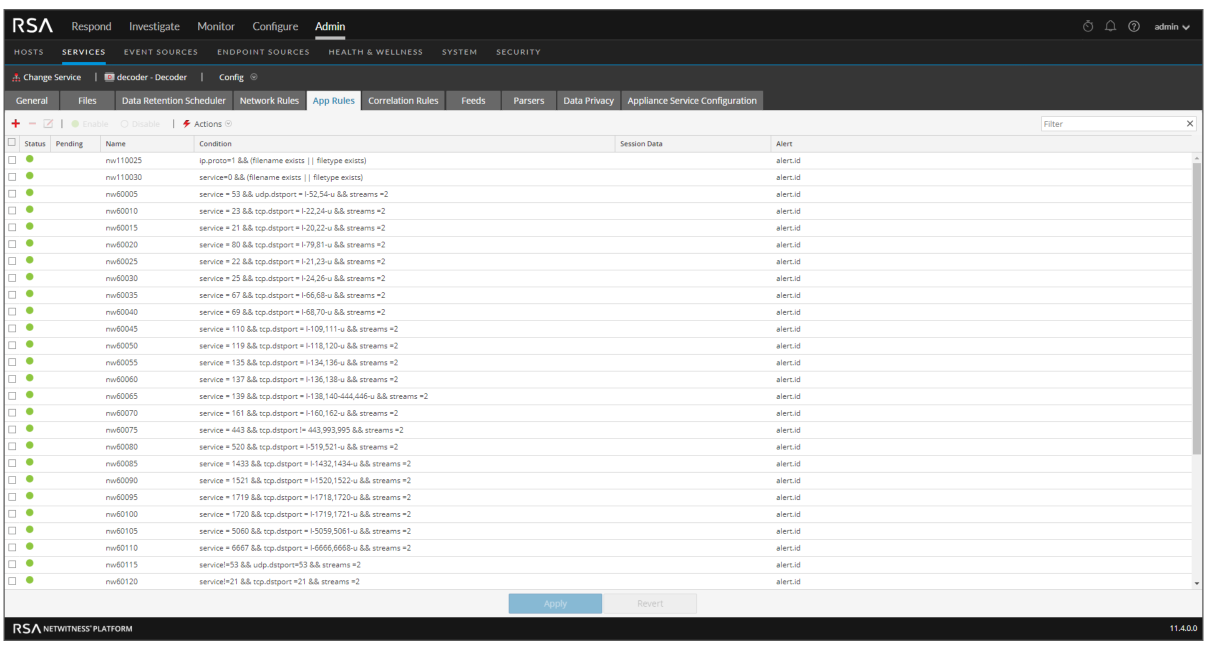
User Role	I want to...	Documentation
Administrator	add or edit application rules	Configure Application Rules

Related Topics

- Decoder and Log Decoder Quick Setup
- Configure Common Settings on a Decoder
- Configure Decoder Rules
- Services Config View - Rules Tabs

Quick Look

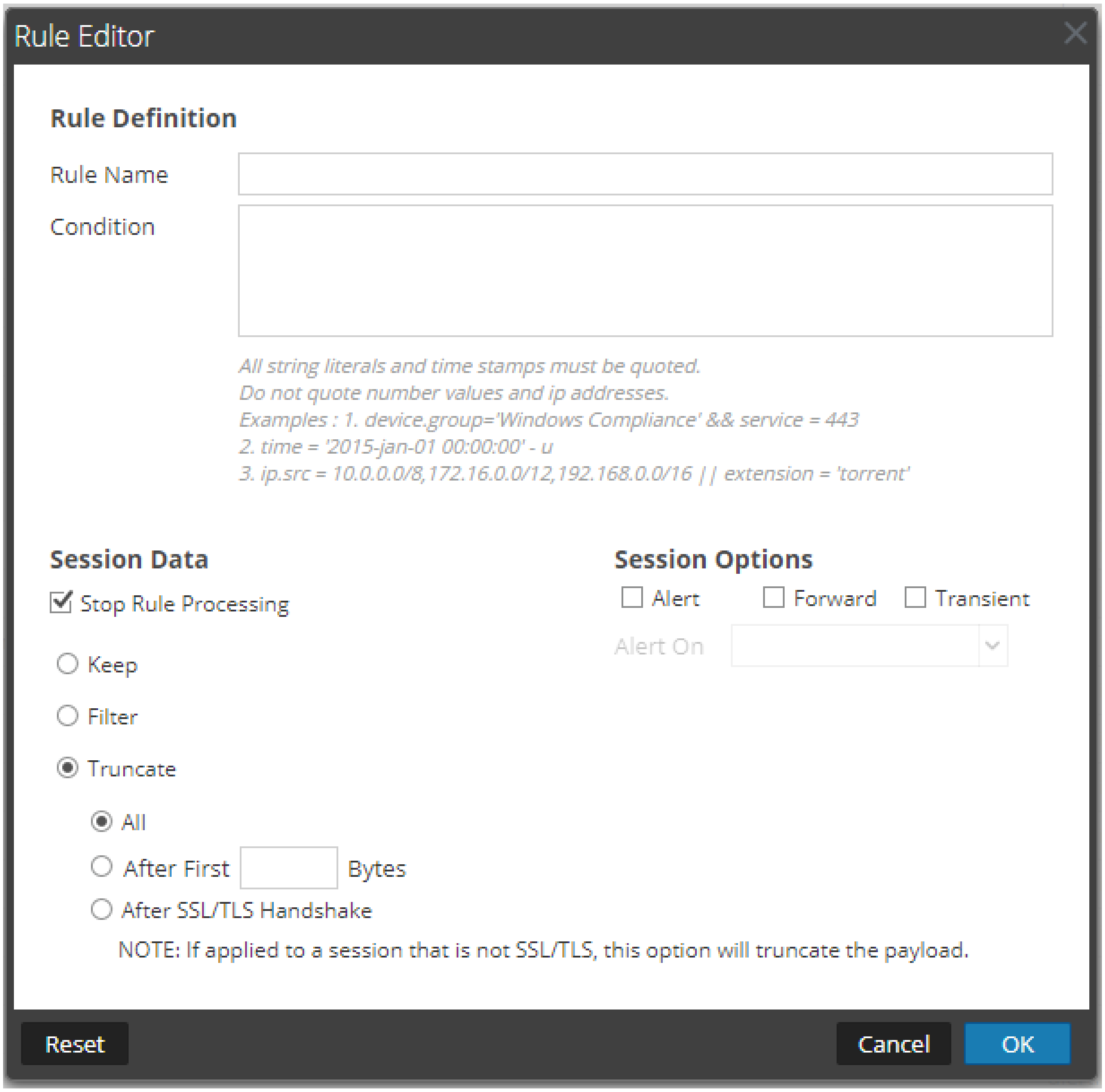
The following figure shows an App Rules tab and the table describes the columns.



Column	Description
Pending	This column indicates whether a rule has pending changes. Rules that are currently active on the Decoder have no indicator. If the rule is new or has been modified, the column contains . Once the rules are applied, the pending indicator is removed.
Name	This is the rule name, a descriptive identifier for the rule.
Condition	This is the definition of the condition that triggers an action when matched.
Session Data	This column displays the Session Data action taken when a packet matches the rule. Possible values are **Filter**, **Keep**, or **Truncate**.
Alert	This column displays the name of the custom alert that the Decoder generates when metadata matches the rule.
Status	This column indicates whether the rule is enabled or disabled with a circle icon. If the circle is filled green, the rule is enabled. If the circle is empty, the rule is disabled.

Rule Editor Dialog

The following figure shows the Rule Editor dialog for an application rule.



The Rule Editor dialog provides the fields and options needed to define an application rule.

Field	Description
Rule Name	The descriptive name that identifies the rule.
Condition	The definition of the condition that triggers an action when matched. You can type directly in the field or build the condition in this field using meta from the Intellisense window actions. As you build the rule definition, Intellisense displays syntax errors and warnings. All string literals and time stamps must be quoted. Do not quote number values and IP addresses. Configure Decoder Rules provides additional details.

The following table describes the Session Data actions and options.

Action	Description
Stop Rule Processing	If checked, further rule evaluation ends if the rule is matched, and the session is saved in accordance with the session action. If not checked, rule evaluation continues until all rules are evaluated.
Keep	The packet payload and associated metadata are saved when they match the rule.
Filter	The packet is not saved when it matches the rule.
Truncate	**Truncate All** – truncates all session payload bytes. The packet payload is not saved when it matches the rule, but packet headers and associated metadata are retained. This is the default truncation option. **Truncate After First <n> Bytes** – truncates the session payload bytes after the specified first <n> bytes, where <n> is an integer. The packet payload is not saved after <n> bytes when it matches the rule, but packet headers and associated metadata are retained. **Truncate SSL/TLS After Handshake** – truncates the payload for all sessions except in the case of an SSL/TLS session, where the SSL exchange is preserved, but the rest of the payload is not saved. This option is for use with SSL parsers.
Alert and **Alert On**	If **Alert** is checked, the packet generates a custom alert when metadata matches the rule. You can select the name of the alert in the **Alert On** field.
Forward	Enables the performance of syslog forwarding when the log matches the rule.
Transient	Prevents the alert metadata that is created from being written to the disk.

The following table describes Rule Editor dialog actions.

Action	Description
Reset	Resets the contents of the dialog to their values before editing; changes are discarded.
Cancel	Cancels any edits and closes the Rule Editor dialog.
OK	Saves the new rule or edited rule, and adds it to the rules grid. The Rule Editor dialog closes.
Save	(Rules with deprecated syntax only) Applies a corrected rule individually to the Decoder service. See Fix Rules with Invalid Syntax.

Correlation Rules Tab

The Correlation Rules tab (**Admin** > **Services** > select a service and click > **View** > **Config** > **Correlation Rules tab**) enables you to manage correlation rules. Basic correlation rules are applied at the session level and alert the user to specific activities that may be occurring in their environment. NetWitness Platform applies correlation rules over a configurable sliding time window.

What do you want to do?

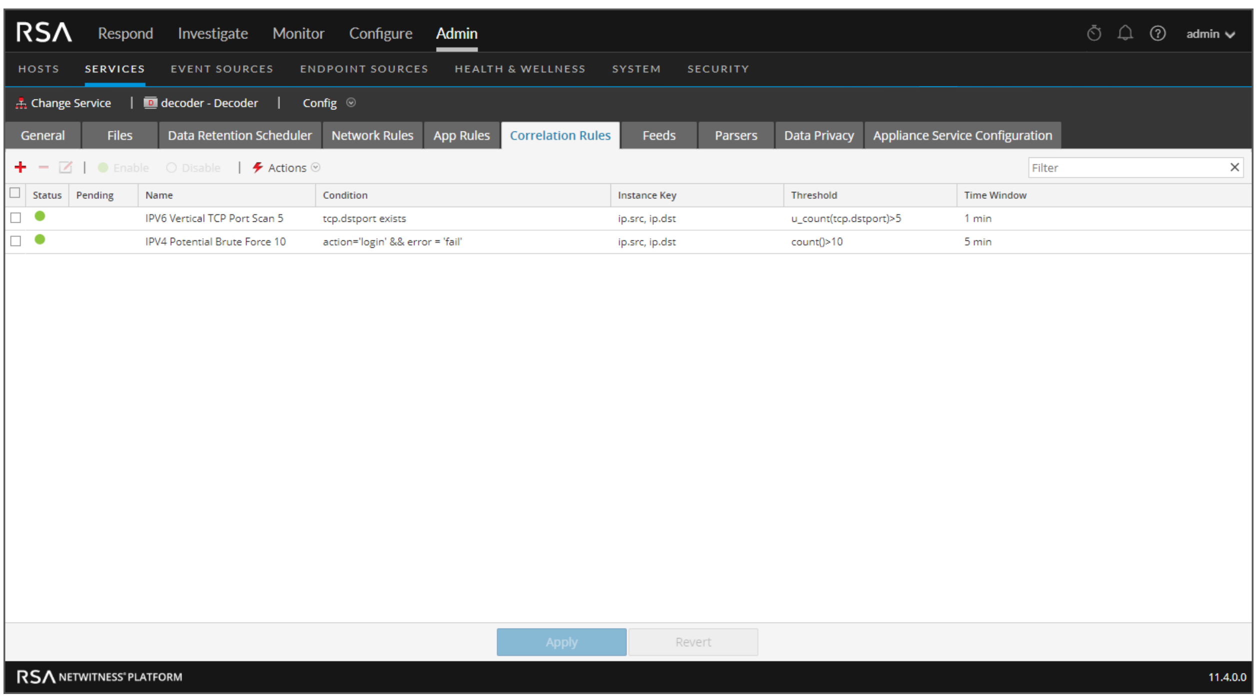
User Role	I want to...	Documentation
Administrator	add or edit a correlation rule	Configure Correlation Rules

Related Topics

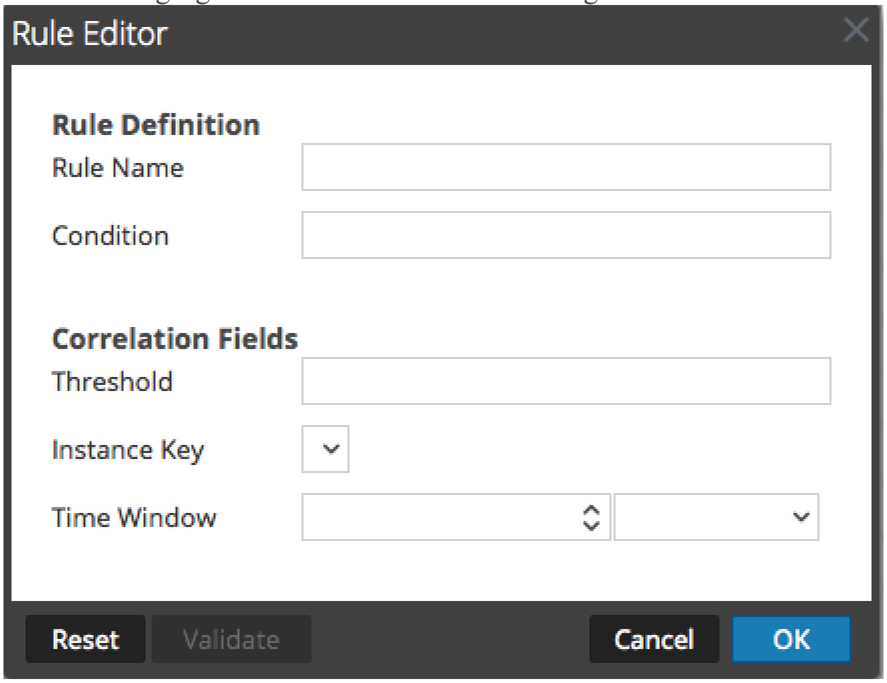
- Configure Common Settings on a Decoder
- Decoder and Log Decoder Quick Setup
- Configure Decoder Rules
- Services Config View - Rules Tabs

Quick Look

The following figure shows the Correlation Rules tab.



The following figure shows the Rule Editor dialog for a correlation rule.



The following table describes the Correlation Rules tab columns.

Column	Description
Pending	This column indicates whether a rule has pending changes. Rules that are currently active on the Decoder have no indicator. If the rule is new or has been modified, the column contains . Once the rules are applied, the pending indicator is removed.
Name	This is the descriptive name for the rule.
Condition	This is the definition of the condition that triggers an action when matched. In conditions, all string literals and time stamps must be quoted. Do not quote number values and IP addresses. [Configure Decoder Rules]() provides additional details.
Instance Key	This is the target indicator to base the event upon. It can be a single primary key, such as ip.src or a compound primary key such as `ip.src,ip.dst`.
Threshold	This is the minimum number of occurrences required to trigger a correlation session and can include a associated key that identifies the meta type that were are counting to determine if the condition is satisfied. The correlation engine cannot use IPv4 or IPv6 as an associated meta type. Use one of these three arguments: • `u_count(associated_key)` = the count of unique values of the specified key. A key is required. • `sum(associated_key)` = the values of the specified key. a key is required. • `count()` = number of sessions, no associated key used. If included, it is ignored.
Time Window	This is the duration in hours, minutes, or seconds within which the threshold must be reached to trigger a correlation session.

Column	Description
Status	This column indicates whether the rule is enabled or disabled with a circle icon. If the circle is filled green, the rule is enabled. If the circle is empty, the rule is disabled.

The **Rule Editor** dialog provides the fields and options needed to define a network rule. The fields correspond exactly to the grid columns.

Action	Description
Reset	Resets the contents of the dialog to their values before editing; changes are discarded.
Cancel	Cancels any edits and closes the Rule Editor Dialog.
OK	Saves the new rule or edited rule, and adds it to the rules grid. The Rule Editor Dialog closes.
Save	(Rules with deprecated syntax only) Applies a corrected rule individually to the Decoder service. See Fix Rules with Invalid Syntax.

Network Rules Tab

The Network Rules tab (**Admin** > **Services** > select a Decoder and click > **View** > **Config** > **Network Rules tab**) enables you to manage network rules. NetWitness Platform applies network rules at the packet level. Network rules consist of rule sets from Layer 2, Layer 3, and Layer 4. Multiple rules can be applied to the Decoder. Rules can be applied to multiple layers (for example, when a network rule filters out specific ports for a specific IP address). Network rules apply only to Network Decoders.

What do you want to do?

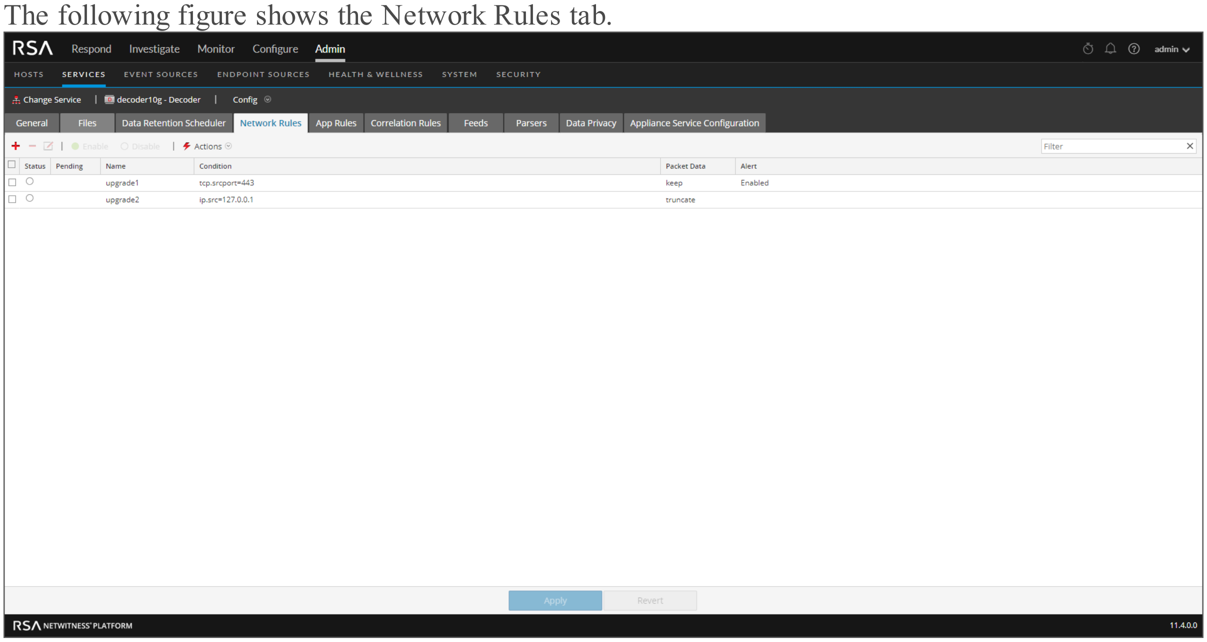
User Role	I want to...	Documentation
Administrator	add, edit, or fix network rules	Configure Network Rules

Related Topics

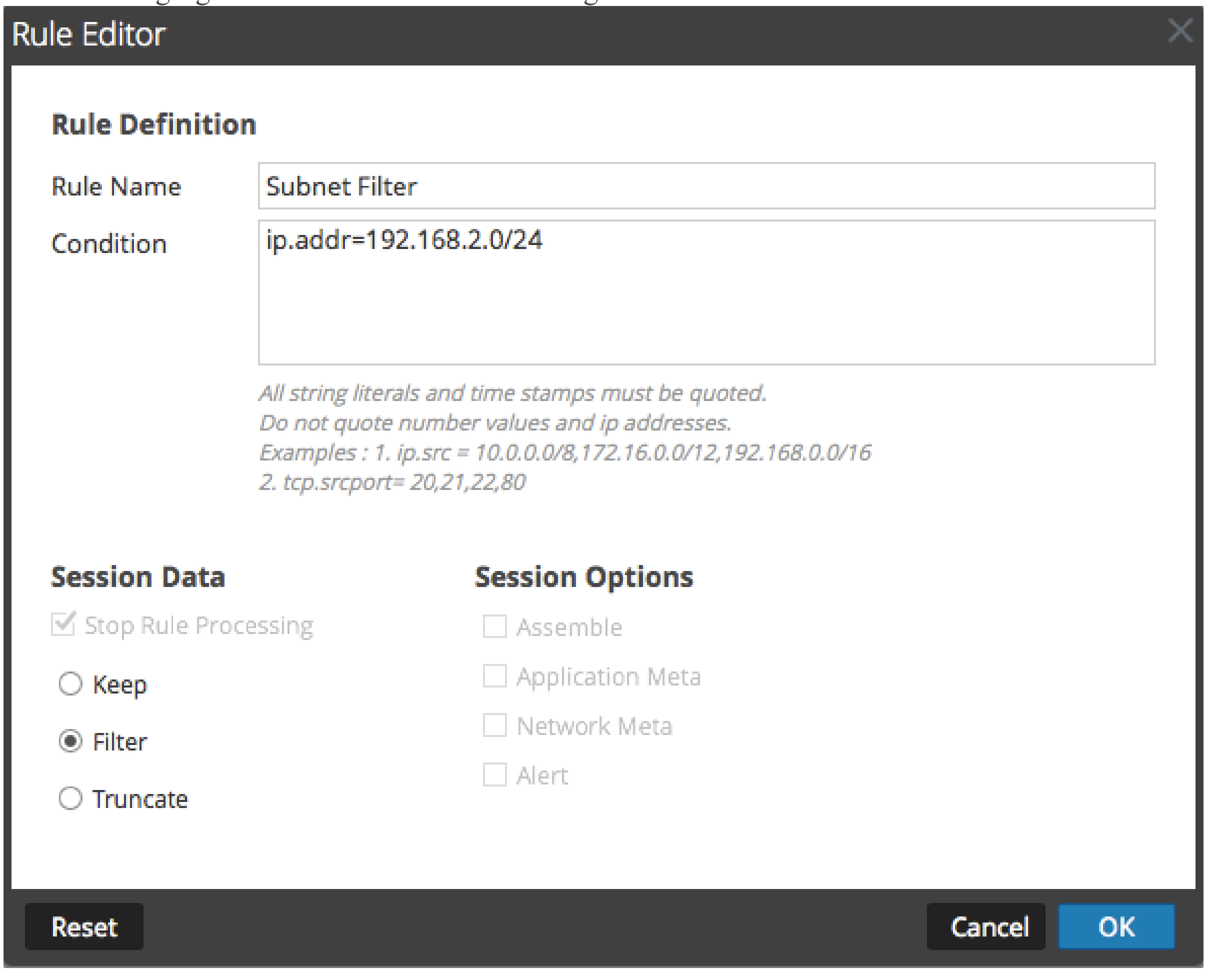
- Decoder and Log Decoder Quick Setup
- Configure Common Settings on a Decoder
- Configure Decoder Rules
- Services Config View - Rules Tabs

Quick Look

The following figure shows the Network Rules tab.



The following figure shows the Rule Editor dialog for a network rule.



The following table describes the columns in the Network Rules grid.

Column	Description
Pending	This column indicates whether a rule has pending changes. Rules that are currently active on the Decoder have no indicator. If the rule is new or has been modified, the column contains . Once the rules are applied, the pending indicator is removed.
Name	This is the rule name, a descriptive identifier for the rule.
Condition	This is the definition of the condition that triggers an action when matched.
Packet Data	This column displays the Session Data action taken when a packet matches the rule. Possible values are **Filter**, **Keep**, or **Truncate**.
Alert	This column indicates whether the Decoder generates a custom alert when metadata matches the rule. Possible values are **Enabled** or **Disabled**.
Status	This column indicates whether the rule is enabled or disabled with a circle icon. If the circle is filled green, the rule is enabled. If the circle is empty, the rule is disabled.

The **Rule Editor** dialog provides the fields and options needed to define a network rule.

The following table describes the Rule Definition fields.

Field	Description
Rule Name	The descriptive name that identifies the rule.
Condition	The definition of the condition that triggers an action when matched. You can type directly in the field or build the condition in this field using meta from the Intellisense window actions. As you build the rule definition, Intellisense displays syntax errors and warnings. In conditions, all string literals and time stamps must be quoted. Do not quote number values and IP addresses. Configure Decoder Rules provides additional details. This section also describes the meta keys that NetWitness Platform supports for use in network rule conditions.

The following table describes the Session Data actions.

Action	Description
Stop Rule Processing	If checked, further rule evaluation ends if the rule is matched, and the session is saved as indicated. If not checked, rule evaluation continues until all rules are evaluated.
Keep	The packet payload and associated meta are saved when they match the rule.
Filter	The packet is not saved when it matches the rule.
Truncate	The packet payload is not saved when it matches the rule, but packet headers and associated meta are retained.

The following table describes the session options.

Action	Description
Assemble	If checked, the assembler assembles the packet chain when it matches the rule.
Network Meta	The packet generates network metadata when it matches the rule.
Application Meta	The packet generates application metadata when it matches the rule.
Alert	The packet generates a custom alert when metadata matches the rule.

The following table describes Rule Editor dialog actions.

Action	Description
Reset	Resets the contents of the dialog to their values before editing; changes are discarded.
Cancel	Cancels any edits and closes the Rule Editor dialog.
OK	Saves the new rule or edited rule, and adds it to the rules grid. The Rule Editor dialog closes.
Save	(Rules with deprecated syntax only) Applies a corrected rule individually to the Decoder service. See Fix Rules with Invalid Syntax.

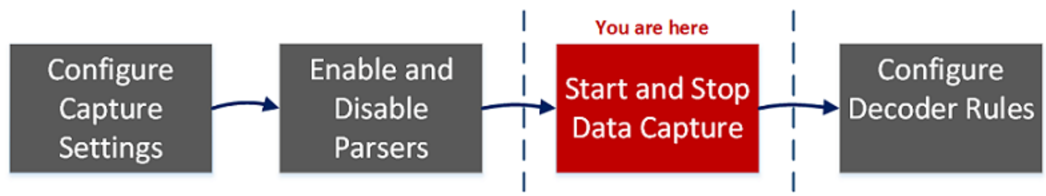
Services System View - Decoders

A Log Decoder is a special type of Decoder, and is configured and managed in a similar way to a Decoder. Therefore, most of the information in this section refers to both types of Decoders. Differences for Log Decoders are noted.

To reach the Services System view, go to **Admin > Services >** select a Decoder or Log Decoder > > **View** > **System**.

Workflow

The following figure depicts the workflow for common Decoder configuration tasks with the steps you can complete in this view highlighted.



What do you want to do?

User Role	I want to...	Documentation
Administrator	configure capture settings	Configure Capture Settings
Administrator	manage parsers and log parsers	Enable and Disable Parsers and Log Parsers
Administrator	start and stop data capture*	Start and Stop Data Capture
Administrator	upload packet capture and log files*	Upload a Log File to a Log Decoder Upload a Packet Capture File
Administrator	reset log stats, perform host tasks, shutdown the service, shutdown the appliance service, and reboot the host*	*Hosts and Services Getting Started Guide*
Administrator	configure rules	Configure Decoder Rules

*You can complete these tasks here.

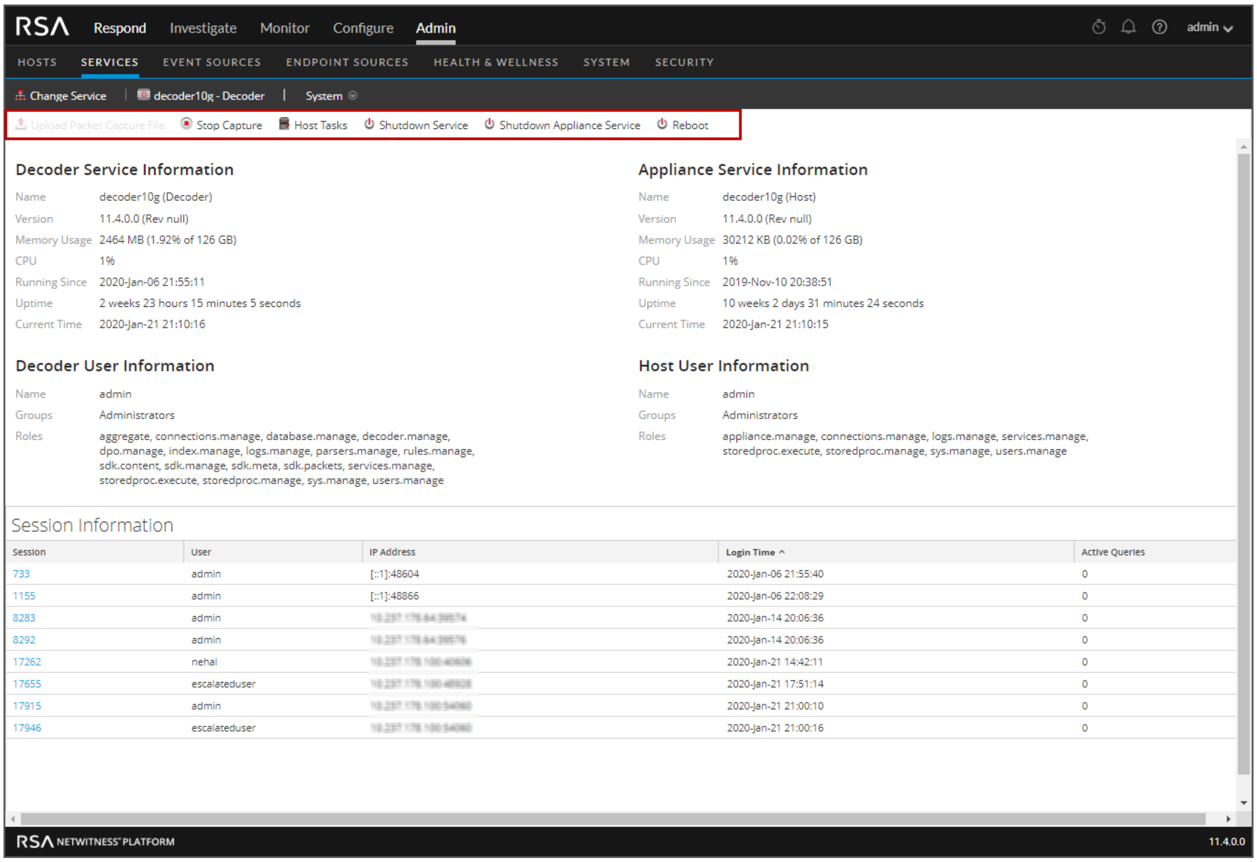
Related Topics

Go to the Master Table of Contents to find all RSA NetWitness Platform 11.x documents.

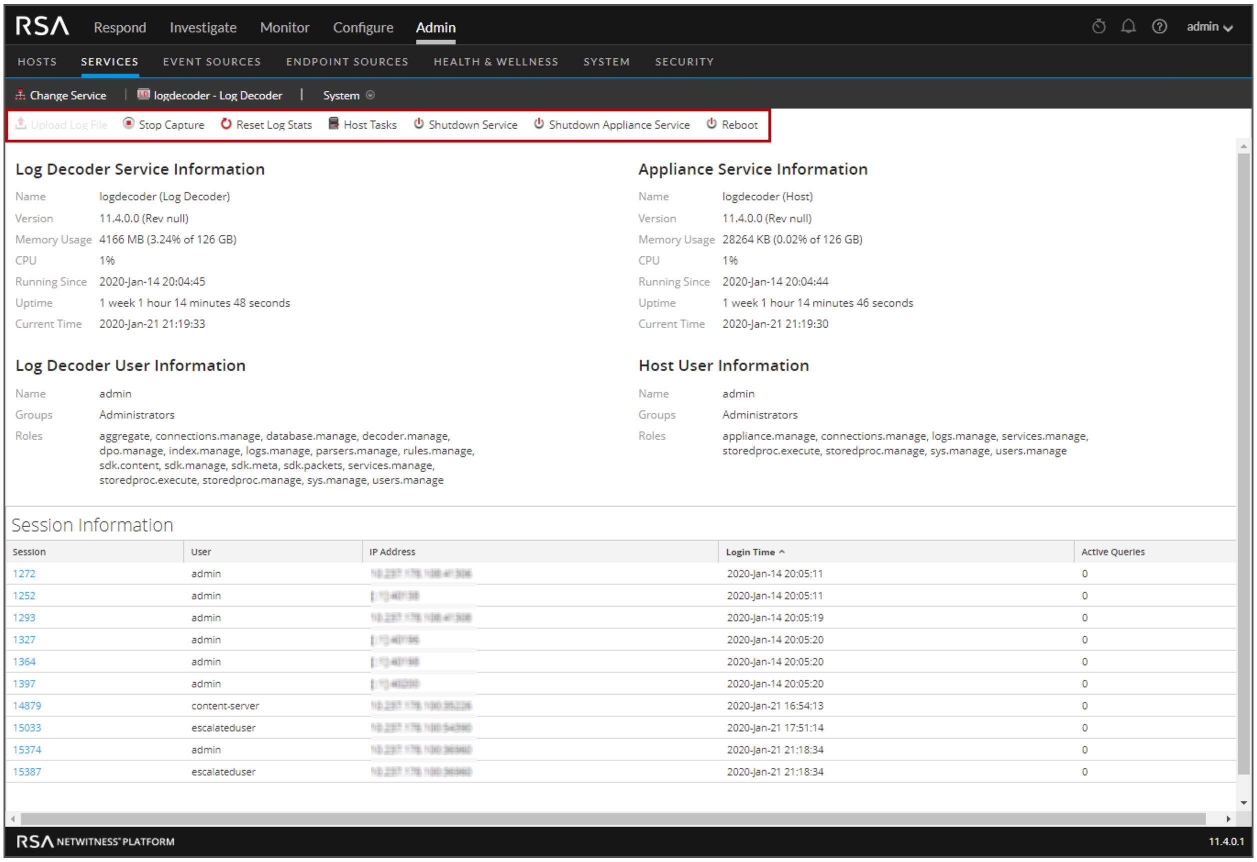
- [Decoder and Log Decoder Quick Setup]
- [Configure Common Settings on a Decoder]
- "Services System View" in the *Hosts and Services Getting Started Guide*

Quick Look

This is an example of the Services System view for a Decoder.

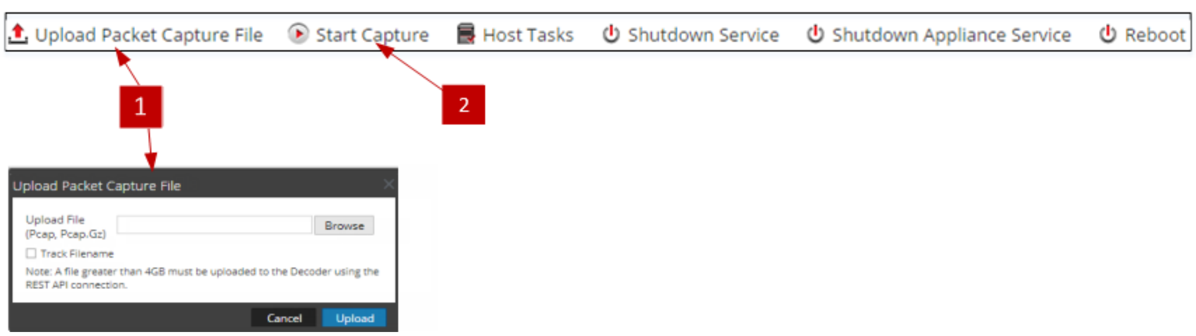


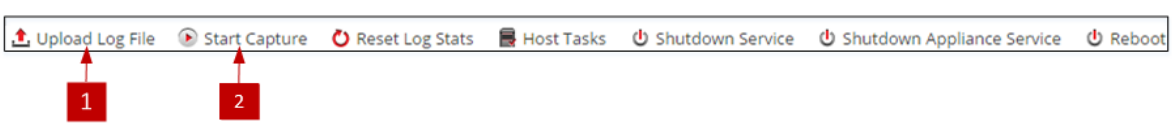
This is an example of the Services System view for a Log Decoder.



Service Info Toolbar

These two toolbars illustrate the options specific to Decoders and Log Decoders.





In addition to the common options in the Services System view toolbar, you can start and stop capture of packets or logs. The upload file options are different for the standard Decoder (packet capture file) and the Log Decoder (log file).

Action	Description
Upload Packet Capture File	Displays a dialog that provides a way to select a packet capture (.pcap) file for upload to the selected Decoder. For more information, see Upload a Packet Capture File. **Note:** This option does not apply to Log Decoders.
Upload Log File	Displays a dialog that provides a way to select a log (.log) file for upload to the selected Log Decoder. For more information, see Upload a Log File to a Log Decoder.
Start/Stop Capture	Starts packet capture on the selected Decoder. When packet capture is in progress, the option in the toolbar changes to Stop Capture, and the option to upload a file is unavailable.