NetWitness® Platform
Version 12.3.1.0

RESTful API User Guide

NETWITNESS

Platform

Contact Information

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

Trademarks

RSA and other trademarks are trademarks of RSA Security LLC or its affiliates ("RSA"). For a list of RSA trademarks, go to https://www.rsa.com/en-us/company/rsa-trademarks. Other trademarks are trademarks of their respective owners.

License Agreement

This software and the associated documentation are proprietary and confidential to RSA Security LLC or its affiliates are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability.

This software is subject to change without notice and should not be construed as a commitment by RSA.

Third-Party Licenses

This product may include software developed by parties other than RSA. The text of the license agreements applicable to third-party software in this product may be viewed on the product documentation page on NetWitness Community. By using this product, a user of this product agrees to be fully bound by terms of the license agreements.

Note on Encryption Technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

Use, copying, and distribution of any RSA Security LLC or its affiliates ("RSA") software described in this publication requires an applicable software license.

RSA believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." RSA MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Miscellaneous

This product, this software, the associated documentations as well as the contents are subject to NetWitness' standard Terms and Conditions in effect as of the issuance date of this documentation and which can be found at https://www.netwitness.com/standard-form-agreements/.

November, 2023

Contents

RESTful API

The RESTful API that ships with NetWitness Platform is a way to programmatically communicate with the NetWitness architecture. It must be enabled by setting `/rest/config/enabled` to **on**, which is the default. The default port for communication is the default port + 100 (for example, on a Concentrator, the default port is **50005**, and the REST API is on port **50105**), but that can be changed by setting the `/rest/config/port` parameter. SSL is controled by the setting in `/sys/config/ssl`. For information about how to perform these tasks, see Access the RESTful API in NetWitness Platform.

The API is based on HTTP and is quite easy to use. The acceptable output formats are:

- text/plain
- text/xml
- text/html
- application/json

The content type that is returned can be controlled through the **HTTP Accept** header. It is possible to set the parameter **force-content-type** to one of the previous values.

The easiest way to begin is to point a browser to the REST port (for details about how to perform these tasks, see Access the RESTful API in NetWitness Platform):

`PATH: http://<hostname or IP>:<REST port>`

This command performs the default operation of `ls`, and returns a listing for the root node tree used by NetWitness:

connections (*)

database (*)

decoder (*)

index (*)

logs (*)

rest (*)

sdk (*)

services (*)

storedproc (*)

sys (*)

users (*)

The user interface that is shown above is enabled by default in all Version 10 and 11 deployments. The user interface with the dark background (shown below) is available in Version 11.3.1 and later, and must be enabled if you want to use it.
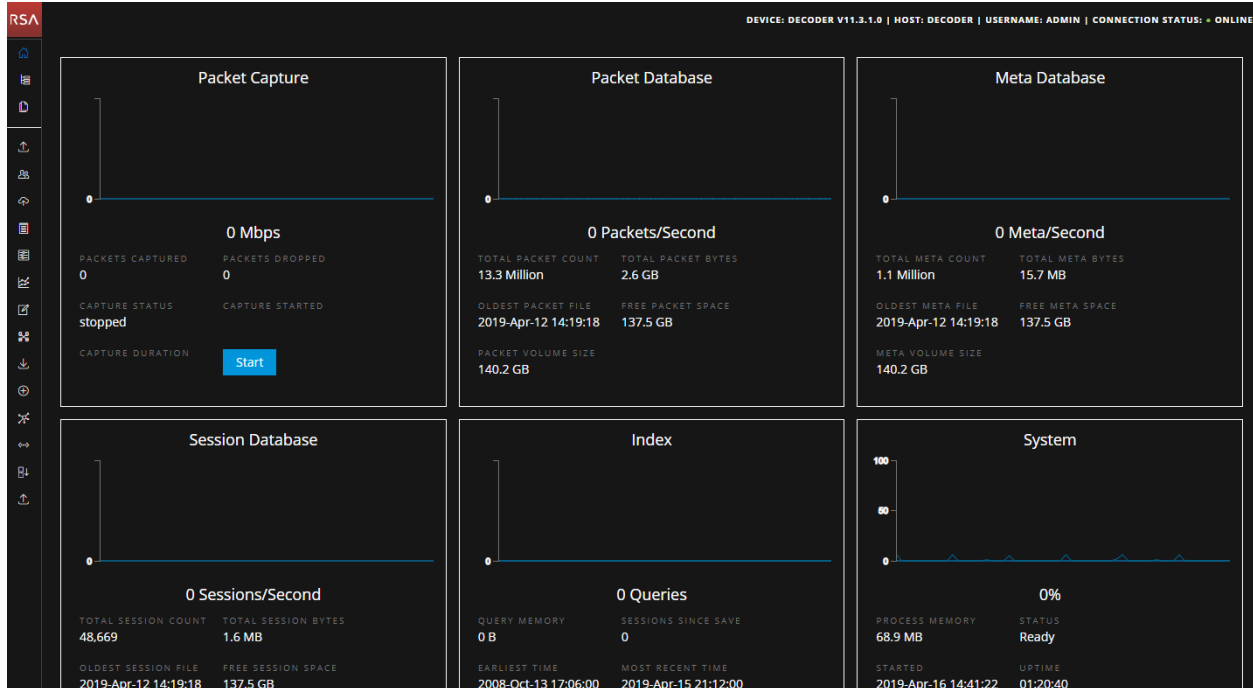
To enable the user interface with the dark background, set `/rest/config/Enable WebSockets Interface (allow.websockets)` to `true` and restart the service. You can change the user interface back to the light background by setting this parameter to `false` and restarting the service.

Usage

The REST API accepts commands by using URL parameters and by POSTing application/json.

The special content type, `application/x-netwitness-string-params`, passes parameters, as plain text, in the format:

```
param1=value1 param2="value \"2\""
```

> **Note:** Quotes, as part of the value, must be preceded by the backslash \ character. Any character can be escaped in this manner, including the backslash itself \\.

The format of the URL consists of the following components:

```
http://<hostname or IP>:<port>/[node1][/node2][…]?msg=<message name>
[&param1=value1][&param2=value2][…]
```

Usage example: the "/logs" Node

The /logs node supports several different messages:

- **ls**–Returns a list of child nodes. It supports the parameters depth and options.

- **mon**–Monitor this node (and possibly descendants) for changes. However, this message is not supported by the REST API because it requires a persistent connection and pipe that cannot be done via REST. Monitoring currently requires the full NetWitness SDK library.

- **pull**–This command pulls logs from the service. It supports two parameters: **count**, which controls how many logs to return, and **id2**, which controls the ending log ID to return. **id2** is optional and when it is not provided, the last log written is returned.

- **info**–Returns detailed node information.

- **help**–The parameters are covered in more detail in Find More Details.

- **count**–A simple command to return the number of child nodes.

- **stopMon**–Stop monitoring the node from a previous mon command (also not supported by REST).

- **download**–A more complicated command to download a large number of log messages with several parameters to control log types and text matching capabilities. Like the **mon** command, this requires more than a simple request/response, which is not supported by the REST node interface.

- **timeroll**–Any log entries that exceed a given age are deleted.

To get a full list of NetWitness messages and parameters, use the help message:

```
http://localhost:50106/logs?msg=help&force-content-type=text/plain
```

This command returns:

```
description: A container node for other node types
security.roles: everyone,logs.manage
message.list: The list of supported messages for this node
ls: [depth:<uint32>] [options:<string>] [exclude:<string>]
mon: [depth:<uint32>]
pull: [id1:<uint64>] [id2:<uint64>] [count:<uint32>] [timeFormat:<string>]
info:
help: [msg:<string>] [op:<string>] [format:<string>]
count:
stopMon:
download: [id1:<uint64>] [id2:<uint64>] [time1:<date-time>] [time2:<date-
time>] op:<string> [logTypes:<string>] [match:<string>] [regex:<string>]
[timeFormat:<string>] [batchSize:<uint32>]
timeRoll: [timeCalc:<string>] [minutes:<uint32>] [hours:<uint32>]
[days:<uint32>] [date:<string>]
debugGen: [count:<uint32>]
```

Example Syntax

To view the last 100 logs:

`http://hostname:50105/logs?msg=pull`

To view the logs in XML format:

`http://hostname:50105/logs?msg=pull&force-content-type=text/xml`

To see the last 10 logs in plain text:

`http://hostname:50105/logs?msg=pull&count=10&force-content-type=text/plain`

Find More Details

For more detailed information about a message (for example the pull message), request help specific to just that message. The **help** message displayed above uses the parameter name **msg**, but in the URL below, **message** is used, an alias for the help **msg** parameter to avoid conflicts with the REST API **msg**.

```
http://localhost:50106/logs?msg=help&message=pull&force-content-
type=text/plain
```

Returns this text:

pull:	Downloads N log entries security.roles: logs.manage parameters: id1 - <uint64, optional> The first log id number to retrieve, this is mutually exclusive with id2 id2 - <uint64, optional> The last log id number that will be sent, defaults to most recent log message when id1 or id2 is not sent count - <uint32, optional, {range:1 to 10000}> The number of logs to pull timeFormat - <string, optional, {enum-one:posix|simple}> The time format used in each log message, default is posix time (seconds since 1970)

Access the RESTful API in NetWitness Platform

This topic describes how to enable the REST API in NetWitness Platform. The REST API must be enabled by setting `/rest/config/enabled` to **on**, which is the default. The default port for communication is the default port + 100 (for example, **50105** for a Concentrator), but that can be changed by setting the `/rest/config/port` parameter. SSL is controled by the setting in `/sys/config/ssl`.

> **Note:** By default, the REST interface accepts **BOTH** SSL and Non-SSL connections on the REST port. By setting `/sys/config/ssl` to on, **ONLY** SSL connections will be accepted on the REST port.

To enable the REST port:

1. In the NetWitness web user interface, go to 🛠 (Admin) > **Services** and select a service, for example, a Concentrator.

2. In the **Host** column, click on the host name. The Hosts page opens, and the IP address of the host is displayed in the **Host** column. Make a note of the IP address.

   > **Note:** If the IP address listed in the Host column is the same as the IP address of the NetWitness web UI, the API is not available for that service.

3. Go to 🛠 (Admin) > **Services**, select the service, and then select **View** > **Config**. Under **System Configuration**, note the port number. You will use this port number as a basis for accessing the API, but you must add 100 to it. For example, if the port number is listed as **50005**, you would enter **50105**.

4. In the browser, type the IP address of the service and append the port number to the IP address as shown here:
   `http://<hostname or IP address>:<port>`

   > **Note:** The URL is HTTP, and not HTTPS.

5. In the Authentication dialog, enter the user name and password and click **Log in**. The root node tree used by NetWitness is displayed:

(Optional) Configure Custom SSL Certificate for the REST Interface

IMPORTANT: As a security best practice, using self-signed certificates is not recommended.

Note: In version 11.6, custom certificates are not supported on Log Collector.

By default, NetWitness Platform provides certificates for all the NetWitness Platform Core Services. For example, Decoder uses the `/etc/netwitness/ng/decoder_cert.pem` certificate.

You can provide your own custom certificate (preferably issued by a certification authority) for the REST ports. The custom certificates must be in the OpenSSL PEM format. Perform the following steps to configure the custom certificates:

1. Rename the custom certificate to `<service name>_rest_cert.pem` and upload it to `/etc/netwitness/ng/`. For example, `decoder_rest_cert.pem`.

2. Rename the private key to `<service>_rest_key.pem` and upload it to `/etc/netwitness/ng`. For example, `decoder_rest_key.pem`.

Note: You can combine the certificate and the private key in a single file and name it as `<service>_rest_cert.pem`. For example, `decoder_rest_cert.pem`.

If custom certificates are not detected, the default NetWitness Platform certificates are used.

Packets

You can retrieve a **pcap** file using the REST service.

`http://<hostname>:<port>/sdk/packets`

If you point a browser to this URL, the web page lets you enter a list of session IDs or a time range. When you click **Submit**, it generates a **pcap** based on the supplied criteria.

Programmatically, using HTTP GET or POST, submit either a **sessions** parameter with a comma-separated list of session IDs and session ranges (#-#) or a **time1** and **time2** parameter. Times must be in the format **YYYY-MM-DD HH:MM:SS**, for example: **2010-Apr-20 09:00:00**.

> **Note:** Since the list of sessions can get quite long, this API accepts the content-type `application/x-netwitness-string-params` for a **POST** command.

Importing Packets

You can import packets to a DECODER using the REST service.

`http://<hostname>:<port>/decoder/import`

If you point a browser to this URL, the web page lets you select a **pcap** file for upload. It also accepts pcap files POSTed to this URL.

REST begins processing incoming data immediately after the HTTP header is parsed. This means import of a **pcap** file occurs quickly and allows for large transfers. There is still a limit, but it is much larger (GBs) and is based on how well the import process can keep up with the client POST coming in. If the client posts a huge pcap (many GBs), it is still possible to overfill the current buffer. Any pcap that is 4 GBs or less should be able to process without issue.

> **Note:** The DECODER cannot be concurrently importing or capturing, or an error results.

Parser/Feed Upload

You can upload Parsers and Feeds using the REST service.

`http://<hostname>:<port>/decoder/parsers/upload`

If you point a browser to this URL, the web page lets you select a parser or feed file for upload.

You can also force a reload by selecting the toggle or providing the parameter `reload=1` on the URL.

Statistics Graph

The REST interface has a built-in statistics graphing tool, which helps you monitor performance for a service during a specified time period. You can use this tool to collect and graph single or multiple statistics from a host during a specific time range. You can also graph real-time statistics.

To access the statistical graphing tool:

1. From the root node tree page, click `sdk`. (For information on accessing the root tree note page, see [Access the RESTful API in NetWitness Platform](#).)

concentrator (*)

connections (*)

database (*)

index (*)

logs (*)

rest (*)

sdk (*)

services (*)

storedproc (*)

sys (*)

users (*)

The options for additional functionality are displayed:

..

config (*)

stats (*)

Additional Functionality

/sdk/app/reports

/sdk/app/sessions

/sdk/app/stats

/sdk/content

/sdk/packets

2. Click `sdk/app/stats`. The Performance Monitor window opens, and statistical graphs are displayed at the bottom of the page.
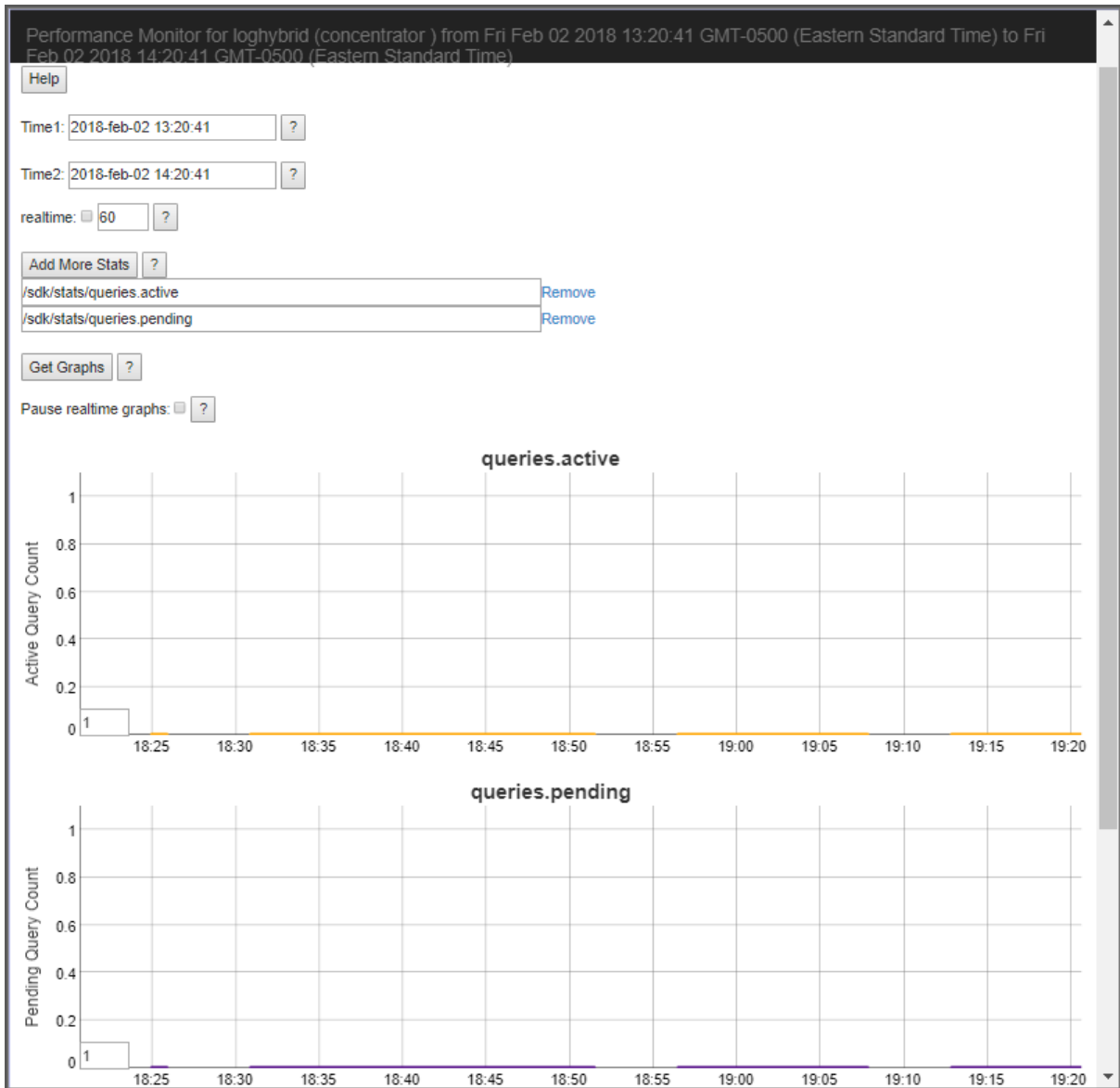
Click the Help buttons for detailed information about this page and the fields it contains.

SDK Commands

All queries on the system are performed by commands sent to the **/sdk** node.

The **/sdk** node has built-in help documentation for each message. To view the help for each command, click on the asterisk (*) beside the sdk node and then choose one of the messages from the drop-down menu. The documentation for the message is displayed in the Output window at the bottom of the screen.

To access the help:

1. From the root node tree page, click `sdk`. (For information on accessing the root tree note page, see Access the RESTful API in NetWitness Platform.)

2. Click the asterisk (*) next to sdk.

Information about the **/sdk** node is displayed.

concentrator (*)

connections (*)

database (*)

index (*)

logs (*)

rest (*)

sdk (*)

services (*)

storedproc (*)

sys (*)

users (*)

Properties for /sdk

[ls ▼] Parameters: [   ] [Send]

Message Help

```
ls: Returns the list of child nodes
security.roles: everyone
parameters:
    depth - <uint32, optional> How many levels deep to return node info, default is 1
    options - <string, optional> What types of nodes to return information about, default is all nodes.  Can be a
number (bitwise mask) or comma separated values like config, stat, folder, session, connection, channel, restart-
needed or pretty-print.
```

Output (or command manual help)

/sdk

The SDK is the primary means to access parsed metadata and the raw data that generated it. There are three main mechanisms for performing queries in the database, the *query*, *values*, and *msearch* commands. Most SDK commands over the RESTful interface override the default expiry of 30 seconds to be unlimited. Why? Because there are already mechanisms in place to cancel long running queries based on configured settings and having a small expiry value only causes confusion.

The following is a brief overview of the commands and what they do:

- **query** - Selects meta from the meta database based on the query that is passed in, possibly using the index for fast retrieval.
- **values** - Returns groups of unique meta values sorted by some criteria. It is optimized to return a subset of the unique values sorted by an aggregate function such as count.
- **msearch** - Takes text search terms as it's input, and returns matching sessions that match the search terms. It can search within indexes, meta, raw packets, or raw logs.
- **packets** - Returns raw packets or log data based on a time range, session ID list or where clause.
- **summary** - Returns name=value pairs of information regarding the active databases of the service.
- **timeline** - Returns session counts for a time period. Usually used for charting sessions over time.
- **deviceId** - Converts a session ID to a equivalent session on a remote service (e.g., which device and session ID was this aggregated from).
- **content** - Convert raw packets data to some other consumable format. Typically used to extract files out of well known protocols like HTTP or SMTP/POP.
- **aliases** - Returns the textual representations of values that are normally integer based (e.g., service 80 is HTTP).

3. To find more specific information, select a property from the **Properties for /sdk** drop-down menu:

The help for the property that you selected is displayed in the Output section:

SDK Commands Further Reference

This guide should be used in conjunction with the SDK documentation, which explains the format of queries and results. This document primarily focuses on how to send queries and parameters via the REST API, not how the queries themselves are formatted. The *Core Database Tuning Guide* explains those concepts in detail. Go to the Master Table of Contents for NetWitness Logs & Packets 11.x to find all NetWitness Platform 11.x documents. All metadata returned via REST is encoded as **UTF-8**.

There is another parameter specific to the REST API called **expiry**. This parameter can be set to the number of seconds to wait for a response before the system returns a timeout error. The default is 30 seconds, which is sufficient for most requests. For queries, the standard SDK sets an infinite timeout. If you set **expiry** to zero (`&expiry=0`), this removes the timeout for that request. It is probably a good idea to set a larger timeout for queries and other requests that may take longer than 30 seconds during normal operations.