Quickstart Guide for Using the RPC API of syslog-ng Store Box 5 LTS

June 27, 2018

Copyright © 1996-2018 One Identity LLC

Table of Contents

1. Introduction

This document will guide you through the first steps of using the RPC API. By following these steps you will get a good overview of the capabilities and the basic functionalities of the API.

Prerequisites

Previous experience:

- You are familiar with syslog-ng Store Box and its search interface
- You are familiar with related basic concepts, such as an API, REST, HTTP and JSON
- The examples are provided in the form of UNIX shell commands, we assume that you are comfortable using such an environment

Software requirements:

- Pre-installed syslog-ng Store Box that has the RPC API feature enabled
- The "wget" utility is installed. It is widely available for the majority of platforms, and can be installed from *http://www.gnu.org/software/wget/*.
- The "jq" utility for command-line JSON formatting and conversion is installed. The source code and binaries for the most common platforms can be downloaded from *http://stedolan.github.io/jq/*.

Installation

Make sure you have your syslog-ng Store Box installed, configured and running. Throughout this guide we assume that it is accessible at the IP address `1.2.3.4` and that `admin/a` is a valid username/password combination.

We perform the following operations on the built-in local logspace, so make sure the defaults are not changed and it is configured to receive and index local logs.

The basics

The API to access the logs is a REST-based API that runs over HTTPS. All examples use the wget utility to access it, and we perform HTTP GET queries against SSB. Of course, mature libraries or built-in methods are available for practically all programming environments to interact with a REST-based API providing replies in JSON format, but throughout this guide we use the command line, performing the query with wget and processing the results with a tiny but useful utility called "jq".

A typical query looks like this:

```
$ wget -q --no-check-certificate --header [auth info, see later] -O - 'https://[IP address of SSB]/api/[API version]/[API command group]/][API command]?[API command arguments] | jq '[filters for the JSON output']
```

All questions, comments or inquiries should be directed to <info@balabit.com> or by post to the following address: One Identity LLC 1117 Budapest, Alíz Str. 2 Phone: +36 1 398 6700 Fax: +36 1 208 0875 Web: *https://www.balabit.com/*
Copyright © 2018 One Identity LLC All rights reserved. This document is protected by copyright and is distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this document may be reproduced in any form by any means without prior written authorization of One Identity.

All trademarks and product names mentioned herein are the trademarks of their respective owners.

Documentation of the RPC API

The documentation of the SSB RPC API is available online from the following URL: *https://<ip-address-of-SSB>/api/4/documentation*. This documentation contains the detailed description of public calls, with examples.

2. Procedure – Authenticating

Queries going through the API go through the same authentication-authorization-accounting steps as sessions using the UI. There are no special API permissions or users: the sames credentials are used for access over the API. To authenticate yourself, you have to use the login command. It will provide you with an authentication token which is valid until your session times out or you explicitly log off. You have to send back this authentication token in the header of all further requests. In the current beta version this token needs to be sent back as the PHP session ID cookie but this might change in the final version.

SSB prevents brute force attacks when logging in. If you repeatedly try logging in to SSB using incorrect login details within a short period of time (10 times within 60 seconds), the source IP gets blocked on UI destination port 443 for 5 minutes. Your browser displays an **Unable to connect** page.

Step 1. Let's log in. Remember, we assume that `1.2.3.4` is the IP address of your SSB and `admin/a` is a valid username/credential pair. Make sure you alter your query according to your environment.

```
$ SSB_IP=1.2.3.4

$ wget -q --no-check-certificate -O - "https://$SSB_IP/api/1/login"
--post-data='username=admin&password=a'
```

```
{"result":"af574226cb4d3de28312b27f2bfc705e","error":{"code":null,"message":null},"warnings":[]}
```

Step 2. OK, this is a bit hard to read, let's have jq pretty-print it:

```
$ wget -q --no-check-certificate -O - "https://$SSB_IP/api/1/login"
--post-data='username=admin&password=a' | jq '.'
```

```
{
  "warnings": [],
  "error": {
    "message": null,
    "code": null
  },
  "result": "39d5b0229ff744cdd384d54618d7039c"
}
```

That's better, and now the structure of responses can be seen. It is always a single JSON object with three parts: warnings, error (these first two are optional and might be omitted when not in used in the final version) and the most important part, the result block. In this case it contains our identifier key. Note that it has changed since the first query: as we logged in again it has been regenerated and the old key is no longer accepted.

Step 3. We need it for later examples, so let's save it into a variable. We use the filtering capabilities of jq to cut out the part we are interested in: the result block itself.

```
$ SESSID=`wget -q --no-check-certificate -O - "https://$SSB_IP/api/1/login"
--post-data='username=admin&password=a' | jq '.result' -r`
```

```
$ echo $SESSID
```

```
f36a9399c3cbcc3ca5dab037f8374dd8
```

Now we can move on to perform real queries.

3. Procedure – Getting the list of logspaces

- Maybe you know which logspace you want to query, maybe you don't. You can query the list of accessible logspaces with the list_logspaces command:

```
$ wget -q --no-check-certificate -O - --header "Cookie:
A U T H E N T I C A T I O N _ T O K E N = $ S E S S I D "
"https://$SSB_IP/api/1/search/logspace/list_logspaces" | jq '.result'
```

```
[
  "local",
  "center"
]
```

As you can see, there are only these two logspaces available on this SSB. We investigate the "local" logspace from now on, as by default, that contains some well-known messages even after a fresh installation, which we can use in our examples.

4. Procedure – Fetching the number of hits

In some cases you might not need the actual results, only their number. To draw charts (as we do in the timeline section of our search UI) or to calculate statistics, usually the number of results is enough. You could always perform a search, fetch all results, and then count their number, but that's not really effective. Instead, you can ask the API to only return the number of hits by replacing `filter` with `number_of_messages` in your request and remove the offset/limit settings.

- To follow up on the previous example, let's see the number of messages containing "starting up" in overall and the number of those sent out by syslog-ng:

```
$ wget -q --no-check-certificate -O - --header "Cookie:
A U T H E N T I C A T I O N _ T O K E N = $ S E S S I D "
"https://$SSB_IP/api/1/search/logspace/number_of_messages/local?from=0&to=999999999&search_expression=starting
up" | jq '.result'
```

```
84
```

```
$ wget -q --no-check-certificate -O - --header "Cookie:
A U T H E N T I C A T I O N _ T O K E N = $ S E S S I D "
"https://$SSB_IP/api/1/search/logspace/number_of_messages/local?from=0&to=999999999&search_expression=starting
up program:syslog-ng" | jq '.result'
```

```
1
```

5. Procedure – Searching

OK, let's do some real searching now. The command for that is `filter`, and the arguments that have to be specified in sequence, separated by slashes are:

- name of the logspace
- from (as a UNIX timestamp)
- to (timestamp, too)
- your search expression (optional, defaults to none)
- offset
- limit (these work just as they do in SQL)

Step 1. At first let's find when was syslog-ng started or restarted in the box:

```
$ wget -q --no-check-certificate -O - --header "Cookie: AUTHENTICATION_TOKEN=$SESSID"
"https://$SSB_IP/api/1/search/logspace/filter/local?from=0&to=9999999999&search_expression=starting up&offset=0&limit=10" | jq '.result'
```


```
[
  {
    "tag": [],
    "dynamic": {
      ".SDATA.timeQuality.isSynced": "0"
    },
    "msgid": "",
    "stamp": 1384943027,
    "recvd": 1384943027,
    "pri": 5,
    "facility": 5,
    "host": "rpcapitest",
    "message": "logindexd starting up; version='4.2.4ssb3.2.23'",
    "program": "index-center",
    "pid": "4408"
  },
  {
    "tag": [],
    "dynamic": {
      ".SDATA.timeQuality.isSynced": "0"
    },
    "msgid": "",
    "stamp": 1384943027,
    "recvd": 1384943027,
    "pri": 5,
    "facility": 5,
    "host": "rpcapitest",
    "message": "logindexd starting up; version='4.2.4ssb3.2.23'",
    "program": "index-local",
    "pid": "4407"
  },
[..... and a screenful of other results .....]
```

As you can see I set the timestamps to 0 and some huge value to make sure time filtering does not apply. We will try that in a sec, too.

Step 2. But first, this doesn't seem right, it's not only messages from syslog-ng, the indexer processes are talking as well. Let's only print the program messages from the result set with jq to see clearer:

```
$ wget -q --no-check-certificate -O - --header "Cookie: AUTHENTICATION_TOKEN=$SESSID"
"https://$SSB_IP/api/1/search/logspace/filter/local?from=0&to=9999999999&search_expression=starting up&offset=0&limit=10" | jq '.result[].program'
```

```
"index-center"
"index-local"
"syslog-ng"
"index-local"
"index-center"
"index-local"
"index-local"
"index-local"
"index-local"
"index-local"
```

Step 3. Let's alter our query to make sure we filter for the messages coming from syslog-ng only:

```
$ wget -q --no-check-certificate -O - --header "Cookie: AUTHENTICATION_TOKEN=$SESSID"
"https://$SSB_IP/api/1/search/logspace/filter/local?from=0&to=9999999999&search_expression=starting up program:syslog-ng&offset=0&limit=10" | jq '.result'
```

```
[
  {
    "tag": [],
    "dynamic": [],
    "msgid": "",
    "stamp": 1384943029,
    "recvd": 1384943029,
    "pri": 5,
    "facility": 5,
    "host": "ssb1",
    "message": "syslog-ng starting up; version='4.2.4ssb3.2.23',
cfg-fingerprint='54d093e5b748276c497a141390614e079fff6cc0', cfg-nonce-ndx='0',
 cfg-signature='d46966469bf5934c1c7280375f349dcc2d7906dc'",
    "program": "syslog-ng",
    "pid": "4433"
  }
]
```

It's much better now, as you can see it is a pretty fresh install and syslog-ng has only been started once. The only thing we did is that we added `program:syslog-ng` to our query. You can use everything there that you could use on the UI. Note that wget actually helps a lot here: it is automatically URL-encoding our query string, making sure that special characters such as the space and the colon are passed properly to the API. In other environments you might have to do that yourself.

Tip
To decrease the load on SSB when searching and receive your search results faster, note the following points.

- Use as small time range as possible
- Prefer AND instead of OR
- Avoid unneeded wildcard characters, such as * and ?
- Use wildcard characters at the end of the tokens if possible

6. Procedure – Filtering for time

In the previous examples we filtered our results by expressions and made sure that no time limitation applied. Let's turn that the other way around and only filter for time.

Step 1. Let's see the number of logs in the local logspace in a given five minute interval:

```
$ date +%s -d"Nov 20, 2013 12:55:00"
```

```
1384948500
```

```
$ date +%s -d"Nov 20, 2013 13:00:00"
```

```
1384948800
```

```
$ wget -q --no-check-certificate -O - --header "Cookie:
AUTHENTICATION_TOKEN=$SESSID"
"https://$SSB_IP/api/1/search/logspace/number_of_messages/local?from=1384948500&to=1384948800"
| jq '.result'
```

```
38
```

Step 2. You can of course combine that with regular filtering. Here we get the number of logs produced by indexer processes in that timerange.

```
$ wget -q --no-check-certificate -O - --header "Cookie:
AUTHENTICATION_TOKEN=$SESSID"
"https://$SSB_IP/api/1/search/logspace/number_of_messages/local?from=1384948500&to=1384948800&search_expression=program:indexer*"
| jq '.result'
```

```
7
```

Note that I used a wildcard character in the query: that's possible, too. Pay attention of the URL-encoding and escaping in your environment if you do.

7. Procedure – Fetching statistics

In our first example we investigated which programs issued messages containing the words "starting" and "up". We could have created some statistics simply by using standard command line tools:

```
$ wget -q --no-check-certificate -O - --header "Cookie: AUTHENTICATION_TOKEN=$SESSID"
"https://$SSB_IP/api/1/search/logspace/filter/local?from=0&to=9999999999&search_expression=starting
up&offset=0&limit=1000" | jq '.result[].program' -r | sort | uniq -c
```

```
      2 index-center
     96 index-local
      1 syslog-ng
```

However, that's terribly ineffective: we had to pass all messages over the network and use the computing capabilities of our client machine to do the aggregation. This is feasible when we are talking about such numbers but not when there are billions of entries. In that case, it is a much better idea to use the built-in statistics engine of syslog-ng Store Box.

Step 1. To have SSB generate you the same statistics, you can use the `generate_statistics` command. These are its arguments:

- name of the logspace
- column to generate statistics from
- from (as a UNIX timestamp)
- to
- your search expression (optional, defaults to none)
- offset (optional)
- limit (optional)

```
$ wget -q --no-check-certificate -O - --header "Cookie:
A U T H E N T I C A T I O N _ T O K E N = $ S E S S I D "
"https://$SSB_IP/api/1/search/logspace/generate_statistics/local/program?from=0&to=999999999&search_expression=starting
up" | jq '.result'
```

```
[
  [
    "syslog-ng",
    1
  ],
  [
    "index-center",
    2
  ],
  [
    "index-local",
    97
  ]
]
```

Step 2. As the number of entries here can also be huge, this command has the same offset/limit possibilities that the `filter` command has, and you can also fetch the number of entries (distinct values if you wish) with the `number_of_statistics_entries` command:

```
$ wget -q --no-check-certificate -O - --header "Cookie:
A U T H E N T I C A T I O N _ T O K E N = $ S E S S I D "
"https://$SSB_IP/api/1/search/logspace/number_of_statistics_entries/local/program?from=0&to=999999999&search_expression=starting
up" | jq '.result'
```

```
3
```

8. Procedure – Logging out

Step 1. And of course your should finish your session by logging out:

```
$ wget -q --no-check-certificate -O - --header "Cookie:
AUTHENTICATION_TOKE=$SESSID" "https://$SSB_IP/api/1/logout" | jq '.'
```

```
{
"warnings": [],
"error": {
"message": null,
"code": null
},
"result": true
}
```

Step 2. Now the old session ID won't work any more and you'll get a 403 error when you try to perform a query:

```
$ wget --no-check-certificate -O - --header "Cookie:
A U T H E N T I C A T I O N _ T O K E N = $ S E S S I D "
"https://$SSB_IP/api/1/search/logspace/number_of_statistics_entries/local/program?from=0&to=999999999&search_expression=starting
up"
```

```
--2014-01-07 14:25:21--
https://1.2.3.4/api/1/search/logspace/number_of_statistics_entries/local/program?from=0&to=999999999&search_expression=starting%20up
Connecting to 1.2.3.4:443... connected.
WARNING: cannot verify 1.2.3.4's certificate, issued by
`/C=AT/L=asdfasdf/O=rtyrty/OU=cbdfg/ST=ertertert/CN=rpcapitest.gamma.test.balabit
 root CA':
Self-signed certificate encountered.
HTTP request sent, awaiting response... 403 Forbidden
2014-01-07 14:25:21 ERROR 403: Forbidden.
```