## Contents

**Preface** .................................................................................. iv

**1 About NexentaStor REST API** ..................................................... 1

Using NexentaStor REST API ...................................................... 1

Synchronous and Asynchronous Calls .......................................... 3

Setting the Limits for Collection of Data ....................................... 5

  Example 1 ............................................................................. 6

  System response: ................................................................. 6

  Example 2 ............................................................................. 7

  System response: ................................................................. 7

Using Swagger to Obtain Authentication Token for a User Account ... 8

  Example to Use the Authentication Token Obtained ................... 9

Using Specific Version of API in URLs ......................................... 9

  Determining Latest API Version Supported by An Appliance .......... 9

  Examples of Sending Requests ................................................. 10

  Accessing Swagger API Documentation .................................... 11

Configuring NexentaStor to Push Real Time Performance Metrics to Carbon Server ........ 11

  List of NexentaStor Statistics .................................................. 11

  Prerequisites for Pushing Statistics ........................................... 12

  Pushing Statistics to Your Carbon Server using REST API .......... 12
Preface

This documentation presents information specific to Nexenta products. The information is for reference purposes and is subject to change.

Intended Audience

This documentation is intended for Network Storage Administrators who are looking for details on enabling data at rest encryption and assumes that you have experience with data storage concepts, such as NAS, SAN, NFS, and ZFS.

This document also assumes that you have prior experience with NexentaStor and its command line interface used for creating, provisioning, and managing virtual and physical storage devices.

Documentation History

The following table lists the released revisions of this documentation.

Product Versions Applicable to this Documentation:

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns-5.3-RESTAPIquickstart-RevA</td>
<td>March, 2020</td>
<td>5.3 GA version</td>
</tr>
</tbody>
</table>

Contacting Support

Send your support questions and requests to support@nexenta.com.

Comments

Your comments and suggestions to improve this documentation are greatly appreciated. Send any feedback to doc.comments@nexenta.com and include the documentation title, number, and revision. Refer to specific pages, sections, and paragraphs whenever possible.
This page intentionally left blank
About NexentaStor REST API

The REST APIs provide programmatic access to managing a NexentaStor 5 appliance through HTTPS requests. It is a robust foundation for the new Nexenta Management Framework, a high-performance, multi-threaded, fault-tolerant management plane that provides a streamlined and simplified, storage-centric management experience.

| Note: | The API online documentation is a development tool and access is disabled by default. Enabling Swagger access in a production environment is not recommended. Regardless of the Swagger Documentation interface is enabled or not, the API is always available for use. |

If you must enable access to the detailed REST API Swagger documentation, follow these steps:

1. Run this CLI command:
   ```bash
   CLI@nexenta> config set rest.useSwagger=true (case-sensitive parameter)
   ```
2. Point your browser to `https://<staticIPaddress>:8443/docs` using the static IP address and password you set up during the NexentaStor installation.
3. Login to try out the APIs.
4. Explore specific APIs by selecting a method, entering part of the URL string, and clicking Try It Out.
Figure 1-1: NexentaStor REST API Interface.
Synchronous and Asynchronous Calls

NexentaStor REST API has two options of running the API calls. Some of the requests are to be done synchronously and some asynchronously. Synchronous modes are meant for requests that will get a quick response so this is basically applied for small requests. Whereas async mode is generally applied for requests that will take longer time to process. When you run an asynchronous call, NexentaStor receives the call, and returns a job ID to help you track the status of the job. This mode also returns a 202 code that implies the call is in progress. Async calls are available for POST, PUT, and DELETE methods.

Figure 1-2: Example of an Async Request.

Click on the 202 button to get a response. The call responds when the request is complete. Or you can check the job status by copying and pasting the job ID in the jobld parameter of the “jobstatus” method and by reading the “done” parameter in the “Response Body”(Figure 1-4).

Figure 1-3: Example to Check an Async Job Status.
Figure 1-4: Example of an Async Job Status.

<table>
<thead>
<tr>
<th>Response Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
</tr>
<tr>
<td>&quot;data&quot;: [</td>
</tr>
<tr>
<td>{</td>
</tr>
</tbody>
</table>
|       "jobId": "4e4ce670-578c-11e7-8ffd-6b9ff89824da",
|       "progress": 0,
|       "originalMethod": "POST",
|       "startTime": "2017-06-22T20:49:37.879Z",
|       "done": true,
|       "finishTime": "2017-06-22T20:49:38.239Z",
|       "href": "/jobStatus/4e4ce670-578c-11e7-8ffd-6b9ff89824da"
|     }, |
|     { |
|       "rel": "self",
|       "href": "/jobStatus?jobId=4e4ce670-578c-11e7-8ffd-6b9ff89824da"
|     } |
|   ] |
|   "links": [ |
|     { |
|       "rel": "self",
|       "href": "/jobStatus?jobId=4e4ce670-578c-11e7-8ffd-6b9ff89824da"
|     } |
| } |
Setting the Limits for Collection of Data

When using the API collection methods to get collection-data, there is a default maximum for one REST API request. The parameter “limit” denotes the default maximum for one REST API request. If this “limit” is not overwritten, the default maximum is 100. In this case you get at most 100 records. If the amount of items to be retrieved, say X, is lesser then 100, then X items are returned. So there is no need to tweak the “limit” parameter to get less than the “limit” values. The maximum value for “limit” is 5000, which means that no more than 5000 items will be returned.

In order to get more than the “limit” values in a single REST API request, you have 2 options:

- tweak “offset” or

  If the amount of items is greater than 5000, then using ‘offset’ is unavoidable, because the rest of items remains unaccessible within the 5000 items ‘limit’ window.

- tweak “limit”

  Tweaking limit will work only if the total amount of items is lesser than 5000. If a request greater than 5000 is passed via REST API, it’s restricted to 5000.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
<th>Parameter Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>offset</td>
<td>100</td>
<td>The first record number of collection, that would be included in result</td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>limit</td>
<td>20</td>
<td>The number of records of collection returned</td>
<td>query</td>
<td>integer</td>
</tr>
</tbody>
</table>

---
The following are 2 examples of how exactly pagination works in the REST API. Note that the request returns the «links» section for meta information, besides the data itself. This provides information about the exact location of the next and previous changes of data, so you just have to follow the links.

**Example 1**

In this example, a GET /storage/filesystems request is issued against a system that has 2048 filesystems. Note that in this example, the limit and offset have their default values.

**System response:**

```json
{
    "data": [
        {
            "path": "shares",
            "pool": "shares",
            "parent": "shares",
            "name": "shares",
            "bytesAvailable": 738662912,
            "bytesLogicalUsed": 34489856,
            "bytesReferenced": 278528,
            "bytesUsed": 184083968,
            "bytesUsedBySnapshots": 0,
            "compressionRatio": 1,
            "gid": "1832792444436969815",
            "mountPoint": "shares",
            "quotaSize": 0,
            "sharedOverNfs": false,
            "sharedOverSmb": false,
            "originalSnapshot": ",",
            "rateLimit": 0,
            "href": "/storage/filesystems/shares"
        }
    ],
    "links": [
        {
            "rel": "next",
            "href": "/storage/filesystems?offset=100&limit=100"
        }
    ]
}
```
Example 2

In this example, a GET /storage/filesystems is requested against my system which has 2048 filesystems and note that the parameters offset is set to 300 and limit is set to 500.

System response:
Using Swagger to Obtain Authentication Token for a User Account

Getting an authentication token for your user account is required for using REST API. You can generate an authentication token, using the Swagger interface, for your account and use it to authenticate anywhere where a password might be required. You also have the option to invalidate individual API tokens on a per-use basis. Authentication token you generate using Swagger remains valid for 30 minutes. Before it expires, you have an option to refresh the session.

To generate an API token for a user account:

1. Point your browser to https://<staticIPaddress>:8443/docs using the static IP address and password you set up during the NexentaStor installation.
2. Login using your account.
3. Click on the “auth” method and select the “Get auth token” API call.
4. Enter the user name and password. Click Try it out.
   This generates the authentication token.

To refresh the session for a user account:

1. Use the following API call under the method “auth”.

Example to Use the Authentication Token Obtained

You can use the authentication token obtained by passing in the "Authorization: Bearer" header in all subsequent requests as shown below.

Example:

curl -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyUmVmIjp7InJvb3QiOiJidXR3X2RldF92YWx1ZSIsInJvb3 RcIjoiQWxhYW0iLCJhdWQiOiJlZGluaWQiLCJpcCI6bW9kZSJ9.fpm1JUa7o14SwGIVS9Z6P0wz_87kVJYQSa116Iog87PU" -k https://172.16.159.162:8443/users

Using Specific Version of API in URLs

NexentaStor 5.3 uses version 1.2.6 of the REST API. The new version 1.2.6 is documented through the swagger web pages. For backwards compatibility, NexentaStor 5.3 continues to support API versions 1.1.X and 1.0.X. The following tables lists the versions of the REST API that are supported by specific versions of NexentaStor and the type of changes were made:

Table 1-1: API Version Corresponding to the NexentaStor version.

<table>
<thead>
<tr>
<th>NexentaStor Version</th>
<th>API Version</th>
<th>Changes in API version</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0.1</td>
<td>v1.0.1</td>
<td>Initial release</td>
</tr>
<tr>
<td>5.0.2</td>
<td>v1.0.2</td>
<td>Addition of new API calls</td>
</tr>
<tr>
<td>5.0.3</td>
<td>v1.0.4</td>
<td>Addition of new API calls</td>
</tr>
<tr>
<td>5.1</td>
<td>v1.1.0</td>
<td>Addition of new API calls</td>
</tr>
<tr>
<td>5.1.1</td>
<td>v1.1.1</td>
<td>Addition of new API calls, removal of obsolete calls, changes to output of existing API calls</td>
</tr>
<tr>
<td>5.2</td>
<td>v1.2.4</td>
<td>Addition of new API calls, removal of obsolete calls, changes to output of existing API calls</td>
</tr>
<tr>
<td>5.2.1</td>
<td>v1.2.5</td>
<td>Addition of new API calls, changes to output of existing API calls</td>
</tr>
<tr>
<td>5.3</td>
<td>v1.2.6</td>
<td>Addition of new API calls, removal of obsolete calls, changes to output of existing API calls</td>
</tr>
</tbody>
</table>

Note: For backwards compatibility, API calls without an explicit version to a NexentaStor 5.3 are treated per version 1.0 of the API. To get access to the latest API methods supported in NexentaStor 5.3, the application must explicitly specify API version 1.2.6.
Determining Latest API Version Supported by An Appliance

Before using the API, verify from NexentaStor appliance that it supports one of the API versions the application uses.

❖ To view the supported API version using curl:

1. Send a request to an arbitrary REST API endpoint. (Use the endpoint for getting the information about the current user)

   Example:
   ```
curl -k -v https://10.3.53.109:8443/auth/user
   ```
   This example will respond with two headers:
   • X-Latest-API-Version
   • X-API-Version

2. The header "X-Latest-API-Version" will appear only If you did not specify the API version (as in the above Example) and if you are not using the latest API version. This header represents the latest supported API version.

3. The header “X-API-Version” indicates the API version used by default.

   ![Example Headers](image)

Examples of Sending Requests

• When sending an API request call using an URL if you need to use a specific version of API, for example 1.2.6, just add the version of the API in the URL. You must add the first two digits 1.2 from the example whereas you can skip the last digit .6 from 1.2.6. When the last digit is omitted, NexentaStor 5.3 REST server will default to the latest API version.

   Example:
   ```
   https://<staticIPaddress>:8443/v1.2.6/settings/properties
   Or
   curl -H -k `accept-version: 1.2.6` -v https://<staticIPaddress>:8443/settings/properties
   ```

• If you do not specify an API version number in the URL, as in this example, NexentaStor 5.3 REST server will default to API version 1.0.

   ```
   curl -k -v https://<staticIPaddress>:8443/settings/properties
   ```

Following are the different ways you can use the REST API in NexentaStor 5.3
• If you specify API version 1.2.6 in the URL and use an old REST call that no longer exists in 1.2.6, the system will respond with a message that the endpoint is not available.

• If you do not specify an API version in the URL and send a REST call that was removed or changed in the current API version 1.2.6, then NexentaStor 5.3 REST server will default to API version 1.0. If the REST call does not exist in 1.0, the system will respond with an error message that the endpoint is not available.

• When managing async jobs, use the same version of API for creating and viewing the results of a job. For example, if you create an ASYNC job (like create a pool) using a certain API version then this specific job should be viewed (managed) using the same version. Avoid mixing API versions for async jobs.

---

**Note:** To ensure an application continues to work with future versions of NexentaStor, it should always explicitly call out the API version it expects the NexentaStor system to use.

---

**Accessing Swagger API Documentation**

Once you install NexentaStor 5.3 and enable swagger using this CLI command, CLI@nexenta> config set rest.useSwagger=true (case-sensitive parameter) and point your browser to https://<staticIPaddress>:8443/docs, you will be pointed to the updated API version 1.2.6

**Configuring NexentaStor to Push Real Time Performance Metrics to Carbon Server**

NexentaStor does not support direct queries of performance metrics via API calls. However, the API can be used to configure the system to push various performance metrics via a Carbon protocol stream to a Carbon server. See Pushing Statistics to Your Carbon Server using REST API for more information.

**List of NexentaStor Statistics**

Currently NexentaStor generates the following statistics. All statistics that are currently exposed have predefined intervals of 15 secs, 30 secs, 1 minute, 30 mins, etc which means the new data is refreshed only at those predefined rates.

• Over all I/O statistics for NFS/SMB protocols (IOPS/bandwidth/R-W avg. latency);

• per NFS/SMB share I/O statistics (IOPS/bandwidth/R-W avg. latency);

• ARC stats (hit ratio);

• per CPU utilisation;

• per disk I/O utilisation (run service time, wait service time);

• per disk I/O latency (in form of histogram, per latency range);

• per FC LUNs I/O stats (IOPS/bandwidth/average latency);

• overall I/O statistics for FC protocol (IOPS/bandwidth/average latency);
• per FC target I/O stats (IOPS/bandwidth/average latency);
• per FC initiator I/O statistics (IOPS/bandwidth/R-W avg. latency);
• per dataset space usage (used/referenced/available);
• per HPR service I/O stats (send/receive speed);
• per iSCSI LUNs I/O stats (IOPS/bandwidth/average latency);
• overall I/O statistics for iSCSI protocol (IOPS/bandwidth/average latency);
• per iSCSI target I/O stats (IOPS/bandwidth/average latency);
• per iSCSI initiator I/O statistics (IOPS/bandwidth/R-W avg. latency);
• per NIC I/O statistics (bandwidth, read/write utilisation);
• per NFS protocol (v3/v4) overall I/O statistics (IOPs);
• per NFS/SMB client I/O statistics (IOPS/bandwidth/R-W avg. latency);
• per zpool generic stats (total space, allocated space, available space, deduplication/compression ratio, bytes used by snapshots);
• per zpool I/O stats (IOPS/bandwidth/R-W avg. latency).

To view and analyze these statistics in an intuitive graphical format, follow the steps explained in the next two sections.

Prerequisites for Pushing Statistics

To investigate aspects of the appliance components operation and performance, ensure that you have the following components set up in your environment.

• Carbon server(s)

To expose these statistics NexentaStor pushes the analytics to a Carbon server in your environment. See appropriate documentation for information on setting up a Carbon server if you don’t have one yet.

• Carbon metrics plain-text protocol that is supported by Carbon

For the transfer of data statistics, you must support Carbon metrics plain-text protocol.

• For more information on the carbon format and graphite, refer to http://graphite.readthedocs.io/en/latest/index.html

Now that you have a Carbon server in your environment, read the contents below to push the statistics collection to your Carbon server.

Pushing Statistics to Your Carbon Server using REST API

To help get your statistics collection into Carbon server, you simply have to register the IP of your Carbon instance using NexentaStor REST API. Once you register the IP address of the Carbon instance(s), NexentaStor pushes the statistics in Carbon compatible format. This registration process facilitates generation of summaries of capacity and performance metrics in real time and historical time. You may view the real/historical-time visual data for a selected NexentaStor appliance.
You can register more than one Carbon analytics servers using the following REST API.

- To register the IP address of the Carbon analytics consumer, do the following:
  1. Enable access to the detailed REST API using this CLI command:
     ```
     CLI@nexenta> config set rest.useSwagger=true (case-sensitive parameter)
     ```
  2. Point your browser to https://<staticIPaddress>:8443/docs using the static IP address and password you set up during the NexentaStor installation.
  3. In the NexentaStor REST API interface, select “analytics”.
  4. Select PUT Request for “Updating the carbon analytics configuration”.
  5. Add the list of IP Carbon analytics server(s) in the servers (array[string]).
     ```
     "servers": ["IP address1:port", "IP address1:port", …]
     ```
  6. For NexentaStor to push the statistics in Carbon compatible format to the Carbon analytics server, select “enabled”:true.
     ```
     Example:
     {
         "enabled": true,
         "servers": ["IP address:port"]
     }
     ```
  7. Click “Try it out!” button to add the Carbon servers.
  8. To stop pushing the NexentaStor statistics to the Carbon analytics server, use the following payload.
     ```
     {
         "enabled": false
     }
     ```
Now that you have pushed the statistics collection to the Carbon server, your statistics collection can be rendered in graphical format using a webapp that is supported by Carbon server.
Global Headquarters
2929 Patrick Henry Drive,
Santa Clara, CA
USA