Getting Started with OnCommand API Services 1.2

Abstract

The goal of this document is to familiarize you with NetApp® OnCommand® API Services. This document will help you learn how to develop sample workflows to manage your storage on the NetApp clusters running ONTAP® using API Services.

TABLE OF CONTENTS

LIST OF FIGURES

1 Overview

NetApp OnCommand API Services is the single REST (Representation State Transfer) end-point to monitor and manage all underlying NetApp storage resources running ONTAP. OnCommand API Services provides API offerings for storage management and control operations, which includes monitoring, storage provisioning and data management, in your NetApp environment. It provides a programmatic way through which third-party applications can issue requests to retrieve information about the storage environment and provision storage resources. The four REST operations Create, Read, Update, and Delete (also known as CRUD) are supported.

This document describes how to leverage the rich set of REST APIs offered by OnCommand API Services to build some key storage management workflows. The use cases that will be covered include:

- Monitoring use case: Identify the bully volume when an aggregate is reporting high latency
- Provisioning and protection use case: Provision a new volume and create a backup
- Active management use case: Take corrective action on a volume running out of space

It is to be noted that no NetApp specific libraries are required for building these use cases. You can create these workflows using any programming language you are comfortable with. This document will not cover language specific API calls. You will find language specific sample codes here. This repository will be updated periodically.

For more details about a list of the APIs offered by OnCommand API Services you can refer the API Programmer's Guide available on the Support site.

1.1 Intended Audience

This document is intended for developers creating applications that interface with the OnCommand API Services software through REST APIs.

The guide is also for storage administrators and architects who want to gain a basic understanding of how the REST APIs provided in OnCommand API Services can be used to build client applications to manage and monitor NetApp storage systems.

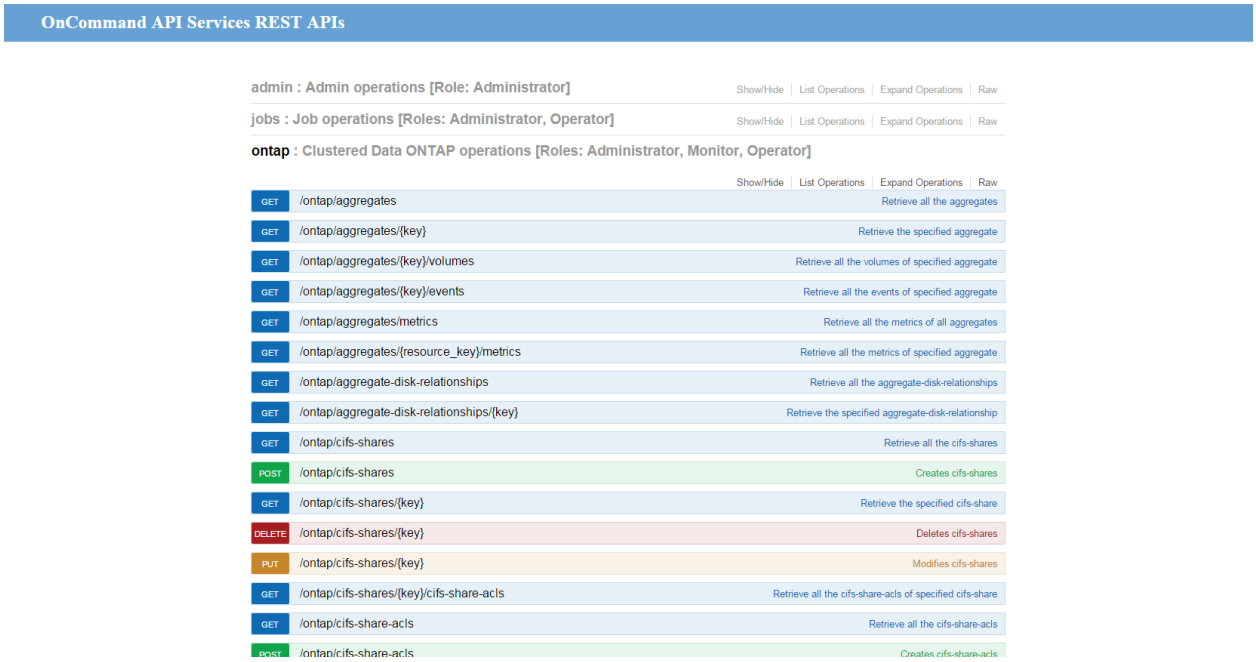
2 Getting Started

To get started, you will need:

1. A 64-bit Linux(RHEL/CentOS) system with a minimum of 12GB RAM and 50GB of free hard disk space to install API Services. For more details, refer to the Instalaltion and Administration Guide available on the Support site.
2. One Data ONTAP test system or a Data ONTAP simulator on which you can test the use cases.
3. Optional : OnCommand Unified Manager (OCUM) 6.3 or higher and OnCommand Performance Manager (OPM) 2.0 or higher. API Services can issue requests to UM or OPM for additional data, such as events and history of performance metrics.

Once you have installed API Services, using the inbuilt Swagger documentation (available at https://<api_server_ip>:<api_server_port>), you can get a list of all the APIs available. It also gives you an option to try the APIs out, before you start coding them into your application. The below images gives you a list of few of the APIs available with OnCommand API Services.

Figure 1) List of APIs available with OnCommand API Services



3 Creating workflows using API Services

Assuming that you have completed the installation and configuration of OnCommand API Services, this section provides a technical overview of how you can accomplish storage management operations by issuing GET, POST, PUT and DELETE HTTP requests to API Services. It is to be noted that API Services identifies all the storage objects using unique resource key. Also, when you enable atomic links while running an API on a storage object by using the query parameter(atomLinks=true), it gives you an URL to all the related storage objects. For example, when you perform a GET on the cluster, enabling atom links, the output you get will contain links to aggregates, storage virtual machines, nodes, cluster peers, network failver groups, network subnets, qos policy groups, storage pools, storage virtual machine peers and events. This accounts for easy navigation from one storage object to all the related storage objects.

3.1 Use case 1: Monitoring

Identify the bully volume when an aggregate is reporting high latency

The steps to be followed include

1. Identify the aggregate that is reporting high latency
2. Get all the volumes that are being served by the aggregate
3. Identify which of these volumes has high IOPs

Let us look at the API calls that you will be making to achieve each of these steps:

1. Identify the aggregate that is reporting high latency

   REST operation: GET

URL:
https://<api_server_ip:api_server_port>/api/1.0/ontap/aggregates/metrics?name=latency&atomLinks=true

Output: Lists all the aggregates and their respective latencies.

Action: Pick the aggregate with highest latency. Using the URL provided in the atom link with the relationship “volumes”, navigate to the volumes in the aggregate.

2. Get all the volumes that are being served by the aggregate

   REST operation: GET

   URL:
   https://<api_server_ip:api_server_port>/api/1.0/ontap/aggregates/{aggregate_key}/volumes?atomLinks=true

   Output: Lists all the volumes in the concerned aggregate.

   Action: For each of the volumes listed, using the URL provided in the atom link with the relationship “metrics”, navigate to the performance metrics of each of the volume. In the URL, add the filter to obtain the total operations on the volume.

3. Identify which of these volumes has high IOPs

   REST operation: GET

   URL:
   https://<api_server_ip:api_server_port>/api/1.0/ontap/volumes/<resource_key>/metrics?name=total_ops&atomLinks=true

   Output: Lists the total operation of the selected volume.

   Action: Repeat this for every volume in the aggregate. Whichever volume has the highest total operations can be considered the bully. As a corrective action, you can issue a volume move request to move it a less loaded aggregate.

   a) Move volume to a less loaded aggregate

      REST operation: POST

      URL: https://<api_server_ip:api_server_port>/api/1.0/ontap/volumes/{key}/jobs/move

      Here, {key} will be the resource key of the volume.

      POST body:

      {

      "destination_aggregate_key ":<aggregate_key of a less loaded aggregate based on the latency obtained in step 1>,

      }

      Output: An asynchronous job is created and the job ID is returned in the header.

      Action: Using the URL in the “Location” field in the HTTP response header, navigate to the newly created job and monitor till completion of the volume move.

3.2 Use case 2: Provisioning and Protection

Provision a new volume and create a backup

The mandatory parameters for creating a new volume includes the storage virtual machine key, aggregate key, name of volume and volume size.

The steps to be followed include

1. Obtain the storage virtual machine key
2. Obtain the aggregate key
3. Using all the mandatory paremeters, issue request to create a new volume
4. Track the newly created job to ensure the task completed successfully
5. Issue request to create a new snapshot

Let us look at the API calls that you will be making to achieve each of these steps:

Before starting the above steps, navigate to the cluster on which you want to create the volume. This will eliminate the possibility of running into Storage Virtual Machines or aggregates with the same names across different clusters

REST operation: GET

URL:
https://<api_server_ip:api_server_port>/api/1.0/ontap/clusters?management_ip=<cluster_ip>&atomLinks=true

Output: Lists the details of the cluster with the selected IP address.

Action: Make a note of the URLs provided in the atom link with the relationship “storage-vms” and “aggregates”, to navigate to the respective SVM and aggregates

1. Obtain the storage virtual machine key

   REST operation: GET

   URL: https://<api_server_ip:api_server_port>/api/1.0/ontap/clusters/<cluster_key>/storage-vms?name=<svm_name>

   Output: Lists all the properties of the concerned SVM

   Action: Make a note of the value corresponding to the “key” field.

2. Obtain the aggregate key

   REST operation: GET

   URL:
   https://<api_server_ip:api_server_port>/api/1.0/ontap/clusters/<cluster_key>/aggregates?name=<aggr_name>

   Output: Lists all the properties of the concerned aggregate.

   Action: Make a note of the value corresponding to the “key” field.

3. Using all the mandatory parameters, issue request to create a new volume

   REST operation: POST

   URL: https://<api_server_ip:api_server_port>/api/1.0/ontap/volumes

   POST body:
   {

   "aggregate_key":<aggregate_key from step 2>,

   "storage_vm_key":<storage_vm_key from step 1>",

   "name":<vol_name>,

"size":<size_in_bytes>

}

Please note that minimum size of volume accepted is 20 MB.
Output: An asynchronous job is created and the job ID is returned in the header.

Action: Using the URL in the “Location” field in the HTTP response header, navigate to the newly created job.

4. Track the newly created job to ensure the task completed successfully

   REST operation: GET

   URL: https://<api_server_ip:api_server_port>/api/1.0/jobs/<job_key>

   Output: Lists the progress of the asynchronous job

   Action: Wait for the job to complete. Once completed, make a note of the volume key that is returned in the “resource_key” field.

5. Issue request to create a new snapshot

   REST operation: POST

   URL: https://<api_server_ip:api_server_port>/api/1.0/ontap/snapshots

   POST body:

   {"volume_key":<volume_key from step 4>,

   "name":<snapshot_name>}
   Output: An asynchronous job is created and the job ID is returned in the header.

   Action: Using the URL in the “Location” field, navigate to the newly created job. Once the job status changes to “COMPLETED”, you can verify that the snapshot has been taken.

3.3 Use case 3: Active Management

Take corrective action on a volume running out of space

In the case of deployments with OCUM in the environment managing the cluster, you can leverage the event APIs. In case of API-S deployments without OCUM, you can use the inventory APIs to determine the size of the volume and then take corrective action. The steps to be followed include

1. Identify the volume running out of space
2. Determine the write operations on the volume
3. If corrective action is required, increase the size of the volume.

Let us look at the API calls that you will be making to achieve each of these steps:

1. Identify the volume running out of space

   a. In deployments with UM:
   REST operation: GET

   URL:
   https://<api_server_ip:api_server_port>/api/1.0/ontap/events?type=volume.space&atomLinks=true

   Output: Lists all the volumes that have reported the almost full event

   Action: For each of the events listed, using the URL provided in the atom link with the relationship “volumes”, navigate to each of the volumes. When you run a GET on each of the

volumes, using the URL provided in the atom link with the relationship “metrics”, navigate to the performance metrics. In the URL, add the filter to obtain the write operations on the volume. Also make a note of the size of the volume.

b. In standalone deployments:
REST operation: GET

URL:
https://<api_server_ip:api_server_port>/api/1.0/ontap/volumes?size_used_percent>95&atomLinks=true

Output: Lists all the volumes that are more than 95% full

Action: For each of the volumes listed, using the URL provided in the atom link with the relationship “metrics”, navigate to the performance metrics of each of the volumes. In the URL, add the filter to obtain the write operations on the volume. Also make a note of the size of the volume

2. Determine the write operations on the volume
REST operation: GET
URL:
https://<api_server_ip:api_server_port>/api/1.0/ontap/volumes/<resource_key>/metrics?name=write_ops
Output: Lists the write operation of the selected volume.
Action: Repeat this for every volume that is almost full. Whichever volume has high write operations runs at a risk running out of space very soon. Make a note of the volume key using the value in the fielf “resource_key”

3. If corrective action is required, double the size of the volume
REST operation: PUT

URL: https://<api_server_ip:api_server_port>/api/1.0/ontap/volumes

PUT body:
{"key":<volume_key from step 2>,

"size":<double of the size(bytes) from step 1>"}
Output: An asynchronous job is created and the job ID is returned in the header.

Action: Using the URL in the “Location” field, navigate to the newly created job. Once the job status changes to “COMPLETED”, you can verify that the volume size has been doubled.

Conclusion

This document describes how you can get started with NetApp OnCommand API Services by trying out a simple set of APIs in a few sample use cases. API Services gives you the flexibility to use a development language of your choice. For users who do not want to start with programming right away, you can use the swagger documentation available with the product to try out the APIs. You can also try other REST clients like cURL and POSTMAN.

Now that you are well versed with using the APIs, you can go integrate it with any third party tool that can consume RESTful APIs.

References

- Installation and Administration Guide
- Installation video

- A video on configuration and how to test the APIs
- Programmer's Guide
- Sample scripts
- Sample powershell scripts
- Splunk integration with API Services
- HP Operations Orchestrator integration with API Services

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.