Getting Started with Rubrik APIs

Module Overview

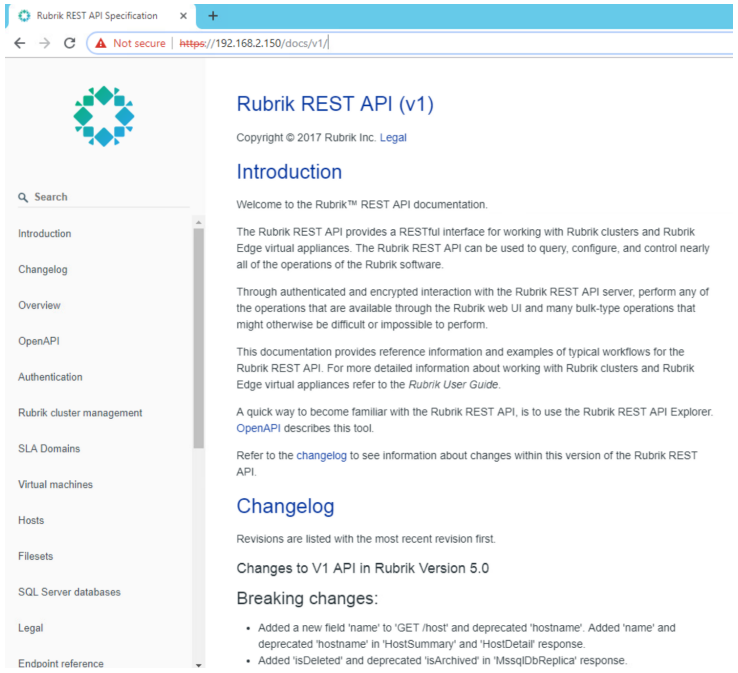
This module will cover the following:

- Overview
- Architecture
- Versioning
- Session Handling
- State and Identifiers
- Pagination
- Error Handling

Lab Resources

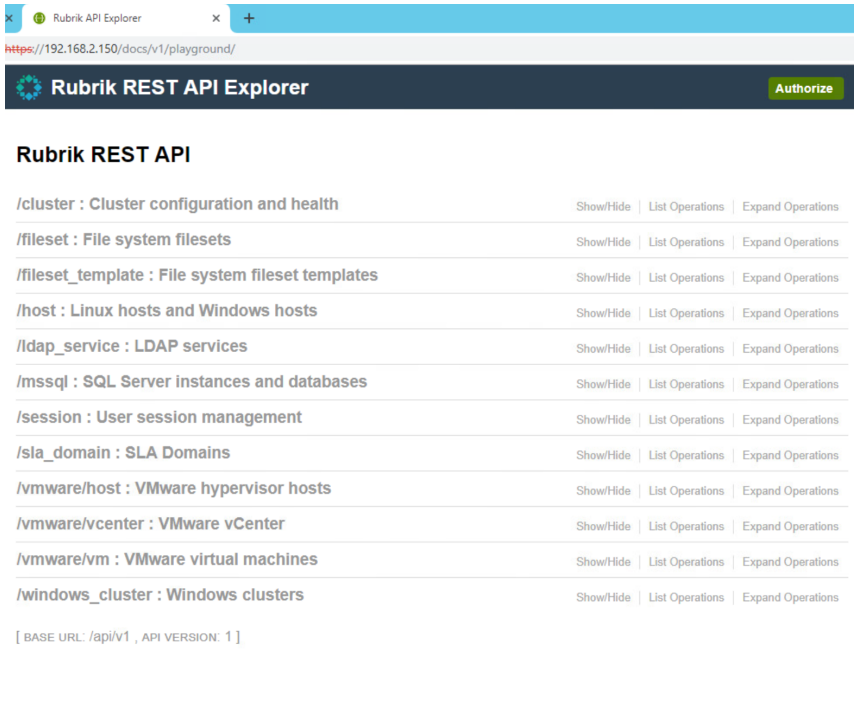
Documentation

https://192.168.2.150/docs/v1



Playground

https://192.168.2.150/docs/v1/playground



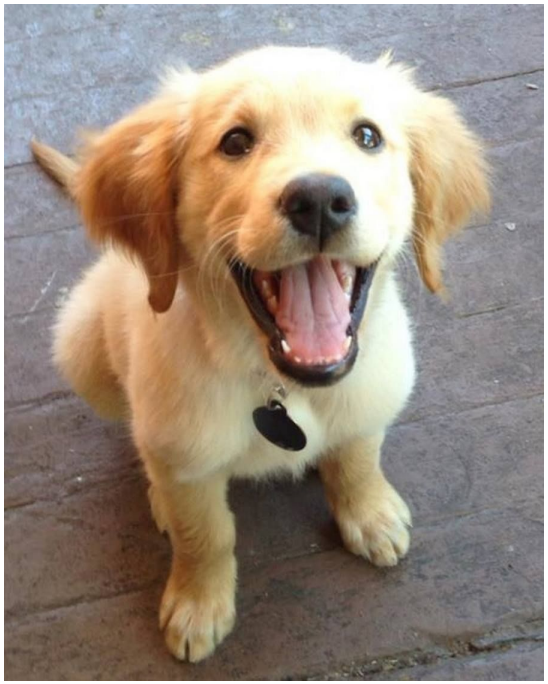
Overview

The API Economy

Most workflows and automation tasks will span across **multiple services.**

APIs are how we get those services to talk to one another.

Leveraging Your Data



- Freedom to use your data and services as desired.
- Integration with any third party services without having to wait for a vendor.
- Ability to generate custom workflows and tasks using languages you prefer.

Key Terms: A Quick Refresh

- Idempotent Responses
- Safe vs Unsafe Requests

Idempotent Responses

A response that does not change when repeated.

- GET information on a resource
  - Until someone updates the resource, you get the same response back.
  - You are not altering the data.
- PUT information into a resource
  - You are providing values for a resource.
  - The response will be identical no matter how many times you PUT values into the resource.

Safe vs Unsafe Requests

A misleading term.

- Safe Requests
  - No values are being changed on the resource.
  - GET and OPTIONS are safe.
- Unsafe Requests
  - Values are (potentially) being changed on the resource.
  - PUT, POST, PATCH, and DELETE are unsafe.

Architecture

Rubrik API History

Created with the 0.1 version of the product to initially handle UI requests across a distributed architecture of nodes.

- There is no “back end” service bus to tap into.
- Any node could potentially be asked to supply a response to a query.

API v1 Released

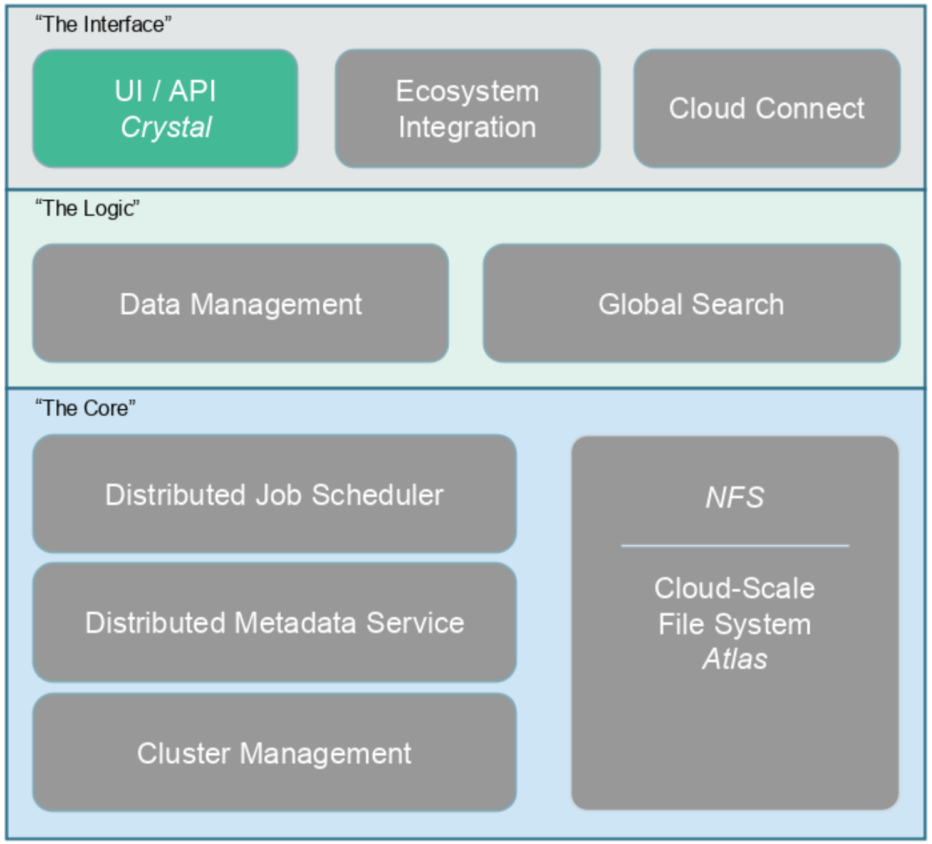
Version 1 of the API (v1) was released with Rubrik Cloud Data Management (RCDM) version 3.1.0.

- Development releases (pre-1.0) of the Rubrik REST API specification made reference to 'swagger’.
- The previous release of the OpenAPI 2.0 Specification was named the Swagger 1.2 Specification.

Crystal

Crystal is the team / project that maintains all things API and UI within Rubrik.

The API is the gateway into all methods of visibility and control over any deployment of a Rubrik cluster.



Services

- Spray
  - Framework for building REST/HTTP-based integration layers on top of Scala.
- Spray-can
  - Lightweight HTTP server
  - Handles REST methods (GET, POST, PUT, PATCH, DELETE)
- Spragger
  - Generates code using Open API Specification (JSON file)

Resources and Parameters

Resource Objects

The Rubrik REST API architecture centers around logical entities called resource objects.

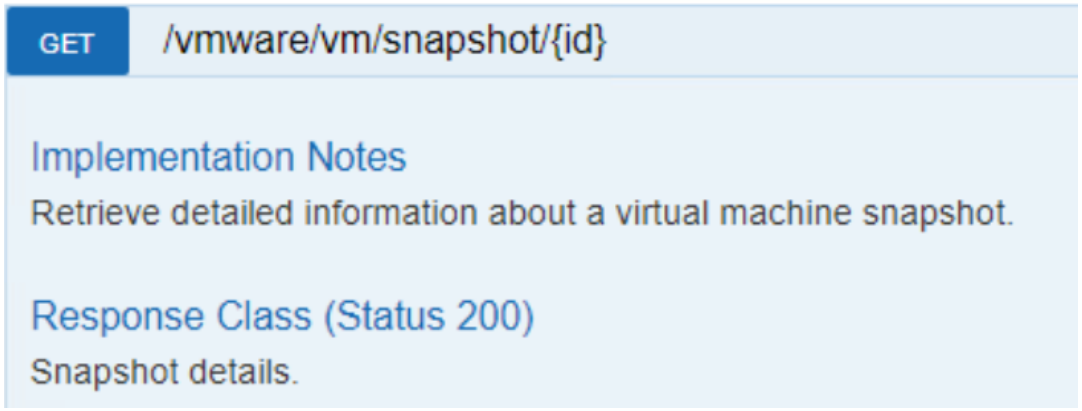
- Resource objects can be discrete entities or can contain a hierarchy of other, dependent, resource objects.
- A dependent, or child, resource object is referenced through the parent object.

Resource Objects: Example

`vmware` – parent resource object

`vm` – child object of `vmware`

`snapshot` – child object of the `vm`



Parameter Types

- Path
  - An argument that is supplied in the URI path
  - Example: `id`
- Query
  - An argument or filter that is supplied in the URI path
  - Example: `force=true`
- Body
  - A JSON payload of `key:value` pairs sent over in the request
  - Example: snapshot details when requesting a deletion

Parameter Types: Example URIs

Path

`https://{node_ip}/api/v1/vmware/vm/snapshot/`**`id_12345`**

Query

`https://{node_ip}/api/v1/vmware/vm/snapshot/id_12345`**`?force=true`**

All About Time

- The Rubrik REST API uses a timestamp to express all time values.
  - The timestamp meets the standards of ISO 8601 and uses Coordinated Universal Time (UTC).

All About Time

- Timestamps take the form: YYYY-MM-DDTHH:MM:SSZ
  - YYYY = four digit year value
  - MM = two digit month value
  - DD = two digit day of the month value
  - HH = two digit hour value using a 24-hour clock
  - MM = two digit minute value
  - SS = two digit second value
  - The T stands for time and the Z stands for Zulu time, another name for UTC.

Async Requests

- Most requests are fulfilled synchronously
  - The Rubrik REST API server uses asynchronous (async) API requests for tasks that take longer to run.
  - By using async requests the Rubrik REST API server avoids blocking the requestor from performing other tasks while the async tasks run.
- You can use the `request_id` to ask the cluster for status on the request.

Async Requests: Response

```
{
  "id": "$request_id",
  "status": "RUNNING",
  "links": [
    {
      "href": "https://$cluster_address/api/v1/vmware/vm/request/$request_id",
      "rel": "self"
    }
  ]
}
```

GET /vmware/vm/request/{id} Get asynchronous request details for VM

Implementation Notes

Get the details of an asynchronous request that involves a VMware virtual machine.

Response Class (Status 200)

Status of an asynchronous request.

Model | Example Value


```
{
  "id": "string",
  "status": "string",
  "progress": 0,
  "startTime": "2019-05-20T08:31:46.651Z",
  "endTime": "2019-05-20T08:31:46.651Z",
  "nodeId": "string",
  "error": {
    "message": "string"
  },
```


Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)	**ID of an asynchronous request.**	path	string

Lab

Lab: Async Requests

https://github.com/RoxieAtRubrik/intro-to-rubrik-apis

Introduction to APIs

Welcome to the Introduction to Rubrik APIs module! This module consists of 1 lesson to help you learn the fundamentals required to interact with Rubrik's REST APIs.

Lessons

To get started, follow the steps outlined in each lesson.

Lesson 1: Async Requests

- Exercise can be found here.

Lesson 2: API Token

- Exercise can be found here.

Versioning

Avoiding Breaking Changes

A commitment to avoid breaking things in the same major version of the API.

- The Rubrik REST API structures the request URL to include the version of the API.
  - This ensures that backwards incompatible improvements in a new version are not unintentionally used with an earlier version of the API.
  - We increment the version number when the new version includes a change that is not backward compatible.

Change Types

Examples of changes that are not backward compatible include:

- Resource type removed
- Required parameter added to an existing resource type
- Required parameter removed from an existing resource type
- Renamed resource or parameter
- Authentication requirement added for a previously unauthenticated resource

Internal Endpoints

The primary purpose of the `/api/internal` base path is to provide endpoints that are used by the Rubrik cluster.

- The evolution and improvement of the Rubrik CDM software can cause changes to these endpoints, removal of these endpoints, or addition of new `/api/internal` endpoints.
- Rubrik does not attempt to make the `/api/internal` endpoints backward compatible.

Stable Endpoints

The `/api/v1` base path provides the most commonly used endpoints.

- Rubrik considers these endpoints to be stable.
- New releases of the Rubrik REST API to provide as much backward compatibility for these endpoints as possible.
- After an upgrade to a new version existing calls to endpoints on the `/api/v1` base path will normally continue to work.

Session Handling

Session

Created whenever a user or service authenticates itself to the Rubrik cluster.

The majority of endpoints require an active, authenticated session in order to respond to requests.

- Exceptions:
  - `/session` (duh)
  - Many of the `/cluster` endpoints that provide version information

Choose Your Adventure

There are a few different routes you can use for generating a session and authenticating to the Rubrik cluster.

- Basic Authentication
  - User credentials for an ad-hoc request
  - User credentials to retrieve a session token for subsequent requests
- API Token
  - For applications and services to authenticate

User Credentials for Ad-Hoc Request

```
curl -k -u username:password -X GET
"https://$cluster_address/api/v1/clus
ter/me"
```

User Credentials for Session Token

- Send a base64 encoded header payload over to `/session`
- Retrieve a token value back in the response
- Use the token as a bearer token for subsequent requests
  - The token remains valid for the session - normally 30 minutes after the last activity.
  - You can close a session and invalidate the session token at any time by making a DELETE call to `/session/{id}` where `{id}` is the session ID or me for the current session ID.

Building the Header: Base64 Encoding

- Generate a base64 encoded string
  - Start with `username:password`
  - Encode using base64
  - Make sure you stick to UTF8
  - Cheat! https://www.base64encode.org/

Example: `dXNlcm5hbWU6cGFzc3dvcmQ=`

Building the Header: Key/Value Pairing

- Build a key/value pair
  - Key = Authorization
  - Value = Basic `{base64_string}`

Example:

`"Authorization":"Basic dXNlcm5hbWU6cGFzc3dvcmQ="`

Building the Header: Send Request

- Send a POST to `/session` with the header payload.
- Receive a response with the token.


```
{
  "id": "f409bab9-7b09-4268-b074-77608a9a932e",
  "organizationId": "Organization:::fc751974-bc34-4f1d-a234-da77d3a2caac",
  "userId": "47ea4a45-1fac-4f0c-be73-1587b2a0c807",
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0N2VhNGE0NS0xZmF
  "expiration": "2019-05-27T08:54:57Z",
  "tag": "WEB"
}
```


Lab

Lab: API Token

https://github.com/RoxieAtRubrik/intro-to-rubrik-apis

Introduction to APIs

Welcome to the Introduction to Rubrik APIs module! This module consists of 1 lesson to help you learn the fundamentals required to interact with Rubrik's REST APIs.

Lessons

To get started, follow the steps outlined in each lesson.

Lesson 1: Async Requests

- Exercise can be found here.

Lesson 2: API Token

- Exercise can be found here.

State and Identifiers

Cluster Span and Control

- Clusters can (and do) work together on tasks.
- They can also see each other's resources when replicating to one another.

Cluster Aliases

Sometimes you just want to use shortcuts!

- `me`
  - Refer to the local resource
- `local`
  - Refer to the local cluster

GET /cluster/{id}

Implementation Notes

Retrieve public information about the Rubrik cluster

Response Class (Status 200)

Information about the cluster

Model | Example Value


```
{
  "id": "string",
  "version": "string",
  "apiVersion": "string",
  "name": "string",
  "timezone": {
    "timezone": "Africa/Johannesburg"
  },
  "geolocation": {
    "address": "string"
```


Response Content Type appli ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	**me**	**ID of the Rubrik cluster or me for self.**	path	string

GET /vmware/vm — Get list of VMs

Implementation Notes

Get summary of all the VMs

Response Class (Status 200)

Virtual machine summary.

Model | Example Value

```
"name": "string",
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
effective_sla_domain_id		Filter by ID of effective SLA Domain.		string
primary_cluster_id		Filter by primary cluster ID, or **local**.	query	string
limit		Limit the number of matches returned	query	integer

GUIDs

There is a non-finite number of systems within a cluster, thus we must be able to handle duplicated resource objects.

Solution: Global Unique Identifiers (GUIDs)

The Value of GUIDs

- Some resource objects may be visible to multiple clusters, such as datastores or virtual machines.
- Some resource objects, such as virtual machines, can move to new parent resources (x-vCenter vMotion).
- Many friendly names are not the key canonical value for tracking a resource object.

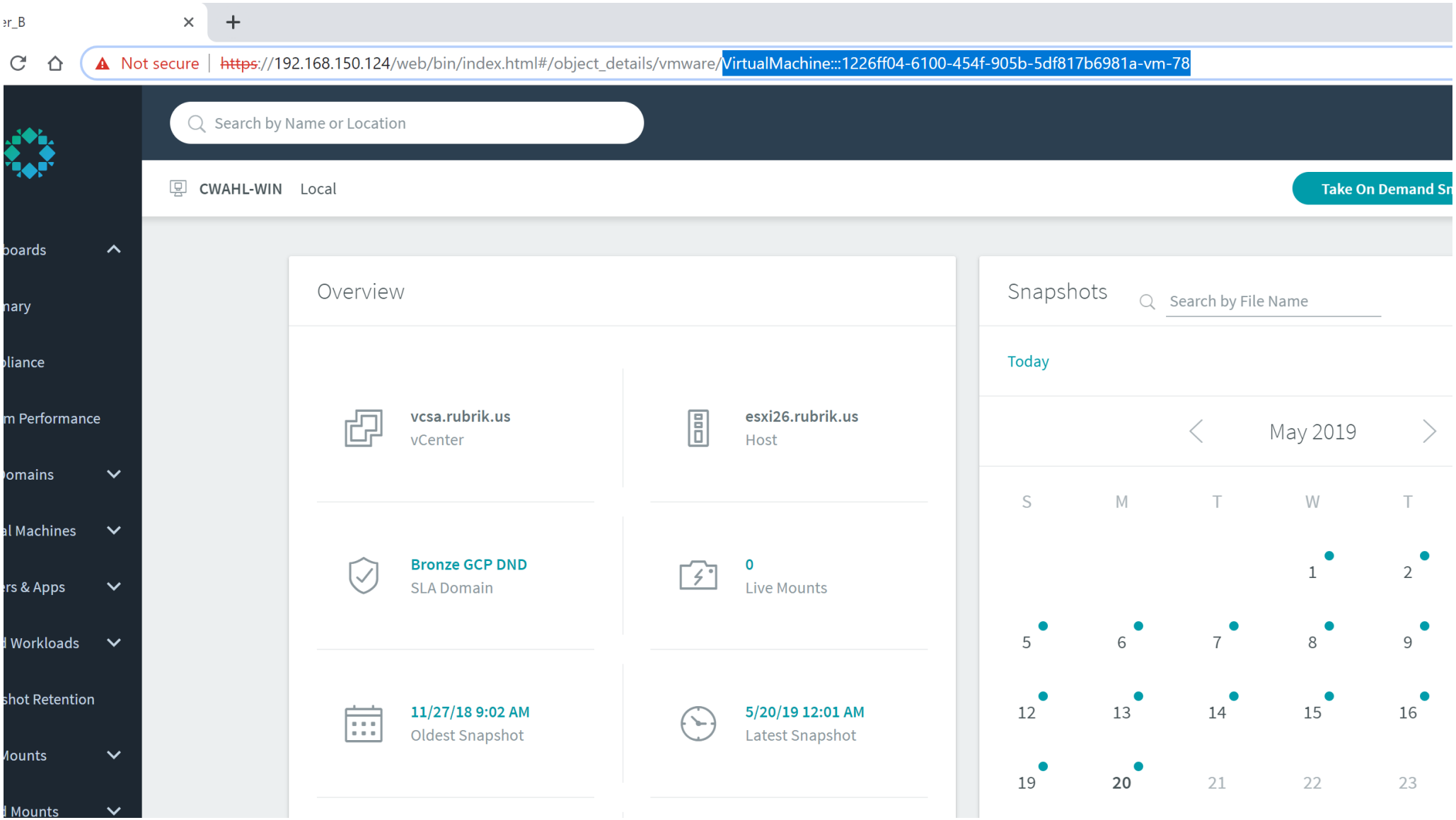
```
    {
      "id": "vCenter:::828ffd57-c67a-4844-8c37-a4079991163b",
      "managedId": "vCenter:::828ffd57-c67a-4844-8c37-a4079991163b",
      "name": "vcsa.xavier.local"
    },
    {
      "id": "DataCenter:::828ffd57-c67a-4844-8c37-a4079991163b-datacenter-21",
      "managedId": "DataCenter:::828ffd57-c67a-4844-8c37-a4079991163b-datacenter-21",
      "name": "Region North"
    },
    {
      "id": "VmwareHost:::828ffd57-c67a-4844-8c37-a4079991163b-host-223",
      "managedId": "VmwareHost:::828ffd57-c67a-4844-8c37-a4079991163b-host-223",
      "name": "esx01.xavier.local"
    }
  ],
  "effectiveSlaSourceObjectId": "VirtualMachine:::828ffd57-c67a-4844-8c37-a4079991163b-vm-327",
  "effectiveSlaSourceObjectName": "windows-2016",
  "configuredSlaDomainId": "6ce1ec85-43dd-468e-a811-77d3a1776ec2",
```

Tip

Easily Find GUIDs in the UI

The URL used to open a resource object displays the GUID

```
https://192.168.150.124/web/bin/index.html#/object_details/vmware/VirtualMachine:::1226ff04-6100-454f-905b-5df817b6981a-vm-78
```


Not secure | https://192.168.150.124/web/bin/index.html#/object_details/vmware/VirtualMachine:::1226ff04-6100-454f-905b-5df817b6981a-vm-78
Search by Name or Location
CWAHL-WIN Local
Take On Demand Sn
Overview
vcsa.rubrik.us
vCenter
esxi26.rubrik.us
Host
Bronze GCP DND
SLA Domain
0
Live Mounts
11/27/18 9:02 AM
Oldest Snapshot
5/20/19 12:01 AM
Latest Snapshot
Snapshots
Search by File Name
Today
May 2019
S M T W T
1 2
5 6 7 8 9
12 13 14 15 16
19 20 21 22 23

Lab

Comparing ID Values

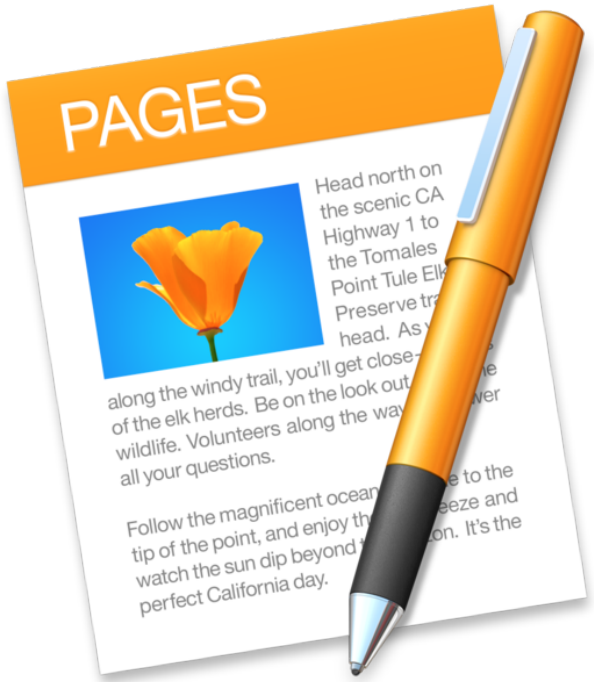
1. Get the Cluster ID value of your Rubrik cluster using “me”
2. Send the request a second time using the full id value.

- Did anything change between the calls?
- Why might you need the cluster ID value?
- Do you notice any other endpoints that can use “me”?
- Can you find any endpoint params that use “local”?

Pagination

Lots of Data!

- Sometimes your request contains a long list of resources.
- Pagination allows you to split up the list of resources into “pages” of responses.

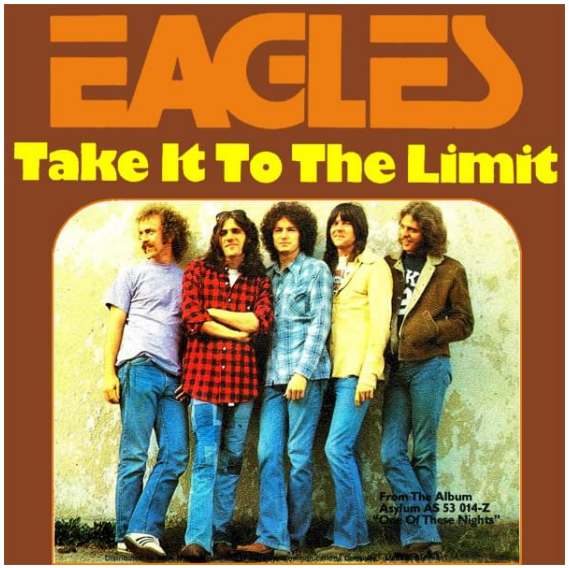


Using Limits

This limits the number of resources that will be retrieved by the API.

```
https://192.168.150.124/api/v1/vmware/vm?limit=1
```

Response with a Limit




```
{
  "hasMore": true,
  "data": [data]
  "total": 571,
  "links": {
    "next": {
      "href": "192.168.150.124/api/v1/vmware/vm?limit=1&offset=1",
      "rel": "next"
    }
  }
}
```

Using Offsets

The offset value tells the API to skip sending *n* records.

The default value is 0, which means the list page that is provided in the response starts with first element in the list.

More Values

To retrieve more values, use the “next” URI or track the limit and offset in your script / program.

```
"next": {
      "href": "192.168.150.124/api/v1/vmware/vm?limit=1&offset=1",
      "rel": "next“
       }
```

Tip

Going Backwards

When using an offset in a list, you will also see a “previous” link.

```
"links": {
    "prev": {
      "href": "192.168.150.124/api/v1/vmware/vm?limit=1&offset=0",
      "rel": "prev"
    },
    "next": {
      "href": "192.168.150.124/api/v1/vmware/vm?limit=1&offset=2",
      "rel": "next"
    }
  }
```

Error Handling

Response Codes

- Every request will return some sort of status code.
- You should always check these and don’t assume the request was successful by default.

Check the Expected Response Class

POST /sla_domain

Implementation Notes

Create a new SLA Domain on a Rubrik cluster by specifying Domain Rules and policies.

Response Class (Status 201)

Summary of newly created SLA Domain.

Model | Example Value

```
{
  "id": "string",
  "primaryClusterId": "string",
  "name": "string",
  "frequencies": [
    {
      "timeUnit": "string"
```

Tip

Common Failures

- 400 - This can be caused by malformed request syntax, invalid request message framing, or deceptive request routing.
- 422 - The Rubrik REST API server sends a response containing the HTTP status code '422 Unproccessable entity' when the request is syntactically correct but has semantic errors.
  - Example: asking for a VM id that doesn’t exist or an archive location that doesn’t exist.

Response Codes

Status code	Description
200 OK	Request succeeded. Not used for a DELETE request, or for a POST request that creates a resource.
201 Created	POST request to create a resource object succeeded.
202 Accepted	Request was successfully accepted for further processing.
204 No Content	Request succeeded and the response body is empty. Used for successful DELETE requests and for successful POST requests that do not return content.
400 Bad Request	Request failed because it was malformed. The request may be garbled, or it may be missing required parameters.
401 Unauthorized	The requestor has insufficient authorization to perform the requested action.
403 Forbidden	The requested action is blocked in the current context.
404 Not Found	The request references a resource object that is unknown to the Rubrik REST API server.
415 Unsupported Media Type	The HTTP header of the request specifies a media type that is not supported by the Rubrik REST API server.
422 Unprocessable Entity	The request specifies a correct media type and contains correct syntax but cannot be processed because of semantical errors.
500 Internal Server Error	The Rubrik REST API server encountered an unhandled error.
503 Service Unavailable Error	The Rubrik REST API server is temporarily unavailable.

Error Details

Request URL

```
https://192.168.150.124/api/v1/sla_domain
```

Response Body

```
The request content was malformed:
com.fasterxml.jackson.core.JsonParseException: Unrecognized token 'asdfasdf': was expecting ('true', 'false' or 'null')
 at [Source: (String)"asdfasdf"; line: 1, column: 17]
```

Response Code

```
400
```

Response Headers


```
{
  "date": "Tue, 21 May 2019 00:24:04 GMT",
  "content-length": "208",
  "content-type": "text/plain; charset=UTF-8"
}
```


Summary

Module Overview

- Overview
- Architecture
- Versioning
- Session Handling
- State and Identifiers
- Pagination
- Error Handling


build
Building the Future of Data Management