REST implementation details

ONTAP Automation

NetApp
September 24, 2021

Table of Contents

REST implementation details

REST establishes a common set of technologies and best practices, however the details of each API can vary based on the choices made during development. You should be aware of the design characteristics of the ONTAP REST API before using it with a live deployment.

REST web services foundation

Representational State Transfer (REST) is a style for creating distributed web applications. When applied to the design of a web services API, it establishes a set of technologies and best practices for exposing server-based resources and managing their states. It uses mainstream protocols and standards to provide a flexible foundation for administering ONTAP clusters.

Resources and state representation

Resources are the basic components of a web-based system. When creating a REST web services application, early design tasks include:

- Identification of system or server-based resources

  Every system uses and maintains resources. A resource can be a file, business transaction, process, or administrative entity. One of the first tasks in designing an application based on REST web services is to identify the resources.

- Definition of resource states and associated state operations

  Resources are always in one of a finite number of states. The states, as well as the associated operations used to affect the state changes, must be clearly defined.

URI endpoints

Every REST resource must be defined and made available using a well-defined addressing scheme. The endpoints where the resources are located and identified use a Uniform Resource Identifier (URI). The URI provides a general framework for creating a unique name for each resource in the network. The Uniform Resource Locator (URL) is a type of URI used with web services to identify and access resources. Resources are typically exposed in a hierarchical structure similar to a file directory.

HTTP messages

Hypertext Transfer Protocol (HTTP) is the protocol used by the web services client and server to exchange request and response messages about the resources. As part of designing a web services application, HTTP methods are mapped to the resources and corresponding state management actions. HTTP is stateless. Therefore, to associate a set of related requests and responses as part of one transaction, additional information must be included in the HTTP headers carried with the request and response data flows.

JSON formatting

Although information can be structured and transferred between a web services client and server in several ways, the most popular option is JavaScript Object Notation (JSON). JSON is an industry standard for

representing simple data structures in plain text and is used to transfer state information describing the resources. The ONTAP REST API uses JSON to format the data carried in the body of each HTTP request and response.

ONTAP REST API versioning

The REST API included with ONTAP 9 is assigned a version number which is independent of the ONTAP release number. You should be aware of the API version included with your release of ONTAP and how this might affect your use of the API.

All versions of ONTAP 9 use the same version of the REST API.

ONTAP 9.6 through ONTAP 9.9.1

Version 1

Related links

ONTAP 9 Release Notes

How to access the ONTAP API

You can access the ONTAP REST API in several different ways.

Network considerations

You can connect to the REST API through the following interfaces:

- Cluster management LIF
- Node management LIF
- SVM management LIF

The LIF you choose to use must be configured to support the HTTPS management protocol. Also, the firewall configuration in your network must allow the HTTPS traffic.

You should always use the cluster management LIF. This will load balance the API requests across all nodes and avoid nodes that are offline or experiencing connectivity issues.

ONTAP API online documentation page

The ONTAP API online documentation page provides an access point when using a web browser. In addition to providing a way to execute individual API calls directly, the page includes a detailed description of the API, including input parameters and other options for each call. The API calls are organized into functional categories. See Summary of the REST resource categories for more information.

The format of the URL used to access the documentation page for the most recent version of the API is:

`https://<cluster_mgmt_ip_address>/docs/api`

Custom software and tools

You can access the ONTAP API using any of several different programming languages and tools. Popular

choices include Python, Java, Curl, and PowerShell. A program, script, or tool that uses the API acts as a REST web services client. Using a programming language enables a deeper understanding of the API and provides an opportunity to automate the ONTAP administration.

The format of the base URL used to directly access the most recent version of the API is:

`https://<cluster_mgmt_ip_address>/api`

To access a specific API version where multiple versions are supported, the format of the URL is:

`https://<cluster_mgmt_ip_address>/api/v1`

Basic operational characteristics

While REST establishes a common set of technologies and best practices, the details of each API can vary based on the design choices.

Request and response API transaction

Every REST API call is performed as an HTTP request to the ONTAP system which generates an associated response to the client. This request/response pair is considered an API transaction. Before using the API, you should be familiar with the input variables available to control a request and the contents of the response output.

Support for CRUD operations

Each of the resources available through the ONTAP REST API is accessed based on the CRUD model:

- Create
- Read
- Update
- Delete

For some of the resources, only a subset of the operations is supported. You should review the ONTAP API documentation page at your ONTAP cluster for more information about each resource.

Object identifiers

Each resource instance or object is assigned a unique identifier when it is created. In most cases, the identifier is a 128-bit UUID. These identifiers are globally unique within a specific ONTAP cluster. After issuing an API call that creates a new object instance, a URL with the associated id value is returned to the caller in the location header of the HTTP response. You can extract the identifier and use it on subsequent calls when referring to the resource instance.

The content and internal structure of the object identifiers can change at any time. You should only use the identifiers on the applicable API calls as needed when referring to the associated objects.

Object instances and collections

Depending on the resource path and HTTP method, an API call can apply to a specific object instance or a

collection of objects.

Synchronous and asynchronous operations

There are two ways that ONTAP performs an HTTP request received from a client.

Synchronous processing

ONTAP performs the request immediately and responds with an HTTP status code of 200 or 201 if it is successful.

Every request using the methods GET, HEAD, and OPTIONS is always performed synchronously. In addition, requests that use POST, PATCH, and DELETE are designed to run synchronously if they are expected to complete in less than two seconds.

Asynchronous processing

If an asynchronous request is valid, ONTAP creates a background task to process the request and a job object to anchor the task. The 202 HTTP status is returned to the caller along with the job object. To determine final success or failure, you must retrieve the state of the job.

Requests that use the methods POST, PATCH, and DELETE are designed to run asynchronously if they are expected to take more than two seconds to complete.

The `return_timeout` query parameter is available with asynchronous API calls and can convert an asynchronous call to complete synchronously. Refer to `How asynchronous processing works using the Job object` for more information.

Security

The security provided with the REST API is based primarily on the existing security features available with ONTAP. The following security is used by the API:

- Transport Layer Security

  All traffic sent over the network between the ONTAP LIF and client is typically encrypted using TLS, based on the ONTAP configuration settings.

- Client authentication

  The same authentication options available with ONTAP System Manager and the Network Manageability SDK can also be used with the ONTAP REST API.

- HTTP authentication

  At an HTTP level, basic authentication is used for the API transactions. An HTTP header with the user name and password in a base64 string is added to each request.

- ONTAP authorization

  ONTAP implements a role-based authorization model. The account you use when accessing the ONTAP REST API or API documentation page should have the proper authority.

Related links

Security using RBAC

Input variables controlling an API request

You can control how an API call is processed through parameters and variables set in the HTTP request.

HTTP methods

The HTTP methods supported by the ONTAP REST API are shown in the following table.

Not all the HTTP methods are available at each of the REST endpoints. Also, both PATCH and DELETE can be used on a collection. See *Object references and access* for more information.

HTTP method	Description
GET	Retrieves object properties on a resource instance or collection.
POST	Creates a new resource instance based on the supplied input.
PATCH	Updates an existing resource instance based on the supplied input.
DELETE	Deletes an existing resource instance.
HEAD	Effectively issues a GET request but only returns the HTTP headers.
OPTIONS	Determine what HTTP methods are supported at a specific endpoint.

Request headers

You must include several headers in the HTTP request.

Content-type

If the request body includes JSON, this header must be set to `application/json`.

Accept

This header should be set to `application/hal+json`. If it is instead set to `application/json` none of the HAL links will be returned except a link needed to retrieve the next batch of records. If the header is anything else aside from these two values, the default value of the `content-type` header in the response will be `application/hal+json`.

Authorization

Basic authentication must be set with the user name and password encoded as a base64 string.

Request body

The content of the request body varies depending on the specific call. The HTTP request body consists of one of the following:

- JSON object with input variables
- Empty JSON object

Filtering objects

When issuing an API call that uses GET, you can limit or filter the returned objects based on any attribute. For example, you can specify an exact value to match:

`<field>=<query value>`

In addition to an exact match, other operators are available to return a set of objects over a range of values. The ONTAP REST API supports the filtering operators shown in the table below.

Operator	Description
=	Equal to
<	Less than
>	Greater than
⇐	Less than or equal to
>=	Greater than or equal to
UPDATE	Or
!	Not equal to
*	Greedy wildcard

You can also return a collection of objects based on whether a specific field is set or not set by using the `null` keyword or its negation `!null` as part of the query.

Any fields that are not set are generally excluded from matching queries.

Requesting specific object fields

By default, issuing an API call using GET returns only the attributes that uniquely identify the object or objects, along with a HAL self link. This minimum set of fields acts as a key for each object and varies based on the object type. You can select additional object properties using the `fields` query parameter in the following ways:

- Common or standard fields

  Specify `fields=*` to retrieve the most commonly used object fields. These fields are typically maintained in local server memory or require little processing to access. These are the same properties returned for an object after using GET with a URL path key (UUID).

- All fields

  Specify `fields=**` to retrieve all the object fields, including those requiring additional server processing to access.

- Custom field selection

  Use `fields=<field_name>` to specify the exact field you want. When requesting multiple fields, the values must be separated using commas without spaces.

As a best practice, you should always identify the specific fields you want. You should only retrieve the set of common fields or all fields when needed. Which fields are classified as common, and returned using `fields=*`, is determined by NetApp based on internal performance analysis. The classification of a field might change in future releases.

Sorting objects in the output set

The records in a resource collection are returned in the default order defined by the object. You can change the order using the `order_by` query parameter with the field name and sort direction as follows:

```
order_by=<field name> asc|desc
```

For example, you can sort the type field in descending order followed by id in ascending order:

```
order_by=type desc, id asc
```

Note the following:

- If you specify a sort field but don't provide a direction, the values are sorted in ascending order.
- When including multiple parameters, you must separate the fields with a comma.

Pagination when retrieving objects in a collection

When issuing an API call using GET to access a collection of objects of the same type, ONTAP attempts to return as many objects as possible based on two constraints. You can control each of these constraints using additional query parameters on the request. The first constraint reached for a specific GET request terminates the request and therefore limits the number of records returned.

If a request ends before iterating over all the objects, the response contains the link needed to retrieve the next batch of records.

Limiting the number of objects

By default, ONTAP returns a maximum of 10,000 objects for a GET request. You can change this limit using the `max_records` query parameter. For example:

```
max_records=20
```

The number of objects actually returned can be less than the maximum in effect, based on the related time constraint as well as the total number of objects in the system.

Limiting the time used to retrieve the objects

By default, ONTAP returns as many objects as possible within the time allowed for the GET request. The default timeout is 15 seconds. You can change this limit using the `return_timeout` query parameter. For example:

```
return_timeout=5
```

The number of objects actually returned can be less than the maximum in effect, based on the related constraint on the number of objects as well as the total number of objects in the system.

Narrowing the result set

If needed, you can combine these two parameters with additional query parameters to narrow the result set. For example, the following returns up to 10 ems events generated after the specified time:

```
time⇒ 2018-04-04T15:41:29.140265Z&max_records=10
```

You can issue multiple requests to page through the objects. Each subsequent API call should use a new time value based on the latest event in the last result set.

Size properties

The input values used with some API calls as well as certain query parameters are numeric. Rather than provide an integer in bytes, you can optionally use a suffix as shown in the following table.

Suffix	Description
KB	KB Kilobytes (1024 bytes) or kibibytes
MB	MB Megabytes (KB x 1024 bytes) or mebibytes
GB	GB Gigabytes (MB x 1024 bytes) or gibibytes
TB	TB Terabytes (GB x 1024 bytes) or tebibytes
PB	PB Petabytes (TB x 1024 bytes) or pebibytes

Related links

Object references and access

Interpreting an API response

Each API request generates a response back to the client. You should examine the response to determine whether it was successful and retrieve additional data as needed.

HTTP status code

The HTTP status codes used by the ONTAP REST API are described below.

Code	Reason phrase	Description
200	OK	Indicates success for calls that do not create a new object.
201	Created	An object is successfully created. The location header in the response includes the unique identifier for the object.
202	Accepted	A background job has been started to perform the request, but has not completed yet.
400	Bad request	The request input is not recognized or is inappropriate.
401	Unauthorized	User authentication has failed.
403	Forbidden	Access is denied due to an authorization (RBAC) error.
404	Not found	The resource referred to in the request does not exist.
405	Method not allowed	The HTTP method in the request is not supported for the resource.
409	Conflict	An attempt to create an object failed because a different object must be created first or the requested object already exists.
500	Internal error	A general internal error occurred at the server.

Response headers

Several headers are included in the HTTP response generated by the ONTAP.

Location

When an object is created, the location header includes the complete URL to the new object including the unique identifier assigned to the object.

Content-type

This will normally be `application/hal+json`.

Response body

The content of the response body resulting from an API request differs based on the object, processing type, and the success or failure of the request. The response is always rendered in JSON.

- Single object

  A single object can be returned with a set of fields based on the request. For example, you can use GET to retrieve selected properties of a cluster using the unique identifier.

- Multiple objects

  Multiple objects from a resource collection can be returned. In all cases, there is a consistent format used, with `num_records` indicating the number of records and records containing an array of the object instances. For example, you can retrieve the nodes defined in a specific cluster.

- Job object

  If an API call is processed asynchronously, a Job object is returned which anchors the background task. For example, the PATCH request used to update the cluster configuration is processed asynchronously and returns a Job object.

- Error object

  If an error occurs, an Error object is always returned. For example, you will receive an error when attempting to change a field not defined for a cluster.

- Empty JSON object

  In certain cases, no data is returned and the response body includes an empty JSON object.

HAL linking

The ONTAP REST API uses HAL as the mechanism to support Hypermedia as the Engine of Application State (HATEOAS). When an object or attribute is returned that identifies a specific resource, a HAL-encoded link is also included allowing you to easily locate and determine additional details about the resource.

Errors

If an error occurs, an error object is returned in the response body.

Format

An error object has the following format:

```
"error": {
"message": "<string>",
"code": <integer>[,
"target": "<string>"]
}
```

You can use the code value to determine the general error type or category, and the message to determine the specific error. When available, the target field includes the specific user input associated with the error.

Common error codes

The common error codes are described in the following table. Specific API calls can include additional error codes.

Code		Description
1	409	An object with the same identifier already exists.
2	400	The value for a field has an invalid value or is missing, or an extra field was provided.
3	400	The operation is not supported.
4	405	An object with the specified identifier cannot be found.
6	403	Permission to perform the request is denied.
8	409	The resource is in use.

How asynchronous processing works using the Job object

After issuing an API request that is designed to run asynchronously, a job object is always created and returned to the caller. The job describes and anchors a background task that processes the request. Depending on the HTTP status code, you must retrieve the state of the job to determine if the request was successful.

Refer to API reference to determine which API calls are designed to be performed asynchronously.

Controlling how a request is processed

You can use the `return_timeout` query parameter to control how an asynchronous API call is processed. There are two possible outcomes when using this parameter.

Timer expires before the request completes

For valid requests, ONTAP returns a 202 HTTP status code along with the job object. You must retrieve the state of the job to determine if the request completed successfully.

Request is completed before the timer expires

If the request is valid and completes successfully before the time expires, ONTAP returns a 200 HTTP status code along with the job object. Because the request is completed synchronously, as indicated by the 200, you do not need to retrieve the job state.

> The default value for the `return_timeout` parameter is zero seconds. Therefore, if you don't include the parameter, the 202 HTTP status code is always returned for a valid request.

Querying the Job object associated with an API request

The Job object returned in the HTTP response contains several properties. You can query the state property in a subsequent API call to determine if the request completed successfully. A Job object is always in one of the following states:

Non-terminal states

- Queued
- Running
- Paused

Terminal states

- Success
- Failure

General procedure for issuing an asynchronous request

You can use the following high-level procedure to complete an asynchronous API call. This example assumes the `return_timeout` parameter is not used, or that the time expires before the background job completes.

1. Issue an API call that is designed to be performed asynchronously.
2. Receive an HTTP response 202 indicating acceptance of a valid request.
3. Extract the identifier for the Job object from the response body.
4. Within a timed loop, perform the following in each cycle:
   a. Get the current state of the Job.
   b. If the Job is in a non-terminal state, perform loop again.
5. Stop when the Job reaches a terminal state (success, failure).

Related links

Workflow 1: Updating the cluster contact and checking job state