Request and response API transaction

ONTAP Select

David Peterson
November 21, 2019

Table of Contents

Request and response API transaction

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

Input variables controlling an API request

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

Request headers

You must include several headers in the HTTP request, including:

- content-type
  If the request body includes JSON, this header must be set to application/json.
- accept
  If the response body will include JSON, this header must be set to application/json.
- authorization
  Basic authentication must be set with the user name and password encoded in 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 (such as, the name of a new cluster)
- Empty

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, there are other operators available to return a set of objects over a range of values. ONTAP Select supports the filtering operators shown below.

Operator	Description
=	Equal to
<	Less than
>	Greater than
⇐	Less than or equal to
>=	Greater than or equal to
	Or

Operator	Description
!	Not equal to
*	Greedy wildcard

You can also return a set 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.

Selecting object fields

By default, issuing an API call using GET returns only the attributes that uniquely identify the object or objects. 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:

- Inexpensive fields
  Specify `fields=*` to retrieve the object fields that are maintained in local server memory or require little processing to access.
- Expensive fields
  Specify `fields=**` to retrieve all the object fields, including those requiring additional server processing to access.
- Custom field selection
  Use `fields=FIELDNAME` 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 inexpensive or expensive fields when needed. The inexpensive and expensive classification is determined by NetApp based on internal performance analysis. The classification for a given field can change at any time.

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`

When including multiple parameters, you must separate the fields with a comma.

Pagination

When issuing an API call using GET to access a collection of objects of the same type, all matching objects are returned by default. If needed, you can limit the number of records returned using the max_records query parameter with the request. For example:
`max_records=20`

If needed, you can combine this parameter with other query parameters to narrow the result set. For example, the following returns up to 10 system events generated after the specified time:
`time⇒ 2019-04-04T15:41:29.140265Z&max_records=10`

You can issue multiple requests to page through the events (or any object type). Each subsequent API call

should use a new time value based on the latest event in the last result set.

Interpreting an API response

Each API request generates a response back to the client. You can 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 Deploy REST API are described below.

Code	Meaning	Description
200	OK	Indicates success for calls that do not create a new object.
201	Created	An object is successfully created; the location response header includes the unique identifier for the object.
202	Accepted	A long-running background job has been started to perform the request, but the operation has not completed yet.
400	Bad request	The request input is not recognized or is inappropriate.
403	Forbidden	Access is denied due to an authorization error.
404	Not found	The resource referred to in the request does not exist.
405	Method not allowed	The HTTP verb in the request is not supported for the resource.
409	Conflict	An attempt to create an object failed because the object already exists.
500	Internal error	A general internal error occurred at the server.
501	Not implemented	The URI is known but is not capable of performing the request.

Response headers

Several headers are included in the HTTP response generated by the Deploy server, including:

- request-id
  Every successful API request is assigned a unique request identifier.
- location
  When an object is created, the location header includes the complete URL to the new object including the unique object identifier.

Response body

The content of the response associated with an API request differs based on the object, processing type, and the success or failure of the request. The response body is 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 all 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 POST request used to deploy a cluster 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 create a cluster with a name that already exists.
- Empty
  In certain cases, no data is returned and the response body is empty. For example, the response body is empty after using DELETE to delete an existing host.