Automation using REST APIs

SnapCenter Software 4.5

NetApp
July 03, 2026

This PDF was generated from https://docs.netapp.com/us-en/snapcenter-45/sc-automation/overview_rest_apis.html on July 03, 2026. Always check docs.netapp.com for the latest.

Table of Contents

Automation using REST APIs . . . 1
- Overview of REST APIs . . . 1
- How to access SnapCenter REST API natively . . . 1
- REST web services foundation . . . 1
  - Resources and state representation . . . 1
  - URI endpoints . . . 2
  - HTTP messages . . . 2
  - JSON formatting . . . 2
- Basic operational characteristics . . . 2
  - Request and response API transaction . . . 2
  - Support for CRUD operations . . . 2
  - Object identifiers . . . 3
  - Object instances and collections . . . 3
  - Synchronous and asynchronous operations . . . 3
  - Security . . . 3
- Input variables controlling an API request . . . 4
  - HTTP methods . . . 4
  - Request headers . . . 4
  - Request body . . . 4
  - Filtering objects . . . 4
  - Requesting specific object fields . . . 5
  - Sorting objects in the output set . . . 6
  - Pagination when retrieving objects in a collection . . . 6
  - Size properties . . . 7
- Interpretation of an API response . . . 7
  - HTTP status code . . . 7
  - Response headers . . . 8
  - Response body . . . 8
  - Errors . . . 9
- Supported REST APIs . . . 10
  - Auth . . . 10
  - Domains . . . 10
  - Jobs . . . 10
  - Settings . . . 10
  - Hosts . . . 10
  - Resources . . . 11
  - Backups . . . 11
  - Clones . . . 11
  - Clonesplit . . . 11
  - Resource Groups . . . 11
  - Policies . . . 11
  - Storage . . . 11
  - Share . . . 12

Plugins . . . 12
Reports . . . 12
Alerts . . . 12
Rbac . . . 12
Configuration . . . 12
CertificateSettings . . . 13
Repository . . . 13
How to access REST APIs using the Swagger API web page . . . 13
Get started with the REST API . . . 14
Hello World . . . 14

Automation using REST APIs

Overview of REST APIs

REST APIs can be used to perform several SnapCenter management operations. REST APIs are exposed through the Swagger web page.

You can access the Swagger web page available at *https://<SnapCenter_IP_address_or_name>:<SnapCenter_port>/swagger/* to display the REST API documentation, as well as to manually issue an API call.

The plug-ins that support REST APIs are:

- Plug-in for Microsoft SQL Server
- Plug-in for SAP HANA Database
- Custom Plug-ins

How to access SnapCenter REST API natively

You can access the SnapCenter REST API directly using any programming language that supports a REST client. Popular language choices include Python, PowerShell, and Java.

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 managing SnapCenter.

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, should 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

While 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 SnapCenter REST API uses JSON to format the data carried in the body of each HTTP request and response.

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 SnapCenter Server system which generates an associated response to the client. This request and 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 SnapCenter 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.

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 SnapCenter Server.

After issuing an API call that creates a new object instance, a URL with the associated ID 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

SnapCenter performs an HTTP request received from a client either synchronously or asynchronously.

Synchronous processing

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

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

Asynchronous processing

If an asynchronous request is valid, SnapCenter creates a background task to process the request and a job object to anchor the task. The HTTP status code 202 is returned to the caller along with the job object. You should retrieve the state of the job to determine success or failure.

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

Security

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

Transport Layer Security

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

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.

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 SnapCenter REST API are shown in the following table.

Not all the HTTP methods are available at each of the REST endpoints.

HTTP method	Description
GET	Retrieves object properties on a resource instance or collection.
POST	Creates a new resource instance based on the supplied input.
DELETE	Deletes an existing resource instance.
PUT	Modifies an existing resource instance.

Request headers

You should include several headers in the HTTP request.

Content-type

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

Accept

This header should be set to *application/json*.

Authorization

Basic authentication should 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

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 SnapCenter 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. 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
```

- If you specify a sort field but do not 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, SnapCenter 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, SnapCenter 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, SnapCenter 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 byes) or tebibytes
PB	PB Petabytes (TB x 1024 byes) or pebibytes

Interpretation of 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 SnapCenter REST API are described below.

Code	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.

Code	Description
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 SnapCenter.

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/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

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

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

Supported REST APIs

The resources available through the SnapCenter REST API are organized in categories, as displayed on the SnapCenter API documentation page. A brief description of each of the resources with the base resource paths is presented below, along with additional usage considerations where appropriate.

Auth

You can use this API call to log into the SnapCenter Server. This API returns a user authorization token that is used to authenticate subsequent requests.

Domains

You can use these API calls to perform the following:

- retrieve all the domains
- retrieve details of a specific domain
- register or unregister a domain
- modify a domain

Jobs

You can use these API calls to perform the following:

- retrieve all the jobs
- retrieve status of a job
- cancel or stop a job

Settings

You can use these API calls to perform the following:

- register, view, modify, or remove a credential
- configure notification settings

Hosts

You can use these API calls to perform the following:

- retrieve host details
- retrieve the plug-in installed and their resource details
- add, remove, or modify a plug-in host
- install or upgrade plug-ins

Resources

You can use these API calls to perform the following:

- retrieve resources
- create, modify, or remove resources
- protect a resource
- back up, restore, or clone a resource

Backups

You can use these API calls to perform the following:

- retrieve backup details
- rename or delete backups

Clones

You can use these API calls to perform the following:

- retrieve clone details
- delete clones

Clonesplit

You can use these API calls to perform the following:

- retrieve the status of a clone split operation
- start or stop a clone split operation

Resource Groups

You can use these API calls to perform the following:

- retrieve details of a resource group
- create, modify, or delete a resource group
- back up a resource group

Policies

You can use these API calls to perform the following:

- retrieve policy details
- create, modify, or delete policies

Storage

You can use these API calls to perform the following:

- retrieve storage details
- create, modify, or delete a storage
- discover resources on a storage
- create or delete a share on the storage

Share

You can use these API calls to perform the following:

- retrieve the details of a share
- create or delete a share on the storage

Plugins

You can use these API calls to retrieve all the plug-ins on a host and perform different operations.

Reports

You can use these API calls to perform the following:

- generate backup, restore, clone, and plug-in reports
- add, run, delete, or modify schedules

Alerts

You can use these API calls to perform the following:

- retrieve all the alerts
- delete alerts

Rbac

You can use these API calls to perform the following:

- retrieve details of users, groups, and roles
- add users
- create, modify, or delete roles
- assign or unassign roles and groups

Configuration

You can use these API calls to perform the following:

- view the configuration settings
- modify the configuration settings

CertificateSettings

You can use these API calls to perform the following:

- view the certificate status
- modify the certificate settings

Repository

You can use these API calls to perform the following:

- backup and restore the NSM repository
- protect and unprotect the NSM repository
- failover
- rebuild the NSM repository

How to access REST APIs using the Swagger API web page

REST APIs are exposed through the Swagger web page. You can access the Swagger web page to display the SnapCenter Server REST APIs, as well as to manually issue an API call. You can use REST APIs to help manage your SnapCenter Server or to perform data protection operations.

You should know the management IP address or domain name of the SnapCenter Server on which you want to execute the REST APIs.

You do not need special permissions to run the REST API client. Any user can access the Swagger web page. The respective permissions on the objects that are accessed via the REST API are based on the user who generates the token to login to the REST API.

Steps

1. From a browser, enter the URL to access the Swagger web page in the format *https://<SnapCenter_IP_address_or_name>:<SnapCenter_port>/swagger/*.

   Ensure that the REST API URL does not have the following characters: +, ., %, and &.

2. In the **Swagger Explore** field, if the Swagger API documentation does not display automatically, type: *https://<SnapCenter_IP_address_or_name>:<SnapCenter_port>/Content/swagger/SnapCenter.yaml*
3. Click **Explore**.

   A list of API resource types or categories are displayed.

4. Click an API resource type to display the APIs in that resource type.

If you encounter unexpected behavior when executing SnapCenter REST APIs, you can use the log files to identify the cause and resolve the problem. You can download the log files from the SnapCenter user interface by clicking **Monitor** > **Logs** > **Download**.

Get started with the REST API

You can quickly get started using the SnapCenter REST API. Accessing the API provides some perspective before you begin using it with the more complex workflow processes on a live setup.

Hello World

You can run a simple command on your system to get started using the SnapCenter REST API and confirm its availability.

What you will need

- Ensure that the Curl utility is available on your system.
- IP address or host name of the SnapCenter Server
- User name and password for an account with authority to access the SnapCenter REST API.

> If your credentials include special characters, you need to format them in a way that is acceptable to Curl based on the shell you are using. For example, you can insert a backslash before each special character or wrap the entire `username:password` string in single quotes.

Step

At the command line interface, run the following to retrieve the plug-in information:

```
curl -X GET -u username:password -k
"https://<ip_address>/api/hosts?fields=IncludePluginInfo"
```

Example:

```
curl -X GET -u admin:password -k
"'https://10.225.87.97/api/hosts?fields=IncludePluginInfo"
```

Copyright information

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at http://www.netapp.com/TM are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.