Workflows

ONTAP Automation

NetApp
April 21, 2024

Table of Contents

Workflows

Prepare to use the workflows

You should be familiar with the structure and format of the workflows before using them with a live ONTAP deployment.

> You should make sure that your ONTAP release supports all the API calls in the workflows you plan to use. See API reference for more information.

Introduction

A *workflow* is a sequence of one or more steps needed to accomplish a specific administrative task or goal. The ONTAP workflows include the core steps and parameters you need to accomplish each task. They provide a starting point for customizing your ONTAP automation environment.

Step types

Each step in an ONTAP workflow is one of the following types:

- REST API call (with details such as curl and JSON examples)
- Perform or invoke another ONTAP workflow
- Miscellaneous related task (such as making a configuration decision)

REST API calls

Most of the workflow steps are REST API calls. These steps use a common format which includes a curl example and other information. See the API reference for more details about the REST API calls.

Single-step workflows

A workflow can contain only one step. These *single-step workflows* are formatted slightly differently than workflows containing multiple steps. For example, the explicit step name is removed. The action or operation should be clear based on the workflow title.

Input variables

The workflows are designed to be as general as possible so they can be used in any ONTAP environment. With this in mind, the REST API calls use variables in the curl examples and other input. The REST API calls can then be easily adapted to different ONTAP environments.

Base URL format

You can access the ONTAP REST API directly through curl or a programming language. In this case, the base URL is different than the URL you use when accessing the ONTAP online documentation or System Manager.

When accessing the API directly, you need to append **api** to the domain or IP address. For example:

`https://ontap.demo-example.com/api`

See How to access the ONTAP REST API for more information.

Common input parameters

There are several input parameters commonly used with most of the REST API calls. These parameters are typically not described in the individual workflows. You should be familiar with the parameters. See Input variables controlling an API request for more information.

If additional parameters are needed for a specific REST API call, they are included in the section **Additional input parameters for the curl example** for each workflow.

Variable format

The ID values and other variables used with the workflow examples are opaque and can vary with each ONTAP cluster. To improve the readability of the examples, actual values are not used. Variables are used instead. This approach, based on a consistent format and set of reserved names, has several benefits including:

- The curl and JSON samples are more readable and easier to understand.
- Because all the keywords use the same format, you can quickly identify them.
- There is no security exposure because the values cannot be copied and reused.

The variables are formatted to be used in a Bash shell environment. Each variable begins with a dollar sign and is enclosed in double quotes as needed. This makes them recognizable to Bash. Upper case is consistently used for the names.

Here are some of the common variable keywords. This list is not exhaustive and additional variables are used as needed. Their meaning should be obvious based on the context.

Keyword	Type	Description
$FQDN_IP	URL	The fully qualified domain name or IP address of the ONTAP management LIF.
$CLUSTER_ID	Path	The UUIDv4 value identifying the ONTAP cluster where the API operations run.
$BASIC_AUTH	Header	The credentials string used for HTTP basic authentication.

JSON input examples

Some of the REST API calls, such as those using POST or PATCH, require JSON input in the body of the request. The JSON input examples are presented separately from the curl examples for clarity. You can use the JSON input examples with one of the techniques described below.

Save to local file

You can copy the JSON input example to a file and save it locally. The curl command refers to the file using the `--data` parameter with the value indicating the file name with a `@` prefix.

Paste into terminal after the curl example

First you need to copy and paste the curl example into a terminal shell. Then edit the example to completely remove the `--data` parameter at the end and replace it with the `--data-raw` parameter. Finally, copy and paste in the JSON example so that it follows the curl command with the updated parameter. You should use single quotes to wrap the JSON input example.

Authentication options

The primary authentication technique available for the REST API is HTTP basic authentication. Beginning with ONTAP 9.14, you also have the option of using the Open Authorization (OAuth 2.0) framework with token-based authentication and authorization.

HTTP basic authentication

When using basic authentication, the user credentials must be included with each HTTP request. There are two options for sending the credentials.

Construct the HTTP request header

You can manually construct the Authorization header and include it with the HTTP requests. This can be done when using a curl command in the CLI or a programming language with your automation code. The high-level steps include:

1. Concatenate the user and password values with a colon:

   ```
   admin:david123
   ```

2. Convert the entire string to base64:

   ```
   YWRtaW46ZGF2aWQxMjM=
   ```

3. Construct the request header:

   ```
   Authorization: Basic YWRtaW46ZGF2aWQxMjM=
   ```

The workflow curl examples include this header with the variable **$BASIC_AUTH** which you need to update before using.

Use a curl parameter

Another option when using curl is to remove the Authorization header and use the curl **user** parameter instead. For example:

```
--user username:password
```

You need to substitute the appropriate credentials for your environment. The credentials are not encoded in base64. When executing the curl command with this parameter, the string is encoded and the Authorization header is generated for you.

OAuth 2.0

When using OAuth 2.0, you need to request an access token from an external authorization server and include it with each HTTP request. The basic high-level steps are described below. Also see Overview of the ONTAP OAuth 2.0 implementation for more details about OAuth 2.0 and how to use it with ONTAP.

Prepare your ONTAP environment

Before using the REST API to access ONTAP, you need to prepare and configure the ONTAP environment. At a high level, the steps include:

- Identify the ONTAP protected resources and clients
- Review the existing ONTAP REST role and user definitions

- Install and configure the authorization server
- Design and configure the client authorization definitions
- Configure ONTAP and enable OAuth 2.0

Request an access token

With ONTAP and the authorization server defined and active, you can make a REST API call using an OAuth 2.0 token. The first step is to request an access token from the authorization server. This is done outside of ONTAP using one of several different techniques based on the server. ONTAP does not issue access tokens or perform redirection.

Construct the HTTP request header

After obtaining an access token, you can construct an Authorization header and include it with the HTTP requests. Regardless of whether you use curl or a programming language to access the REST API, you must include the header with every client request. You can construct the header as follows:

```
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSld …
```

Using the examples with Bash

If you use the workflow curl examples directly, you must update the variables they contain with values appropriate for your environment. You can manually edit the examples or rely on the Bash shell to do the substitution for you as described below.

> ⓘ One advantage of using Bash is that you can set the variable values one time in a shell session instead of once per curl command.

Steps

1. Open the Bash shell provided with Linux or similar operating system.
2. Set the variable values included in the curl example you want to run. For example:

   ```
   CLUSTER_ID=ce559b75-4145-11ee-b51a-005056aee9fb
   ```

3. Copy the curl example from the workflow page and paste it into the shell terminal.
4. Press **ENTER** which will do the following:
   a. Substitute the variable values you set
   b. Execute the curl command

Cluster

Get cluster configuration

You can retrieve the configuration for an ONTAP cluster including specific fields. You might do this as part of assessing the state of the cluster or before updating the configuration.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/cluster

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
fields	Query	No	Select the values you want returned. Examples include `contact` and `version`.

Curl example: Retrieve the cluster contact information

This example illustrates how to retrieve a single field. To get the entire cluster object and configuration, you need to remove the `fields` query parameter.

```
curl --request GET \
--location "https://$FQDN_IP/api/cluster?fields=contact" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "contact": "support@company-demo.com"
}
```

Update cluster contact

You can update the contact information for a cluster. Because the request is processed asynchronously, you also need to determine if the associated background job completed successfully.

Step 1: Update the cluster contact information

You can issue an API call to update the cluster contact information.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
PATCH	/api/cluster

Processing type

Asynchronous

Curl example

```
curl --request PATCH \
--location "https://$FQDN_IP/api/cluster" \
--include \
--header "Content-Type: application/json" \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "contact": "support@company-demo.com"
}
```

JSON output example

A job object is returned. You should save the job identifier to use it in the next step.

```
{ "job": {
    "uuid": "d877f5bb-3aa7-11e9-b6c6-005056a78c89",
    "_links": {
      "self": {
        "href": "/api/cluster/jobs/d877f5bb-3aa7-11e9-b6c6-005056a78c89"
      }
    }
  }
}
```

Step 2: Retrieve the status of the job

Perform the workflow Get job instance and confirm the `state` value is `success`.

Step 3: Confirm the cluster contact information

Perform the workflow Get cluster configuration. You should set the `fields` query parameter to `contact`.

Get job instance

You can retrieve the instance of a specific ONTAP job. You would typically do this to determine if the job and associated operation completed successfully.

> You need the UUID of the job object, which is typically provided after issuing an asynchronous request. Also review Asynchronous processing using the Job object before working with ONTAP internal jobs.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/cluster/jobs/{uuid}

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
$JOB_ID	Path	Yes	Needed to identify the job being requested.

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/cluster/jobs/$JOB_ID" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

The state value and other fields are included in the returned job object. The job in this example was run as part of updating an ONTAP cluster.

```
{
  "uuid": "d877f5bb-3aa7-11e9-b6c6-005056a78c89",
  "description": "PATCH /api/cluster",
  "state": "success",
  "message": "success",
  "code": 0,
  "_links": {
    "self": {
      "href": "/api/cluster/jobs/d877f5bb-3aa7-11e9-b6c6-005056a78c89"
    }
  }
}
```

NAS

File security permissions

Prepare to manage file security and audit policies

You can manage the permissions and audit policies for files available through the SVMs within an ONTAP cluster.

Overview

ONTAP uses System Access Control Lists (SACLs) and Discretionary Access Control Lists (DACLs) to assign permissions to file objects. Beginning with ONTAP 9.9.1, the REST API includes support for managing the SACL and DACL permissions. You can use the API to automate the administration of the file security permissions. In many cases you can use a single REST API call instead of multiple CLI commands or ONTAPI (ZAPI) calls.

> For ONTAP releases prior to 9.9.1, you can automate the administration of the SACL and DACL permissions using the CLI passthrough feature. See Migration considerations and Using the private CLI passthrough with the ONTAP REST API for more information.

Several example workflows are available to illustrate how to manage the ONTAP file security services using the REST API. Before using the workflows and issuing any of the REST API calls, make sure to review Prepare to use the workflows.

If you use Python, also see the script file_security_permissions.py for examples of how to automate some of the file security activities.

ONTAP REST API versus ONTAP CLI commands

For many tasks, using the ONTAP REST API requires fewer calls than the equivalent ONTAP CLI commands or ONTAPI (ZAPI) calls. The table below includes a list of API calls and the equivalent the CLI commands needed for each task.

ONTAP REST API	ONTAP CLI
`GET /protocols/file-security/effective-permissions/`	`vserver security file-directory show-effective-permissions`
`POST /protocols/file-security/permissions/`	1. `vserver security file-directory ntfs create` 2. `vserver security file-directory ntfs dacl add` 3. `vserver security file-directory ntfs sacl add` 4. `vserver security file-directory policy create` 5. `vserver security file-directory policy task add` 6. `vserver security file-directory apply`
`PATCH /protocols/file-security/permissions/`	`vserver security file-directory ntfs modify`

ONTAP REST API	ONTAP CLI
`DELETE /protocols/file-security/permissions/`	1. `vserver security file-directory ntfs dacl remove` 2. `vserver security file-directory ntfs sacl remove`

Related information

- Python script illustrating file permissions
- Simplified management of file-security permissions with ONTAP REST APIs
- Using the private CLI passthrough with the ONTAP REST API

Get the effective permissions for a file

You can retrieve the current effective permissions for a specific file or folder.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/protocols/file-security/effective-permissions/{svm.uuid}/{path}

Processing type

Synchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/protocols/file-security/effective-
permissions/$SVM_ID/$FILE_PATH" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "svm": {
    "uuid": "cf5f271a-1beb-11ea-8fad-005056bb645e",
    "name": "vs1"
  },
  "user": "administrator",
  "type": "windows",
  "path": "/",
  "share": {
    "path": "/"
  },
  "file_permission": [
    "read",
    "write",
    "append",
    "read_ea",
    "write_ea",
    "execute",
    "delete_child",
    "read_attributes",
    "write_attributes",
    "delete",
    "read_control",
    "write_dac",
    "write_owner",
    "synchronize",
    "system_security"
  ],
  "share_permission": [
    "read",
    "read_ea",
    "execute",
    "read_attributes",
    "read_control",
    "synchronize"
  ]
}
```

Get the auditing information for a file

You can retrieve the auditing information for a specific file or folder.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/protocols/file-security/permissions/{svm.uuid}/{path}

Processing type

Synchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/protocols/file-
security/permissions/$SVM_ID/$FILE_PATH" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "svm": {
    "uuid": "9479099d-5b9f-11eb-9c4e-0050568e8682",
    "name": "vs1"
  },
  "path": "/parent",
  "owner": "BUILTIN\\Administrators",
  "group": "BUILTIN\\Administrators",
  "control_flags": "0x8014",
  "acls": [
    {
      "user": "BUILTIN\\Administrators",
      "access": "access_allow",
      "apply_to": {
        "files": true,
        "sub_folders": true,
        "this_folder": true
      },
      "advanced_rights": {
        "append_data": true,
        "delete": true,
        "delete_child": true,
        "execute_file": true,
        "full_control": true,
        "read_attr": true,
        "read_data": true,
        "read_ea": true,
        "read_perm": true,
        "write_attr": true,
        "write_data": true,
        "write_ea": true,
        "write_owner": true,
        "synchronize": true,
        "write_perm": true
      },
      "access_control": "file_directory"
    },
    {
      "user": "BUILTIN\\Users",
      "access": "access_allow",
      "apply_to": {
        "files": true,
        "sub_folders": true,
        "this_folder": true
      },
      "advanced_rights": {
        "append_data": true,
        "delete": true,
        "delete_child": true,
        "execute_file": true,
        "full_control": true,
        "read_attr": true,
        "read_data": true,
        "read_ea": true,
        "read_perm": true,
        "write_attr": true,
        "write_data": true,
        "write_ea": true,
        "write_owner": true,
        "synchronize": true,
        "write_perm": true
      },
      "access_control": "file_directory"
    }
  ],
  "inode": 64,
  "security_style": "mixed",
  "effective_style": "ntfs",
  "dos_attributes": "10",
  "text_dos_attr": "----D---",
  "user_id": "0",
  "group_id": "0",
  "mode_bits": 777,
  "text_mode_bits": "rwxrwxrwx"
}
```

Apply new permissions to a file

You can apply a new security descriptor to a specific file or folder.

Step 1: Apply the new permissions

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/protocols/file-security/permissions/{svm.uuid}/{path}

Processing type

Asynchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request POST --location "https://$FQDN_IP/api/protocols/file-
security/permissions/$SVM_ID/$FILE_PATH?return_timeout=0" --include
--header "Accept */*" --header "Authorization: Basic $BASIC_AUTH" --data
'{ \"acls\": [ { \"access\": \"access_allow\", \"advanced_rights\": {
\"append_data\": true, \"delete\": true, \"delete_child\": true,
\"execute_file\": true, \"full_control\": true, \"read_attr\": true,
\"read_data\": true, \"read_ea\": true, \"read_perm\": true,
\"write_attr\": true, \"write_data\": true, \"write_ea\": true,
\"write_owner\": true, \"write_perm\": true }, \"apply_to\": { \"files\":
true, \"sub_folders\": true, \"this_folder\": true }, \"user\":
\"administrator\" } ], \"control_flags\": \"32788\", \"group\": \"S-1-5-
21-2233347455-2266964949-1780268902-69700\", \"ignore_paths\": [
\"/parent/child2\" ], \"owner\": \"S-1-5-21-2233347455-2266964949-
1780268902-69304\", \"propagation_mode\": \"propagate\"}'
```

JSON output example

```
{
  "job": {
    "uuid": "3015c294-5bbc-11eb-9c4e-0050568e8682",
    "_links": {
      "self": {
        "href": "/api/cluster/jobs/3015c294-5bbc-11eb-9c4e-0050568e8682"
      }
    }
  }
}
```

Step 2: Retrieve the status of the job

Perform the workflow Get job instance and confirm the `state` value is `success`.

Update security descriptor information

You can update a specific security descriptor to a specific file or folder, including the primary owner, group, or control flags.

Step 1: Update the security descriptor

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
PATCH	/api/protocols/file-security/permissions/{svm.uuid}/{path}

Processing type

Asynchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request POST --location "https://$FQDN_IP/api/protocols/file-
security/permissions/$SVM_ID/$FILE_PATH?return_timeout=0" --include
--header "Accept */*" --header "Authorization: Basic $BASIC_AUTH" --data
'{ \"control_flags\": \"32788\", \"group\": \"everyone\", \"owner\":
\"user1\"}'
```

JSON output example

```
{
  "job": {
    "uuid": "6f89e612-5bbd-11eb-9c4e-0050568e8682",
    "_links": {
      "self": {
        "href": "/api/cluster/jobs/6f89e612-5bbd-11eb-9c4e-0050568e8682"
      }
    }
  }
}
```

Step 2: Retrieve the status of the job

Perform the workflow Get job instance and confirm the `state` value is `success`.

Delete an access control entry

You can delete an existing Access Control Entry (ACE) from a specific file or folder. The change propagates to any child objects.

Step 1: Delete the ACE

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
DELETE	/api/protocols/file-security/permissions/{svm.uuid}/{path}

Processing type

Asynchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request DELETE --location "https://$FQDN_IP/api/protocols/file-
security/permissions/$SVM_ID/$FILE_PATH?return_timeout=0" --include
--header "Accept */*" --header "Authorization: Basic $BASIC_AUTH" --data
'{ \"access\": \"access_allow\", \"apply_to\": { \"files\": true,
\"sub_folders\": true, \"this_folder\": true }, \"ignore_paths\": [
\"/parent/child2\" ], \"propagation_mode\": \"propagate\"}'
```

JSON output example

```
{
  "job": {
    "uuid": "3015c294-5bbc-11eb-9c4e-0050568e8682",
    "_links": {
      "self": {
        "href": "/api/cluster/jobs/3015c294-5bbc-11eb-9c4e-0050568e8682"
      }
    }
  }
}
```

Step 2: Retrieve the status of the job

Perform the workflow Get job instance and confirm the `state` value is `success`.

Networking

List the IP interfaces

You can retrieve the IP LIFs assigned to the cluster and SVMs. You might do this to confirm your network configuration or when planning to add another LIF.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/network/ip/interfaces

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
fields	Query	No	Return a limited list of the relevant configuration values.

Curl example: return all LIFs with the default configuration values

```
curl --request GET \
--location "https://$FQDN_IP/api/network/ip/interfaces" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Curl example: Return all LIFs with four specific configuration values

```
curl --request GET \
--location
```

```
"https://$FQDN_IP/api/network/ip/interfaces?fields=name,scope,svm.name,ip.address" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "records": [
    {
      "uuid": "5ded9e38-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_mgmt1",
      "ip": {
        "address": "172.29.151.116"
      },
      "scope": "cluster",
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/5ded9e38-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "bb03c162-999e-11ee-acad-005056ae6bd8",
      "name": "cluster_mgmt",
      "ip": {
        "address": "172.29.186.156"
      },
      "scope": "cluster",
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/bb03c162-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c5ffbd03-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data1",
      "ip": {
        "address": "172.29.186.150"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c5ffbd03-999e-11ee-acad-
005056ae6bd8"
```

```
        }
      }
    },
    {
      "uuid": "c6612abe-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data2",
      "ip": {
        "address": "172.29.186.151"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c6612abe-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c6b21b94-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data3",
      "ip": {
        "address": "172.29.186.152"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c6b21b94-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c7025322-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data4",
      "ip": {
        "address": "172.29.186.153"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c7025322-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c752cc66-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data5",
      "ip": {
        "address": "172.29.186.154"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c752cc66-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c7a03719-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data6",
      "ip": {
        "address": "172.29.186.155"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c7a03719-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "ccd4c59c-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data4_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac5"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/ccd4c59c-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "d9144c30-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data6_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac7"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/d9144c30-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "d961c13b-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data1_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac2"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/d961c13b-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "d9ac8d6a-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data5_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac6"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/d9ac8d6a-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "d9fce1a3-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data2_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac3"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/d9fce1a3-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "da4995a0-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data3_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac4"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/da4995a0-999e-11ee-acad-005056ae6bd8"
        }
      }
    },
    {
      "uuid": "da9e7afd-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_cluster_mgmt_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac8"
      },
      "scope": "cluster",
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/da9e7afd-999e-11ee-acad-005056ae6bd8"
        }
      }
    },
    {
      "uuid": "e6db58b4-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_mgmt1_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:3008::1a0"
      },
      "scope": "cluster",
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/e6db58b4-999e-11ee-acad-005056ae6bd8"
        }
      }
    }
  ],
  "num_records": 16,
  "_links": {
    "self": {
      "href": "/api/network/ip/interfaces?fields=name,scope,svm.name,ip.address"
    }
  }
}
```

Security

Accounts

List the accounts

You can retrieve a list of the accounts. You might do this to assess your security environment or before creating a new account.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/security/accounts

Processing type

Synchronous

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/security/accounts" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "records": [
    {
      "owner": {
        "uuid": "642573a8-9d14-11ee-9330-005056aed3de",
        "name": "vs0",
        "_links": {
          "self": {
            "href": "/api/svm/svms/642573a8-9d14-11ee-9330-
005056aed3de"
          }
        }
      },
      "name": "vsadmin",
      "_links": {
        "self": {
          "href": "/api/security/accounts/642573a8-9d14-11ee-9330-
005056aed3de/vsadmin"
        }
      }
    },
    {
      "owner": {
        "uuid": "fdb6fe29-9d13-11ee-9330-005056aed3de",
        "name": "sti214nscluster-1"
      },
      "name": "admin",
      "_links": {
        "self": {
          "href": "/api/security/accounts/fdb6fe29-9d13-11ee-9330-
005056aed3de/admin"
        }
      }
    },
    {
      "owner": {
        "uuid": "fdb6fe29-9d13-11ee-9330-005056aed3de",
        "name": "sti214nscluster-1"
      },
      "name": "autosupport",
      "_links": {
        "self": {
          "href": "/api/security/accounts/fdb6fe29-9d13-11ee-9330-
005056aed3de/autosupport"
```

```
        }
      }
    }
  ],
  "num_records": 3,
  "_links": {
    "self": {
      "href": "/api/security/accounts"
    }
  }
}
```

Certificates and keys

List the installed certificates

You can list the certificates installed in your ONTAP cluster. You might do this to see if a particular certificate is available or to get the ID of a specific certificate.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/security/certificates

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
max_records	Query	No	Specify the number of records you want returned.

Curl example: Return three certificates

```
curl --request GET \
--location "https://$FQDN_IP/api/security/certificates?max_records=3" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "records": [
    {
      "uuid": "dad822c2-573c-11ee-a310-005056aecc29",
      "name": "vs0_17866DB5C933E2EA",
      "_links": {
        "self": {
          "href": "/api/security/certificates/dad822c2-573c-11ee-a310-
005056aecc29"
        }
      }
    },
    {
      "uuid": "7d8e5570-573c-11ee-a310-005056aecc29",
      "name": "BuypassClass3RootCA",
      "_links": {
        "self": {
          "href": "/api/security/certificates/7d8e5570-573c-11ee-a310-
005056aecc29"
        }
      }
    },
    {
      "uuid": "7dbb2191-573c-11ee-a310-005056aecc29",
      "name": "EntrustRootCertificationAuthority",
      "_links": {
        "self": {
          "href": "/api/security/certificates/7dbb2191-573c-11ee-a310-
005056aecc29"
        }
      }
    }
  ],
  "num_records": 3,
  "_links": {
    "self": {
      "href": "/api/security/certificates?max_records=3"
    },
    "next": {
      "href": "/api/security/certificates?start.svm_id=sti214nscluster-
1&start.uuid=7dbb2191-573c-11ee-a310-005056aecc29&max_records=3"
    }
  }
}
```

Install a certificate

You can install a signed X.509 certificate in your ONTAP cluster. You might do this as part of configuring an ONTAP feature or protocol that requires strong authentication.

Before you begin

You must have the certificate you want to install. You should also make sure any intermediate certificates are installed as needed.

> Before using the JSON input examples included below, make sure to update the `public_certificate` value with the certificate for your environment.

Step 1: Install the certificate

You can issue an API call to install the certificate.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/certificates

Curl example: Install a root CA certificate at the cluster level

```
curl --request POST \
--location "https://$FQDN_IP/api/security/certificates" \
--include \
--header "Content-Type: application/json" \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
    "type": "server_ca",
    "public_certificate":
    "-----BEGIN CERTIFICATE-----
MIID0TCCArkCFGYdznvTVvaY1VZPNfy4yCCyPph6MA0GCSqGSIb3DQEBCwUAMIGk
MQswCQYDVQQGEwJVUzELMAkGA1UECAwCTkMxDDAKBgNVBAcMA1JUUDEWMBQGA1UE
CgwNT05UQVAgRXhhbXBsZTETMBEGA1UECwwKT05UQVAgOS4xNDEcMBoGA1UEAwwT
Ki5vbnRhcC1leGFtcGxlLmNvbTEvMC0GCSqGSIb3DQEJARYgZGF2aWQucGV0ZXJz
b25Ab250YXAtZXhhbXBsZS5jb20wHhcNMjMxMDA1MTUyOTE4WhcNMjQxMDA0MTUy
OTE4WjCBpDELMAkGA1UEBhMCVVMxCzAJBgNVBAgMAk5DMQwwCgYDVQQHDANSVFAx
FjAUBgNVBAoMDU9OVEFQIEV4YW1wbGUxEzARBgNVBAsMCk9OVEFQIDkuMTQxHDAa
BgNVBAMMEyoub250YXAtZXhhbXBsZS5jb20xLzAtBgkqhkiG9w0BCQEWIGRhdmlk
LnBldGVyc29uQG9udGFwLWV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEAxQgy8mhb1Jhkf0D/MBodpzgW0aSp2jGbWJ+Zv2G8BXkp1762
dPHRkv1hnx9JvwkK4Dba05GiCiD5t3gjH/jUQMSFb+VwDbVmubVFnXjkm/4Q7sea
tMtA/ZpQdZbQFZ5RKtdWz7dzzPYEl2x8Q1Jc8Kh7NxERNMtgupGWZZn7mfXKYr4O
N/+vgahIhDibS8YK5rf1w6bfmrik9E2D+PEab9DX/1DL5RX4tZ1H2OkyN2UxoBR6
Fq7l6n1Hi/5yR0Oi1xStN6sO7EPoGak+KSlK41q+EcIKRo0bP4mEQp8WMjJuiTkb
5MmeYoIpWEUgJK7S0M6Tp/3bTh2CST3AWxiNxQIDAQABMA0GCSqGSIb3DQEBCwUA
A4IBAQABfBqOuROmYxdfrj93OyIiRoDcoMzvo8cHGNUsuhnlBDnL2O3qhWEs97s0
mIy6zFMGnyNYa0t4i1cFsGDKP/JuljmYHjvv+2lHWnxHjTo7AOQCnXmQH5swoDbf
o1Vjqz8Oxz+PRJ+PA3dF5/8zqaAR6QreAN/iFR++6nUq1sbbM7w03tthBVMgo/h1
E9I2jVOZsqMFujm2CYfMs4XkZtrYmN6nZA8JcUpDjIWcAVbQYurMnna9r42oS3GB
WB/FE9n+P+FfJyHJ93KGcCXbH5RF2pi3wLlHilbvVuCjLRrhJ8U20I5mZoiXvABc
IpYuBcuKXLwAarhDEacXttVjC+Bq
-----END CERTIFICATE-----"
}
```

Step 2: Confirm the certificate has been installed

Perform the workflow List the installed certificates and confirm the certificate is available.

RBAC

Prepare to use RBAC

You can use the ONTAP RBAC capability in several different ways depending on your environment. A few common scenarios are presented as workflows in this section. In each case the focus is on a specific security and administrative goal.

Before creating any roles and assigning a role to an ONTAP user account, you should prepare by reviewing the major security requirements and options presented below. Also make sure to review the general workflow concepts at Prepare to use the workflows.

What ONTAP release are you using?

The ONTAP release determines what REST endpoints and RBAC features are available.

Identify the protected resources and scope

You need to identify the resources or commands to be protected and the scope (cluster or SVM).

What access should the user have?

After identifying the resources and scope, you need to determine the access level to be granted.

How will the users access ONTAP?

The user can access ONTAP through the REST API or CLI or both.

Is one of the built-in roles sufficient or is a custom role needed?

It is more convenient to use an existing built-in role but you can create a new custom role if needed.

What type of role is needed?

Based on the security requirements and the ONTAP access, you need to choose whether to create a REST or traditional role.

Create roles

Limit access to SVM volume operations

You can define a role to restrict storage volume administration within an SVM.

About this workflow

A traditional role is first created to initially allow access to all the major volume administration functions except cloning. The role is defined with the following characteristics:

- Able to perform all CRUD volume operations including get, create, modify, and delete
- Cannot create a volume clone

You can then optionally update the role as needed. In this workflow, the role is changed in the second step to allow the user to create a volume clone.

Step 1: Create the role

You can issue an API call to create the RBAC role.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/roles" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "role1",
  "owner": {
    "name": "cluster-1",
    "uuid": "852d96be-f17c-11ec-9d19-005056bbad91"
  },
  "privileges": [
      { "path": "volume create", "access": "all" },
      { "path": "volume delete", "access": "all" }
    ]
}
```

Step 2: Update the role

You can issue an API call to update the existing role.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM that contains the role definition.
$ROLE_NAME	Path	Yes	This is the name of the role within the SVM to be updated.

Curl example

```
curl --request POST \
--location
"https://$FQDN_IP/api/security/roles/$SVM_ID/$ROLE_NAME/priveleges" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "path": "volume clone",
  "access": "all"
}
```

Enable adminstration of data protection

You can provide a user with limited data protection capabilities.

About this workflow

The traditional role created is defined with the following characteristics:

- Able to create and delete snapshots as well as update SnapMirror relationships
- Cannot create or modify higher level objects such as volumes or SVMs

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/roles" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "role1",
  "owner": {
    "name": "cluster-1",
    "uuid": "852d96be-f17c-11ec-9d19-005056bbad91"
  },
  "privileges": [
      {"path": "volume snapshot create", "access": "all"},
      {"path": "volume snapshot delete", "access": "all"},
      {"path": "volume show", "access": "readonly"},
      {"path": "vserver show", "access": "readonly"},
      {"path": "snapmirror show", "access": "readonly"},
      {"path": "snapmirror update", "access": "all"}
  ]
}
```

Allow generation of ONTAP reports

You can create a REST role to provide users with the ability to generate ONTAP reports.

About this workflow

The role created is defined with the following characteristics:

- Able to retrieve all storage object information related to capacity and performance (such as volume, qtree, LUN, aggregates, node, and SnapMirror relationships)
- Cannot create or modify higher level objects (such as volumes or SVMs)

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/roles" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "rest_role1",
  "owner": {
    "name": "cluster-1",
    "uuid": "852d96be-f17c-11ec-9d19-005056bbad91"
  },
  "privileges": [
      {"path": "/api/storage/volumes", "access": "readonly"},
      {"path": "/api/storage/qtrees", "access": "readonly"},
      {"path": "/api/storage/luns", "access": "readonly"},
      {"path": "/api/storage/aggregates", "access": "readonly"},
      {"path": "/api/cluster/nodes", "access": "readonly"},
      {"path": "/api/snapmirror/relationships", "access": "readonly"},
      {"path": "/api/svm/svms", "access": "readonly"}
  ]
}
```

Create a user with a role

You can use this workflow to create a user with an associated REST role.

About this workflow

This workflow includes the typical steps needed to create a custom REST role and associate it with a new user account. Both the user and role have an SVM scope and are associated with a specific data SVM. Some of the steps may be optional or need to change depending on your environment.

Step 1: List the data SVMs in the cluster

Perform the following REST API call to list the SVMs in the cluster. The UUID and name of each SVM are provided in the output.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/svm/svms

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/svm/svms?order_by=name" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

After you finish

Select the desired SVM from the list where you will create the new user and role.

Step 2: List the users defined to the SVM

Perform the following REST API call to list the users defined in the SVM you selected. You can identify the SVM through the owner parameter.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/security/accounts

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/security/accounts?owner.name=dmp" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

After you finish

Based on the users already defined in the SVM, choose a unique name for the new user.

Step 3: List the REST roles defined to the SVM

Perform the following REST API call to list the roles defined in the SVM you selected. You can identify the SVM through the owner parameter.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/security/roles

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/security/roles?owner.name=dmp" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

After you finish

Based on the roles already defined in the SVM, choose a unique name for the new role.

Step 4: Create a custom REST role

Perform the following REST API call to a create a custom REST role in the SVM. The role initially has only one privilege which establishes a default access of **none** so that all access is denied.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/roles" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "dprole1",
  "owner": {
    "name": "dmp",
    "uuid": "752d96be-f17c-11ec-9d19-005056bbad91"
  },
  "privileges": [
      {"path": "/api", "access": "none"},
  ]
}
```

After you finish

Optionally perform step 3 again to display the new role. You can also display the roles at the ONTAP CLI.

Step 5: Update the role by adding more privileges

Perform the following REST API call to modify the role by adding privileges as needed.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles/{owner.uuid}/{name}/privileges

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	The UUID of the SVM that contains the role definition.
$ROLE_NAME	Path	Yes	The name of the role within the SVM to be updated.

Curl example

```
curl --request POST \
--location
"https://$FQDN_IP/api/security/roles/$SVM_ID/$ROLE_NAME/privileges" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "path": "/api/storage/volumes",
  "access": "readonly"
}
```

After you finish

Optionally perform step 3 again to display the new role. You can also display the roles at the ONTAP CLI.

Step 6: Create a user

Perform the following REST API call to a create a user account. The role **dprole1** created above is associated with the new user.

> You can create the user without a role. In this case, the user is assigned a default role (either `admin` or `vsadmin`) depending on whether the user is defined with cluster or SVM scope. You'll need to modify the user to assign a different role.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/accounts

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/accounts" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "owner": {"uuid":"daf84055-248f-11ed-a23d-005056ac4fe6"},
  "name": "david",
  "applications": [
      {"application":"ssh",
       "authentication_methods":["password"],
       "second_authentication_method":"none"}
  ],
  "role":"dprole1",
  "password":"netapp123"
}
```

After you finish

You can sign in to the SVM management interface using the credentials for the new user.

Storage

List the aggregates

You can retrieve a list of aggregates in the cluster. You might do this to assess utilization and performance.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/storage/disks

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
node.name	Query	No	Can be used to identify the node each aggregate is attached to.

Curl example: return all aggregates with the default configuration values

```
curl --request GET \
--location "https://$FQDN_IP/api/storage/aggregates" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Curl example: return all aggregates with a specific configuration value

```
curl --request GET \
--location "https://$FQDN_IP/api/storage/aggregates?fields=node.name" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "records": [
    {
      "uuid": "760d8137-fc59-47da-906a-cc28db0a1c1b",
      "name": "sti214_vsim_sr027o_aggr1",
      "node": {
        "name": "sti214-vsim-sr027o"
      },
      "_links": {
        "self": {
          "href": "/api/storage/aggregates/760d8137-fc59-47da-906a-
cc28db0a1c1b"
        }
      }
    }
  ],
  "num_records": 1,
  "_links": {
    "self": {
      "href": "/api/storage/aggregates?fields=node.name"
    }
  }
}
```

List the disks

You can retrieve a list of disks in the cluster. You might do this to locate one or more spares to use as part of creating an aggregate.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/storage/disks

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
state	Query	No	Can be used to identify the spare disks available for new aggregates.

Curl example: return all the disks

```
curl --request GET \
--location "https://$FQDN_IP/api/storage/disks" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Curl example: return spare disks

```
curl --request GET \
--location "https://$FQDN_IP/api/storage/disks?state=spare" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "records": [
    {
      "name": "NET-1.20",
      "state": "spare",
      "_links": {
        "self": {
          "href": "/api/storage/disks/NET-1.20"
        }
      }
    },
    {
      "name": "NET-1.12",
      "state": "spare",
      "_links": {
        "self": {
          "href": "/api/storage/disks/NET-1.12"
        }
      }
    },
    {
      "name": "NET-1.7",
      "state": "spare",
      "_links": {
        "self": {
          "href": "/api/storage/disks/NET-1.7"
        }
      }
    }
  ],
  "num_records": 3,
  "_links": {
    "self": {
      "href": "/api/storage/disks?state=spare"
    }
  }
}
```

Support

EMS

Prepare to manage EMS support services

You can configure Event Management System (EMS) processing for an ONTAP cluster as well as retrieve EMS messages as needed.

Overview

There are several example workflows available that illustrate how to use the ONTAP EMS services. Before using the workflows and issuing any of the REST API calls, make sure to review Prepare to use the workflows.

If you use Python, also see the scripy events.py for examples of how to automate some of the EMS-related activities.

ONTAP REST API versus ONTAP CLI commands

For many tasks, using the ONTAP REST API requires fewer calls than the equivalent ONTAP CLI commands. The table below includes a list of API calls and the equivalent the CLI commands needed for each task.

ONTAP REST API	ONTAP CLI
GET /support/ems	`event config show`
POST /support/ems/destinations	1. `event notification destination create` 2. `event notification create`
`GET /support/ems/events`	`event log show`
`POST /support/ems/filters`	1. `event filter create -filter-name <filtername>` 2. `event filter rule add -filter-name <filtername>`

Related information

- Python script illustrating EMS
- ONTAP REST APIs: Automate Notification of High-Severity Events

List the EMS log events

You can retrieve all event notification messages or only those with specific characteristics.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/support/ems/events

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
fields	Query	No	Used to request specific fields to be included in the response.
max_records	Query	No	Can be used to limit the number of records returned in a single request.
log_message	Query	No	Used to search for a specific text value and only return the matching messages.
message.severity	Query	No	Limit the returned messages to those with a specific severity such as `alert`.

Curl example: Return the latest message and the name value

```
curl --request GET \
--location
"https://$FQDN_IP/api/support/ems/events?fields=message.name&max_records=1
" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Curl example: Return a message containing specific text and severity

```
curl --request GET \
--location
"https://$FQDN_IP/api/support/ems/events?log_message=*disk*&message.severi
ty=alert" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "records": [
    {
      "node": {
        "name": "malha-vsim1",
        "uuid": "da4f9e62-9de3-11ec-976a-005056b369de",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/da4f9e62-9de3-11ec-976a-
005056b369de"
          }
        }
      },
      "index": 4602,
      "time": "2022-03-18T06:37:46-04:00",
      "message": {
        "severity": "alert",
        "name": "raid.autoPart.disabled"
      },
      "log_message": "raid.autoPart.disabled: Disk auto-partitioning is
disabled on this system: the system needs a minimum of 4 usable internal
hard disks.",
      "_links": {
        "self": {
          "href": "/api/support/ems/events/malha-vsim1/4602"
        }
      }
    }
  ],
  "num_records": 1,
  "_links": {
    "self": {
      "href":
"/api/support/ems/events?log_message=*disk*&message.severity=alert&max_rec
ords=1"
    },
    "next": {
      "href": "/api/support/ems/events?start.keytime=2022-03-
18T06%3A37%3A46-04%3A00&start.node.name=malha-
vsim1&start.index=4602&log_message=*disk*&message.severity=alert"
    }
  }
}
```

Get the EMS configuration

You can retrieve the current EMS configuration for an ONTAP cluster. You might do this before updating the configuration or creating a new EMS notification.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/support/ems

Processing type

Synchronous

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/support/ems" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "proxy_url": "https://proxyserver.mycompany.com",
  "proxy_user": "proxy_user",
  "mail_server": "mail@mycompany.com",
  "_links": {
    "self": {
      "href": "/api/resourcelink"
    }
  },
  "pubsub_enabled": "1",
  "mail_from": "administrator@mycompany.com"
}
```

Create an EMS notification

You can use the following workflow to create a new EMS notification destination to receive selected event messages.

Step 1: Configure the system-wide email settings

You can issue the following API call to configure the system-wide email settings.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
PATCH	/api/support/ems

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
mail_from	Query	Yes	Sets the `from` field in the notification email messages.
mail_server	Query	Yes	Configures the target SMTP mail server.

Curl example

```
curl --request PATCH \
--location
"https://$FQDN_IP/api/support/ems?mail_from=administrator@mycompany.com&ma
il_server=mail@mycompany.com" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Step 2: Define a message filter

You can issue an API call to define a filter rule to match the messages.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/support/ems/filters

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
Filter	Body	Yes	Includes the values for the filter configuration.

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/support/ems/filters" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "test-filter",
  "rules.type": ["include"],
  "rules.message_criteria.severities": ["emergency"]
}
```

Step 3: Create a message destination

You can issue an API call to create a message destination.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/support/ems/destinations

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
Destination configuration	Body	Yes	Includes the values for the event destination.

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/support/ems/destinations" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "test-destination",
  "type": "email",
  "destination": "administrator@mycompany.com",
  "filters.name": ["important-events"]
}
```

SVM

List the SVMs

You can list the Storage Virtual Machines (SVMs) defined within an ONTAP cluster. You might do this as part of finding the identifier for a specific SVM or to assure name uniqueness before creating a new SVM.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/svm/svms

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/svm/svms" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "records": [
    {
      "uuid": "71bd74f8-40dc-11ee-b51a-005056aee9fa",
      "name": "vs0",
      "_links": {
          "self": {
            "href": "/api/svm/svms/71bd74f8-40dc-11ee-b51a-005056aee9fa"
          }
      }
    }
  ],
  "num_records": 1,
  "_links": {
    "self": {
      "href": "/api/svm/svms"
    }
  }
}
```

Copyright information

Trademark information