Ansible Modules for Dell EMC PowerMax

Product Guide

1.2

September 2020

Notes, cautions, and warnings

NOTE: A NOTE indicates important information that helps you make better use of your product.

CAUTION: A CAUTION indicates either potential damage to hardware or loss of data and tells you how to avoid the problem.

WARNING: A WARNING indicates a potential for property damage, personal injury, or death.

1

Introduction

This chapter contains the following topics:

Topics:

Product overview

The Ansible Modules for Dell EMC PowerMax are used to automate and orchestrate the configuration, and deployment of the Dell EMC PowerMax arrays. The capabilities of Ansible modules are managing volumes, storage groups, ports, port groups, hosts, host groups, masking views, snapshots, SRDF pairings and state, gather high level facts about the arrays and RDF group configuration. The options available for each capability are list, show, create, delete, and modify.

The Ansible modules are called by tasks within the Ansible playbooks. The *Idempotency* feature is enabled for all the modules. The *Idempotency* feature enables the playbook to be run multiple times and hence supports fault tolerance. The modules use Unisphere RESTAPI to interface with the PowerMax arrays.

List of Ansible Modules for Dell EMC PowerMax

The following are the list of modules:

- Gather facts module
- Storage group module
- Volume module
- Host module
- Host group module
- Port module
- Port group module
- Masking view module
- Snapshot module
- SRDF module
- RDF Group module

2

Configure Ansible

This chapter contains the following topics:

Topics:

Software prerequisites

This table provides information about the software prerequisites for the Ansible Modules for Dell EMC PowerMax.

Prerequisites

Table 1. Software prerequisites

Ansible Modules	Unisphere Version	PowerMaxOS	Red Hat Enterprise Linux	Python version	Python library version	Ansible
v1.0	9.0	5978.221.221 5978.444.444	7.5	2.7.12	3.0.0.14	2.6 or later
v1.1	9.0 9.1	5978.221.221 5978.444.444	7.5	2.7.12 3.5.2	3.1.x	2.6 or later
v1.2	9.1 and above	5978.221.221 5978.444.444 5978.665.665	7.5 7.6, 7.7, 7.8, and 8.2	2.7 or later	9.1.0.0 and above	2.8 or later

NOTE: Unisphere Version 9.1 is compatible with PowerMax Python library version 9.1.x.x. Similarly, Unisphere version above 9.1 only works with Python library version greater than 9.1.x.x.

Set up the Ansible host

The Ansible server must be configured to write and run Ansible playbooks.

About this task

Do the following before you run playbooks on Ansible modules for Dell EMC PowerMax:

Steps

1. Install Python library for Unisphere.
   Run the following command to install PyU4V:

```
sudo apt install python-pip
pip install PyU4V
```

   For more information, see PyU4V Documentation.

2. Create the `dellemc` folder in one of the following folders if it is not available:
   - For Python 2.7 `/usr/lib/python2.7/site-packages/ansible/modules/storage`
   - For Python 3.5 `/usr/lib/python3.5/site-packages/ansible/modules/storage`
3. Create the dell folder in one of the following folders if it is not available:
   - For Python 2.7 `/usr/lib/python2.7/site-packages/ansible/module_utils/storage`
   - For Python 3.5 `/usr/lib/python3.5/site-packages/ansible/module_utils/storage`
4. Copy the Ansible modules to the appropriate locations in the virtual machine.
   a. Copy *dellemc_ansible_powermax_utils.py* and *_init_.py* from the `/utils` to one of the following locations:
      - For Python 2.7 `/usr/lib/python2.7/site-packages/ansible/module_utils/storage/dell`
      - For Python 3.5 `/usr/lib/python3.5/site-packages/ansible/module_utils/storage/dell`
   b. Copy all the module python files from the `/library` folder to one of the following:
      - For Python 2.7 `/usr/lib/python2.7/site-packages/ansible/modules/storage/dellemc`
      - For Python 3.5 `/usr/lib/python3.5/site-packages/ansible/modules/storage/dellemc/`
   c. Copy the *dellemc_powermax.py* from `/doc_fragments` to one of the following:
   - For Python 2.7 `/usr/lib/python2.7/site-packages/ansible/plugins/doc_fragments`
   - For Python 3.5 `/usr/lib/python3.5/site-packages/ansible/plugins/doc_fragments/`

   NOTE: The path may vary depending on the Python library version and the operating system.

3

Ansible modules for Dell EMC PowerMax

This chapter presents the following topics:

Topics:

Gather Facts module

The Gather Facts module displays a list of specific entities in PowerMax array. The Gather facts module is used with Ansible to register values that are used in conditional statements within the playbooks.

The Gather Facts module supports two sets of operations.

- Operations on the Unisphere server
- Operations on the PowerMax array registered under an Unisphere server

On the Unisphere host, the Gather Facts module lists the registered storage arrays that are managed by the Unisphere host.

On the PowerMax array, the Gather Facts module lists the following entities:

- Arrays in an Unisphere host
- Volumes in a specific array
- Storage groups in a specific array
- Hosts in a specific array
- Host groups in a specific array
- Storage resource pools in a specific array
- Ports in a specific array
- Port groups in a specific array
- Masking views in a specific array
- RDF groups in a specific array
- Health status of the array
- Support for generic filters for PowerMax array entities

Get list of volumes

The user can get the list of volumes and TDEV volumes in the storage array by running the appropriate playbook.

The syntax of the playbook is as follows:

```
- name: Get list of volumes
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
```

```
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    tdev_volumes: "True"
    gather_subset:
       - vol
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of arrays

The user can get the list of registered storage arrays that are managed by the Unisphere host by running the appropriate playbook.

The syntax for the playbook is shown as follows:

```
- name: Get array list
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of storage group

The user can get the list of storage groups in the array by running the appropriate playbook.

The syntax for the playbook is shown as follows:

```
- name: Get list of Storage groups
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    gather_subset:
       - sg
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of storage resource pool

The user can get the list of storage resource pools, and the details of each storage resource pool in the array by running the appropriate playbook.

The syntax for the playbook is shown as follows:

```
- name: Get list of Storage Resource Pools
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
```

```
    serial_no: "{{serial_no}}"
    gather_subset:
       - srp
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of port groups

The user can get the list of port groups in the array by running the appropriate playbook.

The syntax for the playbook is shown as follows:

```
- name: Get list of Port Groups
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    gather_subset:
       - pg
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of hosts

The user can get the list of hosts in the array by running the appropriate playbook.

The syntax for the playbook is shown as follows:

```
- name: Get list of Hosts
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    gather_subset:
       - host
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of host groups

The user can get the list of host groups in the array by running the appropriate playbook.

The syntax for the playbook is shown as follows:

```
- name: Get list of Host Groups
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
```

```
    gather_subset:
       - hg
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of ports

The user can get the list of ports in the array by running the appropriate playbook.

The syntax for the playbook is shown as follows:

```
- name: Get list of Ports
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    gather_subset:
       - port
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of masking views

The user can get the list of masking views available in the storage array by running the appropriate playbook.

The syntax for the playbook is shown as follows:

```
- name: Get list of Maskng Views
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    gather_subset:
       - mv
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of RDF groups

The user can get the list of RDF groups in the array by running the appropriate playbook.

The syntax for the playbook is shown as follows:

```
- name: Get list of all RDF Groups of given PowerMax/VMAX Storage System
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    serial_no: "{{serial_no}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    gather_subset:
       - rdf
  register: subset_result
```

```
- debug:
    var: subset_result
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get array health status

The user can get the health status of the storage arrays that are managed by the Unisphere host by running the appropriate playbook.

The syntax of the playbook is as follows:

```
- name: Get array health status
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}
    "universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    gather_subset:
       - health
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get list of volumes and storage groups with filter

The user can get the list of volumes and storage groups, with appropriate filter, in the arrays that are managed by the Unisphere host by running the appropriate playbook.

The syntax of the playbook is as follows:

```
- name: Get list of volumes and storage groups with filter
  dellemc_powermax_gatherfacts:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    gather_subset:
      - vol
      - sg
    filters:
      - filter_key: "tdev"
        filter_operator: "equal"
        filter_value: "True"
      - filter_key: "cap_gb"
        filter_operator: "equal"
        filter_value: "5"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Gather Facts module parameters

The following table provides the information about the parameters that are displayed on the console, when the user runs the *playbook* using the Gather facts module:

Table 2. Parameters

Parameter	Choices/default	Comments
`unispherehost`		IP or FQDN of the Unisphere host. This parameter is mandatory.
`universion`	• 91 • 92	The version of the Unisphere software. This parameter is optional.
`verifycert`	• True • False	To validate the SSL certificate. • True - Verify the SSL certificate. • False - Do not verify the SSL certificate. This parameter is mandatory.
`user`		The user name to access the Unisphere server. The user name can be encrypted using Ansible vault. This parameter is mandatory.
`password`		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
`serial_no`		The serial number of the PowerMax array. If this parameter is absent, the script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is optional.
`gather_subset`	• vol • srp • sg • pg • host • hg • port • mv • rdf • health	List of string variables to specify the PowerMax entities for which the information is required. Required only if `serial_no` is present. • vol - volumes • srp - storage resource pools • sg - storage groups • pg - port groups • host - hosts • hg - host groups • port - ports • mv - masking views • rdf - RDF groups • health - health status of a specific PowerMax array
`tdev_volumes`	• True • False	This parameter supports a boolean variable. By setting the value of this parameter to *True*, only TDEV Volumes will be listed. The default value of this parameter is *True*.
`filters`		
`filter_key`		Name identifier of the filter.

Table 2. Parameters (continued)

Parameter	Choices/default	Comments
		NOTE: Filters are applicable only when it is a valid `filter_key`, else the SDK will ignore it and return the usual result. If a valid `filter_key` is passed and the given condition does not match, then an empty list is generated in the output.
`filter_operator`	• equal • greater • lesser • like	Operation to be performed on filter key.
`filter_value`		Value of the filter key. **NOTE:** When the `filter_value` is given as *True* or *False*, then the `filter_operator` is ignored irrespective of its value, because the SDK treats the value as bool type. filters: - filter_key: tdev filter_operator: equal filter_value: True

Storage group module

The Storage group module manages the storage groups available on the PowerMax storage array.

The storage group module has the following functionalities:

- List the volumes of a storage group.
- Create a storage group.
- Delete an existing storage group.
- Add existing volumes to a storage group.
- Remove existing volumes from a storage group.
- Create volumes in a storage group.
- Modify the storage group attributes.
- Add child storage group to a parent storage group.
- Remove child storage group from a parent storage group.
- Add new volumes to SRDF protected storage group.
- Remove volumes from SRDF protected storage group.

Get details of storage group including the volumes

The user can get the details of a storage group in the array. The details include the list of volumes within the storage group.

The syntax of the playbook is shown as follows:

```
- name: Get storage group details including volumes
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
```

```
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "ansible_sg"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Create empty storage group

The user can create an empty storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create empty storage group
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "foo"
    service_level: "Diamond"
    srp: "SRP_1"
    compression: True
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete empty storage group

The user can delete an empty storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete the storage Group
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "foo"
    state: "absent"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Add existing volumes to the existing storage group

The user can add one or more existing volumes to a storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Adding existing volume(s) to existing SG
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
```

```
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "foo"
    state: "present"
    volumes:
      - vol_id: "00028"
      - vol_id: "00018"
      - vol_id: "00025"
    vol_state: "present-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Add new volumes to SRDF protected storage group

The user can add volumes to a SRDF protected storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Add volumes to SRDF protected storage group
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name}}"
    state: "present"
    volumes:
       - vol_name: "{{vol_name1}}"
         size: 1
         cap_unit: "GB"

       - vol_name: "{{vol_name2}}"
         size: 1
         cap_unit: "GB"
    vol_state: "present-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

NOTE: Add volumes to SRDF protected storage group is supported from PowerMax foxtail_version (v5978.444.444) onwards.

Create new volumes for existing storage group

The user can create new volumes for an existing storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create new volumes for existing SG
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "foo"
    state: "present"
    volumes:
      - vol_name: "foo"
        size: 1
```

```
        cap_unit: "GB"
      - vol_name: "bar"
        size: 1
        cap_unit: "GB"
    vol_state: "present-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Remove volumes from storage group

The user can remove one or multiple volumes from a storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Remove volume(s) from existing SG
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "foo"
    state: "present"
    volumes:
      - vol_id: "00028"
      - vol_id: "00018"
      - vol_name: "ansible-vol"
    vol_state: "absent-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Remove volumes from SRDF protected storage group

The user can remove multiple volumes from a SRDF protected storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Remove volumes from SRDF protected storage group
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name}}"
    state: "present"
    volumes:
       - vol_name: "{{vol_name1}}"
       - vol_name: "{{vol_name2}}"
    vol_state: "absent-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Add child storage group to parent storage group

The user can add a child storage group to a parent storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Adding child SG to parent SG
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "parent_sg"
    state: "present"
    child_storage_groups:
      - "pie"
      - "bar"
    child_sg_state: "present-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Remove child storage group from parent storage group

The user can remove a child storage group from a parent storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Removing child SG from parent SG
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "parent_sg":
    state: "present"
    child_storage_groups:
      - "pie"
      - "bar"
    child_sg_state: "absent-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Rename storage group

The user can rename a storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Rename Storage Group
  dellemc_powermax_storagegroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "ansible_sg":
```

```
    new_sg_name: "ansible_sg_renamed"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Storage Group module parameters

The following table lists the parameters that must be set before the user runs the playbook for the Storage Group module:

Table 3. Parameters

Parameter	Choices/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.
verifycert	• True • False	To validate the SSL certificate. • True - Verifies the SSL certificate. • False - Specifies that the SSL certificate should not be verified. This parameter is mandatory.
user		The user name to access the Unisphere server. The user name can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
serial_no		The serial number of the PowerMax array. If this parameter is absent, the script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is mandatory.
sg_name		The name of the storage group. This parameter is mandatory.
new_sg_name		The new name of the storage group.
service_level		The name of the service level
srp		The name of the storage resource pool. This parameter is ignored if the service_level is not specified. The default value is the name of the default storage resource pool of the array.
compression	• true • false The default value is true.	Defines if the compression is enable for a storage group. This parameter is unavailable if the service_level is not specified.
state	• absent • present	Defines whether the storage group should exist or not. This parameter is mandatory.

Table 3. Parameters (continued)

Parameter	Choices/default	Comments
volumes		This parameter lists the volumes present in the storage group. Each volume has four attributes. • vol_name • size • cap_unit • vol_id Use either volume name or volume ID to identify the volumes. Both cannot be used simultaneously to identify the volumes. For new volumes that are added to storage group, the name and size must be provided. This parameter is optional.
vol_state	• present-in-group • absent-in-group	Defines the state of the volumes inside the storage group.
child_storage_groups		Lists the child storage group.
child_sg_state	• present-in-group • absent-in-group	Describes the state of child storage group in a parent storage group.

Volume module

The Volume module manages the storage volumes on PowerMax arrays.

The volume module has the following capabilities:

- Create volume.
- Expand volume.
- Rename volume.
- Delete volume using the volume ID.
- Move volumes between storage groups.
- Get volume details.

The volume module has the following capabilities using WWN:

- Expand volume.
- Rename volume.
- Delete volume using the volume ID.
- Move volumes between storage groups.
- Get volume details.
- Add new volume to SRDF protected storage group.
- Expand volume which is part of SRDF protected storage group.

Create volume

The user can create a volume in the storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create volume
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
```

```
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    vol_name: "{{vol_name}}"
    sg_name: "{{sg_name}}"
    size: 1
    cap_unit: "{{cap_unit}}"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Expand volume

The user can expand the size of the volumes by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Expand volume size
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    size: 3
    cap_unit: "{{cap_unit}}"
    vol_id: "0059B"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Rename volume

The user can rename the volume by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Rename volume
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    new_name: "{{new_vol_name}}"
    vol_id: "0059B"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete volume

The user can delete the volume by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete volume
  dellemc_powermax_volume:
```

```
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    vol_id: "0059B"
    state: 'absent'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Move volumes between storage group

The user can move the volumes from one storage group to another by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Move volume between storage group
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    vol_name: "{{vol_name}}"
    sg_name: "{{sg_name}}"
    new_sg_name: "{{new_sg_name}}"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Add new volume to SRDF protected storage group

The user can add a volume to the SRDF protected storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Add new volume to SRDF protected storage group
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    vol_name: "{{vol_name}}"
    sg_name: "{{sg_name}}"
    size: 1
    cap_unit: "{{cap_unit}}"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

NOTE: Add volume to SRDF protected storage group is supported from PowerMax foxtail_version (v5978.444.444) onwards.

Expand volume of SRDF protected storage group using volume name

The user can expand volume of SRDF protected storage group using volume name by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Expand volume of SRDF protected storage group using volume name
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    vol_name: "{{vol_name}}"
    sg_name: "{{sg_name}}"
    size: 8
    cap_unit: "{{cap_unit}}"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

NOTE: Expand volume of SRDF protected storage group using volume name is supported from PowerMax foxtail_version (v5978.444.444) onwards.

Expand volume of SRDF protected storage group using volume identifier

The user can expand volume of SRDF protected storage group using identifier by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
  - name: Expand volume of SRDF protected storage group using volume identifier
    dellemc_powermax_volume:
      unispherehost: "{{unispherehost}}"
      universion: "{{universion}}"
      verifycert: "{{verifycert}}"
      user: "{{user}}"
      password: "{{password}}"
      serial_no: "{{serial_no}}"
      vol_id: "{{vol_id}}"
      size: 8
      cap_unit: "{{cap_unit}}"
      state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

NOTE: Expand volume of SRDF protected storage group using volume identifier is supported from PowerMax foxtail_version (v5978.444.444) onwards.

Get volume using WWN

The user can get details of a volume using WWN by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Get volume details
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
```

```
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    vol_wwn: "{{result.volume_details.wwn}}"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Expand volume using WWN

The user can expand the size of the volumes using WWN by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Expand volume using wwn
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    size: 21
    cap_unit: "{{cap_unit}}"
    vol_wwn: "{{result.volume_details.wwn}}"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Move volumes between storage groups using WWN

The user can move the volumes using WWN, from one storage group to another by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Move volume between Storage group
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    vol_wwn: "{{result.volume_details.wwn}}"
    sg_name: "{{sg_name}}"
    new_sg_name: "{{new_sg_name}}"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Rename volume using WWN

The user can rename the volume using WWN by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Rename volume
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
```

```
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    new_name: "Ansible_Testing_Renamed"
    vol_wwn: "{{result.volume_details.wwn}}"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete volume using WWN

The user can delete the volume using WWN by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete volume
  dellemc_powermax_volume:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    vol_wwn: "{{result.volume_details.wwn}}"
    state: "absent"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Volume module parameters

The parameters of the Volume module are listed as follows:

Table 4. Parameters

Parameter	Choices/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.
verifycert	• True • False	To validate the SSL certificate. • True - Verifies the SSL certificate. • False - Specifies that the SSL certificate should not be verified. This parameter is mandatory.
user		The user name to access the Unisphere server. The user name can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
serial_no		The serial number of the PowerMax array. If this parameter is absent, the

Table 4. Parameters (continued)

Parameter	Choices/default	Comments
		script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is mandatory.
vol_name		The name of the volume.
new_name		The new identifier for the volume.
sg_name		The name of the current storage group.
new_sg_name		The name of the target storage group for moving volumes from one storage group to another.
size		The desired size of the volume. This parameter is mandatory when you create or expand the volume. Shrinking of volumes is not supported.
cap_unit	• MB • GB • TB The default unit is *GB*.	Volume capacity unit.
vol_id		The native id of the volume. This parameter is optional.
state	• absent • present	Defines whether the volume should exist in the storage group. This parameter is mandatory.
vol_wwn		The external WWN of the volume. This parameter is optional.

Host module

The host module manages the host within the PowerMax array.

The host module has the following functions:

- Create host with initiators and host flags.
- Add initiators to the host.
- Remove initiators from the host.
- Modify host flag values.
- Rename host.
- Delete host.

Create host

The user can create a host by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create host
    dellemc_powermax_host:
      unispherehost: "{{unispherehost}}"
      universion: "{{universion}}"
      verifycert: "{{verifycert}}"
      user: "{{user}}"
      password: "{{password}}"
```

```
      serial_no: "{{serial_no}}"
      host_name: "{{host_name}}"
      initiators:
      - 10000090fa7b4e85
      host_flags:
          spc2_protocol_version: true
          consistent_lun: true
          volume_set_addressing: 'unset'
          disable_q_reset_on_ua: false
          openvms: 'unset'
      state: 'present'
      initiator_state: 'present-in-host'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get details of the host

The user can get details of the host by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
-name: Get host details
   dellemc_powermax_host:
     unispherehost: "{{unispherehost}}"
     universion: "{{universion}}"
     verifycert: "{{verifycert}}"
     user: "{{user}}"
     password: "{{password}}"
     serial_no: "{{serial_no}}"
     host_name: "{{host_name}}"
     state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Add initiator to host

The user can add an initiator to the host by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Adding initiator to host
    dellemc_powermax_host:
      unispherehost: "{{unispherehost}}"
      universion: "{{universion}}"
      verifycert: "{{verifycert}}"
      user: "{{user}}"
      password: "{{password}}"
      serial_no: "{{serial_no}}"
      host_name: "{{host_name}}"
      initiators:
      - 10000090fa3d303e
      initiator_state: 'present-in-host'
      state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Remove initiators from host

The user can remove initiators from the host by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Removing initiator from host
    dellemc_powermax_host:
      unispherehost: "{{unispherehost}}"
      universion: "{{universion}}"
      verifycert: "{{verifycert}}"
      user: "{{user}}"
      password: "{{password}}"
      serial_no: "{{serial_no}}"
      host_name: "{{host_name}}"
      initiators:
      - 10000090fa3d303e
      initiator_state: 'absent-in-host'
      state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Modify host flags

The user can modify the host flags by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Modify flags of host
    dellemc_powermax_host:
      unispherehost: "{{unispherehost}}"
      universion: "{{universion}}"
      verifycert: "{{verifycert}}"
      user: "{{user}}"
      password: "{{password}}"
      serial_no: "{{serial_no}}"
      host_name: "{{host_name}}"
      host_flags:
          spc2_protocol_version: unset
          consistent_lun: unset
          volume_set_addressing: true
          disable_q_reset_on_ua: false
          openvms: false
          avoid_reset_broadcast: true
      state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Rename host

The user can rename the host by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Rename host
    dellemc_powermax_host:
      unispherehost: "{{unispherehost}}"
      universion: "{{universion}}"
      verifycert: "{{verifycert}}"
      user: "{{user}}"
      password: "{{password}}"
      serial_no: "{{serial_no}}"
```

```
      host_name: "{{host_name}}"
      new_name: "{{new_host_name}}"
      state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete host

The user can delete the host by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete host
    dellemc_powermax_host:
      unispherehost: "{{unispherehost}}"
      universion: "{{universion}}"
      verifycert: "{{verifycert}}"
      user: "{{user}}"
      password: "{{password}}"
      serial_no: "{{serial_no}}"
      host_name: "{{new_host_name}}"
      state: 'absent'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Host module parameters

The parameters of the Host module are listed as follows:

Table 5. Parameters

Parameter	Choice/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.
verifycert	• True • False	To validate the SSL certificate. • True - Verifies the SSL certificate. • False - Specifies that the SSL certificate should not be verified. This parameter is mandatory.
user		The user name to access the Unisphere server. The user name can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
serial_no		The serial number of the PowerMax array. If this parameter is absent, the script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is mandatory.

Table 5. Parameters (continued)

Parameter	Choice/default	Comments
`host_name`		The name of the host. The following conditions are applicable when naming the host: • Do not use special characters except "_." • Case sensitive for RestAPI calls. This parameter is mandatory.
`new_name`		The new name of the host when you rename the host. The following conditions must be met when you enter a new name: • Do not use special characters except "_." • Case sensitive for RestAPI calls.
`initiators`		Lists the initiator *WWN* or *IQN* that needs to be added to or removed from the host.
`host_flags`	• yes • no • unset The default parameter is *unset*.	Enter as in *yaml* dictionary. The `host_flags` are optional. All the `host_flags` are listed below. • volume_set_addressing • disable_q_reset_on_ua • environ_set • avoid_reset_broadcast • openvms • scsi_3 • spc2_protocol_version • scsi_support1 • consistent_lun
`state`	• absent • present	Defines whether the host must exist in the system. • absent - indicates that the host must not exist in the system. • present - indicates that the host must exist in the system. This parameter is mandatory.
`initiator_state`	• present-in-host • absent-in-host	Defines whether the initiator must be available in the host. • present-in-host - indicates that the initiator must be present in the host. • absent-in-host - indicates that the initiator must not be present in the host. This parameter is mandatory for the following conditions: • Create a host with initiators. • Add initiators to a host. • Remove initiators from a host

Host group module

The host group module manages the host group within the PowerMax array. A host group is a container for multiple hosts and enables simple configuration of clusters.

The functions of the host group module are as follows:

- Create host group with hosts.
- Add hosts to a host group.
- Remove hosts from a host group.
- Rename host group.
- Modify the host flags of a host group.
- Delete host group.

Create host group

The user can create a host group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create host group
  dellemc_powermax_hostgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    hostgroup_name: "{{hostgroup_name}}"
    hosts:
    - Ansible_Testing_host
    state: 'present'
    host_state: 'present-in-group'
    host_flags:
        spc2_protocol_version: true
        consistent_lun: true
        disable_q_reset_on_ua: false
        openvms: 'unset'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get details of host group

The user can get the details of the host group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Get host group details
  dellemc_powermax_hostgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    hostgroup_name: "{{hostgroup_name}}"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Add host to host group

The user can add host to a host group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Adding host to host group
  dellemc_powermax_hostgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    hostgroup_name: "{{hostgroup_name}}"
    hosts:
    - Ansible_Testing_host2
    state: 'present'
    host_state: 'present-in-group'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Remove host from host group

The user can remove a host from the host group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Removing host from host group
  dellemc_powermax_hostgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    hostgroup_name: "{{hostgroup_name}}"
    hosts:
    - Ansible_Testing_host2
    state: 'present'
    host_state: 'absent-in-group'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Modify host group flags

The user can modify the host flags by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Modify flags of host group
  dellemc_powermax_hostgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    hostgroup_name: "{{hostgroup_name}}"
    host_flags:
        spc2_protocol_version: unset
        disable_q_reset_on_ua: false
```

```
        openvms: false
        avoid_reset_broadcast: true
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Rename host group

The user can rename the host group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Rename host group
  dellemc_powermax_hostgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    hostgroup_name: "{{hostgroup_name}}"
    new_name: "Ansible_Testing_hostgroup2"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete host group

The user can delete the host group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete host group
  dellemc_powermax_hostgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    hostgroup_name: "Ansible_Testing_hostgroup2"
    state: 'absent'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Host Group module parameters

The parameters of the Host Group module are listed as follows with an example:

Table 6. Parameters

Parameter	Choice/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.

Table 6. Parameters (continued)

Parameter	Choice/default	Comments
verifycert	• True • False	To validate the SSL certificate. • True - Verifies the SSL certificate. • False - Specifies that the SSL certificate should not be verified. This parameter is mandatory.
user		The user name to access the Unisphere server. The user name can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
serial_no		The serial number of the PowerMax array. If this parameter is absent, the script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is mandatory.
host_group_name		The name of the host group. The following conditions must be met when you enter the name: • Do not use special characters except "_" • Case sensitive for RestAPI calls. This parameter is mandatory.
new_name		The new name of the host group when you rename the host group. The following conditions must be met when you enter a new name: • Do not use special characters except "_" • Case sensitive for RestAPI calls.
hosts		Lists of the host names that are added to the host group or removed from host group. You can create an empty host group.
host_state	• present-in-group • absent-in-group	Defines whether the host must be available in the host group. • present-in-group - indicates that the host must be present in the host group. • absent-in-group - indicates that the host must not be present in the host group.
host_flags	• yes • no • unset (default)	Enter as in *yaml* dictionary. All the host_flags are listed below. • volume_set_addressing • disable_q_reset_on_ua • environ_set

Table 6. Parameters (continued)

Parameter	Choice/default	Comments
		• avoid_reset_broadcast • openvms • scsi_3 • spc2_protocol_version • scsi_support1 • consistent_lun
state	• absent • present	Defines whether the host group must be present in the system. • absent - The host must not be present in the system. • present - The host must be present in the system. This parameter is mandatory.

Port module

The ports available on the PowerMax array are managed by the Port module. The Port module lists the details of single or multiple ports.

Get details of single/multiple ports

The user can get the details of single or multiple ports by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Get details of single/multiple ports
  dellemc_powermax_port:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{array_id}}"
    ports:
    - director_id: "FA-1D"
      port_id: "5"
    - director_id: "SE-1F"
      port_id: "29"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Port module parameters

The parameters for the Port module are listed as follows with an example:

Table 7. Parameters

Parameter	Choice/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.

Table 7. Parameters (continued)

Parameter	Choice/default	Comments
verifycert	• True • False	To validate the SSL certificate. • True - Verifies the SSL certificate. • False - Specifies that the SSL certificate should not be verified. This parameter is mandatory.
user		The user name to access the Unisphere server. The user name can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
serial_no		The serial number of the PowerMax array. If this parameter is absent, the script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is mandatory.
ports		Lists the port director and the port id.

Port Group module

The Port Group module manages the port group in the PowerMax array.

The port group management module has the following functions:

- Create a port group with ports.
- Create empty port group.
- Add ports to a port group.
- Remove ports from a port group.
- Rename a port group.
- Remove a port group.

Create port group without ports

The user can create a port group without ports by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create port group without ports
  dellemc_powermax_portgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{array_id}}"
    portgroup_name: "{{portgroup_name}}"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Create port group with ports

The user can create a port group with ports by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create port group with ports
  dellemc_powermax_portgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{array_id}}"
    portgroup_name: "{{portgroup_name}}"
    state: "present"
    ports:
    - director_id: "FA-1D"
      port_id: "5"
    - director_id: "FA-2D"
      port_id: "5"
    port_state: "present-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Add ports to port group

The user can add ports to a port group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Add ports to port group
  dellemc_powermax_portgroup:
   unispherehost: "{{unispherehost}}"
   universion: "{{universion}}"
   verifycert: "{{verifycert}}"
   user: "{{user}}"
   password: "{{password}}"
   serial_no: "{{array_id}}"
   portgroup_name: "{{portgroup_name}}"
   state: "present"
   ports:
   - director_id: "FA-2D"
     port_id: "8"
   - director_id: "FA-2D"
     port_id: "9"
   port_state: "present-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Remove ports from port group

The user can remove ports from the port group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Remove ports from port group
  dellemc_powermax_portgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
```

```
    password: "{{password}}"
    serial_no: "{{array_id}}"
    portgroup_name: "{{portgroup_name}}"
    state: "present"
    ports:
    - director_id: "FA-2D"
    port_id: "8"
    - director_id: "FA-2D"
    port_id: "9"
    port_state: "absent-in-group"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Rename port group

The user can rename the port group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Modify port group
  dellemc_powermax_portgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{array_id}}"
    portgroup_name: "{{portgroup_name}}"
    state: "present"
    new_name: "{{new_name}}"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete port group

The user can delete a port group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete port group
  dellemc_powermax_portgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{array_id}}"
    portgroup_name: "{{portgroup_name}}"
    state: "absent"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Port Group module parameters

The parameters for the Port Group module are listed as follows with an example:

Table 8. Parameters

Parameter	Choice/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.
verifycert	• True • False	To validate the SSL certificate. • True - Verifies the SSL certificate. • False - Specifies that the SSL certificate should not be verified. This parameter is mandatory.
user		The user name to access the Unisphere server. The user name can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
serial_no		The serial number of the PowerMax array. If this parameter is absent, the script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is mandatory.
portgroup_name		The name of the port group. The following conditions must be met when you enter the name: • Do not use special characters except "_" • Case sensitive for RestAPI calls. This parameter is mandatory.
state	• absent • present	Defines whether the port group must be present in the system. • absent: The port group must not be present in the system. • present: The port group must be present in the system. This parameter is mandatory.
ports		Lists the port director and ports that are added to the port group or removed from port group.
port_state	• present-in-group • absent-in-group	Defines whether the port must be available in the port group. • present-in-group: indicates that the port must be present in the port group.

Table 8. Parameters (continued)

Parameter	Choice/default	Comments
		• absent-in-group: indicates that the port must not be present in the port group.
new_name		The new name of the port group when you rename the port group. The following conditions must be met when you enter a new name: • Do not use special characters except "_." • Case sensitive for RestAPI calls.

Masking View module

The Masking View module manages the masking views on the PowerMax array.

The masking view module has the following functions:

- Create masking view with a port group, storage group, host, or host group.
- Modify masking view.
- Delete masking view.

For creating a masking view, the port groups, storage groups, hosts, and host groups must be present on the array. Once the masking view is created, only the name of the masking view can be changed. The entities such as port group, storage group, hosts, or host group cannot be changed. The masking view can be created either for a host or a host group, but not for both. The host name or a host group name must be provided to create masking view.

Create masking view with host group

The user can create a masking view with a host group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create MV with hostgroup
  dellemc_powermax_maskingview:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    mv_name: "{{mv_name}}"
    portgroup_name: "Ansible_Testing_portgroup"
    hostgroup_name: "Ansible_Testing_hostgroup"
    sg_name: "Ansible_Testing_SG"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Create masking view with host

The user can create a masking view with host by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create MV with host
```

```
  dellemc_powermax_maskingview:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    mv_name: "{{mv_name}}"
    portgroup_name: "Ansible_Testing_portgroup"
    host_name: "Ansible_Testing_host"
    sg_name: "Ansible_Testing_SG"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Rename masking view

The user can rename the masking view by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Rename host masking view
  dellemc_powermax_maskingview:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    mv_name: "{{mv_name}}"
    new_mv_name: "Ansible_Testing_mv_renamed"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete masking view

The user can delete the masking view by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete host masking view
  dellemc_powermax_maskingview:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    mv_name: "Ansible_Testing_mv_renamed"
    state: "absent"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Masking View module parameters

The parameters for the Masking View module are listed as follows with an example:

Table 9. Parameters

Parameter	Choice/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.
verifycert	• True • False	To validate the SSL certificate. • True - Verifies the SSL certificate. • False - Specifies that the SSL certificate should not be verified. This parameter is mandatory.
user		The user name to access the Unisphere server. The user name can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
serial_no		The serial number of the PowerMax array. If this parameter is absent, the script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is mandatory.
mv_name		The name of the masking view. The following conditions must be met when you enter the name: • Do not use special characters except "_" • Case sensitive for RestAPI calls. This parameter is mandatory.
new_mv_name		The new name of the masking view when you rename the existing masking view. The following conditions must be met when you provide the new name for masking view: • Do not use special characters except "_" • Case sensitive for REST API calls.
portgroup_name		The name of the existing port group.
hostgroup_name		The name of the existing host group. Use this parameter to create cluster export.
host_name		The name of the existing host. Use this parameter to create an exclusive host or to export to a host.

Table 9. Parameters (continued)

Parameter	Choice/default	Comments
sg_name		The name of the existing storage group.
state	• absent • present	Defines whether the masking view must be present on the system. This parameter is mandatory.

Snapshot module

The Snapshot module manages the snapshots available on the PowerMax array.

The snapshot module has the following functions:

- Create a storage group snapshot.
- Get details of a storage group snapshot.
- Rename a storage group snapshot.
- Change storage group snapshot link status.
- Delete storage group snapshot.

Create snapshot for a storage group

The user can create a snapshot for a storage group by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create a Snapshot for a Storage Group
  dellemc_powermax_snapshot:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "ansible_sg"
    snapshot_name: "ansible_sg_snap"
    ttl: "2"
    ttl_unit: "days"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get details of storage group snapshot

The user can get details of a storage group snapshot by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Get Storage Group Snapshot details
  dellemc_powermax_snapshot:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "ansible_sg"
    snapshot_name: "ansible_sg_snap"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete storage group snapshot

The user can delete a storage group snapshot by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete Storage Group Snapshot
  dellemc_powermax_snapshot:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "ansible_sg"
    snapshot_name: "ansible_sg_snap"
    generation: 1
    state: "absent"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Rename storage group snapshot

The user can rename the storage group snapshot by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Rename Storage Group Snapshot
  dellemc_powermax_snapshot:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "ansible_sg"
    snapshot_name: "ansible_sg_snap"
    new_snapshot_name: "ansible_snap_new"
    generation: 0
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Change snapshot link status to linked

The user can change the link status of the snapshot to *linked* by running the appropriate playbook

The syntax of the playbook is shown as follows:

```
- name: Change Snapshot Link Status to Linked
  dellemc_powermax_snapshot:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "ansible_sg"
```

```
    snapshot_name: "ansible_snap_new"
    generation: 1
    target_sg_name: "ansible_sg_target"
    link_status: "linked"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Change snapshot link status to unlinked

The user can change the link status of the snapshot to *unlinked* by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Change Snapshot Link Status to UnLinked
  dellemc_powermax_snapshot:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "ansible_sg"
    snapshot_name: "ansible_snap_new"
    generation: 1
    target_sg_name: "ansible_sg_target"
    link_status: "unlinked"
    state: "present"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Snapshot module parameters

The parameters for the Snapshot module are listed as follows with an example:

Table 10. Parameters

Parameter	Choice/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.
verifycert	• True • False	To validate the SSL certificate. • True - Verifies the SSL certificate. • False - Specifies that the SSL certificate should not be verified. This parameter is mandatory.
user		The user name to access the Unisphere server. The user name can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.

Table 10. Parameters (continued)

Parameter	Choice/default	Comments
`serial_no`		The serial number of the PowerMax array. If this parameter is absent, the script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is mandatory.
`sg_name`		The name of the storage group. This parameter is mandatory.
`snapshot_name`		The name of the snapshot. This parameter is mandatory.
`new_snapshot_name`		The new name of the snapshot.
`generation`		The generation number of the Snapshot. Generation is mandatory for link, unlink, rename, and delete operations. Optional for *Get snapshot details* operation. Create snapshot creates a snapshot with generation number 0. Rename is supported only for generation number 0.
`target_sg_name`		The target storage group name.
`link_status`	• linked • unlinked	Defines the link status of the snapshot.
`ttl`		The Time To Live (TTL) value for the Snapshot. If the TTL value is not specified, the storage group snapshot details would be returned. However, to create a storage group snapshot, the TTL must be specified. If the storage group snapshot does not have a TTL value, specify the TTL value as "None."
`ttl_unit`	• hours • days	The unit for TTL. If no `ttl_unit` is specified, 'days' is taken as default value.
`state`	• absent • present	Defines whether the snapshot must exist on the storage array. This parameter is mandatory.

SRDF Module

The SRDF Module manages the SRDF links in Dell EMC PowerMax arrays. The SRDF Module supports the following functionalities:

- Create SRDF links with the following replication modes:
  - Synchronous
  - Asynchronous
  - Adaptive copy
  - Active mode
- Get SRDF pair states for a given storage group and RDFG number.
- Modify SRDF link mode.
- Perform the following operations:
  - Establish
  - Restore

  - Swap
  - Failover
  - Resume
  - Suspend
  - Split
  - Failback
  - Setbias
- Metro configurations: Set bias operation, change resiliency between bias and witness.
- Run create and update operations Asynchronously (by default) or synchronously.
- Get Job details for a given Job ID.
- Delete SRDF links.
- Concurrent SRDF configuration support.
- Online Device expansion support

NOTE: The following features are not supported:

- The Star and Cascaded SRDF configurations
- The Invalidate, Ready, Not Ready, Move Pairs, and Refresh SRDF pairing operations
- Migration use cases through the SRDF Module

Create SRDF links

The user can create SRDF links by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create and establish storagegroup SRDF/S pairing
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    remote_serial_no: "{{remote_serial_no}}"
    sg_name: "{{sg_name}}"
    srdf_mode: 'Synchronous'
    srdf_state: 'Establish'
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Create and establish a storage group SRDF/a pairing

The user can create and establish a storage group SRDF/a pairing by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
  - name: Create and establish storagegroup SRDF/a pairing
    register: Job_details_body
    dellemc_powermax_srdf:
      unispherehost: "{{unispherehost}}"
      universion: "{{universion}}"
      verifycert: "{{verifycert}}"
      user: "{{user}}"
      password: "{{password}}"
      serial_no: "{{serial_no}}"
      sg_name: "{{sg_name}}"
      remote_serial_no: "{{remote_serial_no}}"
      srdf_mode: 'Asynchronous'
```

```
      srdf_state: 'Establish'
      state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Create a storage group SRDF/s pair in suspended mode

The user can create a storage group SRDF/s pair in the default suspended mode by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create storagegroup SRDF/s pair in default suspended mode as an
        Synchronous task
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name2}}"
    remote_serial_no: "{{remote_serial_no}}"
    state: 'present'
    srdf_mode: 'Synchronous'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get SRDF details

The user can get SRDF details by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Get SRDF details
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name}}"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Modify SRDF mode

The user can modify SRDF mode by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Modify SRDF mode
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
```

```
    sg_name: "{{sg_name}}"
    srdf_mode: 'Synchronous'
    state: 'present'
```

You can modify the `srdf_mode` parameter to any of the following modes:

- Synchronous
- Asynchronous
- Adaptive copy
- Active

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Failover SRDF link

The user can perform failover operation by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Failover SRDF link
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name}}"
    srdf_state: 'Failover'
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Failback SRDF link

The user can perform failback operation by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Failback SRDF link
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{ serial_no }}"
    sg_name: "{{sg_name}}"
    rdfg_no: "{{rdfg_no}}"
    srdf_state: "Failback"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Create storage group Metro SRDF pair with Witness for resiliency

The user can create a storage group Metro SRDF pair with Witness for resiliency by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create storagegroup Metro SRDF pair with Witness for resiliency
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name}}"
    remote_serial_no: "{{remote_serial_no}}"
    state: 'present'
    srdf_mode: 'Active'
    srdf_state: 'Establish'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Get SRDF Job status

The user can get SRDF Job status by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Get SRDF Job status
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    job_id: "{{job_id}}"
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Establish SRDF link

The user can establish the SRDF link by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Establish SRDF link
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name}}"
    srdf_state: 'Establish'
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Create and Establish concurrent SRDF link

The user can establish the concurrent SRDF link (one at a time) by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Create and Establish SRDF pairing with remote array 1
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    remote_serial_no: "{{remote_serial_no_1}}"
    sg_name: "{{sg_name}}"
    srdf_state: 'Establish'
    srdf_mode: 'Synchronous'
    state: 'present'

- name: Create and Establish SRDF pairing with remote array 2
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    remote_serial_no: "{{remote_serial_no_2}}"
    sg_name: "{{sg_name}}"
    srdf_state: 'Establish'
    srdf_mode: 'Adaptive Copy'
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Suspend SRDF link

The user can suspend the SRDF link by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Suspend SRDF link
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name2}}"
    srdf_state: 'Suspend'
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Suspend concurrent SRDF link

The user can suspend the concurrent SRDF link (one at a time) by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Suspend SRDF link
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name2}}"
    srdf_state: 'Suspend'
    state: 'present'
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete SRDF link

The user can delete the SRDF links by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete SRDF link
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{serial_no}}"
    sg_name: "{{sg_name}}"
    state: 'absent'
```

NOTE: Only links in Suspended states can be deleted.

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

Delete concurrent SRDF link

The user can delete the concurrent SRDF link (one at a time) by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Delete SRDF link
  dellemc_powermax_srdf:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: "{{r1_serial_no}}"
    sg_name: "{{sg_name1}}"
    rdfg_no: "{{rdfGroupNumber}}"
    state: 'absent'
    wait_for_completion: True
```

NOTE: Only links in Suspended states can be deleted.

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

SRDF module parameters

The following table provides information about the parameters that are displayed on the console, when the user runs the playbook using SRDF Module:

Table 11. Parameters

Parameter	Choices/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.
verifycert	• True • False	To validate the SSL certificate. • True - indicates that the SSL certificate should be verified. • False - indicates that the SSL certificate should not be verified. This parameter is mandatory.
user		The username to access the Unisphere server. The username can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
serial_no		The serial number of the source PowerMax or VMAX array (primary array), when protecting a storage group. This parameter is mandatory. **NOTE:** You can issue srdf_state operations from a primary or remote array.
remote_serial_no		A 12-Digit serial number of the remote PowerMAX or VMAX array (remote). This parameter is mandatory when creating an SRDF pair.
sg_name		Name of the Storage group. SRDF pairings are managed at a storage group level. This parameter is required to identify the SRDF link. This parameter is mandatory.
srdf_mode	• Active • Adaptive Copy • Synchronous • Asynchronous	The replication mode of the SRDF pair. This parameter is mandatory when creating an SRDF pair. It can be modified by specifying one of the following values: • Active • Adaptive Copy • Synchronous • Asynchronous
state	• present • absent	Define whether the SRDF pairing should exist or not.

Table 11. Parameters (continued)

Parameter	Choices/default	Comments
		• present - indicate that the SRDF pairing should exist in the system. • absent - indicate that the SRDF pairing should not exist in the system. This parameter is mandatory.
`srdf_state`	• Establish • Restore • Swap • Failover • Resume • Suspend • Split • Failback • Setbias	Desired state of the SRDF pairing. This parameter is optional. While creating a new SRDF pair, the allowed values are **`Establish`** and **`Suspend`**. If the `srdf_state` parameter is not specified, the pair will be created in the **`Suspend`** state.
`new_rdf_group`	• True • False The default value is false.	PowerMax has a limited number of RDF groups. If this flag is set to *True*, and the RDF groups are exhausted, then SRDF link creation will fail.
`rdfg_number`		The RDF group number. This parameter is optional for each call. For the **`create`** operation, if specified, the array will reuse the RDF group, or display an error. For the **`modify`** and **`delete`** operations, if the RFD group number is not specified, the storage group is protected by multiple RDF Groups. This causes an error.
`job_id`		Job ID of an Asynchronous task. This parameter is used to get the details of a job.
`wait_for_completion`	• True • False The default value is false.	Flag to indicate if the operation should be run synchronously or asynchronously. • True - Synchronous • False - Asynchronously The default value is False. All create and update operations will be run asynchronously by default.
`witness`	• True • False	Flag to specify use of Witness for a Metro configuration. The flag can be set only for modifying the `srdf_state` parameter to either Establish, Suspend or Restore. • True - To use Witness • False - To use Bias **NOTE:** It is recommended to set this parameter for SRDF Metro in a production environment. Use *Unipshere for PowerMAX* UI or REST API to configure this parameter. **NOTE:** Set `witness` to **`true`** when you create a Metro configuration.

RDF Group module

The RDF Group module provides details of the RDF group and the list of volumes available on a specified RDF Group. The RDF group module supports the Get details of RDF Groups and volumes function.

Get RDF group and volumes details

The user can get details of an RDF group and the list of volumes in a specific RDF Group, by running the appropriate playbook.

The syntax of the playbook is shown as follows:

```
- name: Get RDF Group detail and Volumes
  dellemc_powermax_rdfgroup:
    unispherehost: "{{unispherehost}}"
    universion: "{{universion}}"
    verifycert: "{{verifycert}}"
    user: "{{user}}"
    password: "{{password}}"
    serial_no: {{ serial_no }}
    rdfgroup_number: "{{rdfgroup_number}}"
  register: "rdfgroup_result"
```

The parameters must be set before the user runs the playbook. See the Parameters table for more information about the parameters.

RDF Group module parameters

The parameters for the RDF Group module are listed as follows with an example:

Table 12. Parameters

Parameter	Choice/default	Comments
unispherehost		IP or FQDN of the Unisphere host. This parameter is mandatory.
universion	• 91 • 92	The version of the Unisphere software. This parameter is optional.
verifycert	• True • False	To validate the SSL certificate. • True - Verifies the SSL certificate. • False - Specifies that the SSL certificate should not be verified. This parameter is mandatory.
user		The username to access the Unisphere server. The username can be encrypted using Ansible vault. This parameter is mandatory.
password		The password to access the Unisphere server. The password can be encrypted using Ansible vault. This parameter is mandatory.
serial_no		The serial number of the PowerMax array. If this parameter is absent, the script lists all the serial numbers of the registered array on the specified Unisphere host. This parameter is mandatory.

Table 12. Parameters (continued)

Parameter	Choice/default	Comments
rdfgroup_number		The serial number of the RDF Group, used to get the details of an RDF Group and list the volumes of the specified RDF Group.