

# Ansible Modules for Dell EMC VPLEX

## Product Guide

1.3

## 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.

<b>Chapter 1: Introduction.....</b>	<b>6</b>
Product overview.....	6
<b>Chapter 2: Configure Ansible.....</b>	<b>8</b>
Software prerequisites.....	8
Steps to install the Ansible module .....	8
Auto installation through installer utility.....	8
Manual installation.....	10
Steps to retrieve CA certificate from VPLEX.....	14
Steps to run playbooks in Ansible collections.....	16
<b>Chapter 3: Ansible modules for Dell EMC VPLEX.....</b>	<b>18</b>
Gather Facts module.....	18
Get list of storage arrays.....	19
Get list of storage volumes.....	19
Get list of storage volumes with filters (operator: equal).....	19
Get list of extents.....	20
Get list of devices.....	20
Get list of distributed devices.....	21
Get list of virtual volumes.....	21
Get list of virtual volumes with filters (operator: greater and lesser).....	22
Get list of distributed virtual volumes.....	22
Get list of consistency groups.....	23
Get list of distributed consistency groups.....	23
Get list of ports.....	24
Get list of BE ports.....	24
Get list of initiators.....	25
Get list of storage views.....	25
Get list of device migration jobs.....	26
Get list of extent migration jobs.....	26
Get list of array management providers.....	27
Gather Facts module parameters.....	27
Storage volume module.....	33
Claim storage volume.....	33
Unclaim storage volume.....	34
Update or modify storage volume.....	36
Set thin rebuild.....	37
List ITLs.....	39
Storage volume module parameters.....	41
Extent module.....	45
Create extent with storage volume name.....	45
Create extent with storage volume ID.....	46
Get extent.....	46
Rename extent.....	47

Delete extent with extent name.....	48
Delete extent with storage volume name.....	49
Extent module parameters.....	49
Device module.....	54
Create a RAID 1, RAID 0, and RAID C device.....	54
Get device from cluster.....	56
Add an extent to the device.....	56
Remove an extent from the device.....	57
Rename device.....	57
Update transfer size of the device.....	58
Add or Remove local mirror to the device.....	59
Add or Remove remote mirror from the device.....	60
Delete device.....	61
Device module parameters.....	62
Distributed device module.....	72
Create a distributed device.....	72
Rename a distributed device.....	73
Update the rule set name of a distributed device.....	74
Get the details of distributed device.....	74
Delete a distributed device.....	75
Distributed device module parameters.....	75
Virtual volume module.....	80
Create virtual volume.....	80
Cache invalidate on virtual volume.....	81
Get virtual volume using name or System ID.....	82
Enable remote access using name or System ID.....	83
Disable remote access using name or System ID.....	84
Rename virtual volume using name or System ID.....	85
Expand virtual volume with device using name or System ID.....	86
Expand virtual volume through backend storage volume expansion.....	87
Delete virtual volume using name or System ID.....	88
Virtual volume module parameters.....	89
Distributed virtual volume module.....	96
Create distributed virtual volume.....	96
Get distributed virtual volume using name or System ID.....	97
Rename a distributed virtual volume using name or System ID.....	98
Expand distributed virtual volume using name or System ID.....	100
Delete distributed virtual volume using name or System ID.....	101
Distributed virtual volume module parameters.....	102
Consistency group module.....	107
Create a consistency group.....	107
Add virtual volumes to consistency group.....	108
Remove virtual volumes from consistency group.....	108
Rename consistency group.....	109
Delete consistency group.....	109
Get consistency group.....	110
Consistency group module parameters.....	110
Distributed consistency group module.....	116
Create a distributed consistency group.....	116
Resume a distributed consistency group.....	117

Get a distributed consistency group.....	117
Add or remove distributed virtual volumes to a distributed consistency group.....	118
Update the detach rule of a distributed consistency group.....	119
Disable or enable auto-resume-at loser.....	120
Rename a distributed consistency group.....	121
Delete a distributed consistency group.....	121
Distributed consistency group module parameters.....	122
Port module.....	132
Get port.....	132
Enable port.....	132
Disable port.....	133
Port module parameters.....	133
Initiator module.....	137
Register an initiator.....	137
Get details of an initiator.....	138
Rename initiator.....	138
Unregister an initiator.....	139
Rediscover initiators.....	140
Initiator module parameters.....	141
Storage View module.....	147
Get details of a storage view.....	148
Create a storage view.....	148
Delete a storage view.....	149
Rename a storage view.....	149
Add ports to a storage view.....	150
Add initiators to a storage view.....	150
Add virtual volumes to a storage view.....	151
Remove ports from a storage view.....	152
Remove initiators from a storage view.....	152
Remove virtual volumes from a storage view.....	153
Storage view module parameters.....	153
Data migration module.....	163
Device migration.....	163
Extent migration.....	169
Data migration module parameters.....	174
Rediscover array module.....	186
Rediscover array.....	187
Get array.....	187
Rediscover array module parameters.....	188
Maps module.....	190
Get map for given virtual volume.....	190
Get map for given distributed virtual volume.....	190
Get map for given device.....	191
Get map for given distributed device.....	191
Get map for given extent.....	192
Get map for given storage volume.....	192
Maps module parameters.....	193

<b>Chapter 4: Sample playbooks.....</b>	<b>198</b>
---	------------

# Introduction

This chapter contains the following topics:

## Topics:

- [Product overview](#)

## Product overview

The Ansible Modules for Dell EMC VPLEX are used to automate and orchestrate the configuration of resources and provision storage from the VPLEX system.

The Ansible modules can manage Storage views, Initiators, Ports, Consistency groups, Virtual volumes, Devices, Extents, Storage volumes, Distributed devices, Distributed virtual volumes, Distributed consistency groups, Data migration jobs, get the hierarchy of the storage entity using Maps module, and also able to get information of the configured resources through the gather facts module. In this release, the filtering mechanism is implemented in the gather facts module as an improvement. 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. The modules use VPLEX Python SDK to interface with the VPLEX.


### List of Ansible Modules for Dell EMC VPLEX

- [Gather facts](#)
- [Storage volume](#)
- [Extent](#)
- [Device](#)
- [Distributed device](#)
- [Virtual volume](#)
- [Distributed virtual volume](#)
- [Consistency group](#)
- [Distributed consistency group](#)
- [Port](#)
- [Initiator](#)
- [Storage view](#)
- [Data migration \(mobility\)](#)
- [Rediscover array](#)
- [Maps](#)

**Table 1. Common parameters**

Parameter name	Choice or default	Type	Mandatory/ Optional Parameter	Description
vplexhost	N/A	str	Mandatory	The IP/FQDN of VPLEX server.
vplexuser	N/A	str	Mandatory	The name of the user used to authenticate with the VPLEX.
vplexpassword	N/A	str	Mandatory	The password of the user used to authenticate with the VPLEX.
verifycert	N/A	bool	Mandatory	Specifies whether to verify the SSL certificate for VPLEX Ansible commands.

**Table 1. Common parameters (continued)**

Parameter name	Choice or default	Type	Mandatory/ Optional Parameter	Description
ssl_ca_cert	N/A	str	Optional	Path of SSL CA certificate file to be verified when verifycert is set to True. It is required only when verifycert is specified as True.  <b>NOTE:</b> If this value is set to true, then see the <a href="#">Steps to retrieve CA certificate from VPLEX</a> .
debug	true	bool	Optional	It specifies log or does not log the debug statements in the ansible module log file (dellemc_ansible_vplex.log).
vplex_timeout	Default : 30 sec	int	Optional	It is a network connectivity timeout value to connect to the VPLEX host in seconds.

# Configure Ansible

This chapter contains the following topics:

## Topics:

- [Software prerequisites](#)
- [Steps to install the Ansible module](#)
- [Steps to run playbooks in Ansible collections](#)

## Software prerequisites

Provides information about the software prerequisites for the Ansible Modules for Dell EMC VPLEX.

**Table 2. Software prerequisites**

Ansible modules	VPLEX version	Red Hat Enterprise Linux	Python version	VPLEX Python SDK version	Ansible version
v1.3	6.2	7.5 ,8.8	2.7.18,3.6	6.2	2.9,2.11
	7.0	7.6, 9.4	3.6.9,3.13	7.0	2.10,2.18
	7.1	9.4	3.13	7.1	2.18
	8.0	9.4	3.13	8.0	2.18
	8.0.1	9.4	3.13	8.0.1	2.18
	9.0	9.4	3.13	9.0	2.18

## Steps to install the Ansible module

### Auto installation through installer utility

Ansible modules can be installed manually or using auto installer utility. With the help of installer, the user can choose the type of installation (copy modules to python library or install collections).

#### About this task

Before using the auto installer script, the Internet connectivity should be stable. For Red Hat Enterprise systems, the subscription manager should be enabled.

1. If the user downloads the tar file **dellemc-vplex-1.3.0.tar.gz**, then transfer the **dellemc-vplex-1.3.0.tar.gz** Ansible code .zip file to the external host or the management server.

**NOTE:** If the user is at VPLEX version 6.x, then the user can only run the Ansible script using an external host. If the user is at metro node version 7.x, then the user can run the Ansible scripts from an external host or from within the VPLEX management server.

2. Create a directory with the name **ansible\_vplex**, move the tar file into the created directory, then extract the transferred code in the created directory using the following commands:

```
mkdir ansible_vplex
mv dellemc-vplex-1.3.0.tar.gz ansible_vplex/
```



```
cd ansible_vplex/  
tar -xvf dellemc-vplex-1.3.0.tar.gz
```

## Steps

1. As per user inputs, the installer will setup the virtual environment and install all required packages, including the latest vplexapi library. The installer.sh can be found in the tools directory of ansible\_vplex repository, and launched using following command:

```
root@newdevm:~/new_folder/ansible_vplex/tools# bash installer.sh  
please choose the python version: (recommended: 1)  
1) python3.13  
2) quit  
#? 1  
please choose the ansible version: (recommended: 1)  
1) ansible2.18  
2) quit  
#? 1  
Please choose the installation method: (recommended: 1)  
1) use_collections  
2) skip_collections  
3) quit  
#? 1  
Please choose VPLEX python sdk: (recommended: 2)  
Info: sdk_6.2 is supported only up to Python 3.6  
1) sdk_6.2  
2) sdk_7.0  
3) sdk_8.0  
4) sdk_8.1  
5) sdk_9.0  
6) quit  
#? 2
```

2. For a Non-VPLEX host, as a prerequisite you must install Python 3.13, virtualenv, and pip.
3. For VPLEX SDK versions other than 7.0, you must manually copy the appropriate SDK folder and set the PYTHONPATH accordingly.
4. Installer verifies that all the modules are installed as expected and accessible in the user environment.
5. Once the installation is completed, installer provides information regarding the installation path.
6. Installer suggests commands to activate the virtual environment and export environment variable to use the modules.

```
Info: Installed collections can be found in /root/.py3.13_ans2.18/collections  
Info: Sample playbooks can be found in /root/new_folder/ansible_vplex/docs/  
  
Info: Run below command to activate virtualenv and run playbooks...  
  
    source /root/.py3.13_ans2.18/bin/activate  
    export ANSIBLE_COLLECTIONS_PATH=/root/.py3.13_ans2.18/collections  
    export PYTHONPATH=/root/Downloads/python-vplex/vplexapi-7.0.0  
  
Info: To deactivate virtualenv run command: deactivate
```

- NOTE:** Once the installer starts running, it asks the user for python version, ansible version, and installation type (use\_collections or skip\_collections). As per the user inputs, installer setup the virtual environment and install all required packages, including the vplexapi library. For the VPLEX Management server, the ansible and python versions are present in the setup by default, the installer place the modules in the respective path based on the execution type (skip\_collections: copy modules to python library, or use\_collections: install collections), and enable the users to run the playbook only through setting the environment.
- NOTE:** On running script on vplex host, the script automatically detects the installed versions of Python and Ansible, and installs the latest collections from the Ansible repository.

## Manual installation

Before installing the VPLEX python SDK, two packages should be installed:

1. urllib3 - pip install urllib3
2. certifi - pip install certifi

## For VPLEX SDK 9.0 (metro node-External host configuration)

The VPLEX API python package is installed on the VPLEX Management Server.

### Install Dell EMC VPLEX python SDK

Follow the below steps:

1. Download the SDK Tarball. External clients can download the SDK from the following location:

```
https://<system>/apipackages/python/vplexapi_v2.tgz
```

2. Extract the Tarball. Run the following command to extract the contents:

```
tar -xvf vplexapi_v2.tgz
```

3. Set the Python Path. Export the path of the extracted vplexapi to the PYTHONPATH environment variable:

```
export PYTHONPATH="${PYTHONPATH}<path-to-extracted-vplexapi>"
```

For example,

```
root@dev-vm-14:~# export PYTHONPATH="${PYTHONPATH}:/root/vplex_2/vplexapi_v2"
root@dev-vm-14:~# echo $PYTHONPATH
{}:/root/vplex_2/vplexapi_v2
```

### Steps to install Ansible collections

1. Download the tar build from [Git hub](#), or from [Ansible galaxy portal](#) . To install the collection anywhere in the system, follow the command:

```
ansible-galaxy collection install dellemc-vplex-1.3.tar.gz -p ./collections
```

2. Set the environment variable:

```
export ANSIBLE_COLLECTIONS_PATHS=$ANSIBLE_COLLECTIONS_PATHS:<install_path>/collections.
```

#### Example:

```
(env_14) root@dev-vm-14:~/install_collection# ls
dellemc-vplex-1.3.tar.gz
(env_14) root@dev-vm-14:~/install_collection# ansible-galaxy collection install dellemc-
vplex-1.3.tar.gz -p ./collections
Starting galaxy collection install process
[WARNING]: The specified collections path '/root/install_collection/collections' is not
part of the configured Ansible collections paths
'/root/.ansible/collections:/usr/share/ansible/collections'. The installed collection
will not be picked up in an Ansible run, unless within a
playbook-adjacent collections directory.
Process install dependency map
Starting collection install process
Installing 'dellemc.vplex:1.3' to '/root/install_collection/collections/
ansible_collections/dellemc/vplex'
dellemc.vplex:1.3 was installed successfully

(env_14) root@dev-vm-14:~/install_collection# cd /root/install_collection/collections/
ansible_collections/dellemc/vplex
(env_14) root@dev-vm-14:~/install_collection/collections/ansible_collections/dellemc/
vplex# ls
CHANGELOG.md    dellemc_ansible_vplex.log  LICENSE          playbooks
README.md       roles                      tools
CHANGELOG.rst   docs                      MANIFEST.json    playbooks_manual_testing
```

```

requirements.txt  sanity_logs
changelogs       FILES.json                      meta           plugins
requirements.yml tests
(env_14) root@dev-vm-14:~/install_collection/collections/ansible_collections/dellemc/
vplex# export ANSIBLE_COLLECTIONS_PATHS=$ANSIBLE_COLLECTIONS_PATHS:/root/
install_collections/collections/
(env_14) root@dev-vm-14:~/install_collection/collections/ansible_collections/dellemc/
vplex# echo $ANSIBLE_COLLECTIONS_PATHS
:/root/install_collections/collections/
(env_14) root@dev-vm-14:~/install_collection/collections/ansible_collections/dellemc/
vplex#

```

## For VPLEX SDK 7.0 (metro node-External host configuration)

The VPLEX API python package is installed on the VPLEX Management Server.

### Install Dell EMC VPLEX python SDK

The client is available from the locations:

1. External clients can download a tarball from <https://<system>/apipackages/python/vplexapi.tgz>.
  - a. Untar the file with the `tar -xvf python-vplex-main.tar` command. It creates a directory that is called "python-vplex-main"
  - b. Export the python path with the vplexapi:
    - i. `export PYTHONPATH="{ $PYTHONPATH }:<path of above untar'd vplexapi>"` **Example:**

```

[root@localhost ~]# export PYTHONPATH="/root/python-vplex-main/vplexapi-7.0.x.x"
[root@localhost ~]# echo $PYTHONPATH
/root/python-vplex-main/vplexapi-7.0.x.x
[root@localhost ~]#

```

**i** **NOTE:** This command works only on the current execution terminal. In order to make it persistent, update the same export command in `$HOME/.bashrc` file followed by system reboot.

- ii. To run the Ansible playbooks, the host server must be configured.
- iii. Do the following before you run playbooks on Ansible modules for Dell EMC VPLEX.

### Steps to install Ansible collections

1. Download the tar build from [Git hub](#), or from [Ansible galaxy portal](#). To install the collection anywhere in the system, follow the command:

```
ansible-galaxy collection install dellemc-vplex-1.3.0.tar.gz -p ./collections
```

2. Set the environment variable:

```
export ANSIBLE_COLLECTIONS_PATHS=$ANSIBLE_COLLECTIONS_PATHS:<install_path>/collections.
```

#### Example:

```

[root@localhost install_collections]# ls
dellemc-vplex-1.3.tar.gz
[root@localhost install_collections]# ansible-galaxy collection install dellemc-
vplex-1.3.tar.gz -p ./collections
Starting galaxy collection install process
[WARNING]: The specified collections path '/root/install_collections/collections' is not
part of the configured Ansible collections paths
'/root/install_collections:/root/collections_testing/install_collections/collections'.
The installed collection won't be picked up in an Ansible run.
Process install dependency map
Starting collection install process
Installing 'dellemc-vplex-1.3.tar.gz' to '/root/install_collections/collections/
ansible_collections/dellemc/vplex'
dellemc.vplex (1.3) was installed successfully
[root@localhost install_collections]# ls

```

```
collections    dellemc-vplex-1.3.tar.gz

[root@localhost install_collections]# ls
collections    dellemc-vplex-1.3.tar.gz
[root@localhost install_collections]# cd collections/ansible_collections/dellemc/vplex/
[root@localhost vplex]# pwd
/root/install_collections/collections/ansible_collections/dellemc/vplex
[root@localhost vplex]# export ANSIBLE_COLLECTIONS_PATHS=$ANSIBLE_COLLECTIONS_PATHS:/
root/install_collections/collections/
[root@localhost vplex]# echo $ANSIBLE_COLLECTIONS_PATHS
:/root/collections_testing/install_collections/collections:/root/install_collections/
collections/
[root@localhost vplex]#
```

## For VPLEX SDK 7.0 (metro node-within management server)

Internal clients can import from `/opt/emc/VPlex/vplexapi` and can run the Ansible playbook from the VPLEX itself. See the following steps:

1. `export PYTHONPATH="/opt/emc/VPlex/vplexapi"`
2. Download the tar build from [Git hub](#), or from [Ansible galaxy portal](#).
3. Copy the .tar file to the VPLEX setup:

```
$ mkdir $HOME/testing_ansible_collections
# copy the .tar file from ansible collections repository to $HOME/
testing_ansible_collections.
```

4. Before running the playbook, ensure that the variable `stdout_callback = debug` is set in `/etc/ansible/ansible.cfg` to enable detailed playbook execution logs.

## Steps to install Ansible collections


1. To install the collection anywhere in the system, follow the command:

```
ansible-galaxy collection install dellemc-vplex-1.3.0.tar.gz -p ./collections
```

2. Set the environment variable:

```
export ANSIBLE_COLLECTIONS_PATHS=$ANSIBLE_COLLECTIONS_PATHS:<install_path>/collections.
```

### Example:

 **NOTE:** This Export command works only on the current execution terminal. Within the VPLEX management server, for each session, the Ansible collections path should be set using the export command before the execution of the playbook.

See the following steps:

```
service@director-1-1-a:~> export PYTHONPATH="/opt/emc/VPlex/vplexapi"
service@director-1-1-a:~> mkdir testing_ansible_collections

<< Copy/download the "dellemc-vplex-x.x.x.tar.gz" into $HOME/testing_ansible_collections
directory >>

service@director-1-1-a:~/testing_ansible_collections> ls
dellemc-vplex-1.3.tar.gz

service@director-1-1-a:~/testing_ansible_collections> ansible-galaxy collection install
dellemc-vplex-1.3.tar.gz -p ./collections
[WARNING]: The specified collections path '/home/service/testing_ansible_collections/
collections' is not part of the configured Ansible collections paths
'/home/service/.ansible/collections:/usr/share/ansible/collections'. The installed
collection won't be picked up in an Ansible run.
Process install dependency map
Starting collection install process
Installing 'dellemc-vplex-1.3.tar.gz' to '/home/service/testing_ansible_collections/
collections/ansible_collections/dellemc/vplex'
service@director-1-1-a:~/testing_ansible_collections> ls
collections    dellemc-vplex-1.3.tar.gz

service@director-1-1-a:~/testing_ansible_collections> cd collections/ansible_collections/
```

```

dell EMC/vplex/docs/samples
service@director-1-1-a:~/testing_ansible_collections/collections/ansible_collections/
dell EMC/vplex/docs/samples> pwd
/home/service/testing_ansible_collections/collections/ansible_collections/dell EMC/vplex/
docs/samples
service@director-1-1-a:~/testing_ansible_collections/collections/ansible_collections/
dell EMC/vplex/docs/samples> export ANSIBLE_COLLECTIONS_PATHS=$ANSIBLE_COLLECTIONS_PATHS:/
home/service/testing_ansible_collections/collections/

service@director-1-1-a:~/testing_ansible_collections/collections/ansible_collections/
dell EMC/vplex/docs/samples> ansible-playbook get_unclaimed_volumes.yml

```

## For VPLEX SDK 6.2

**NOTE:** For the VPLEX setup 6.2, the Ansible scripts can be performed only from a host and not from the VPLEX management server.

### Install Dell EMC VPLEX python SDK

The following are the steps to install VPLEX python SDK 6.2 in the host machine:

1. Download the tar from [Git hub](#) into the corresponding host system (supported OS: RHEL 7.x and RHEL 8.x).
2. Untar the file with the `tar -xvf vplexapi.tgz` command. It creates a directory inside which `python-vplex-main` is created.
3. **NOTE:** Ensure that there is no space after the `=`, else the command will fail.

Export the python path with the `vplexapi`:

- a. `export PYTHONPATH="<path of above untar'd vplexapi-6.2.0.3>".`

#### Example:

**NOTE:** This command works only on the current execution terminal. To make it persistent, update the same export command in `$HOME/.bashrc` file followed by the system reboot.

```

[root@localhost ~]# export PYTHONPATH="/root/python-vplex-main/vplexapi-6.2.0.3"
[root@localhost ~]# echo $PYTHONPATH
/root/python-vplex-main/vplexapi-6.2.0.3
[root@localhost ~]#

```

4. To run the Ansible playbooks, the host server must be configured.

### Steps to install Ansible collections

1. Download the tar build from [Git hub](#), or from [Ansible galaxy portal](#). To install the collection anywhere in the system, follow the command:

```
ansible-galaxy collection install dell EMC-vplex-1.3.0.tar.gz -p ./collections
```

2. Set the environment variable:

```
export ANSIBLE_COLLECTIONS_PATHS=$ANSIBLE_COLLECTIONS_PATHS:<install_path>/collections
```

#### Example:

```

[root@localhost install_collections]# ls
dell EMC-vplex-1.3.tar.gz
[root@localhost install_collections]# ansible-galaxy collection install dell EMC-
vplex-1.3.tar.gz -p ./collections
Starting galaxy collection install process

Process install dependency map
Starting collection install process
Installing 'dell EMC.vplex:1.3' to '/root/install_collections/collections/
ansible_collections/dell EMC/vplex'

```

```

dellemc.vplex (1.3) was installed successfully
[root@localhost install_collections]# ls
collections  dellemc-vplex-1.3.tar.gz

[root@localhost install_collections]# ls
collections  dellemc-vplex-1.3.tar.gz
[root@localhost install_collections]# cd collections/ansible_collections/dellemc/vplex/
[root@localhost vplex]# pwd
/root/install_collections/collections/ansible_collections/dellemc/vplex
[root@localhost vplex]# export ANSIBLE_COLLECTIONS_PATHS=$ANSIBLE_COLLECTIONS_PATHS:/
root/install_collections/collections/
[root@localhost vplex]# echo $ANSIBLE_COLLECTIONS_PATHS
:/root/collections_testing/install_collections/collections:/root/install_collections/
collections/
[root@localhost vplex]#

```

## Steps to retrieve CA certificate from VPLEX

This section is optional. If the customer wants to set **verifycert** attribute to **True** in the playbook, then the following steps must be performed:

### For VPLEX SDK 9.0 (metro node-External host configuration)

To retrieve CA certificate from VPLEX and copy it to the Ansible host machine, follow these steps:

1. Log in to VPLEX CLI with the valid credentials.
2. Run the `ls` command in `"/home/service"` to locate `ssl.pem`.
3. Copy the file `ssl.pem` into the Ansible host machine: `scp ssl.pem user@<ansible-host>:/execution_directory_path`.

#### Example:

```

service@director-1-1-a:~> ls
bin  ssl.p12  ssl.pem  tools
service@director-1-1-a:~> ll
total 24
drwxr-x--- 3 service users 4096 Nov 19 12:34 bin
-rw----- 1 service users 3990 Nov 19 12:36 ssl.p12
-rw-r----- 1 service users 8236 Nov 19 12:36 ssl.pem
drwxr-x--- 3 service users 4096 Nov 24 06:46 tools
service@director-1-1-a:~> scp ssl.pem root@10.226.81.252:/root/
root@10.226.81.252's password:
ssl.pem
service@director-1-1-a:~>

```

### For VPLEX SDK 7.0 (metro node-within management server)

To retrieve CA certificate from VPLEX for VPLEX management server, follow these steps:

1. Log in to VPLEX CLI with the valid credentials.
2. Run the `ls` command and find the `ssl.pem`.
3. Specify the path of the file including the filename in the playbook for `ssl_ca_cert` variable, and run the playbook.

#### Example:

```

service@director-1-1-a:~> ls
bin  ssl.p12  ssl.pem  tools
service@director-1-1-a:~> ll
total 24
drwxr-x--- 3 service users 4096 Nov 19 12:34 bin
-rw----- 1 service users 3990 Nov 19 12:36 ssl.p12
-rw-r----- 1 service users 8236 Nov 19 12:36 ssl.pem

```

```
drwxr-x--- 3 service users 4096 Nov 24 06:46 tools
```

```
<< Specify the file path including the filename for "ssl_ca_cert" variable along with  
"verifycert" set to True in the ansible playbook to be executed >>
```

```
service@director-1-1-a:~> ansible-playbook gatherfacts.yml
```

```
# Gather Facts
```

```
---
```

```
- name: List the storage objects of VPLEX
```

```
  hosts: localhost
```

```
  connection: local
```

```
  vars:
```

```
    # Variable parameters
```

```
    vplexhost: <*****>
```

```
    vplexuser: <*****>
```

```
    vplexpassword: <*****>
```

```
    cluster_name: 'cluster-1'
```

```
    # Constant parameters
```

```
    verifycert: true
```

```
    ssl_ca_cert: '/home/service/ssl.pem'
```

```
collections:
```

```
  - dellemc.vplex
```

```
tasks:
```

```
  # This task returns the list of all clusters present in VPLEX.
```

```
  - name: List of clusters in VPLEX
```

```
    dellemc_vplex_gatherfacts:
```

```
      vplexhost: "{{ vplexhost }}"
```

```
      vplexuser: "{{ vplexuser }}"
```

```
      vplexpassword: "{{ vplexpassword }}"
```

```
      verifycert: "{{ verifycert }}"
```

```
      ssl_ca_cert: "{{ ssl_ca_cert }}"
```

```
    register: clusters
```

```
  # This task displays the clusters name.
```

```
  - debug:
```

```
    msg: "{{ clusters['Clusters'] }}"
```

```
  # This task returns the list of all storage objects present in a cluster.
```

```
  - name: List of all storage objects in a given cluster
```

```
    dellemc_vplex_gatherfacts:
```

```
      vplexhost: "{{ vplexhost }}"
```

```
      vplexuser: "{{ vplexuser }}"
```

```
      vplexpassword: "{{ vplexpassword }}"
```

```
      verifycert: "{{ verifycert }}"
```

```
      ssl_ca_cert: "{{ ssl_ca_cert }}"
```

```
      cluster_name: "{{ cluster_name }}"
```

```
    gather_subset:
```

```
      - stor_array
```

```
      - stor_vol
```

```
      - be_port
```

```
      - port
```

```
      - initiator
```

```
      - stor_view
```

```
      - virt_vol
```

```
      - cg
```

```
      - device
```

```
      - extent
```

```
      - dist_device
```

```
      - dist_cg
```

```
      - dist_virt_vol
```

```
      - device_mig_job
```

```
      - extent_mig_job
```

```
      - amp
```

```
    register: vplex_object
```

```
  # List the storage object names present in the cluster
```

```
  - debug:
```

```
    msg: "{{ vplex_object }}"
```

```
# This task returns the list of storage volumes that are unclaimed
- name: List of all storage volumes that are unclaimed in a given cluster
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "{{ cluster_name }}"
    ssl_ca_cert: "{{ ssl_ca_cert }}"
    gather_subset:
      - stor_vol
    filters:
      - filter_key: "use"
        filter_operator: "equal"
        filter_value: "unclaimed"
    register: volume

# List the storage volume names present in the cluster
- debug:
    msg: "{{ volume }}"
```

## For VPLEX SDK 6.2

To retrieve CA certificate from VPLEX and copy it to the Ansible host machine, follow these steps:

1. Log in to VPLEX CLI with the valid credentials.
2. `cd /etc/ipsec.d/cacerts`
3. Copy the file `strongswanCert.pem` into the Ansible host machine: `scp -r strongswanCert.pem user@ansible-host>:/execution_directory_path.`

### Example:

```
service@satellite-1:~> cd /etc/ipsec.d/cacerts/
service@satellite-1:/etc/ipsec.d/cacerts> ll
total 4
-rw-rw---- 1 root groupSvc 1655 Mar 14 2020 strongswanCert.pem
service@satellite-1:/etc/ipsec.d/cacerts> scp -r strongswanCert.pem
root@10.227.50.57:/root/
root@10.227.50.57's password:
strongswanCert.pem
service@satellite-1:/etc/ipsec.d/cacerts>
```

## Steps to run playbooks in Ansible collections

To use any Ansible module, ensure that the importing of proper Fully Qualified Collection Name (FQCN) must be embedded in the playbook.

See the following example:

```
collections:
  - dellemc.vplex
```

### Example:

```
# Rediscover StorageArray

---
- name: Rediscover StorageArray Tests
  hosts: localhost
  connection: local
  vars:
    # Variable parameters
    vplexhost: <*****>
    vplexuser: <*****>
    vplexpassword: <*****>
```



```
cluster_name: "cluster-1"
# Constant parameters
verifycert: false
array_name: "array_name"

collections:
  - dellemc.vplex

tasks:
  # Get StorageArray
  - name: Get StorageArray
```

For generating Ansible documentation for a specific module, embed the FQCN before the module name. See the following example:

```
ansible-doc dellemc.vplex.dellemc_vplex_gatherfacts
```

# Ansible modules for Dell EMC VPLEX

This chapter contains the following topics:

## Topics:

- [Gather Facts module](#)
- [Storage volume module](#)
- [Extent module](#)
- [Device module](#)
- [Distributed device module](#)
- [Virtual volume module](#)
- [Distributed virtual volume module](#)
- [Consistency group module](#)
- [Distributed consistency group module](#)
- [Port module](#)
- [Initiator module](#)
- [Storage View module](#)
- [Data migration module](#)
- [Rediscover array module](#)
- [Maps module](#)

## Gather Facts module

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

The gather facts module helps to access the inventory of Dell EMC storage objects.

As an improvement, the filtering mechanism is implemented in the gather facts module so that the user can provide a specific key, value, and operator for obtaining filtered details of the storage objects. For the valid `filter_key`, see the RESTAPI guide, and the fields under the response of each storage entity are the supported `filter_keys`.

Objects in the inventory include:

- Storage Arrays
- Storage Volumes
- Extents
- Devices
- Distributed Devices
- Virtual Volumes
- Distributed Virtual Volumes
- Consistency Groups
- Distributed Consistency Groups
- Ports
- BE Ports
- Initiators
- Storage Views
- Device migration jobs
- Extent migration jobs
- Array Management Providers (AMP)

## Get list of storage arrays

To get the list of connected storage arrays from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

To get list of all storage array in a given cluster, storage arrays should be present in VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Storage Arrays
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - stor_array
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in subset and the filter keys, then a list of storage array names is listed-Success with "changed": False. Got storage arrays from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for storage array-Failure with "changed": False. Execution fails with the error message stating as "Could not get storage arrays".

## Get list of storage volumes

To get the list of storage volumes from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

To get list of all storage volumes in a given cluster, storage volumes should be present in VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Storage Volumes
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - stor_vol
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in subset and the filter keys, then a list of storage volume names is listed-Success with "changed": False. Got storage volumes from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for storage volume-Failure with "changed": False. Execution fails with the error message stating as "Could not get storage volumes".

## Get list of storage volumes with filters (operator: equal)

The user can get the list of unclaimed storage volumes with filters (operator: equal).

To get list of all storage volumes using the filter operator from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

To get unclaimed storage volumes in a given cluster, unclaimed storage volumes should be present in the VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of unclaimed storage volumes
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - stor_vol
    filters:
      - filter_key: use
        filter_operator: "equal"
        filter_value: "unclaimed"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in subset and the filter keys, then a list of storage volume names is listed-Success with "changed": False. Got unclaimed storage volumes from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for storage volume-Failure with "changed": False. Execution fails with the error message stating as "Could not get storage volumes".

## Get list of extents

To get the list of extent from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

To get list of all extent in a given cluster, extent should be present in the VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Extents
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - extent
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in subset and the filter keys, then a list of extent is listed-Success with "changed": False. Got extents from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for extent-Failure with "changed": False. Execution fails with the error message stating as "Could not get extents".

## Get list of devices

To get the list of devices from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

To get list of all devices in a given cluster, devices should be present in the VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Devices
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - device
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in subset and the filter keys, then a list of devices is listed-Success with "changed": False. Got devices from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for device-Failure with "changed": False. Execution fails with the error message stating as "Could not get devices".

## Get list of distributed devices

To get the list of distributed devices present in the VPLEX metro setup, run the appropriate playbook.

### Prerequisite

To get list of all distributed devices in a given cluster, it should be present in the VPLEX setup. The cluster name is not required for distributed entities.

The syntax of the task is as follows:

```
- name: Get list of Distributed Devices
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    gather_subset:
      - dist_device
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in subset and the filter keys, then a list of distributed devices is listed-Success with "changed": False. Got distributed device details from VPLEX.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for distributed device-Failure with "changed": False. Execution fails with the error message stating as "Could not get distributed devices".

## Get list of virtual volumes

To get the list of virtual volumes from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

To get list of all virtual volumes in a given cluster, virtual volumes should be present in VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Virtual Volumes
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - virt_vol
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

1. If there is a match in subset and the filter keys, then a list of virtual volumes is listed-Success with "changed": False. Got virtual volumes from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for virtual volume-Failure with "changed": False. Execution fails with the error message stating as "Could not get virtual volumes".

## Get list of virtual volumes with filters (operator: greater and lesser)

The user can get the list of virtual volumes within the specified size range using filters (operator: greater and lesser).

#### Prerequisite

To get list of all virtual volumes in a given cluster, virtual volumes should be present in VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of virtual volumes within the specified size range
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - virt_vol
    filters:
      - filter_key: capacity
        filter_operator: "greater"
        filter_value: "10GB"
      - filter_key: capacity
        filter_operator: "lesser"
        filter_value: "50GB"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

1. If there is a match in subset and the filter keys, then a list of virtual volumes is listed-Success with "changed": False. Got virtual volumes from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for virtual volume-Failure with "changed": False. Execution fails with the error message stating as "Could not get virtual volumes".

## Get list of distributed virtual volumes

To get the list of distributed virtual volumes present in the VPLEX metro setup, run the appropriate playbook.

### Prerequisite

To get list of all distributed virtual volumes in a given cluster, it should be present in the VPLEX setup. The cluster name is not required for distributed entities.

The syntax of the task is as follows:

```
- name: Get list of Distributed Virtual Volumes
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    gather_subset:
      - dist_virt_vol
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in subset and the filter keys, then a list of distributed virtual volumes is listed-Success with "changed": False. Got distributed virtual volumes from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for distributed virtual volume-Failure with "changed": False. Execution fails with the error message stating as "Could not get distributed virtual volumes".

## Get list of consistency groups

To get the list of consistency groups from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

To get list of all consistency groups in a given cluster, consistency groups should be present in VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Consistency Groups
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - cg
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in subset and the filter keys, then a list of consistency groups is listed-Success with "changed": False. Got consistency groups from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for consistency group-Failure with "changed": False. Execution fails with the error message stating as "Could not get consistency groups".

## Get list of distributed consistency groups

To get the list of distributed consistency groups present in the VPLEX metro setup, run the appropriate playbook.

### Prerequisite

To get list of all distributed consistency groups in a given cluster, it should be present in the VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Distributed Consistency Groups
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    gather_subset:
      - dist_cg
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

1. If there is a match in subset and the filter keys, then a list of distributed consistency groups is listed-Success with "changed": False. Got distributed consistency group details from VPLEX.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for distributed consistency group-Failure with "changed": False. Execution fails with the error message stating as "Could not get distributed consistency groups".

## Get list of ports

To get the list of front end ports from the specific VPLEX cluster, run the appropriate playbook.

#### Prerequisite

To get list of all ports in a given cluster, ports should be present in the VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Ports
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - port
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

1. If there is a match in subset and the filter keys, then a list of ports is listed-Success with "changed": False. Got ports from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for ports-Failure with "changed": False. Execution fails with the error message stating as "Could not get ports".

## Get list of BE ports

To get the list of back end ports from the specific VPLEX cluster, run the appropriate playbook.

#### Prerequisite

To get list of all back end ports in a given cluster, back end ports should be present in VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Back End Ports
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
```



```

vpsexpassword: "{{ vplexpassword }}"
verifycert: "{{ verifycert }}"
cluster_name: "cluster-1"
gather_subset:
  - be_port

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in subset and the filter keys, then a list of BE ports is listed-Success with "changed": False. Got back end ports from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for BE ports-Failure with "changed": False. Execution fails with the error message stating as "Could not get BE ports".

## Get list of initiators

To get the list of initiator ports from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

The initiator should be present in VPLEX setup.

The syntax of the task is as follows:

```

- name: Get list of Initiators
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - initiator

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

With the appropriate playbook syntax, on trying to run the playbook, the following are the expected output of the initiators:

1. If there is a match in the subset and the filter keys, then a list of initiators is listed -Success with "changed":False. Got the initiator details.
2. If there is no match in subset and filter keys, an empty list is returned-Success with "changed":False. Return empty list.
3. If the `filter_key` is not supported for initiators-Failure with "changed": False. Execution fails with an error message stating that "Could not get the initiators".

## Get list of storage views

To get the list of storage views from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

To get list of all storage views in a given cluster, storage views should be present in VPLEX setup.

The syntax of the task is as follows:

```

- name: Get list of Storage Views
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"

```

```
gather_subset:
  - stor_view
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

1. If there is a match in subset and the filter keys, then a list of storage views is listed-Success with "changed": False. Got storage views from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for storage views-Failure with "changed": False. Execution fails with the error message stating as "Could not get storage views".

## Get list of device migration jobs

To get the list of device migration jobs present in VPLEX metro setup, run the appropriate playbook.

#### Prerequisite

To get list of all device migration jobs in a given cluster, it should be present in the VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Device migration jobs
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    gather_subset:
      - device_mig_job
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

1. If there is a match in subset and the filter keys, then a list of device migration jobs is listed-Success with "changed": False. Got device migration job details from VPLEX.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for device migration jobs-Failure with "changed": False. Execution fails with the error message stating as "Could not get device migration jobs".

## Get list of extent migration jobs

To get the list of extent migration jobs present in VPLEX metro setup, run the appropriate playbook.

#### Prerequisite

To get list of all extent migration jobs in a given cluster, it should be present in VPLEX setup.

The syntax of the task is as follows:

```
- name: Get list of Extent migration jobs
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    gather_subset:
      - extent_mig_job
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

1. If there is a match in subset and the filter keys, then a list of extent migration jobs is listed-Success with "changed": False. Got extent migration job details.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for extent migration jobs-Failure with "changed": False. Execution fails with the error message stating as "Could not get extent migration jobs".

## Get list of array management providers

To get the list of array management providers (AMPs) from the specific VPLEX cluster, run the appropriate playbook.

### Prerequisite

To get list of all AMPs in a given cluster, AMPs should be present in the VPLEX setup.


The syntax of the task is as follows:

```
- name: Get list of Array Management Provider
  dellemc_vplex_gatherfacts:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    gather_subset:
      - amp
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If there is a match in the subset and the filter keys, then a list of array management providers is listed-Success with "changed": False. Got AMPs from cluster-1.
2. If there is no match in subset and the filter keys, an empty list is returned-Success with "changed": False. Returns empty list.
3. If the `filter_key` is not supported for AMPs-Failure with "changed": False. Execution fails with the error message stating as "Could not get array management providers".

 **NOTE:** AMP is supported only in the Vplex 6.0 and its minor versions and it is not supported in the Metro Node 7.0 and above.

## Gather Facts module parameters

The following table provides information about the gather facts module parameters with the examples:

**Table 3. Gather facts module parameters**

Parameter name	Choices or Default	Type	Mandatory/Optional Parameter	Description
vplexhost	N/A	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	N/A	str	Mandatory	The username to access the VPLEX server.
vplexpassword	N/A	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"> <li>• True - Verifies the SSL certificate</li> <li>• False - Specified that the SSL certificate should not be verified.</li> </ul>
ssl_ca_cert	N/A	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dellemc_ansible_vplex.log).

**Table 3. Gather facts module parameters (continued)**

Parameter name	Choices or Default	Type	Mandatory/Optional Parameter	Description
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	N/A	str	Optional	<p>Name of the cluster.</p> <p><b>NOTE:</b> The <code>cluster_name</code> is not required for Distributed Devices, Distributed Virtual Volumes, Distributed Consistency Groups, Extent Migrations, and Device Migrations.</p> <p>If the user does not specify the <code>cluster_name</code> for the storage elements, excluding for the above specified distributed entries the gather facts module returns basic information of clusters.</p>
gather_subset	<ul style="list-style-type: none"> <li>• stor_array</li> <li>• stor_vol</li> <li>• port</li> <li>• be_port</li> <li>• initiator</li> <li>• stor_view</li> <li>• virt_vol</li> <li>• cg</li> <li>• device</li> <li>• extent</li> <li>• dist_device</li> <li>• dist_cg</li> <li>• dist_virt_vol</li> <li>• device_mig_job</li> <li>• extent_mig_job</li> <li>• amp</li> </ul>	array	Optional	<p>List of string variables to specify the VPLEX entities for which the information is required. If <code>gather_subset</code> is not provided, the gather facts module returns list of clusters.</p> <ul style="list-style-type: none"> <li>• stor_array - storage arrays</li> <li>• stor_vol - storage volumes</li> <li>• port - ports</li> <li>• be_port - back end ports</li> <li>• initiator - initiators</li> <li>• stor_view - storage views</li> <li>• virt_vol - virtual volumes (local)</li> <li>• cg - consistency groups (local)</li> <li>• device - devices (local)</li> <li>• extent - extents</li> <li>• dist_device - distributed devices</li> <li>• dist_cg - distributed consistency groups</li> <li>• dist_virt_vol - distributed virtual volumes</li> <li>• device_mig_job - device migration jobs</li> <li>• extent_mig_job - extent migration jobs</li> <li>• amp - array management providers</li> </ul>
filters <ul style="list-style-type: none"> <li>• filter_key: &lt;str&gt;</li> <li>• filter_operator: &lt;str&gt;</li> <li>• filter_value: &lt;str&gt;</li> </ul>	N/A	List of dictionaries	Optional	<p>The 'filters' is a list of dictionaries. Each element of the list is a dictionary which has the following three keys:</p> <ul style="list-style-type: none"> <li>• filter_key: &lt;str&gt; value is the response key</li> <li>• filter_operator: &lt;str&gt; any value from the supported list of operators given.               <ol style="list-style-type: none"> <li>1. greater</li> <li>2. lesser</li> <li>3. equal</li> <li>4. greater-equal</li> <li>5. lesser-equal</li> </ol> </li> <li>• filter_value: &lt;str&gt; value</li> </ul> <p><b>NOTE:</b> For valid <code>filter_key</code>, see the RESTAPI guide for the supported fields that can be determined from the response of each storage entity. For example, in storage volume, use, capacity, thin_capable, name, sort_by, and so on are some of the supported properties for fetching the storage volume.</p>

## Sample output

### Without filtering

```
[root@localhost playbooks]# ansible-playbook dellemc_vplex_gatherfacts_tests.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [List the storage objects of VPLEX]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [List of clusters in VPLEX]
*****
ok: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "msg": [
    "cluster-1",
    "cluster-2"
  ]
}

TASK [List of all storage objects in a given cluster]
*****
ok: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "msg": {
    "ArrayManagementProviders": [],
    "BackEndPorts": [
      {
        "address": "0xc001445a80e00800",
        "director": "director-1-1-A",
        "name": "IO-02",
        "role": "back-end",
        "status": "up"
      },
      {
        "address": "0xc001445a80e00900",
        "director": "director-1-1-A",
        "name": "IO-03",
        "role": "back-end",
        "status": "up"
      },
      {
        "address": "0xc001445a80e10800",
        "director": "director-1-1-B",
        "name": "IO-02",
        "role": "back-end",
        "status": "up"
      },
      {
        "address": "0xc001445a80e10900",
        "director": "director-1-1-B",
        "name": "IO-03",
        "role": "back-end",
        "status": "up"
      }
    ]
  }
}
```

```

    },
    {
        "address": "0xc001445a80e20800",
        "director": "director-2-1-A",
        "name": "IO-02",
        "role": "back-end",
        "status": "up"
    },
    {
        "address": "0xc001445a80e20900",
        "director": "director-2-1-A",
        "name": "IO-03",
        "role": "back-end",
        "status": "up"
    },
    {
        "address": "0xc001445a80e30800",
        "director": "director-2-1-B",
        "name": "IO-02",
        "role": "back-end",
        "status": "up"
    },
    {
        "address": "0xc001445a80e30900",
        "director": "director-2-1-B",
        "name": "IO-03",
        "role": "back-end",
        "status": "up"
    }
},
"ConsistencyGroups": [],
"DeviceMigrationJob": [
    "D_x0144_1_1__196_1_77",
    "dev_inter_job_1"
],
"Devices": [
    "ADtestuser01_1",
    "C1_Local_00",
    "C1_Local_01",
    "MIGRATE_D_x0144_1_1__196_1_77",
    "add_test_1",
    "ansible_virt_vol_dev",
    "dev_ansible_demo_3"
],
"DistributedConsistencyGroups": [
    "DR_CG_002",
    "DR_CG_001",
    "ansible_test_cg"
],
"DistributedDevices": [
    "DR1_10GB_003",
    "DR1_10GB_004",
    "DR1_10GB_005",
    "DR1_10GB_006"
],
"DistributedVirtualVolumes": [
    "DR1_10GB_003_vol",
    "DR1_10GB_004_vol",
    "DR1_10GB_005_vol",
    "DR1_10GB_006_vol"
],
"Extents": [
    "ext_log_test_new",
    "extent_68ccf098007c0b818e44aa65920f1445_1",
    "extent_68ccf098007ca5a21c8dbfcafff778ea3_1",
    "extent_68ccf098007db6ec6c27d0208576922e_1",
    "extent_68ccf098007e43edde32fb509bfe6ee6_1",
    "extent_68ccf098007f53eca31f915947b7e274_1",
    "extent_68ccf098007fd6d3757fd24d264f8486_1",
    "extent_68ccf09800807ad45dc927417189cf84_1",
    "extent_prov_cluster-1_0",
    "extent_prov_cluster-1_1",
    "extent_ps_cluster-1_0",

```

```

        "extent_ps_cluster-1_1",
        "extent_sv_1-1_0",
        "extent_sv_1606207897_0",
        "extent_sv_1606207897_1",
        "extent_sv_2_1",
        "extent_sv_cluster-1_0",
        "extent_sv_cluster-1_1"
    ],
    "Initiators": [
        {
            "name": "RHEL-dsveg092",
            "type": "default"
        },
        {
            "name": "ansible-init1",
            "type": "default"
        },
        {
            "name": "dsveg165Rhel02",
            "type": "default"
        },
        {
            "name": "ansible-init2",
            "type": "default"
        },
        {
            "name": "dsveg165Rhel01",
            "type": "default"
        },
        {
            "name": "UNREGISTERED-0x10000000c9b82e35"
        }
    ],
    "Ports": [
        "P0000000002D6000E0-IO-00",
        "P0000000002D6000E1-IO-00",
        "P0000000002D6000E0-IO-01",
        "P0000000002D6000E1-IO-01"
    ],
    "StorageArrays": [
        "DellEMC-PowerStore-4PGJBX2",
        "DellEMC-PowerStore-4PFLBX2"
    ],
    "StorageViews": [
        "Dsveg165Rhel",
        "rhel-dsveg092",
        "ansible-storview"
    ],
    "StorageVolumes": [
        "DellEMC-PowerStore-4PFLBX2_LUN_0x0101",
        "DellEMC-PowerStore-4PFLBX2_LUN_0x0103",
        "DellEMC-PowerStore-4PFLBX2_LUN_0x0104",
        "DellEMC-PowerStore-4PFLBX2_LUN_0x0107",
        "DellEMC-PowerStore-4PFLBX2_LUN_0x0108",
        "DellEMC-PowerStore-4PFLBX2_LUN_0x0109",
        "DellEMC-PowerStore-4PFLBX2_LUN_0x010a",
        "DellEMC-PowerStore-4PFLBX2_LUN_0x010c",
        "DellEMC-PowerStore-4PFLBX2_LUN_0x010d",
        "sv_1606216152_cluster-1_1",
        "sv_1606817471_cluster-1_0",
        "sv_1606817471_cluster-1_1"
    ],
    "VirtualVolumes": [
        "device_DellEMC-PowerStore-4PFLBX2_LUN_0x0147_1_2_vol",
        "device_DellEMC-PowerStore-4PFLBX2_LUN_0x014c_1_4_vol",
        "device_DellEMC-PowerStore-4PFLBX2_LUN_0x0150_1_5_vol",
        "device_DellEMC-PowerStore-4PFLBX2_LUN_0x0151_1_6_vol",
        "device_DellEMC-PowerStore-4PFLBX2_LUN_0x0152_1_7_vol",
        "device_DellEMC-PowerStore-4PFLBX2_LUN_0x0153_1_8_vol",
        "device_DellEMC-PowerStore-4PFLBX2_LUN_0x0155_1_9_vol",
        "device_DellEMC-PowerStore-4PFLBX2_LUN_0x0157_1_10_vol",
        "rh92-50GB_25_vol",
        "rh92-50GB_2_vol",
    ]
}

```

```

        "rh92-50GB_3_vol",
        "rh92-50GB_4_vol"
    ],
    "changed": false,
    "failed": false
}
}

PLAY RECAP
*****
*****
localhost      : ok=5    changed=0    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## With filtering

```

(py3_ans_10) [root@localhost gf]# ansible-playbook dellemc_vplex_gatherfacts_tests.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [List the storage objects of VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [List of all storage volumes that are unclaimed in a given cluster]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "msg": {
    "ArrayManagementProviders": [],
    "BackEndPorts": [],
    "ConsistencyGroups": [],
    "DeviceMigrationJobs": [],
    "Devices": [],
    "DistributedConsistencyGroups": [],
    "DistributedDevices": [],
    "DistributedVirtualVolumes": [],
    "ExtentMigrationJobs": [],
    "Extents": [],
    "Initiators": [],
    "Ports": [],
    "StorageArrays": [],
    "StorageViews": [],
    "StorageVolumes": [
      "VPD83T3:60000970000197700282533030363730",
      "VPD83T3:60000970000197700282533030363731",
      "VPD83T3:60000970000197700282533030363732",
      "VPD83T3:60000970000197700282533030363733",
      "VPD83T3:60000970000197700282533030363734",
      "VPD83T3:60000970000197700282533030394331",
      "VPD83T3:60000970000197700282533030394332",
      "VPD83T3:60000970000197700282533030394333",
      "VPD83T3:60000970000197700282533030394334",
      "VPD83T3:60000970000197700282533030394335",
      "VPD83T3:60000970000197700282533030394338",
      "VPD83T3:60000970000197700282533030394339",
      "VPD83T3:60000970000197900206533032394638",
      "VPD83T3:60000970000197900206533032394639",
      "VPD83T3:60000970000197900206533032394641",
      "VPD83T3:60000970000197900206533032394642",
      "VPD83T3:60000970000197900206533032394643",
      "VPD83T3:60000970000197900206533032394644",
      "VPD83T3:60000970000197900206533032394645",
    ]
  }
}

```



```

        "VPD83T3:60000970000197900206533032394646",
        "VPD83T3:60000970000197900206533032413030",
        "VPD83T3:60000970000197900206533032413031",
        "VPD83T3:60000970000197900206533032413032",
        "VPD83T3:60000970000197900206533032413033",
        "VPD83T3:60000970000197900206533032413034",
        "VPD83T3:60000970000197900206533032413035",
        "VPD83T3:60000970000197900206533032413036",
        "VPD83T3:60000970000197900206533032413037",
        "VPD83T3:60000970000197900206533032413038",
        "VPD83T3:60000970000197900206533032413041",
        "VPD83T3:60000970000197900206533032413042",
        "VPD83T3:60000970000197900206533032413043",
        "VPD83T3:60000970000197900206533032413044",
        "VPD83T3:60000970000197900206533032413045",
        "VPD83T3:60000970000197900206533032413046",
        "VPD83T3:60000970000197900206533032413130",
        "VPD83T3:60000970000197900206533032413131",
        "VPD83T3:60000970000197900206533032413132",
        "VPD83T3:60000970000197900206533032413133",
        "VPD83T3:60000970000197900206533032413134",
        "VPD83T3:60000970000197900206533032413135",
        "VPD83T3:60000970000197900206533032413136",
        "VPD83T3:60000970000197900206533032413137",
        "VPD83T3:60000970000197900206533032413138",
        "VPD83T3:60000970000197900206533032413139",
        "VPD83T3:60000970000197900206533032413141",
        "VPD83T3:60000970000197900206533032413142"
    ],
    "VirtualVolumes": [],
    "changed": false,
    "failed": false
}
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=0    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Storage volume module

The storage volume module manages the storage volumes in the VPLEX.

The module has the following capabilities:

- Claim storage volume
- Unclaim storage volume
- Rename storage volume
- Set thin rebuild
- List ITLs

## Claim storage volume

To claim volume, run appropriate playbook.

### Claim Storage Volume by name

#### Prerequisite

To claim the storage volume in a given cluster, storage volume name should be present in the VPLEX setup in unclaimed state.

The syntax of task is shown as follows:

```
- name: Claim Storage Volume
  dell EMC_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_name: "ansible_stor_vol"
    claimed_state: "claimed"
    state: "present"
```

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Success with "changed": True. Display the storage volume details about the use status as claimed along with other properties.
2. If the storage volume is present as claimed - Success with "changed": False. Display the storage volume details about the use status as claimed along with other properties.
3. If the storage volume is present as used - Success with "changed": False. Display the storage volume details about the use status as used along with other properties.
4. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol\_name> from the specific cluster".

## Claim Storage Volume by ID

### Prerequisite

To claim the storage volume in a given cluster, storage volume ID should be present in the VPLEX setup in unclaimed state.

The syntax of task is shown as follows:

```
- name: Claim Storage Volume
  dell EMC_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_id: "VPD83T3:68ccf098009d68af56e98e31d8c8fd84"
    claimed_state: "claimed"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Success with "changed": True. Display the storage volume details about the use status as claimed along with other properties.
2. If the storage volume is present as claimed - Success with "changed": False. Display the storage volume details about the use status as claimed along with other properties.
3. If the storage volume is present as used - Success with "changed": False. Display the storage volume details about the use status as used along with other properties.
4. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol\_id> from the specific cluster".

## Unclaim storage volume

To unclaim volume, run appropriate playbook.

## Unclaim Storage Volume by name

### Prerequisite

To unclaim the storage volume in a given cluster, storage volume name should be present in the VPLEX setup in claimed state.

The syntax of task is shown as follows:

```
- name: Unclaim Storage Volume
  dellemc_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_name: "ansible_stor_vol"
    claimed_state: unclaimed
    state: present
```

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Success with "changed": False. Display the storage volume details about the use status as unclaimed along with other properties.
2. If the storage volume is present as claimed - Success with "changed": True. Display the storage volume details about the use status as unclaimed along with other properties.
3. If the storage volume is present as used - Failure with "changed": False. Exits with the failure message stating as "Could not unclaim the storage volume <storage\_vol\_name> from the specific cluster, as volume is not claimed".
4. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol\_name> from the specific cluster".

## Unclaim Storage Volume by ID

### Prerequisite

To unclaim the storage volume in a given cluster, storage volume ID should be present in the VPLEX setup in claimed state.

The syntax of task is shown as follows:

```
- name: Unclaim Storage Volume
  dellemc_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_id: "VPD83T3:68ccf098009d68af56e98e31d8c8fd84"
    claimed_state: "unclaimed"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Success with "changed": False. Display the storage volume details about the use status as unclaimed along with other properties.
2. If the storage volume is present as claimed - Success with "changed": True. Display the storage volume details about the use status as unclaimed along with other properties.
3. If the storage volume is present as used - Failure with "changed": False. Exits with the failure message stating as "Could not unclaim the storage volume <storage\_vol\_name> from the specific cluster, as volume is not claimed".
4. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol\_id> from the specific cluster".

## Update or modify storage volume

To rename the storage volume, run appropriate playbook.

### Update or modify storage volume by name

#### Prerequisite

To update the storage volume in a given cluster, storage volume name should be present in the VPLEX setup in claimed state.

The syntax of task is shown as follows:

```
- name: Update Storage Volume
  dell EMC_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_name: "ansible_stor_vol"
    new_storage_volume_name: "new_ansible_st_name_vol"
    claimed_state: "claimed"
    state: "present"
```

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Failure with "changed": False. Exits with the failure message stating as "Unclaimed Storage volume can not be renamed".
2. If the storage volume is present as claimed or used - Success with "changed": True. Display the storage volume details.
3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol\_name> from the specific cluster".

### Update or modify volume by ID

#### Prerequisite

The syntax of task is shown as follows:

To update the storage volume in a given cluster, storage volume ID should be present in the VPLEX setup in claimed state.

```
- name: Update Storage Volume
  dell EMC_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_id: "VPD83T3:68ccf098009d68af56e98e31d8c8fd84"
    new_storage_volume_name: "new_ansible_st_id_vol"
    claimed_state: claimed
    state: present
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Failure with "changed": False. Exits with the failure message stating as "Unclaimed Storage volume can not be renamed".
2. If storage volume is present as claimed or used - Success with "changed": True. Display the storage volume details.
3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol\_id> from the specific cluster".

## Set thin rebuild

To enable thin\_rebuild of storage volume, run appropriate playbook.

### Set thin rebuild to true by name

#### Prerequisite

To set thin\_rebuild the storage volume in a given cluster, storage volume name should be present in VPLEX setup in claimed state.

The syntax of task is shown as follows:

```
- name: Set thin rebuild Storage Volume
  dell EMC_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_name: "ansible_stor_vol"
    thin_rebuild: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Failure with "changed": False. Exits with the failure message stating as "Could not update thin\_rebuild for <storage\_vol> in specific cluster as it is unclaimed".
2. If the storage volume is present as claimed or used - Success with "changed": True. Display the storage volume details.
3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol> from the specific cluster".

### Set thin rebuild to true by ID

#### Prerequisite

To set thin\_rebuild the storage volume in a given cluster, storage volume ID should be present in the VPLEX setup in claimed state.

The syntax of task is shown as follows:

```
- name: Set thin rebuild Storage Volume
  dell EMC_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_id: "VPD83T3:68ccf098009d68af56e98e31d8c8fd84"
    thin_rebuild: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Failure with "changed": False. Exits with the failure message stating as "Could not update thin\_rebuild for <storage\_vol> in specific cluster as it is unclaimed".
2. If the storage volume is present as claimed or used - Success with "changed": True. Display the storage volume details.

3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol> from the specific cluster".

## Set thin rebuild to false by name

### Prerequisite

To set thin\_rebuild to false the storage volume in a given cluster, storage volume name should be present in VPLEX setup in claimed state.

The syntax of task is shown as follows:

```
- name: Set thin rebuild Storage Volume
  dellemc_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_name: "ansible_stor_vol"
    thin_rebuild: false
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Failure with "changed": False. Exits with the failure message stating as "Could not update thin\_rebuild for <storage\_vol> in specific cluster as it is unclaimed".
2. If storage volume is present as claimed or used - Success with "changed": True. Display the storage volume details.
3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol> from the specific cluster".

## Set thin rebuild to false by ID

### Prerequisite

To set thin\_rebuild to false the storage volume in a given cluster, storage volume ID should be present in the VPLEX setup in claimed state.

The syntax of task is shown as follows:

```
- name: Set thin rebuild Storage Volume
  dellemc_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_id: "VPD83T3:68ccf098009d68af56e98e31d8c8fd84"
    thin_rebuild: false
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Failure with "changed": False. Exits with the failure message stating as "Could not update thin\_rebuild for <storage\_vol> in specific cluster as it is unclaimed".
2. If the storage volume is present as claimed or used - Success with "changed": True. Display the storage volume details.
3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol> from the specific cluster".

## List ITLs

To see storage volume details with or without ITLs list, run appropriate playbook.

The syntax of task is shown as follows:

### Get ITLs list in volume details through name

#### Prerequisite

To get ITLs list of storage volume in a given cluster, storage volume name should be present in the VPLEX setup in claimed state.

The syntax of task is shown as follows:

```
- name: List ITL's of Storage Volume
  dellemc_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_name: "ansible_stor_vol"
    get_itls: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Success with "changed": False. Display the storage volume ITLs details only.
2. If the storage volume is present as claimed or used - Success with "changed": False. Display the storage volume ITLs details only.
3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol> from the specific cluster".

### List ITLs - Get ITLs list in volume details through ID

#### Prerequisite

To get ITLs list of storage volume in a given cluster, storage volume ID should be present in the VPLEX setup in claimed state.

The syntax of task is shown as follows:

```
- name: List ITL's of Storage Volume
  dellemc_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_id: "VPD83T3:68ccf098009d68af56e98e31d8c8fd84"
    get_itls: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Success with "changed": False. Display the storage volume ITLs details only.
2. If the storage volume is present as claimed or used - Success with "changed": False. Display the storage volume ITLs details only.

3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol> from the specific cluster".

## Remove ITLs list in volume details through name

### Prerequisite

To get ITLs list of storage volume in a given cluster, storage volume name should be present in the VPLEX setup in claimed state, and parameter `get_itls` should be passed as false.

The syntax of task is shown as follows:

```
- name: List ITL's of Storage Volume
  dellemc_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_name: "ansible_stor_vol"
    get_itls: false
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Success with "changed": False. Display the storage volume details except ITLs list.
2. If the storage volume is present as claimed or used - Success with "changed": False. Display the storage volume details except ITLs list.
3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol> from the specific cluster".

## Remove ITLs list in volume details through ID

### Prerequisite

To get ITLs list of storage volume in a given cluster, storage volume ID should be present in the VPLEX setup in claimed state and parameter `get_itls` should be passed as false.

The syntax of task is shown as follows:

```
- name: List ITL's of Storage Volume
  dellemc_vplex_storage_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_id: "VPD83T3:68ccf098009d68af56e98e31d8c8fd84"
    get_itls: false
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the storage volume is present as unclaimed - Success with "changed": False. Display the storage volume details except ITLs list.
2. If the storage volume is present as claimed or used - Success with "changed": False. Display the storage volume details except ITLs list.



3. If the storage volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage volume <storage\_vol> from the specific cluster".

## Storage volume module parameters

The parameters for the storage volume module are listed.

**Table 4. Parameters for the storage volume module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameter	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The user name to access the VPLEX server.
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Mandatory	To validate the SSL certificate. If it is True, it verifies the SSL certificate. If it is False, it does not verify the SSL certificate.
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dell EMC_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	-	str	Mandatory	Name of the cluster.
storage_volume_name	-	str	Optional	Name of a specific instance of the resource.
storage_volume_id	-	str	Optional	ID of specific storage volume.
new_storage_volume_name	-	str	Optional	The new name for renaming storage volume.
get_itls	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Optional	To get the ITL's list of the storage volume.
thin_rebuild	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Optional	This parameter allows to change the value of thin_rebuild.
claimed_state	<ul style="list-style-type: none"><li>• claimed</li><li>• unclaimed</li></ul>	str	Optional	The state of a specific storage volume that is either claimed or unclaimed.
state	<ul style="list-style-type: none"><li>• present</li><li>• absent</li></ul>	str	Mandatory	The state of a specific storage volume.

## Sample output

### Claim storage volume

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook claim_storage_volume_tests.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'
```

```

PLAY [Perform Storage Volume module operations on VPLEX]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Claim Storage Volume]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "claim_vol": {
    "changed": true,
    "failed": false,
    "storage_details": {
      "application_consistent": false,
      "block_count": 524640,
      "block_size": 4096,
      "capacity": 2148925440,
      "health_indications": [],
      "health_state": "ok",
      "io_status": "alive",
      "itls": [
        {
          "initiator": "0x5000144270124b11",
          "lun": "211",
          "target": "0x5000097378091458"
        },
        {
          "initiator": "0x5000144270124b10",
          "lun": "211",
          "target": "0x5000097378091458"
        },
        {
          "initiator": "0x5000144260124b11",
          "lun": "211",
          "target": "0x5000097378091458"
        },
        {
          "initiator": "0x5000144260124b10",
          "lun": "211",
          "target": "0x5000097378091458"
        }
      ],
      "largest_free_chunk": 2148925440,
      "name": "VPD83T3:60000970000197200581533031424232",
      "operational_status": "ok",
      "provision_type": "legacy",
      "storage_array_family": "symmetrix",
      "storage_array_name": "EMC-SYMMETRIX-197200581",
      "storage_volumetype": "normal",
      "system_id": "VPD83T3:60000970000197200581533031424232",
      "thin_capable": true,
      "thin_rebuild": true,
      "use": "claimed",
      "used_by": [],
      "vendor_specific_name": "EMC"
    }
  }
}

PLAY RECAP
*****

```

```
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0
```

## Get a storage volume

```
(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook get_st_vol.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Storage Volume module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [List ITL's list of storage volume]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "itls_list_vol": {
    "changed": false,
    "failed": false,
    "storage_details": [
      {
        "initiator": "0x5000144260124b11",
        "lun": "0x010b",
        "target": "0x5000097378091458"
      },
      {
        "initiator": "0x5000144260124b10",
        "lun": "0x010b",
        "target": "0x5000097378091458"
      },
      {
        "initiator": "0x5000144270124b11",
        "lun": "0x010b",
        "target": "0x5000097378091458"
      },
      {
        "initiator": "0x5000144270124b10",
        "lun": "0x010b",
        "target": "0x5000097378091458"
      }
    ]
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=0    unreachable=0    failed=0
```

## Rename storage volume

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook rename_storage_volume_tests.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available
```

```
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
```

```
PLAY [Perform Storage Volume module operations on VPLEX]
```

```
*****
*****
```

```
TASK [Gathering Facts]
```

```
*****
*****
*****
```

```
ok: [localhost]
```

```
TASK [Rename Storage Volume]
```

```
*****
*****
```

```
changed: [localhost]
```

```
TASK [debug]
```

```
*****
*****
*****
```

```
ok: [localhost] => {
  "rename_vol": {
    "changed": true,
    "failed": false,
    "storage_details": {
      "application_consistent": false,
      "block_count": 524640,
      "block_size": 4096,
      "capacity": 2148925440,
      "health_indications": [],
      "health_state": "ok",
      "io_status": "alive",
      "largest_free_chunk": 2148925440,
      "name": "ansible_storvol_new",
      "operational_status": "ok",
      "provision_type": "legacy",
      "storage_array_family": "symmetrix",
      "storage_array_name": "EMC-SYMMETRIX-197200581",
      "storage_volumetype": "normal",
      "system_id": "VPD83T3:60000970000197200581533031424232",
      "thin_capable": true,
      "thin_rebuild": true,
      "use": "claimed",
      "used_by": [],
      "vendor_specific_name": "EMC"
    }
  }
}
```

```
PLAY RECAP
```

```
*****
*****
*****
```

```
localhost                : ok=3    changed=1    unreachable=0    failed=0
```

## Unclaim storage volume

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook unclaim_storage_volume_tests.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source
```

```
[WARNING]: No inventory was parsed, only implicit localhost is available
```

```
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
```

```
PLAY [Perform Storage Volume module operations on VPLEX]
```

```
*****
*****
```

```

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Unclaim Storage Volume - Idempotency]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "unclaim_vol_idem": {
    "changed": true,
    "failed": false,
    "storage_details": {
      "application_consistent": false,
      "block_count": 524640,
      "block_size": 4096,
      "capacity": 2148925440,
      "health_indications": [],
      "health_state": "ok",
      "io_status": "alive",
      "largest_free_chunk": 2148925440,
      "name": "VPD83T3:60000970000197200581533031424232",
      "operational_status": "ok",
      "provision_type": "legacy",
      "storage_array_family": "symmetrix",
      "storage_array_name": "EMC-SYMMETRIX-197200581",
      "storage_volumetype": "traditional",
      "system_id": "VPD83T3:60000970000197200581533031424232",
      "thin_capable": true,
      "thin_rebuild": false,
      "use": "unclaimed",
      "used_by": [],
      "vendor_specific_name": "EMC"
    }
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Extent module

The extent module manages the extents in VPLEX.

The manage extent module has the following functions:

- Create an extent
- Get extent from cluster
- Rename an extent
- Delete an extent

## Create extent with storage volume name

To create an extent, run the appropriate playbook.

### Prerequisite

To create an extent, a storage volume name in claimed state should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Create an extent with storage_volume_name
  dellemc_vplex_extent:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    extent_name: "ansible_ext_name"
    storage_volume_name: "ansible_storvol"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent is not present- Success with "changed": True. A new extent is created.
2. If extent is present with the same storage volume (Idempotency) - Success with "changed": False. No change to the extent as it is created.
3. If extent name is present with storage volume, and it is used through the different extent - Failure with "changed": False. Exits with the failure message stating as "given storage volume is in use".

## Create extent with storage volume ID

To create an extent, run the appropriate playbook.

### Prerequisite

To create an extent, a storage volume ID in claimed state should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Create an extent with storage_volume_id
  dellemc_vplex_extent:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    extent_name: "ansible_ext_ID"
    storage_volume_id: "VPD83T3:60000970000197200581533030353735"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent is not present- Success with "changed": True. A new extent is created.
2. If extent is present with the same storage volume (Idempotency) - Success with "changed": False. No change to the extent as it is created.
3. If extent name is present with storage volume, and it is used through the different extent - Failure with "changed": False. Exits with the failure message stating as "given storage volume is in use".

## Get extent

To get the extent details, run the appropriate playbook.

### Prerequisite

To get an extent, extent with same name should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Get extent
  dellemc_vplex_extent:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    extent_name: "extent_ansible_storvol_1"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent is present- Success with "changed": False. Got extents from cluster-1.
2. If extent is not present- Failure with "changed": False. Exits with the failure message stating as "Extent is not present".

## Rename extent

To rename the extent, run the appropriate playbook.

### Rename extent with extent name

#### Prerequisite

1. To rename the extent, extent should be present in the the VPLEX setup.
2. To rename extent name, a new extent name should not be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Rename extent with extent name
  dellemc_vplex_extent:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    extent_name: "extent_ansible_storvol_1"
    new_extent_name: "ansible_ext_update_name"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent is present- Success with "changed": True. Displays the corresponding extent details.
2. If extent is present with the same storage volume (Idempotency) - Success with "changed": False. No change to the extent as it is renamed.
3. If extent is not present- Failure with "changed": False. Exits with the failure message stating as "Extent is not present".

## Rename extent with storage volume name

#### Prerequisite

1. To rename the extent, extent should be present in the VPLEX setup.

2. To rename extent with storage volume name, an extent should be present over that storage volume name in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Rename extent
  dell EMC_vplex_extent:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_name: "ansible_storvol_1"
    new_extent_name: "ansible_ext_update_name"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent is present- Success with "changed": True. Extent name details are updated.
2. If extent is present with the same storage volume (Idempotency) - Success with "changed": False. No change to the extent as it is renamed.
3. If extent is not present- Failure with "changed": False. Exits with the failure message stating as "Extent is not present".

## Rename extent with storage volume ID

### Prerequisite

1. To rename the extent, the extent should be present in VPLEX setup.
2. To rename extent with storage volume ID, an extent should be present over that storage volume ID in VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Rename extent
  dell EMC_vplex_extent:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_id: "VPD83T3:6000097000019720058153303143"
    new_extent_name: "ansible_ext_update_id"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent is present- Success with "changed": True. Extent name details are updated.
2. If extent is present with the same storage volume (Idempotency) - Success with "changed": False. No change to the extent as it is renamed.
3. If extent is not present- Failure with "changed": False. Exits with the failure message stating as "Extent is not present".

## Delete extent with extent name

To delete the extent, run the appropriate playbook.

### Prerequisite



To delete the extent, extent with same name should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Delete an extent with extent_name
  dellemc_vplex_extent:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    extent_name: "extent_ansible_storvol_1"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent is present- Success with "changed": True. Extent is deleted.
2. If extent is not present (Idempotency) - Success with "changed": False. Exits without fail.

## Delete extent with storage volume name

To delete an extent, run the appropriate playbook.

### Prerequisite

To delete the extent, extent on given storage volume name should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Delete an extent with storage_volume_name
  dellemc_vplex_extent:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_volume_name: "ansible_storvol"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent is present- Success with "changed": True. Extent is deleted.
2. If extent is not present (Idempotency) - Success with "changed": False. Exits without fail.

## Extent module parameters

The parameters for the extent module are listed.

**Table 5. Parameters for the extent module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The user name to access the VPLEX server.
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.

**Table 5. Parameters for the extent module (continued)**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
verifycert	<ul style="list-style-type: none"> <li>True</li> <li>False</li> </ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"> <li>True - Verifies the SSL certificate</li> <li>False - Specified that the SSL certificate should not be verified.</li> </ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>True</li> <li>False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dell EMC_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	-	str	Mandatory	Name of the cluster.
storage_volume_name	-	str	Optional	Storage volume name to create the extent. <b>NOTE:</b> Any one of the parameters storage_volume_name or storage_volume_id or extent_name is required.
extent_name	-	str	Optional	The name of a specific instance of the resource. It is required for creating an extent.
storage_volume_id	-	str	Optional	Storage volume ID to create the extent.
new_extent_name	-	str	Optional	The value to be used while renaming the extent.
state	present/absent	str	Mandatory	The state of the extent.

## Sample output

### Create extent

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook create_extent_tests.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Details of the VPLEX host]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Create an Extent with storage volume name]
*****
changed: [localhost]

TASK [debug]
*****
```

```

ok: [localhost] => {
  "create_extent_name": {
    "changed": true,
    "extent_details": {
      "application_consistent": "False",
      "block_count": 2621760.0,
      "block_offset": 0.0,
      "block_size": 4096.0,
      "capacity": 10738728960.0,
      "health_indications": [],
      "health_state": "ok",
      "io_status": "alive",
      "itls": [
        "0x5000144270124b11/0x5000097378091458/76",
        "0x5000144270124b10/0x5000097378091458/76",
        "0x5000144260124b11/0x5000097378091458/76",
        "0x5000144260124b10/0x5000097378091458/76"
      ],
      "name": "ansible_extent_name",
      "operational_status": "ok",
      "storage_array_family": "symmetrix",
      "storage_volume": "/vplex/v2/clusters/cluster-1/storage_volumes/
VPD83T3%3A60000970000197200581533030353632",
      "storage_volumetype": "normal",
      "system_id": "SLICE:f0124b3da38e31f1",
      "underlying_storage_block_size": 512.0,
      "use": "claimed",
      "used_by": [],
      "vendor_specific_name": "EMC"
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Get an extent

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook get_extents.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Details of the VPLEX host]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get extent details]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "get_extent": {
    "changed": false,
    "extent_details": {
      "application_consistency_group_modulename": "False",

```

```

        "block_count": 524640.0,
        "block_offset": 0.0,
        "block_size": 4096.0,
        "capacity": 2148925440.0,
        "health_indications": [],
        "health_state": "ok",
        "io_status": "alive",
        "itls": [
            "0x5000144260124b11/0x5000097378091458/185",
            "0x5000144260124b10/0x5000097378091458/185",
            "0x5000144270124b11/0x5000097378091458/185",
            "0x5000144270124b10/0x5000097378091458/185"
        ],
        "name": "eext_4",
        "operational_status": "ok",
        "storage_array_family": "symmetrix",
        "storage_volume": "/vpflex/v2/clusters/cluster-1/storage_volumes/
vi_ex_test_6_1",
        "storage_volumetype": "normal",
        "system_id": "SLICE:e0124b74540343ec",
        "underlying_storage_block_size": 512.0,
        "use": "claimed",
        "used_by": [],
        "vendor_specific_name": "EMC"
    },
    "failed": false
}
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=0    unreachable=0    failed=0

```

## Rename extent

```

(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook rename_extent_tests.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Details of the VPLEX host]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Rename Extent using storage volume ID]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "rename_extent": {
    "changed": true,
    "extent_details": {
      "application_consistent": "False",
      "block_count": 1310880.0,
      "block_offset": 0.0,
      "block_size": 4096.0,
      "capacity": 5369364480.0,

```

```

        "health_indications": [],
        "health_state": "ok",
        "io_status": "alive",
        "itls": [
            "0x5000144270124b11/0x5000097378091458/37",
            "0x5000144270124b10/0x5000097378091458/37",
            "0x5000144260124b11/0x5000097378091458/37",
            "0x5000144260124b10/0x5000097378091458/37"
        ],
        "name": "ansible_ext_update_id",
        "operational_status": "ok",
        "storage_array_family": "symmetrix",
        "storage_volume": "/vplex/v2/clusters/cluster-1/storage_volumes/
Symm0581_053B",
        "storage_volumetype": "normal",
        "system_id": "SLICE:f0124b3da38e2959",
        "underlying_storage_block_size": 512.0,
        "use": "claimed",
        "used_by": [],
        "vendor_specific_name": "EMC"
    },
    "failed": false
}
}

PLAY RECAP
*****
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Delete extent

```

(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook delete_extent_tests.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Details of the VPLEX host]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Delete an Extent with extent name]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "delete_extent": {
    "changed": true,
    "extent_details": null,
    "failed": false
  }
}

PLAY RECAP
*****
*****

```

```
*****
localhost : ok=3    changed=1    unreachable=0    failed=0
```

## Device module

The device module manages the local devices in the VPLEX.

The manage device module has the following functions:

- Create a device
- Get device from cluster
- Add extent to the device
- Remove extent from the device
- Rename a device
- Delete a device

## Create a RAID 1, RAID 0, and RAID C device

### Create a RAID 1 device

To create a device, run the appropriate playbook.

#### Prerequisite

1. The device name should not be present in the VPLEX, and it should not contain any special characters. Also, the length should not be more than 63 characters.
2. For RAID 1 device, **stripe\_depth** is not required. If provided, it throws error.
3. The extent that is provided should be in 'claimed' state and it should be present in the VPLEX.

The syntax of the task is shown as follows:

```
- name: Create raid-1 device
  dell EMC vplex device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    geometry: "raid-1"
    device_name: "ansible-test"
    extents: ["extent_1", "extent_2"]
    extent_state: "present-in-device"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the extent is in claimed state, then device name is valid - Success with "changed": True. Device is created.
2. If the provided device name exists (Idempotency) - Success with "changed": False. Device is created.
3. If device name is invalid - Failure with "changed": False. Execution fails with the error message "Device name should start with an alphabet or '\_' and only alphanumeric characters and -\_ are allowed".
4. If stripe\_depth is provided - Failure with "changed": False. Execution fails with the error message "stripe\_depth is not required for raid-1".
5. If extent is not in the claimed state - Failure with "changed": False. Execution fails with the error message "Could not create device in cluster-1 due to error: One or more of the elements is in use.".

## Create a RAID 0 device

To create a device, run the appropriate playbook.

### Prerequisite

1. The device name should not be present in the VPLEX, and it should not contain any special characters. Also, the length should not be more than 63 characters.
2. The extent that is provided should be in 'claimed' state and it should be present in the VPLEX.

The syntax of the task is shown as follows:

```
- name: Create raid-0 device
  dell EMC vplex device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    geometry: "raid-0"
    stripe_depth: "4KB"
    device_name: "ansible-test"
    extents: ["extent_1","extent_2"]
    extent_state: "present-in-device"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the extent is in the claimed state, stripe\_depth is provided and device name is valid - Success with "changed": True. Device is created.
2. If the provided device name exists (Idempotency) - Success with "changed": False. Device is created.
3. If device name is invalid - Failure with "changed": False. Execution fails with the error message "Device name should start with an alphabet or '\_' and only alphanumeric characters and -\_ are allowed".
4. If extent is not in the claimed state - Failure with "changed": False. Execution fails with the error message "Could not create device in cluster-1 due to error: One or more of the elements is in use.".

## Create a RAID C device

To create a device, run the appropriate playbook.

### Prerequisite

1. The device name should not be present in the VPLEX, and it should not contain any special characters. Also, the length should not be more than 63 characters.
2. The extent that is provided should be in 'claimed' state and it should be present in the VPLEX.

The syntax of the task is shown as follows:

```
- name: Create raid-c device
  dell EMC vplex device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    geometry: "raid-c"
    device_name: "ansible-test"
    extents: ["extent_1","extent_2"]
    extent_state: "present-in-device"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the extent is in the claimed state, and device name is valid - Success with "changed": True. Device is created.
2. If the provided device name exists (Idempotency) - Success with "changed": False. Device is created.
3. If device name is invalid - Failure with "changed": False. Execution fails with the error message "Device name should start with an alphabet or '\_' and only alphanumeric characters and -\_ are allowed".
4. If stripe\_depth is provided - Failure with "changed": False. Execution fails with the error message "stripe\_depth is not required for raid-c".
5. If extent is not in the claimed state - Failure with "changed": False. Execution fails with the error message "Could not create device in cluster-1 due to error: One or more of the elements is in use."

## Get device from cluster

To get details of a device using device\_name, run the appropriate playbook.

### Prerequisite

The device should be present in the VPLEX.

The syntax of the task is shown as follows:

```
- name: Get device from cluster
  dellemc_vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    device_name: "ansible-test"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).


### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the device is present- Success with "changed": False. Displays the corresponding device details.
2. If the device is not present- Failure with "changed": False. Execution fails with the error message "Could not get the device".

## Add an extent to the device

To add an extent to the device, run the appropriate playbook.

 **NOTE:** This task is supported only for RAID 1 device. It is not supported for RAID 0 and RAID C devices.

### Prerequisite

1. The device should be present in the VPLEX.
2. One or more extents that are provided should be in the 'claimed' state and should be present in the VPLEX.
3. The size of one or more extents that are provided should be greater or equal to the size of the device.
4. The geometry of the device should be only 'RAID 1' .

The syntax of the task is shown as follows:

```
- name: Add an extent to device
  dellemc_vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    device_name: "ansible-test"
    extents: ["extent_1"]
```



```
extent_state: "present-in-device"
state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).


### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If all the prerequisites are satisfied - Success with "changed": True. Extent is added to the device.
2. If the extent is already added to the device (Idempotency) - Success with "changed": False. Extent is added to the device.
3. If extent is in used state - Failure with "changed": False. Execution fails with the error message "Extent is used by another device in cluster-1".
4. If extent size < device size - Failure with "changed": False. Execution fails with the error message "Could not attach extent to device in cluster-1. The size of the device is greater than the extent".
5. If the device is not RAID 1 - Failure with "changed": False. Execution fails with the error message "Add or Remove extent is supported only on RAID 1 device".

## Remove an extent from the device

To remove an extent to the device, run the appropriate playbook.

 **NOTE:** This task is supported only for RAID 1 device. It is not supported for RAID 0 and RAID C devices.

### Prerequisite

1. The device should be present in the VPLEX.
2. If one or more extents that are provided must be attached to the device (that is they should be in the used state and the device over them should be the same as the device name provided).
3. The health indication of the device should not be in a rebuilding state.
4. The geometry of the device should be only RAID 1.

The syntax of the task is shown as follows:

```
- name: Remove an extent from Device
  dell EMC_vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    device_name: "ansible-test"
    extents: ["extent_1"]
    extent_state: "absent-in-device"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If all the prerequisites are satisfied - Success with "changed": True. Extent is removed from the device.
2. If the extent is already removed from the device, or not added to the device (Idempotency) - Success with "changed": False. Extent is removed from the device.
3. If extent is to be removed when the device is in rebuilding state - Failure with "changed": False. Execution fails with the error message "Could not remove extent since the device is in rebuild state".

## Rename device

To rename the device, run the appropriate playbook.

### Prerequisite

1. The device should be present in the VPLEX.

2. A new device name should not be present in the VPLEX. It should be completely a new value.
3. It should not contain special characters and cannot be greater than 63 characters.

The syntax of the task is shown as follows:

```
- name: Rename a local device
  dellemc_vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    device_name: "ansible-test"
    new_device_name: "ansible-test-new"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the device name and new name are different, and new name does not exist in the VPLEX - Success with "changed": True. Device is renamed.
2. If the device name and new name are same (Idempotency) - Success with "changed": False. Device is renamed.
3. If the new device name is invalid - Failure with "changed": False. Execution fails with the error message "Could not rename the device".
4. If the new name exists in VPLEX - Failure with "changed": False. Execution fails with the error message "New Device name already exists in cluster-1".

## Update transfer size of the device

To update the transfer size of the device, run the appropriate playbook.

### Prerequisite

1. The device should be present in the VPLEX.
2. The geometry of the device should be always RAID 1.
3. The transfer size should be always **>=40KB** and **<=128MB**, and it should be multiple of 4 KB.

The syntax of the task is shown as follows:

```
- name: Update transfer_size of a local device
  dellemc_vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    device_name: "ansible_device_1"
    transfer_size: "40960"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If all the prerequisites are satisfied - Success with "changed": True. Updated the transfer size of the device.
2. If the provided transfer size is equal to the transfer\_size of the device (Idempotency) - Success with "changed": False. Transfer size is updated for the device.
3. If provided transfer size is of invalid range - Failure with "changed": False. Execution fails with the error message "Specified transfer size 135266304 is too high/Specified transfer size 1 is too low".

## Add or Remove local mirror to the device

Describes about adding or removing the local mirror to the device.

### Add local mirror to the device

To add local mirror to the device, run the appropriate playbook.

#### Prerequisite

1. The device and mirror device should be present in the VPLEX.
2. The mirror should be present in the same cluster as the device.
3. The geometry of the device should always be RAID 1.
4. The mirror should not have virtual volume over it, but the parent device can have virtual volume over it.
5. The mirror should not be added to the device, or part of any other device.
6. The size of the mirror device that is provided should be greater or equal to the size of the device.

The syntax of the task is shown as follows:

```
- name: Add local mirror to device
  dell EMC_vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    device_name: "ansible_device_1"
    mirror_name: "local_mirror_dev_1"
    mirror_state: "present-in-device"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If all the prerequisites are satisfied - Success with "changed": True. Local mirror is added to the device.
2. If the local mirror is already added to the device (Idempotency) - Success with "changed": False. Local mirror is added to the device.
3. If the mirror is attached to some other device - Failure with "changed": False. Execution fails with the error message "Could not update the device due to error: Mirror is in use".
4. If the device is not RAID 1 - Failure with "changed": False. Execution fails with the error message "Could not update the device due to error: Parent is not RAID 1".
5. If mirror size > device size - Failure with "changed": False. Execution fails with the error message "The mirror device capacity should not be lesser than the device dev\_10gb\_1 capacity".

### Remove local mirror to the device

To remove local mirror to the device, run the appropriate playbook.

#### Prerequisite

1. The device and mirror device should be present in the VPLEX.
2. The geometry of the device should always be RAID 1.
3. The mirror that is provided should be attached to the proposed device.
4. The main device should not be in a rebuilding state.

The syntax of the task is shown as follows:

```
- name: Remove local mirror from device
  dell EMC_vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
```

```

verifycert: "{{ verifycert }}"
cluster_name: "cluster-1"
device_name: "ansible_device_1"
mirror_name: "local_mirror_dev_1"
mirror_state: "absent-in-device"
state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If all the prerequisites are satisfied - Success with "changed": True. Local mirror is removed from the device.
2. If the local mirror is already removed from the device or not added to the device (Idempotency) - Success with "changed": False. Local mirror is removed from the device.
3. If local mirror is to be removed when the device is in a rebuilding state - Failure with "changed": False. Execution fails with the error message "Could not remove mirror since the device is in rebuild state".

## Add or Remove remote mirror from the device

### Add remote mirror to the device

To add remote mirror to the device, run the appropriate playbook.

#### Prerequisite

1. The device and mirror device should be present in the VPLEX.
2. The device and remote mirror can be of any geometry.
3. The size(capacity) of the remote mirror should be greater than or equal to the size(capacity) of parent device.
4. Only one remote mirror can be added to the parent device.
5. The remote mirror should not have virtual volume over it, but the parent device can have a virtual volume over it.
6. There should be no device with same name as the remote mirror name in its cluster (for example, if the user has a device\_1 as remote mirror from cluster-2, then there should not be a device with the same name that is device\_1 in cluster-1).

The syntax of the task is shown as follows:

```

- name: Add remote mirror to device
  dell EMC vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    device_name: "ansible_device_1"
    target_cluster: "cluster-2"
    mirror_name: "remote_mirror_dev_1"
    mirror_state: "present-in-device"
    state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If all the prerequisites are satisfied - Success with "changed": True. Remote mirror is added to the device.
2. If the remote mirror is already added to the device (Idempotency) - Success with "changed": False. Remote mirror is added to the device.
3. If mirror is attached to some other device - Failure with "changed": False. Execution fails with the error message "Could not update the device due to error: Mirror is in use".
4. If device name and mirror name are same - Failure with "changed": False. Execution fails with the error message "Could not update the device due to error: Unable to attach mirror to the device".

5. If mirror size > device size - Failure with "changed": False. Execution fails with the error message "The mirror device capacity should not be lesser than the device dev\_10gb\_1 capacity".

## Remove remote mirror to the device

To remove remote mirror to the device, run the appropriate playbook.

### Prerequisite

1. The device and mirror device should be present in the VPLEX.
2. To remove the remote mirror, the remote mirror should be attached to the parent device (that is distributed device because once user adds remote mirror to parent device, it becomes distributed device with name same as the parent device).
3. The parent device should not be in a rebuilding state.

The syntax of the task is shown as follows:

```
- name: Remove remote mirror from device
  dellemc_vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    device_name: "ansible_device_1"
    target_cluster: "cluster-2"
    mirror_name: "remote_mirror_dev_1"
    mirror_state: "absent-in-device"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If all the prerequisites are satisfied - Success with "changed": True. Remote mirror is removed from the device.
2. If the Remote mirror is already removed from the device, or not added to the device (Idempotency) - Success with "changed": False. Remote mirror is removed from the device.
3. If remote mirror is to be removed when device is in a rebuilding state - Failure with "changed": False. Execution fails with error message "Could not remove mirror since the device is in rebuild state".

## Delete device

To delete the device, run the appropriate playbook.

### Prerequisite

1. The device should be present in the VPLEX.
2. The health indication or status should not be in rebuilding state.

The syntax of the task is shown as follows:

```
- name: Delete device from cluster
  dellemc_vplex_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    device_name: "ansible-test"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the device is present - Success with "changed": True. Device is deleted.
2. If the device is deleted or does not exist in the VPLEX (Idempotency) - Success with "changed": False. Device is deleted.
3. If device is in the rebuilding state - Failure with "changed": False. Execution fails with the error message "Could not delete the device in rebuilding state".

## Device module parameters

The parameters for the local device module are listed.

**Table 6. Parameters for the local device module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	User name to access the VPLEX server.
vplexpassword	-	str	Mandatory	Password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Mandatory	To validate the SSL certificate: <ul style="list-style-type: none"> <li>• True - Verifies the SSL certificate.</li> <li>• False - Specified that the SSL certificate should not be verified.</li> </ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dellmc_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	-	str	Mandatory	Name of the cluster.
device_name	-	str	Mandatory	Name of the device. Device name can only contain letters, numbers _ or - and less than 60 characters.
geometry	<ul style="list-style-type: none"> <li>• raid-1</li> <li>• raid-0</li> <li>• raid-c</li> </ul> default: raid-1	str	-	Geometry for the new device. If no geometry specified then raid-1 set by default.
stripe_depth	-	str	-	Size of the stripe_depth if geometry is raid-0. It must be specified while creating raid-0 device.
extents	<ul style="list-style-type: none"> <li>• present-in-device</li> <li>• absent-in-device</li> </ul>	list	Optional	Extent names while creating a new device. It is required to be specified while creating a device or while adding or removing extent from the device.
extent_state	-	str	Optional	To determine whether to add /remove extent from the device. This is required to be specified while creating a device, or while adding or removing extent from the device. <ul style="list-style-type: none"> <li>• present-in-device- Add extent to the device</li> <li>• absent-in-device- Remove extent from the device</li> </ul>
new_device_name	-	str	Optional	The new name of the device. It is required to be specified while re-naming the device. New device name can only contains letters, numbers _ or - and fewer than 60 characters.

**Table 6. Parameters for the local device module (continued)**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
mirror_name	-	str	Optional	The name of the mirror device (local or remote) should be added to the device.
mirror_state	<ul style="list-style-type: none"> <li>present-in-device</li> <li>absent-in-device</li> </ul>	str	Optional	To add or remove local or remote mirror to or from the device. <ul style="list-style-type: none"> <li>present-in-device- Add mirror to the device</li> <li>absent-in-device- Remove mirror from the device</li> </ul>
transfer_size	-	int	Optional	To update the transfer size of the device, change the rebuilding time. Valid range for transfer size is 40960-134217728 bytes that is 40K-128M , and it should be multiples ok 4K. Default: 128KB(131072).
target_cluster	-	str	Optional	The name of the target cluster where the mirror is present.
state	<ul style="list-style-type: none"> <li>Present</li> <li>Absent</li> </ul>	str	Mandatory	To determine whether a device exists or not. <ul style="list-style-type: none"> <li>present - The device must be present in the system</li> <li>absent - The device must not be present in the system</li> </ul>

## Sample output

### Create device

```
(py3_ans2_9) [root@dsvej252 devices]# ansible-playbook create.yml
[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Device operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Create a raid-1 device]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "create_device": {
    "changed": true,
    "device_details": {
      "application_consistent": false,
      "block_count": 2621440.0,
      "block_size": 4096.0,
      "capacity": 10737418240.0,
      "geometry": "raid-1",
      "health_indications": [
        "rebuilding"
      ],
      "health_state": "minor-failure",
      "locality": "local",
```

```

        "name": "ansible-test",
        "operational_status": "degraded",
        "rebuild_allowed": true,
        "rebuild_progress": "0",
        "rebuild_status": "rebuilding",
        "rebuild_type": "full",
        "service_status": "running",
        "storage_array_family": "powerstore",
        "system_id": "ansible-test",
        "thin_capable": true,
        "top_level": true,
        "transfer_size": 131072.0,
        "visibility": "local"
    },
    "failed": false
}
}

```

#### PLAY RECAP

```

*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

### Get a device

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook get_device.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

```

```

[WARNING]: No inventory was parsed, only implicit localhost is available

```

```

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

```

#### PLAY [Testing Device operations]

```

*****
*****

```

#### TASK [Gathering Facts]

```

*****
*****

```

```

ok: [localhost]

```

#### TASK [Get device from cluster]

```

*****
*****

```

```

ok: [localhost]

```

#### TASK [debug]

```

*****
*****

```

```

ok: [localhost] => {
  "get_device": {
    "changed": false,
    "device_details": {
      "application_consistent": false,
      "block_count": 2621760.0,
      "block_size": 4096.0,
      "capacity": 10738728960.0,
      "geometry": "raid-1",
      "health_indications": [
        "rebuilding"
      ],
      "health_state": "minor-failure",
      "locality": "local",
      "name": "ansible_dev_new",
      "operational_status": "degraded",
      "rebuild_allowed": true,
      "rebuild_eta": "129",
      "rebuild_progress": "36",
      "rebuild_status": "rebuilding",
      "rebuild_type": "full",

```



```

        "service_status": "running",
        "storage_array_family": "symmetrix",
        "system_id": "ansible_dev_new",
        "thin_capable": true,
        "top_level": true,
        "transfer_size": 40960.0,
        "visibility": "local"
    },
    "failed": false
}
}

PLAY RECAP
*****
*****
localhost                : ok=4    changed=0    unreachable=0    failed=0

```

## Add extents to a device

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook add_extents.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Device operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Build a list of extents for update]
*****
*****
ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Add extent to device]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "add_extent": {
    "changed": true,
    "device_details": {
      "application_consistent": false,
      "block_count": 2621760.0,
      "block_size": 4096.0,
      "capacity": 10738728960.0,
      "geometry": "raid-1",
      "health_indications": [
        "rebuilding"
      ],
      "health_state": "minor-failure",
      "locality": "local",
      "name": "ansible_dev_new",
      "operational_status": "degraded",
      "rebuild_allowed": true,
      "rebuild_progress": "0",
      "rebuild_status": "rebuilding",
      "rebuild_type": "full",
      "service_status": "running",
      "storage_array_family": "symmetrix",

```

```

        "system_id": "ansible_dev_new",
        "thin_capable": true,
        "top_level": true,
        "transfer_size": 40960.0,
        "visibility": "local"
    },
    "failed": false
}
}

PLAY RECAP
*****
*****
localhost                : ok=4    changed=1    unreachable=0    failed=0

```

## Remote extents from device

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook remove_extents.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Device operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Build a list of extents for update]
*****
*****
ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Remove extent from Device]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "remove_extent": {
    "changed": true,
    "device_details": {
      "application_consistent": false,
      "block_count": 2621760.0,
      "block_size": 4096.0,
      "capacity": 10738728960.0,
      "geometry": "raid-1",
      "health_indications": [],
      "health_state": "ok",
      "locality": "local",
      "name": "ansible_dev_new",
      "operational_status": "ok",
      "rebuild_allowed": true,
      "rebuild_status": "done",
      "rebuild_type": "full",
      "service_status": "running",
      "storage_array_family": "symmetrix",
      "system_id": "ansible_dev_new",
      "thin_capable": true,
      "top_level": true,
      "transfer_size": 40960.0,
      "visibility": "local"
    }
  }
}

```

```

    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=4    changed=1    unreachable=0    failed=0

```

## Rename device

```

(py3_ans2_9) [root@dsvej252 devices]# ansible-playbook rename.yml
[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Device operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Rename device]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "rename_device": {
    "changed": true,
    "device_details": {
      "application_consistent": false,
      "block_count": 2621440.0,
      "block_size": 4096.0,
      "capacity": 10737418240.0,
      "geometry": "raid-1",
      "health_indications": [],
      "health_state": "ok",
      "locality": "local",
      "name": "ansible-test-new",
      "operational_status": "ok",
      "rebuild_allowed": true,
      "rebuild_status": "done",
      "rebuild_type": "full",
      "service_status": "running",
      "storage_array_family": "powerstore",
      "system_id": "ansible-test-new",
      "thin_capable": true,
      "top_level": true,
      "transfer_size": 131072.0,
      "visibility": "local"
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0
skipped=0                rescued=0    ignored=0

```

## Update transfer size of the device

```
(py3_ans2_9) [root@localhost devices]# ansible-playbook transfer_size.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Device operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Update Transfer size of a device]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "transfer_size": {
    "changed": true,
    "device_details": {
      "application_consistent": false,
      "block_count": 1310880.0,
      "block_size": 4096.0,
      "capacity": 5369364480.0,
      "geometry": "raid-1",
      "health_indications": [],
      "health_state": "ok",
      "locality": "local",
      "name": "sas_par_dev_1",
      "operational_status": "ok",
      "rebuild_allowed": true,
      "rebuild_status": "done",
      "rebuild_type": "full",
      "service_status": "running",
      "storage_array_family": "symmetrix",
      "system_id": "sas_par_dev_1",
      "thin_capable": true,
      "top_level": true,
      "transfer_size": 40960.0,
      "visibility": "local"
    },
    "failed": false
  }
}

PLAY RECAP
*****
localhost          : ok=3    changed=1    unreachable=0    failed=0
skipped=0          rescued=0    ignored=0
```

## Add local mirror to the device

```
(py3_ans2_9) [root@localhost devices]# ansible-playbook add_local_mirror.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Device operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]
```

```

TASK [Add mirror to Device]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "add_mirror": {
    "changed": true,
    "device_details": {
      "application_consistent": false,
      "block_count": 1310880.0,
      "block_size": 4096.0,
      "capacity": 5369364480.0,
      "geometry": "raid-1",
      "health_indications": [
        "rebuilding"
      ],
      "health_state": "minor-failure",
      "locality": "local",
      "name": "sas_par_dev_1",
      "operational_status": "degraded",
      "rebuild_allowed": true,
      "rebuild_progress": "0",
      "rebuild_status": "rebuilding",
      "rebuild_type": "full",
      "service_status": "running",
      "storage_array_family": "symmetrix",
      "system_id": "sas_par_dev_1",
      "thin_capable": true,
      "top_level": true,
      "transfer_size": 131072.0,
      "visibility": "local"
    },
    "failed": false
  }
}

PLAY RECAP
*****
localhost          : ok=3    changed=1    unreachable=0    failed=0
skipped=0          rescued=0    ignored=0

```

## Remove local mirror to the device

```

(py3_ans2_9) [root@localhost devices]# ansible-playbook remove_local_mirror.yml

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Device operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Remove mirror from Device]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "add_mirror": {

```

```

    "changed": true,
    "device_details": {
      "application_consistent": false,
      "block_count": 1310880.0,
      "block_size": 4096.0,
      "capacity": 5369364480.0,
      "geometry": "raid-1",
      "health_indications": [],
      "health_state": "ok",
      "locality": "local",
      "name": "sas_par_dev_1",
      "operational_status": "ok",
      "rebuild_allowed": true,
      "rebuild_status": "done",
      "rebuild_type": "full",
      "service_status": "running",
      "storage_array_family": "symmetrix",
      "system_id": "sas_par_dev_1",
      "thin_capable": true,
      "top_level": true,
      "transfer_size": 131072.0,
      "visibility": "local"
    },
    "failed": false
  }
}

```

#### PLAY RECAP

```

*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

### Add remote mirror to the device

```

(py3_ans2_9) [root@localhost devices]# ansible-playbook add_remote_mirror.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

```

#### PLAY [Testing Device operations]

```

*****
*****

```

#### TASK [Gathering Facts]

```

*****
*****
ok: [localhost]

```

#### TASK [Add remote mirror to Device]

```

*****
*****
changed: [localhost]

```

#### TASK [debug]

```

*****
*****
ok: [localhost] => {
  "add_mirror": {
    "changed": true,
    "device_details": {
      "application_consistent": false,
      "block_count": 1310880.0,
      "block_size": 4096.0,
      "capacity": 5369364480.0,
      "geometry": "raid-1",
      "health_indications": [],
      "health_state": "ok",
      "locality": "local",
      "name": "sas_par_dev_12021Jan25_124405",
      "operational_status": "ok",
      "rebuild_allowed": true,
      "rebuild_status": "done",
      "rebuild_type": "full",

```

```

        "service_status": "running",
        "storage_array_family": "symmetrix",
        "system_id": "sas_par_dev_12021Jan25_124405",
        "thin_capable": true,
        "top_level": true,
        "transfer_size": 40960.0,
        "visibility": "global"
    },
    "failed": false
}
}

PLAY RECAP
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Remove remote mirror to the device

```

(py3_ans2_9) [root@localhost devices]# ansible-playbook remove_remote_mirror.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Device operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Remove mirror from Device]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "rem_mirror": {
    "changed": true,
    "device_details": {
      "capacity": 5369364480,
      "geometry": "raid-1",
      "health_indications": [],
      "health_state": "ok",
      "name": "sas_par_dev_1",
      "operational_status": "ok",
      "rebuild_allowed": true,
      "rebuild_status": "done",
      "rebuild_type": "full",
      "rule_set_name": "cluster-1-detaches",
      "service_status": "running",
      "storage_array_family": "symmetrix",
      "thin_capable": true
    },
    "failed": false
  }
}

PLAY RECAP
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Delete device

```
(py3_ans2_9) [root@dsvej252 devices]# ansible-playbook delete.yml
[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Device operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Delete device]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "delete_device": {
    "changed": true,
    "device_details": null,
    "failed": false
  }
}

PLAY RECAP
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0
```

## Distributed device module

The distributed device module manages the distributed devices in VPLEX metro setup.

Distributed Device playbook is supported in legacy systems VPLEX 6.2, but not supported in VPLEX 7.0 and above versions due to device rule-set deprecation.

The manage distributed device module has the following functions:

- Create a distributed device
- Rename a distributed device
- Update the rule set name of a distributed device
- Get the details of a distributed device
- Delete a distributed device

## Create a distributed device

To create a distributed device, run the appropriate playbook.

### Prerequisite

1. To create a distributed device, distributed device with same name should not be present in the VPLEX setup.
2. The source device and target device should not be from the same cluster.
3. Both source device and target device should not contain virtual volumes over it, and both should be top-level devices.
4. Size of target device must be greater or equal to the size of source device.



The syntax of the task is shown as follows:

```
- name: Create a Distributed device
  dell EMC_vplex_distributed_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_device_name: "ansible_dist_dev"
    target_cluster: "cluster-2"
    target_device: "ansible_dev_1"
    source_cluster: "cluster-1"
    source_device: "ansible_dev_2"
    rule_set: "cluster-1-detaches"
    sync: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed device is not present- Success with "changed": True. A new distributed device is created.
2. If distributed device is present with same name, source and target device (Idempotency) - Success with "changed": False. No change to the distributed device as it is created.
3. If given source or target device has virtual volume over it - Failure with "changed": False. Exits with the failure message stating as " given source/target device has virtual volume in cluster<cluster-name>".

## Rename a distributed device

To rename a distributed device, run the appropriate playbook.

### Prerequisite

1. To rename the distributed device, distributed device with same name should be present in the VPLEX setup.
2. The new name should not have special characters other than \_ and - , and its length should not be greater than 63 characters.
3. There should be no distributed device present with the name with which the distributed device is going to be renamed.

The syntax of the task is shown as follows:

```
- name: Rename Distributed device
  dell EMC_vplex_distributed_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_device_name: "ansible_dist_dev"
    new_distributed_device_name: "new_ansible_dist_dev"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed device is present- Success with "changed": True. The details of distributed device name are updated.
2. If new distributed device name is given as same as old name (Idempotency) - Success with "changed": False. No change to the distributed device as it is displayed with the same name.
3. If distributed device is not present- Failure with "changed": False. Exits with the failure message stating as "Resource not found".

## Update the rule set name of a distributed device

To update the rule set name of the distributed device, run the appropriate playbook.

### Prerequisite

1. To update rule set of the distributed device, distributed device with same name should be present in the VPLEX setup.
2. The rule set name should be either cluster-1-detaches or cluster-2-detaches .

The syntax of the task is shown as follows:

```
- name: Update Rule set name of Distributed device
  dellemc_vplex_distributed_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_device_name: "ansible_dist_dev"
    rule_set: "cluster-2-detaches"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed device is present, and rule set provided is different - Success with "changed": True. The details of distributed device name are updated.
2. If distributed device is present, and rule set provided is same (Idempotency) - Success with "changed": False. No change to the distributed device as it is renamed.
3. If distributed device is not present- Failure with "changed": False. Exits with the failure message stating as "Resource not found".

## Get the details of distributed device

To get details of the distributed device, run the appropriate playbook.

### Prerequisite

To get a distributed device, distributed device with same name should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Get details of Distributed device
  dellemc_vplex_distributed_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_device_name: "ansible_dist_dev"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed device is present- Success with "changed": False. Displays the corresponding distributed device details.
2. If distributed device is not present- Failure with "changed": False. Exits with the failure message stating as "Resource not found".

## Delete a distributed device

To delete the distributed device, run the appropriate playbook.

### Prerequisite

1. To delete the distributed device, distributed device with same name should be present in the VPLEX setup.
2. It should not be in a rebuilding state.
3. Its virtual volume should not be exported to storage view, and it should not be part of any distributed consistency group.

The syntax of the task is shown as follows:

```
- name: Delete a Distributed device
  dellemc_vplex_distributed_device:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_device_name: "ansible_dist_dev"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

1. If distributed device is present- Success with "changed": True. Distributed device is deleted.
2. If distributed device is not present (Idempotency) - Success with "changed": False. Exits without fail.
3. If distributed device is in the rebuilding state - Failure with "changed": False. Execution fails with the error message Could not delete the distributed device as it is in rebuilding state.

## Distributed device module parameters

The parameters for the distributed device module are listed.

**Table 7. Parameters for the distributed device module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The user name to access the VPLEX server.
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Mandatory	To validate the SSL certificate: <ul style="list-style-type: none"><li>• True - Verifies the SSL certificate.</li><li>• False - Specified that the SSL certificate should not be verified.</li></ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True".
debug	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dellemc_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
distributed_device_name	-	str	Mandatory	Name of the distributed device. It should not have any special characters, and the length should not exceed more than 63 characters.

**Table 7. Parameters for the distributed device module (continued)**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
source_device	-	str	Optional	It is the name of the source device. It should not have any virtual volume. It is used in creating distributed device.
source_cluster	-	str	Optional	It is the name of the source cluster. It is used in creating distributed device.
target_device	-	str	Optional	It is the name of the target device. It should not have virtual volume and the size of target_device should be greater than or equal to size of the source_device. It is used in creating distributed device.
rule_set	-	str	Optional	To specify the detach rule. It is used in creating distributed device and updating the rule set.
sync	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	To synchronize the data in both source and target devices. It is used as an optional parameter for creating distributed device.
new_distributed_device_name	-	str	Optional	To rename the existing distributed device name. It should not have any special characters, and the length should not exceed more than 63 characters.
state	<ul style="list-style-type: none"> <li>• Present</li> <li>• Absent</li> </ul>	str	Mandatory	To determine whether device exists or not.

## Sample output

### Create a distributed device

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook create_dist_device.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Simple provisioning workflow for Vplex]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Create a Distributed device]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "create_dd": {
    "changed": true,
    "dist_device_details": {
      "capacity": 5369364480,
      "devices": [
        "/vplex/v2/clusters/cluster-1/devices/comb_1",
        "/vplex/v2/clusters/cluster-2/devices/vir_2_1"
      ]
    }
  }
}
```

```

        "geometry": "raid-1",
        "health_indications": [
            "rebuilding"
        ],
        "health_state": "minor-failure",
        "name": "add_test_dd",
        "operational_status": "degraded",
        "rebuild_allowed": true,
        "rebuild_progress": 0,
        "rebuild_status": "rebuilding",
        "rebuild_type": "full",
        "rule_set_name": "cluster-1-detaches",
        "service_status": "running",
        "storage_array_family": "symmetrix",
        "thin_capable": true
    },
    "failed": false
}
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Get a distributed device

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook get_dist_dd.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Simple provisioning workflow for VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get details of Distributed device]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
    "get_dd": {
        "changed": false,
        "dist_device_details": {
            "capacity": 5369364480,
            "devices": [
                "/vplex/v2/clusters/cluster-1/devices/
ansible_virt_vol_dev2020Nov27_090952"
            ],
            "geometry": "raid-1",
            "health_indications": [],
            "health_state": "ok",
            "name": "ansible_test_dd_dev",
            "operational_status": "ok",
            "rebuild_allowed": true,
            "rebuild_status": "done",
            "rebuild_type": "full",
            "rule_set_name": "cluster-2-detaches",
            "service_status": "running",
            "storage_array_family": "symmetrix",

```

```

        "thin_capable": true
    },
    "failed": false
}
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=0    unreachable=0    failed=0

```

## Update the rule set of distributed device

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
update_rule_set.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Simple provisioning workflow for VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Update Rule set name of Distributed device]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "rule_set": {
    "changed": true,
    "dist_device_details": {
      "capacity": 5369364480,
      "devices": [
        "/vplex/v2/clusters/cluster-1/devices/
ansible_virt_vol_dev2020Nov27_090952"
      ],
      "geometry": "raid-1",
      "health_indications": [],
      "health_state": "ok",
      "name": "ansible_test_dd_dev",
      "operational_status": "ok",
      "rebuild_allowed": true,
      "rebuild_status": "done",
      "rebuild_type": "full",
      "rule_set_name": "cluster-2-detaches",
      "service_status": "running",
      "storage_array_family": "symmetrix",
      "thin_capable": true
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Rename a distributed device

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook rename_dist_device.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Simple provisioning workflow for VPlex]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Rename Distributed device]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "rename_dd": {
    "changed": true,
    "dist_device_details": {
      "capacity": 5369364480,
      "devices": [
        "/vpflex/v2/clusters/cluster-1/devices/comb_1",
        "/vpflex/v2/clusters/cluster-2/devices/vir_2_1"
      ],
      "geometry": "raid-1",
      "health_indications": [
        "rebuilding"
      ],
      "health_state": "minor-failure",
      "name": "new_dd_test_name",
      "operational_status": "degraded",
      "rebuild_allowed": true,
      "rebuild_eta": 8,
      "rebuild_progress": 94,
      "rebuild_status": "rebuilding",
      "rebuild_type": "full",
      "rule_set_name": "cluster-1-detaches",
      "service_status": "running",
      "storage_array_family": "symmetrix",
      "thin_capable": true
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0
```

## Delete a distributed device

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook delete_dist_device.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'
```

```

PLAY [Simple provisioning workflow for VPLEX]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Delete Distributed device]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "delete_dd": {
    "changed": true,
    "dist_device_details": null,
    "failed": false
  }
}

PLAY RECAP
*****
localhost                  : ok=3    changed=1    unreachable=0    failed=0

```

## Virtual volume module

The virtual volume module manages the virtual volumes in the VPLEX.

The manage virtual volume module has the following functions:

- Create a virtual volume
- Cache-invalidate on virtual volume
- Get virtual volume by using name/System ID
- Enable remote access of a virtual volume by using name/System ID
- Disable remote access of virtual volume by using name/System ID
- Rename a virtual volume by using name/System ID
- Delete a virtual volume by using name/System ID
- Expand virtual volume with devices by using name/System ID
- Expand virtual volume through backend array storage volume expansion

## Create virtual volume

To create a virtual volume, run the appropriate playbook.

### Prerequisite

To create a virtual volume in a given cluster, underlying supporting device should be present in the VPLEX.

The syntax of the task is shown as follows:

```

- name: Create a virtual volume
  dell EMC vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_name: "ansible_virt_dev_vol"

```



```
supporting_device_name: "ansible_dev"
state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If virtual volume is present - Success with "changed": False. Display the virtual volume details.
2. If supporting device is present - Success with "changed": True. Display the virtual volume details.
3. If supporting device is not present - Failure with "changed": False. Exits with the failure message stating as Could not find the supporting device in cluster-1.
4. If supporting device is not free - Failure with "changed": False. Exits with the failure message stating as The supporting device already has a virtual volume on it.

## Create virtual volume with wait\_for\_rebuild set to False

The syntax is as follows:

```
- name: Create a virtual volume
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_name: "ansible_virt_dev_vol"
    supporting_device_name: "ansible_dev"
    wait_for_rebuild: false
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

## Cache invalidate on virtual volume

To cache-invalidate a virtual volume, run the appropriate playbook.

### Prerequisite

To refresh cache of virtual volume in a given cluster, virtual volume with same name should be present in VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Perform cache invalidate
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    cache_invalidate: true
    virtual_volume_name: "ansible_cache_vir"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If cache invalidate is performed on virtual volume - Success with "changed": True.
2. If the service status of virtual volume is unexported, and cache-invalidate is performed - Success with "changed": False.

3. If the virtual-volume\_name is invalid -Failure with "changed": False. Exits with the failure message stating as "Could not get the virtual\_volume from cluster".
4. If cache invalidate is performed on the virtual volumes where version of VPLEX is more than 6.2 -Failure with "changed": False. Exits with the failure message stating as "To perform cache invalidate the VPLEX version should be 6.2 or lesser".
5. If cache invalidate is performed on virtual volume when the director communication status is not "ok" -Failure with "changed": False. Exits with the failure message stating as "For cache invalidate operation, directors communication status must be ok".

## Get virtual volume using name or System ID

To get the details of virtual volume, run the appropriate playbook.

The syntax of the task is as follows:

### Get details of virtual volume using name

#### Prerequisite

To get virtual volume in a given cluster, virtual volume with same name should be present in VPLEX setup.

```
- name: Get virtual Volume by using name
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_name: "ansible_virt_dev_vol"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": False. Displays the virtual volume details to the user.
2. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster".

### Get details of virtual volume using System ID

#### Prerequisite

To get virtual volume in a given cluster, virtual volume with same ID should be present in VPLEX setup.

```
- name: Get virtual Volume by using ID
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_id: "ansible_virt_id_vol"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": False. Displays the virtual volume details to the user.

2. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster".

## Enable remote access using name or System ID

To enable remote access for a virtual volume, run the appropriate playbook.

The syntax of the task is as follows:

### Enable remote access using name

#### Prerequisite

To enable remote access of virtual volume in a given cluster, virtual volume with same name should be present in VPLEX setup.

```
- name: Enable remote access using name
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_name: "ansible_virt_dev_vol"
    remote_access: "enable"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Changes the "visibility" property of the virtual volume to "global".
2. If the remote access is already enabled for the virtual volume ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster" .

### Enable remote access using System ID

#### Prerequisite

To enable remote access of virtual volume in a given cluster, virtual volume with same ID should be present in VPLEX setup.

```
- name: Enable remote access using ID
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_id: "ansible_virt_id_vol"
    remote_access: "enable"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Changes the "visibility" property of the virtual volume to "global".

2. If the remote access is already enabled for the virtual volume ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster" .

## Disable remote access using name or System ID

To disable remote access for a virtual volume, run the appropriate playbook.

The syntax of the task is as follows:

### Disable remote access using name

#### Prerequisite

To disable remote access of virtual volume in a given cluster, virtual volume with same name should be present in the VPLEX setup.

```
- name: Disable remote access using name
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_name: "ansible_virt_dev_vol"
    remote_access: "disable"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Changes the "visibility" property of the virtual volume to "local".
2. If the remote access is already disabled for the virtual volume ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster".

### Disable remote access using System ID

#### Prerequisite

Virtual Volume for which the remote access to be disabled should be present in the VPLEX, and also the **visibility** property should be in the **global** state.

```
- name: Disable remote access using ID
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_id: "ansible_virt_id_vol"
    remote_access: "disable"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Changes the "visibility" property of the virtual volume to "local".
2. If the remote access is already disabled for the virtual volume ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster".

## Rename virtual volume using name or System ID

To rename a virtual volume, run the appropriate playbook.

The syntax of the task is as follows:

### Rename virtual volume using name

#### Prerequisite

To rename virtual volume in a given cluster, virtual volume with same name should be present in the VPLEX setup

```
- name: Rename virtual volume using name
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_name: "ansible_virt_dev_vol"
    new_virtual_volume_name: "new_ansible_virt_dev_vol"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Changes the virtual volume name with the user new name.
2. If the virtual volume is visible with the new name ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the new name is name of any other virtual volume - Failure with "changed": False. Exits with the failure message stating as "The name is already used by some other virtual volume".
4. If the new name is specified with invalid characters, or more than maximum length of characters - Failure with "changed": False. Exits with the failure message stating as "Invalid characters or length of new name exceeds the maximum size".
5. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster".

### Rename virtual volume using System ID

#### Prerequisite

To rename virtual volume in a given cluster, virtual volume with same ID should be present in the VPLEX setup.

```
- name: Rename virtual volume using ID
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_id: "ansible_virt_id_vol"
```

```
new_virtual_volume_name: "new_ansible_virt_id_vol"
state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Changes the virtual volume name with the user new name.
2. If the virtual volume is visible with the new name ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the new name is name of any other virtual volume - Failure with "changed": False. Exits with the failure message stating as "The name is already used by some other virtual volume" .
4. If the new name is specified with invalid characters, or more than maximum length of characters - Failure with "changed": False. Exits with the failure message stating as "Invalid characters or length of new name exceeds the maximum size".
5. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster" .

## Expand virtual volume with device using name or System ID

To expand a virtual volume with a device, run the appropriate playbook. This type of expansion is supported only for VPLEX 6.2.

The syntax of the task is as follows:

### Expand virtual volume with device using name

#### Prerequisite

To expand virtual volume in a given cluster, virtual volume with same name and devices in additional devices list should be present in the VPLEX setup.

```
- name: Expand virtual volume with Device
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_name: "ansible_virt_dev_vol"
    expand: true
    additional_devices: ["ansible_add_dev"]
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Expands the virtual volume with the additional device capacity.
2. If the virtual volume is already expanded with the same additional device ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the additional device is used through another virtual volume - Failure with "changed": False. Exits with the failure message stating as "Device is already used by another virtual volume".
4. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster".

## Expand virtual volume with device using System ID

### Prerequisite

To expand virtual volume in a given cluster, virtual volume with same ID and devices in additional devices list should be present in the VPLEX setup.

```
- name: Expand virtual volume with Device
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_id: "ansible_virt_id_vol"
    expand: true
    additional_devices: ["ansible_add_id_dev"]
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Expands the virtual volume with the additional device capacity.
2. If the virtual volume is already expanded with the same additional device ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the additional device is used through another virtual volume - Failure with "changed": False. Exits with the failure message stating as "Device is already used by another virtual volume".
4. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume <virtual\_volume\_name> from the specific cluster".

## Expand virtual volume through backend storage volume expansion

To expand a virtual volume after the expansion of backend array storage volume, run the appropriate playbook.

The syntax of the task is as follows:

## Expand virtual volume through backend storage volume expansion using name

### Prerequisite

To expand virtual volume in a given cluster through backend storage, virtual volume with the same name and with the expandable capacity greater than 0 should be present in the VPLEX setup.

```
- name: Expand virtual volume using backend storage volume expansion
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_name: "ansible_virt_dev_vol"
    expand: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Expands the virtual volume with the sum of original capacity and the expandable\_capacity.
2. If the virtual volume is already with 0 bytes in the expandable\_capacity ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as Could not get virtual volume <virtual\_volume\_name> from the specific cluster.

## Expand virtual volume through backend storage volume expansion using System ID

### Prerequisite

To expand virtual volume in a given cluster through backend storage, virtual volume with the same ID and with the expandable capacity greater than 0 should be present in the VPLEX setup.

```
- name: Expand virtual volume using backend storage volume expansion
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_id: "ansible_virt_id_vol"
    expand: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Expands the virtual volume with the sum of original capacity and the expandable\_capacity.
2. If the virtual volume is already with 0 bytes in the expandable\_capacity ( Idempotency ) - Success with "changed": False. No changes happen to the virtual volume.
3. If the virtual volume is absent - Failure with "changed": False. Exits with the failure message stating as Could not get virtual volume <virtual\_volume\_name> from the specific cluster.

## Delete virtual volume using name or System ID

To delete a virtual volume, run the appropriate playbook.

The syntax of the task is as follows:

### Delete virtual volume using name

#### Prerequisite

To delete virtual volume in a given cluster, virtual volume with same name should be present in the VPLEX setup.

```
- name: Delete virtual volume using name
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_name: "ansible_virt_dev_vol"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result



On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Deletes the virtual volume from the VPLEX.
2. If the virtual volume is used in any of consistency groups or storage views - Failure with "changed": False. Exits with the failure message stating as "Virtual Volume is used by <another storage entity>".
3. If the virtual volume is absent - Success with "changed": False. No changes in the VPLEX as the virtual volume is not present as expected.

## Delete virtual volume using System ID

### Prerequisite

To delete virtual volume in a given cluster, virtual volume with same ID should be present in the VPLEX setup.

```
- name: Delete virtual volume using ID
  dellemc_vplex_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    virtual_volume_id: "ansible_virt_id_vol"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volume is present - Success with "changed": True. Deletes the virtual volume from the VPLEX.
2. If the virtual volume is used in any of consistency groups or storage views - Failure with "changed": False. Exits with the failure message stating as "Virtual Volume is used by <another storage entity>".
3. If the virtual volume is absent - Success with "changed": False. No changes in the VPLEX as the virtual volume is not present as expected.


## Virtual volume module parameters

The parameters for the manage virtual volume module are listed.

**Table 8. Parameters for manage virtual volume module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	User name to access the VPLEX server.
vplexpassword	-	str	Mandatory	Password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Mandatory	To validate the SSL certificate. If it is True, it verifies the SSL certificate. If it is False, it does not verify the SSL certificate.
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dellemc_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	-	str	Mandatory	Name of the cluster.

**Table 8. Parameters for manage virtual volume module (continued)**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
virtual_volume_name	-	str	Optional	Name of a specific instance of the resource virtual volume.
virtual_volume_id	-	str	Optional	ID of a specific virtual volume.
new_virtual_volume_name	-	str	Optional	The new name for renaming virtual volume.
supporting_device_name	-	str	Optional	The name of the supporting device on which virtual volume is created.
thin_enable	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul> The default value is True.	bool	Optional	To update thin enable value, while creating virtual volume. It is used in creating virtual volume.
wait_for_rebuild	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul> The default value is True.	bool	Optional	To specify the creation of virtual volume during the rebuild is in progress on the supporting_device. If set to <b>True</b> , then the creation of virtual volume fails during the rebuild is in progress on the supporting_device, if set to <b>False</b> , it proceeds with the creation of the virtual volume irrespective of the rebuild state.
expand	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	To perform the expand operation on the virtual volume.
remote_access	<ul style="list-style-type: none"> <li>• Enable</li> <li>• Disable</li> </ul>	str	Optional	To specify either to enable or disable remote access.
additional_devices	-	list	Optional	Target device list to expand virtual volume. <div>  <b>NOTE:</b> Virtual volume expand operation is not supported in release 1.1.                         </div>
cache_invalidate	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	To perform cache invalidate operation on the virtual volume.
state	<ul style="list-style-type: none"> <li>• Present</li> <li>• Absent</li> </ul>	str	Mandatory	The state of a specific virtual volume. For delete virtual volume state is absent. For remaining operations state should be present.

## Sample output

### Create virtual volume

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook create_virtual_volume_tests.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Virtual Volume module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
```

```

ok: [localhost]

TASK [create virtual volume]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "create_vol": {
    "changed": true,
    "failed": false,
    "storage_details": {
      "additional_devs": [],
      "block_count": 1310880,
      "block_size": 4096,
      "capacity": 5369364480,
      "expandable": true,
      "expandable_capacity": 0,
      "expansion_method": "storage-volume",
      "health_indications": [],
      "health_state": "ok",
      "locality": "local",
      "mirrors": [],
      "name": "ansible_virt_vol_dev_vol",
      "operational_status": "ok",
      "service_status": "unexported",
      "storage_array_family": "symmetrix",
      "supporting_device": "/vplex/v2/clusters/cluster-1/devices/ansible_dev_1",
      "system_id": "ansible_dev_1_vol",
      "thin_enabled": "enabled",
      "visibility": "local",
      "vpd_id": "VPD83T3:6000144000000010f0124b3da38e31f5"
    }
  }
}

PLAY RECAP
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Cache invalidate on virtual volume

```

(demo) [root@localhost playbooks]# ansible-playbook
cache_6in.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Virtual Volume module operations on VPLEX]
**

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [cache invalidate virtual volume]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "create_vol": {
    "changed": true,

```

```

        "failed": false,
        "storage_details": {
            "additional_devs": [],
            "block_count": 524640,
            "block_size": 4096,
            "capacity": 2148925440,
            "expandable": true,
            "expandable_capacity": 0,
            "expansion_method": "storage-volume",
            "health_indications": [],
            "health_state": "ok",
            "locality": "local",
            "mirrors": [],
            "name": "ansible_vir_2",
            "operational_status": "ok",
            "recoverpoint_protection_at": [],
            "service_status": "inactive",
            "storage_array_family": "symmetrix",
            "supporting_device": "/vplex/v2/clusters/cluster-1/devices/vplex_",
            "system_id": "ansible_vir_2",
            "thin_enabled": "disabled",
            "visibility": "local",
            "vpd_id": "VPD83T3:6000144000000010f0124b3da38e29ba"
        }
    }
}

```

#### PLAY RECAP

```

*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

### Get virtual volume

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook get_vv.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

```

```

[WARNING]: No inventory was parsed, only implicit localhost is available

```

```

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

```

#### PLAY [Perform Virtual Volume module operations on VPLEX]

```

*****
*****

```

##### TASK [Gathering Facts]

```

*****
*****

```

```

ok: [localhost]

```

##### TASK [Get Virtual Volume]

```

*****
*****

```

```

ok: [localhost]

```

##### TASK [debug]

```

*****
*****

```

```

ok: [localhost] => {
    "vol_details": {
        "changed": false,
        "failed": false,
        "storage_details": {
            "additional_devs": [],
            "block_count": 524640,
            "block_size": 4096,
            "capacity": 2148925440,
            "consistency_group": "/vplex/v2/clusters/cluster-1/consistency_groups/
Yash_3",
            "expandable": true,

```

```

        "expandable_capacity": 0,
        "expansion_method": "storage-volume",
        "health_indications": [],
        "health_state": "ok",
        "locality": "local",
        "mirrors": [],
        "name": "ansible_vol_2",
        "operational_status": "ok",
        "service_status": "unexported",
        "storage_array_family": "symmetrix",
        "supporting_device": "/vplex/v2/clusters/cluster-1/devices/
dev_new_123_extent_0",
        "system_id": "ansible_vol_2",
        "thin_enabled": "disabled",
        "visibility": "local"
    }
}
}

```

#### PLAY RECAP

```

*****
*****
localhost                : ok=3    changed=0    unreachable=0    failed=0

```

### Enable remote access of virtual volume

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook remote_access.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Virtual Volume module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Enable Virtual Volume remote access]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "remote_access": {
    "changed": true,
    "failed": false,
    "storage_details": {
      "additional_devs": [],
      "block_count": 524640,
      "block_size": 4096,
      "capacity": 2148925440,
      "consistency_group": "/vplex/v2/clusters/cluster-1/consistency_groups/
Yash_3",
      "expandable": true,
      "expandable_capacity": 0,
      "expansion_method": "storage-volume",
      "health_indications": [],
      "health_state": "ok",
      "locality": "local",
      "mirrors": [],
      "name": "ansible_vol_2",
      "operational_status": "ok",
      "service_status": "unexported",

```

```

        "storage_array_family": "symmetrix",
        "supporting_device": "/vplex/v2/clusters/cluster-1/devices/
dev_new_123_extent_0",
        "system_id": "ansible_vol_2",
        "thin_enabled": "disabled",
        "visibility": "global"
    }
}
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Disable remote access of virtual volume

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
disable_remote_access.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Virtual Volume module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Disable Remote Access of Virtual Volume]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
    "disable_remote_access": {
        "changed": true,
        "failed": false,
        "storage_details": {
            "additional_devs": [],
            "block_count": 524640,
            "block_size": 4096,
            "capacity": 2148925440,
            "consistency_group": "/vplex/v2/clusters/cluster-1/consistency_groups/
Yash_3",
            "expandable": true,
            "expandable_capacity": 0,
            "expansion_method": "storage-volume",
            "health_indications": [],
            "health_state": "ok",
            "locality": "local",
            "mirrors": [],
            "name": "ansible_vol_2",
            "operational_status": "ok",
            "service_status": "unexported",
            "storage_array_family": "symmetrix",
            "supporting_device": "/vplex/v2/clusters/cluster-1/devices/
dev_new_123_extent_0",
            "system_id": "ansible_vol_2",
            "thin_enabled": "disabled",
            "visibility": "local"
        }
    }
}

```

```

}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Rename virtual volume

```

(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook rename_virtual_volume_tests.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Virtual Volume module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
*****
ok: [localhost]

TASK [Rename Virtual Volume]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
*****
ok: [localhost] => {
  "rename_vol": {
    "changed": true,
    "failed": false,
    "storage_details": {
      "additional_devs": [],
      "block_count": 1310880,
      "block_size": 4096,
      "capacity": 5369364480,
      "expandable": true,
      "expandable_capacity": 0,
      "expansion_method": "storage-volume",
      "health_indications": [],
      "health_state": "ok",
      "locality": "local",
      "mirrors": [],
      "name": "ansible_virt_vol_dev_vol_new",
      "operational_status": "ok",
      "service_status": "unexported",
      "storage_array_family": "symmetrix",
      "supporting_device": "/vplex/v2/clusters/cluster-1/devices/ansible_dev_1",
      "system_id": "ansible_virt_vol_dev_vol",
      "thin_enabled": "enabled",
      "visibility": "local",
      "vpd_id": "VPD83T3:6000144000000010f0124b3da38e31f5"
    }
  }
}

PLAY RECAP
*****
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Delete virtual volume

```
(py3_ans2 7) [root@dsvej252 playbooks]# ansible-playbook delete_virtual_volume_tests.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Virtual Volume module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
*****
ok: [localhost]

TASK [Delete Virtual Volume]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
*****
ok: [localhost] => {
  "del_vol": {
    "changed": true,
    "failed": false,
    "storage_details": {}
  }
}

PLAY RECAP
*****
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0
```

## Distributed virtual volume module

The distributed virtual volume module manages the distributed virtual volumes in the VPLEX metro setup.

The manage distributed virtual volume module has the following functions:

- Create a distributed virtual volume
- Get distributed virtual volume by using name/system ID
- Rename distributed virtual volume by using name/system ID
- Expand distributed virtual volume by using name/system ID
- Delete distributed virtual volume by using name/system ID

## Create distributed virtual volume

To create a distributed virtual volume, run the appropriate playbook.

### Prerequisite

1. To create a distributed virtual volume, the provided name should not be present in the VPLEX.
2. Distributed device should be present in the VPLEX, and it should not have any distributed virtual volume on top of it.
3. The distributed virtual volume name should not have length more than 63 characters, and it should not have any special characters other than - and \_.



The syntax of the task is shown as follows:

## Create distributed virtual volume with wait\_for\_rebuild set to True

```
- name: Create Distributed virtual volume
  dellemc_vplex_distributed_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_device_name: "ansible_test_dev"
    thin_enable: true
    wait_for_rebuild: true
    distributed_virtual_volume_name: "ansible_dist_vv"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

## Create distributed virtual volume with wait\_for\_rebuild set to False

```
- name: Create Distributed virtual volume
  dellemc_vplex_distributed_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_device_name: "ansible_test_dev"
    thin_enable: true
    wait_for_rebuild: false
    distributed_virtual_volume_name: "ansible_dist_vv"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed virtual volume is created - Success with "changed": True. The distributed virtual volume is created.
2. If distributed virtual volume is already created (Idempotency) - Success with "changed": False. No change in the distributed virtual volume as it is created.
3. If distributed device is used through another distributed virtual volume - Failure with "changed": False. Exits with the failure message stating "Could not create distributed virtual volume as distributed device as virtual volume on top of it".
4. If distributed device is invalid - Failure with "changed": False. Exits with the failure message stating "Could not get the distributed device".

## Get distributed virtual volume using name or System ID

### Get distributed virtual volume using name

To get a distributed virtual volume details using name, run the appropriate playbook.

#### Prerequisite

The distributed virtual volume should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Get Distributed virtual volume by name
  dellemc_vplex_distributed_virtual_volume:
```

```

vplexhost: "{{ vplexhost }}"
vplexuser: "{{ vplexuser }}"
vplexpassword: "{{ vplexpassword }}"
verifycert: "{{ verifycert }}"
distributed_virtual_volume_name: "ansible_dist_vv"
state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the distributed virtual volume is present- Success with "changed": False. Displays the corresponding distributed virtual volume details.
2. If the distributed virtual volume name is invalid- Failure with "changed": False. Execution fails with error message "Could not get the distributed virtual volume".

## Get distributed virtual volume using System ID

To get a distributed virtual volume details using System ID, run the appropriate playbook.

### Prerequisite

The distributed virtual volume should be present in the VPLEX.

The syntax of the task is as follows:

```

- name: Get Distributed virtual volume by system ID
  dellemc_vplex_distributed_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_virtual_volume_id: "ansible_dist_id"
    state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the distributed virtual volume is present- Success with "changed": False. Displays the corresponding distributed virtual volume details.
2. If the distributed virtual volume id is invalid- Failure with "changed": False. Execution fails with error message "Could not get the distributed virtual volume".

## Rename a distributed virtual volume using name or System ID

### Rename a distributed virtual volume using name

To rename a distributed virtual volume name with valid name using distributed virtual volume name as input parameter, run the appropriate playbook.

### Prerequisite

1. The distributed virtual volume should be present in the VPLEX.
2. A new name should not be present in the VPLEX and should not have special characters other than - or \_ . Also, the length should not be greater than 63 characters.

The syntax of the task is as follows:

```

- name: Rename Distributed virtual volume using name
  dellemc_vplex_distributed_virtual_volume:

```

```

vplexhost: "{{ vplexhost }}"
vplexuser: "{{ vplexuser }}"
vplexpassword: "{{ vplexpassword }}"
verifycert: "{{ verifycert }}"
distributed_virtual_volume_name: "ansible_dist_vv"
new_distributed_virtual_volume_name: "ansible_upd_dist_vv"
state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed virtual volume is renamed to a new name- Success with "changed": True. The distributed virtual volume is renamed.
2. If the name of distributed virtual volume and a new name are same (Idempotency) - Success with "changed": False. Distributed virtual volume is already displayed with the same new name.
3. If the new name of distributed virtual volume is invalid - Failure with "changed": False. Execution fails with error message "Could not rename the distributed virtual volume".
4. If the new name of distributed virtual volume is used in the VPLEX - Failure with "changed": False. Execution fails with error message "Distributed virtual volume with the new name already exists".

## Rename a distributed virtual volume System ID

To rename a distributed virtual volume name with valid name using system ID as input parameter, run the appropriate playbook.

### Prerequisite

1. The distributed virtual volume should be present in the VPLEX.
2. A new name should not be present in the VPLEX and should not have special characters other than - or \_ . Also, the length should not be greater than 63 characters.

The syntax of the task is as follows:

```

- name: Rename Distributed virtual volume using system ID
  dell EMC vplex distributed_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_virtual_volume_id: "ansible_dist_vv_id"
    new_distributed_virtual_volume_name: "ansible_upd_dist_vv_id"
    state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed virtual volume is renamed to a new name- Success with "changed": True. The distributed virtual volume is renamed.
2. If the name of distributed virtual volume and a new name are same (Idempotency) - Success with "changed": False. Distributed virtual volume is already displayed with the same new name.
3. If the new name of distributed virtual volume is invalid - Failure with "changed": False. Execution fails with error message "Could not rename the distributed virtual volume".
4. If the new name of distributed virtual volume is used in the VPLEX - Failure with "changed": False. Execution fails with error message "Distributed virtual volume with the new name already exists".

# Expand distributed virtual volume using name or System ID

## Expand distributed virtual volume using name

To expand a distributed virtual volume name using distributed virtual volume name as input parameter, run the appropriate playbook.

### Prerequisite

Distributed virtual volume on which the operation is to be performed should have the **expandable\_capacity > 0** bytes and the operation can be performed through giving the valid distributed\_virtual\_volume\_name.. The expansion is possible only by performing back-end storage volume expansion. Use the Ansible module of the corresponding storage array that contains the storage volumes on which the distributed virtual volume has been created.

The syntax of the task is as follows:

```
- name: Expand Distributed virtual volume using name
  dell EMC_vplex_distributed_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_virtual_volume_name: "ansible_dist_vv"
    expand: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Expand distributed virtual volume through giving the valid name - Success with "changed": True. Expands the distributed virtual volume with the capacity visible in "expandable\_capacity" field along with the original size.
2. Expand distributed virtual volume through giving the valid name ( Idempotency ) - Success with "changed": False. If the "expandable\_capacity" is 0 bytes, then no expand operation happens to the distributed virtual volume and it remains unchanged.
3. Expand distributed virtual volume through giving the name which is invalid - Failure with "changed": False. Exits with failure message stating as "Could not find the distributed virtual volume".

## Expand distributed virtual volume using System ID

To expand a distributed virtual volume name using System ID as input parameter, run the appropriate playbook.

### Prerequisite

Distributed virtual volume on which the operation is to be performed should have the **expandable\_capacity > 0** bytes and the operation can be performed through giving the valid distributed\_virtual\_volume\_id. The expansion is possible only by performing back-end storage volume expansion. Use the Ansible module of the corresponding storage array that contains the storage volumes on which the distributed virtual volume has been created.

The syntax of the task is as follows:

```
- name: Expand Distributed virtual volume using system ID
  dell EMC_vplex_distributed_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_virtual_volume_id: "ansible_dist_vv_id"
    expand: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Expand distributed virtual volume through giving the valid ID - Success with "changed": True. Expands the distributed virtual volume with the capacity visible in "expandable\_capacity" field along with the original size.
2. Expand distributed virtual volume through giving the valid ID ( Idempotency ) - Success with "changed": False. If the "expandable\_capacity" is 0 bytes, then no expand operation happens to the distributed virtual volume and it remains unchanged.
3. Expand distributed virtual volume through giving invalid ID - Failure with "changed": False. Exits with failure message stating as "Could not find the distributed virtual volume" .

## Delete distributed virtual volume using name or System ID

To delete a distributed virtual volume using distributed virtual volume name as input parameter, run the appropriate playbook.

### Delete distributed virtual volume using name

#### Prerequisite

1. The distributed virtual volume should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Delete Distributed virtual volume using name
  dellemc_vplex_distributed_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_virtual_volume_name: "ansible_dist_vv"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed virtual volume is deleted- Success with "changed": True. The distributed virtual volume is deleted.
2. If distributed virtual volume is already deleted, or does not exist in the VPLEX (Idempotency) - Success with "changed": False. Distributed virtual volume is deleted.
3. If the distributed virtual volume is used in any of distributed consistency groups or storage views - Failure with "changed": False. Exits with the failure message stating as Distributed Virtual Volume is used by <another storage entity>.

### Delete distributed virtual volume using System ID

To delete a distributed virtual volume using system ID as input parameter, run the appropriate playbook.

#### Prerequisite

1. The distributed virtual volume should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Delete Distributed virtual volume using system ID
  dellemc_vplex_distributed_virtual_volume:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_virtual_volume_id: "ansible_dist_vv_id"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed virtual volume is deleted- Success with "changed": True. The distributed virtual volume is deleted.
2. If distributed virtual volume is already deleted, or does not exist in the VPLEX (Idempotency) - Success with "changed": False. Distributed virtual volume is deleted.
3. If the distributed virtual volume is used in any of distributed consistency groups or storage views - Failure with "changed": False. Exits with the failure message stating as Distributed Virtual Volume is used by <another storage entity>.

## Distributed virtual volume module parameters

The parameters for the manage distributed virtual volume module are listed.

**Table 9. Parameters for the manage distributed virtual volume module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The user name to access the VPLEX server.
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"> <li>• True - Verifies the SSL certificate</li> <li>• False - Specified that the SSL certificate should not be verified</li> </ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dell EMC_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
distributed_virtual_volume_name	-	str	Optional	The name of a specific distributed virtual volume. For all the operations, it can be used. This parameter is mutually exclusive with distributed_virtual_volume_id.
distributed_device_name	-	str	Optional	The name of specific distributed device on which virtual volume should be created. It is used for creating distributed virtual volume. It should not have virtual volume above it.
thin_enable	-	bool	Mandatory	To update thin enable value, while creating distributed virtual volume. It is a boolean value.
distributed_virtual_volume_id	-	str	Optional	The system ID of a specific distributed virtual volume. It is used to perform operations on distributed virtual volume based on system ID. It is mutually exclusive with distributed_virtual_volume_name.
wait_for_rebuild	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul> The default value is True.	bool	Optional	To specify the creation of virtual volume when rebuild is in progress on the distributed device. If set to 'True', then the creation of virtual volume fails during the rebuild is in progress on the distributed device. If set to 'False', it proceeds with the creation of the virtual volume irrespective of the rebuild state.

**Table 9. Parameters for the manage distributed virtual volume module (continued)**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
new_distributed_virtual_volume_name	-	str	-	The new name of the distributed virtual volume. The new_distributed_virtual_volume_name can only contains letters, numbers _ or - and fewer than 63 characters.
expand	-	bool	-	The expand operation on distributed volume name happens only on this parameter is set. It is a Boolean value - true, performs the expand operation The expand operation on the specified volume happens only when "expandable_capacity" value is "greater than 0 bytes"
state	<ul style="list-style-type: none"> <li>Present</li> <li>Absent</li> </ul>	str	-	To specify which operation to be done on a distributed virtual volume. To delete, it should be absent. For all other operations it should be present. It takes two values either: <ul style="list-style-type: none"> <li>Present</li> <li>Absent</li> </ul>

## Sample output

### Create distributed virtual volume

```
[root@centos76 playbooks]# ansible-playbook create_dist_vv.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Distributed Virtual Volumes of Vplex]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Create a distributed virtual volume]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "create_dist_vv": {
    "changed": true,
    "dist_vv_details": {
      "block_count": 1310880,
      "block_size": 4096,
      "capacity": 5369364480,
      "expandable": true,
      "expandable_capacity": 0,
      "expansion_method": "storage-volume",
      "health_indications": [],
      "health_state": "ok",
      "locality": "distributed",
      "name": "ansible_test_dd_dev_vol",
      "operational_status": "ok",
      "recoverpoint_protection_at": [],
      "service_status": "unexported",
      "storage_array_family": "symmetrix",
      "supporting_device": "/vplex/v2/distributed_storage/distributed_devices/
```

```

    ansible_test_dd_dev",
      "system_id": "ansible_test_dd_dev_vol",
      "thin_enabled": "enabled",
      "visibility": "global",
      "vpd_id": "VPD83T3:6000144000000010f0124b3da38e31e8"
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Get distributed virtual volume

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook get_dist_vv.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Distributed Virtual Volumes of Vplex]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get details of distributed virtual volume by using its name]
*****
**
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "get_dist_vv_name": {
    "changed": false,
    "dist_vv_details": {
      "block_count": 1310880,
      "block_size": 4096,
      "capacity": 5369364480,
      "consistency_group": "/vplex/v2/distributed_storage/
distributed_consistency_groups/test_123",
      "expandable": true,
      "expandable_capacity": 0,
      "expansion_method": "storage-volume",
      "health_indications": [],
      "health_state": "ok",
      "locality": "distributed",
      "name": "vir_vol_1",
      "operational_status": "ok",
      "service_status": "unexported",
      "storage_array_family": "symmetrix",
      "supporting_device": "/vplex/v2/distributed_storage/distributed_devices/
test1_dd",
      "system_id": "ansible_update_dist_vtvfff",
      "thin_enabled": "enabled",
      "visibility": "global",
      "vpd_id": "VPD83T3:6000144000000010f0124b3da38e2810"
    },
    "failed": false
  }
}

```



```
PLAY RECAP
*****
*****
localhost                : ok=3    changed=0    unreachable=0    failed=0
```

## Expand distributed virtual volume

```
[root@localhost playbooks]# ansible-playbook expand_dist_vv.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Distributed Virtual Volumes of Vplex]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Expand distributed virtual volume by using its name]
*****
ok: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "expand_dist_vv_name": {
    "changed": true,
    "dist_vv_details": {
      "block_count": 5243040,
      "block_size": 4096,
      "capacity": 21475491840,
      "expandable": true,
      "expandable_capacity": 0,
      "expansion_method": "storage-volume",
      "health_indications": [],
      "health_state": "ok",
      "locality": "distributed",
      "name": "ansible_test_dist_dev_vol",
      "operational_status": "ok",
      "service_status": "unexported",
      "storage_array_family": "symmetrix",
      "supporting_device": "/vplex/v2/distributed_storage/distributed_devices/
ansible_test_dist_dev",
      "system_id": "ansible_test_dist_dev_vol",
      "thin_enabled": "enabled",
      "visibility": "global",
      "vpd_id": "VPD83T3:600014400000010f0124b3da38e222c"
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

## Rename distributed virtual volume

```
[root@centos76 playbooks]# ansible-playbook rename_dist_vv.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Distributed Virtual Volumes of Vplex]
```

```

*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Rename distributed virtual volume by using its name]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "rename_dist_vv_name": {
    "changed": true,
    "dist_vv_details": {
      "block_count": 1310880,
      "block_size": 4096,
      "capacity": 5369364480,
      "expandable": true,
      "expandable_capacity": 0,
      "expansion_method": "storage-volume",
      "health_indications": [],
      "health_state": "ok",
      "locality": "distributed",
      "name": "ansible_update_dist_vv",
      "operational_status": "ok",
      "recoverpoint_protection_at": [],
      "service_status": "unexported",
      "storage_array_family": "symmetrix",
      "supporting_device": "/vplex/v2/distributed_storage/distributed_devices/
ansible_test_dd_dev",
      "system_id": "ansible_test_dd_dev_vol",
      "thin_enabled": "enabled",
      "visibility": "global",
      "vpd_id": "VPD83T3:6000144000000010f0124b3da38e31e8"
    },
    "failed": false
  },
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Delete distributed virtual volume

```

[root@centos76 playbooks]# ansible-playbook delete_dist_vv.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Distributed Virtual Volumes of Vplex]
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Delete distributed virtual volume by using its name]
*****
*****
changed: [localhost]

TASK [debug]

```

```

*****
*****
ok: [localhost] => {
  "delete_dist_vv_name": {
    "changed": true,
    "dist_vv_details": null,
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Consistency group module

The consistency group module manages the consistency groups in VPLEX.

The consistency group module has the following functionalities:

- Create a consistency group
- Add virtual volumes to the consistency group
- Remove virtual volumes from consistency group
- Rename consistency group
- Delete a consistency group
- Get consistency group

## Create a consistency group

To create a consistency group, run the appropriate playbook.

### Prerequisite

1. The consistency group name should not be present in the VPLEX.
2. It should not have special characters in its name and length should not be more than 63 characters.

The syntax of the task is as follows:

```

- name: Create CG
  dell EMC_vplex_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    cg_name: "ansible_cg"
    state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the consistency group name is not present in VPLEX - Success with "changed": True. Consistency group is created.
2. If there is a consistency group with same name (Idempotency)- Success with "changed": False. Consistency group is already created.
3. If the name is invalid, or contain more than 63 characters, or contain special characters - Failure with "changed": False. Execution fails with error message "Could not create the consistency group".

## Add virtual volumes to consistency group

To add the virtual volumes to the consistency group, run the appropriate playbook.

### Prerequisite

1. The consistency group name should be present in the VPLEX.
2. The virtual volumes should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Add virtual volumes to CG
  dellemc_vplex_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    cg_name: "ansible_cg"
    virtual_volumes: "ansible_vv_1"
    virtual_volume_state: "present-in-cg"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the consistency group name and the virtual volumes are present in VPLEX - Success with "changed": True. Virtual volumes are added to the consistency group.
2. If the virtual volumes are already present in the consistency group (Idempotency) - Success with "changed": False. Virtual volumes are already added to consistency group.
3. If the consistency group name is invalid or virtual volumes are invalid or used - Failure with "changed": False. Execution fails with error message "Could not add virtual volumes to the consistency group".

## Remove virtual volumes from consistency group

To remove the virtual volumes from the consistency group, run the appropriate playbook.

### Prerequisite

1. The consistency group name should be present in the VPLEX.
2. The virtual volumes should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Remove virtual volumes from CG
  dellemc_vplex_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    cg_name: "ansible_cg"
    virtual_volumes: "ansible_vv_1"
    virtual_volume_state: "absent-in-cg"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the virtual volumes are present in the consistency group in VPLEX - Success with "changed": True. Virtual volumes are removed from consistency group.

2. If the virtual volumes are not present in the consistency group (Idempotency) - Success with "changed": False. Virtual volumes are not present in the consistency group.
3. If consistency group name is invalid, or virtual volumes are invalid - Failure with "changed": False. Execution fails with error message "Could not remove virtual volumes to the consistency group".

## Rename consistency group

To rename the consistency group, run the appropriate playbook.

1. The consistency group name should be present in the VPLEX.
2. A new name should not be present in the VPLEX and should not have special characters. Also, the length should not be greater than 63 characters.

### Prerequisite

The syntax of the task is as follows:

```
- name: Rename CG
  dellemc_vplex_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    cg_name: "ansible_cg"
    new_cg_name: "ansible_cg_new"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the consistency group name and new name are different, and new name does not exist in the VPLEX - Success with "changed": True. Consistency group is renamed.
2. If the consistency group name and new name are same (Idempotency) - Success with "changed": False. Consistency group is already displayed with the same new name.
3. If new consistency group name is invalid - Failure with "changed": False. Execution fails with error message "Could not rename the consistency group".

## Delete consistency group

To delete the consistency group, run the appropriate playbook.

### Prerequisite

The consistency group name should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Delete CG
  dellemc_vplex_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    cg_name: "ansible_cg"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If consistency group is present- Success with "changed": True. Consistency group is deleted.
2. If consistency group is not present (Idempotency) - Success with "changed": False. Consistency group is not present in the VPLEX.
3. If consistency group name is invalid - Failure with "changed": False. Consistency group is not present in the VPLEX.

## Get consistency group

To get the consistency group, run the appropriate playbook.

### Prerequisite

The consistency group name should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Get CG from cluster
  dellemc_vplex_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    cg_name: "ansible_cg"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If consistency group is present- Success with "changed": False. Displays the corresponding consistency group details.
2. If consistency group is not present- Failure with "changed": False. Execution fails with error message "Could not get details of consistency group".

## Consistency group module parameters

Parameters for the consistency group module are listed.

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

**Table 10. Parameters for the consistency group module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameter	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The user name to access the VPLEX server.
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"> <li>• True - Verifies the SSL certificate</li> <li>• False - Specified that the SSL certificate should not be verified</li> </ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dellemc_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.

**Table 10. Parameters for the consistency group module (continued)**

Parameter name	Choice or default	Type	Mandatory/Optional Parameter	Description
cluster_name	-	str	Mandatory	Name of the cluster.
cg_name	-	str	Mandatory	Name of the consistency group. The consistency group name can only contains letters, numbers _ or - and fewer than 63 characters.
virtual_volumes	-	list	-	List of virtual volumes to add/remove from the consistency group.
virtual_volume_state	<ul style="list-style-type: none"> <li>present-in-cg</li> <li>absent-in-cg</li> </ul>	str	-	To determine whether add /remove virtual volumes. <ul style="list-style-type: none"> <li>present-in-cg - Add virtual volumes to the consistency group.</li> <li>absent-in-cg - Remove virtual volumes from the consistency group.</li> </ul>
new_cg_name	-	str	Optional	The new name of the consistency group. It is required to be specified while re-naming the consistency group. The new_cg_name can only contains letters, numbers _ or - and fewer than 63 characters.
state	<ul style="list-style-type: none"> <li>present</li> <li>absent</li> </ul>	str	Mandatory	To determine whether consistency group exists or not. <ul style="list-style-type: none"> <li>present - The consistency group must be present in the system.</li> <li>absent - The consistency group must not be present in the system.</li> </ul>

## Sample output

### Create a consistency group

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook create_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Consistency group operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Create Consistency group]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "create_cg": {
    "cg_details": {
      "auto_resume_at_loser": true,
      "name": "ansible_cg",
      "operational_status": [
        {
```

```

        "cluster": "cluster-1",
        "details": [],
        "summary": "ok"
    },
    {
        "cluster": "cluster-2",
        "details": [],
        "summary": "unknown"
    }
],
"read_only": false,
"storage_at_clusters": [],
"virtual_volumes": [],
"visibility": [
    "/vplex/v2/clusters/cluster-1"
]
},
"changed": true,
"failed": false
}
}

```

#### PLAY RECAP

```

*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

### Get a consistency group

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook get_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

```

```

[WARNING]: No inventory was parsed, only implicit localhost is available

```

```

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

```

#### PLAY [Testing Consistency group operations]

```

*****
*****

```

#### TASK [Gathering Facts]

```

*****
*****

```

```

ok: [localhost]

```

#### TASK [Get Consistency group]

```

*****
*****

```

```

ok: [localhost]

```

#### TASK [debug]

```

*****
*****

```

```

ok: [localhost] => {
  "get_cg": {
    "cg_details": {
      "auto_resume_at_loser": true,
      "name": "ansible_cg",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "unknown"
        }
      ],
      "read_only": false,

```



```

        "storage_at_clusters": [],
        "virtual_volumes": [],
        "visibility": [
            "/vplex/v2/clusters/cluster-1"
        ]
    },
    "changed": false,
    "failed": false
}
}

PLAY RECAP
*****
*****
localhost                : ok=4    changed=0    unreachable=0    failed=0

```

## Add virtual volumes to a consistency group

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook add_vv_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Consistency group operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Build a list of virtual volumes]
*****
*****
ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Add virtual volumes to Consistency group]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
    "add_vol_cg": {
        "cg_details": {
            "auto_resume_at_loser": true,
            "name": "ansible_cg",
            "operational_status": [
                {
                    "cluster": "cluster-1",
                    "details": [],
                    "summary": "ok"
                },
                {
                    "cluster": "cluster-2",
                    "details": [],
                    "summary": "unknown"
                }
            ],
            "read_only": false,
            "storage_at_clusters": [],
            "virtual_volumes": [
                "/vplex/v2/clusters/cluster-1/virtual_volumes/vir_1",
                "/vplex/v2/clusters/cluster-1/virtual_volumes/vir_2"
            ]
        }
    }
}

```

```

        "visibility": [
            "/vplex/v2/clusters/cluster-1"
        ]
    },
    "changed": true,
    "failed": false
}
}

PLAY RECAP
*****
*****
localhost                : ok=4    changed=1    unreachable=0    failed=0

```

## Remove virtual volumes from a consistency group

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook remove_vv_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Consistency group operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Build a list of virtual volumes]
*****
*****
ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Remove virtual volumes from Consistency group]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "remove_vol_cg": {
    "cg_details": {
      "auto_resume_at_loser": true,
      "name": "ansible_cg",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "unknown"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [],
      "virtual_volumes": [],
      "visibility": [
        "/vplex/v2/clusters/cluster-1"
      ]
    },
    "changed": true,
  }
}

```

```

        "failed": false
    }
}

PLAY RECAP
*****
*****
localhost                : ok=4    changed=1    unreachable=0    failed=0

```

## Rename a consistency group

```

(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook rename_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Consistency group operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Rename Consistency group]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "rename_dr_cg": {
    "cg_details": {
      "auto_resume_at_loser": true,
      "name": "ansible_cg_new",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "unknown"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [],
      "virtual_volumes": [],
      "visibility": [
        "/vplex/v2/clusters/cluster-1"
      ]
    },
    "changed": true,
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Delete a consistency group

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook delete_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Consistency group operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Delete Consistency group]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "delete_cg": {
    "cg_details": null,
    "changed": true,
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0
```

# Distributed consistency group module

The distributed consistency group module manages the distributed consistency groups in the VPLEX metro setup.

The manage distributed consistency group module has the following functionalities:

- Create a distributed consistency group
- Resume a distributed consistency group
- Get a distributed consistency group
- Add/Remove the distributed virtual volumes to a distributed consistency group
- Update the detach rule of a distributed consistency group
- Disable/Enable Auto-resume-at-loser
- Rename distributed consistency group
- Delete a distributed consistency group

## Create a distributed consistency group

To create a distributed consistency group, run the appropriate playbook.

### Prerequisite

1. The distributed consistency group name should not be present in the VPLEX.
2. It should not have special characters in its name and length should not be more than 63 characters.

The syntax of the task is as follows:

```
- name: Create a distributed cg
  dellemc_vplex_distributed_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_cg_name: "ansible_d_cg"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the distributed consistency group name is not present in VPLEX - Success with "changed": True. Distributed consistency group is created.
2. If there is already a distributed consistency group with same name (Idempotency) - Success with "changed": False. Distributed consistency group is created.
3. If the name is invalid - Failure with "changed": False. Execution fails with error message "Could not create the distributed consistency group".

## Resume a distributed consistency group

To resume a distributed consistency group, run the appropriate playbook.

#### Prerequisite

The distributed consistency group should be present in the VPLEX and when the WAN COM or cluster link is disabled.

The syntax of the task is as follows:

```
- name: Resume I/O on virtual volumes in distributed cg
  dellemc_vplex_distributed_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_cg_name: "test_dr_cg"
    resume_at: "cluster-1"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If cluster link is disabled and both clusters are suspended, then I/O is resumed on in one of the clusters - Success with "changed": True. I/O is resumed on the distributed virtual volumes that are part of distributed consistency group.
2. If I/O is already resumed on any of the clusters on distributed consistency group (Idempotency) - Success with "changed": False.
3. If distributed consistency group has no distributed virtual volumes on it, and trying to resume when cluster link or WAN COM is disabled, then it fails with error message "Could not resume as distributed consistency group do not have virtual volume on it".

## Get a distributed consistency group

To get a distributed consistency group, run the appropriate playbook.

#### Prerequisite

The distributed consistency group should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Get a distributed cg
  dellemc_vplex_distributed_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_cg_name: "ansible_dr_cg"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed consistency group is present- Success with "changed": False. Displays the corresponding distributed consistency group details.
2. If distributed consistency group is not present- Failure with "changed": False. Execution fails with error message "Could not get details of distributed consistency group".

## Add or remove distributed virtual volumes to a distributed consistency group

### Add distributed virtual volumes to a distributed consistency group

To add distributed virtual volumes to a distributed consistency group, run the appropriate playbook.

#### Prerequisite

1. The distributed consistency group should be present in the VPLEX.
2. The virtual volumes should be global locality(that is distributed virtual volumes), and it should not be part of any other distributed consistency groups.

The syntax of the task is as follows:

```
- name: Add distributed volumes to distributed cg
  dellemc_vplex_distributed_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_cg_name: "test_cg"
    distributed_virtual_volumes: ["test_vol_1","test_vol_2"]
    distributed_virtual_volume_state: "present-in-cg"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the prerequisites are satisfied - Success with "changed": True. Virtual volumes are added to distributed consistency group.
2. If the distributed consistency group name has the same list of virtual volumes (Idempotency) - Success with "changed": False. Virtual volumes are already added to distributed consistency group.
3. Else - Failure with "changed": False. Execution fails with error message "Could not add virtual volumes to the distributed consistency group".

### Remove distributed virtual volumes from a distributed consistency group

To remove distributed virtual volumes from a distributed consistency group, run the appropriate playbook.

### Prerequisite

1. The distributed consistency group should be present in the VPLEX.
2. The distributed consistency group should have the virtual volumes.

The syntax of the task is as follows:

```
- name: Remove distributed volumes from distributed cg
dell EMC_vplex_distributed_consistency_group:
  vplexhost: "{{ vplexhost }}"
  vplexuser: "{{ vplexuser }}"
  vplexpassword: "{{ vplexpassword }}"
  verifycert: "{{ verifycert }}"
  distributed_cg_name: "test_cg"
  distributed_virtual_volumes: ["test_vol_1","test_vol_2"]
  distributed_virtual_volume_state: "absent-in-cg"
  state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the prerequisites are satisfied - Success with "changed": True. Distributed virtual volumes are removed from distributed consistency group.
2. If the distributed virtual volume is removed, or the virtual volume does not belong to any other distributed consistency group (Idempotency) - Success with "changed": False. Virtual volumes are removed from distributed consistency group.
3. If the given distributed virtual volumes are part of any other distributed consistency group - Failure with "changed": False. Execution fails with error message "Could not remove distributed virtual volumes from the distributed consistency group".

## Update the detach rule of a distributed consistency group

To update the detach rule of a distributed consistency group, run the appropriate playbook.

### Prerequisite

1. The distributed consistency group should be present in the VPLEX.
2. Detach rule must be in one of **no\_automatic\_winner**, **cluster-1**, and **cluster-2**.

The syntax of the task is as follows:

```
- name: Update the detach rule of distributed cg
dell EMC_vplex_distributed_consistency_group:
  vplexhost: "{{ vplexhost }}"
  vplexuser: "{{ vplexuser }}"
  vplexpassword: "{{ vplexpassword }}"
  verifycert: "{{ verifycert }}"
  distributed_cg_name: "test_cg"
  detach_rule: "cluster-1"
  state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the provided detach rule and the detach rule present for distributed consistency group are different - Success with "changed": True. Detach rule is set for distributed consistency group.
2. If the provided detach rule and the detach rule present for distributed consistency group are same (Idempotency) - Success with "changed": False. Provided detach rule is same as present.
3. If distributed consistency group is not present - Failure with "changed": False. Execution fails with error message "Could not update distributed consistency group due to reason: Resource not found".

## Disable or enable auto-resume-at loser

### Disable auto-resume-at loser

To disable auto-resume-at-loseauto-resume-at-loser of a distributed consistency group, run the appropriate playbook.

#### Prerequisite

The distributed consistency group should be present in the VPLEX, and auto-resume-at-loser should be in the enabled state.

The syntax of the task is as follows:

```
- name: Disable auto-resume-at-loser for distributed cg
  dellemc_vplex_distributed_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_cg_name: "test_cg"
    auto_resume_at_loser: false
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If auto\_resume\_at\_loser is in enabled state - Success with "changed": True. Auto resume at loser is disabled for distributed consistency group.
2. If auto\_resume\_at\_loser is already in disabled state (Idempotency) - Success with "changed": False. Auto resume at loser is disabled.
3. If distributed consistency group is not present - Failure with "changed": False. Execution fails with error message "Could not update distributed consistency group due to reason: Resource not found".

### Enable auto-resume-at loser

To enable auto-resume-at-loser of a distributed consistency group, run the appropriate playbook.

#### Prerequisite

The distributed consistency group should be present in the VPLEX, and auto-resume-at-loser should be in the disabled state.

The syntax of the task is as follows:

```
- name: Enable auto-resume-at-loser for distributed cg
  dellemc_vplex_distributed_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_cg_name: "test_cg"
    auto_resume_at_loser: true
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If auto\_resume\_at\_loser is in disabled state - Success with "changed": True. Auto resume at loser is enabled for distributed consistency group.
2. If auto\_resume\_at\_loser is already in enabled state (Idempotency) - Success with "changed": False. Auto resume at loser is enabled.



3. If distributed consistency group is not present - Failure with "changed": False. Execution fails with error message "Could not update distributed consistency group due to reason: Resource not found".

## Rename a distributed consistency group

To update the name of the existing distributed consistency group, run the appropriate playbook.

### Prerequisite

1. The distributed consistency group should be present in the VPLEX
2. It should not have special characters in its name and length should not be more than 63 characters.

The syntax of the task is as follows:

```
- name: Rename a distributed cg
  dellemc_vplex_distributed_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_cg_name: "test_cg"
    new_distributed_cg_name: "test_cg_newname"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the distributed consistency group name and new name are different, and new name does not exist in the VPLEX - Success with "changed": True. Distributed consistency group is renamed.
2. If the distributed consistency group name and new name are same (Idempotency) - Success with "changed": False. Distributed consistency group is already renamed.
3. If new distributed consistency group name is invalid - Failure with "changed": False. Execution fails with error message "Could not rename the distributed consistency group".

## Delete a distributed consistency group

To delete a distributed consistency group, run the appropriate playbook.

### Prerequisite

The distributed consistency group should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Delete a distributed cg
  dellemc_vplex_distributed_consistency_group:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    distributed_cg_name: "test_cg"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed consistency group is present- Success with "changed": False. Distributed consistency group is deleted.
2. If distributed consistency group is not present (Idempotency) - Success with "changed": False. Distributed consistency group is not present in the VPLEX.

## Distributed consistency group module parameters

The parameters for the distributed consistency group module are listed.

**Table 11. Parameters for the distributed consistency group module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameter	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The user name to access the VPLEX server.
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>True</li> <li>False</li> </ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"> <li>True - Verifies the SSL certificate</li> <li>False - Specified that the SSL certificate should not be verified</li> </ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>True</li> <li>False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dellemc_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
distributed_cg_name	-	str	Mandatory	Name of the distributed consistency group.
distributed_virtual_volumes	-	list	Optional	List of distributed virtual volumes.
distributed_virtual_volume_state	<ul style="list-style-type: none"> <li>present-in-cg</li> <li>absent-in-cg</li> </ul>	str	Optional	State of distributed virtual volumes.
new_distributed_cg_name	-	str	Optional	Name of the new distributed consistency group.
detach_rule	-	str	Optional	Detach rule of the distributed consistency group.
auto_resume_at_loser	<ul style="list-style-type: none"> <li>True</li> <li>False</li> </ul>	bool	Optional	Specifies whether auto-reume-at-loser is enabled or disabled.
resume_at	-	str	Optional	Specifies which cluster to resume I/O, when the cluster link is disabled.
state	<ul style="list-style-type: none"> <li>Present</li> <li>Absent</li> </ul>	str	Mandatory	This is the state of a distributed consistency group.

## Sample output

### Create a distributed consistency group

```
(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook create_dist_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Distributed consistency group operations]
```

```

*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Create a distributed cg]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "create_dr_cg": {
    "changed": true,
    "d_cg_details": {
      "auto_resume_at_loser": true,
      "detach_rule": {
        "type": "no_automatic_winner"
      },
      "name": "ansible_dr_cg",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "ok"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ],
      "virtual_volumes": [],
      "visibility": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ]
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Resume a distributed consistency group

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook resume_dist_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Distributed consistency group operations]
*****
*****

TASK [Gathering Facts]

```

```

*****
*****
ok: [localhost]

TASK [Resume I/O on vv in distributed cg]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "resume_dr_cg": {
    "changed": true,
    "d_cg_details": {
      "auto_resume_at_loser": true,
      "detach_rule": {
        "type": "no_automatic_winner"
      },
      "name": "test_123",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [
            "cluster-departure"
          ],
          "summary": "suspended"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ],
      "virtual_volumes": [
        "/vplex/v2/distributed_storage/distributed_virtual_volumes/vir_vol_1"
      ],
      "visibility": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ]
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Get a distributed consistency group

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook get_dist_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Distributed consistency group operations]
*****
*****

TASK [Gathering Facts]

```

```

*****
*****
ok: [localhost]

TASK [Get a distributed cg]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "get_dr_cg": {
    "changed": false,
    "d_cg_details": {
      "auto_resume_at_loser": true,
      "detach_rule": {
        "type": "no_automatic_winner"
      },
      "name": "ansible_dr_cg",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "ok"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ],
      "virtual_volumes": [],
      "visibility": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ]
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=0    unreachable=0    failed=0

```

## Add distributed virtual volumes to a distributed consistency group

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
add_dist_vv_dist_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Distributed consistency group operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

```

```

TASK [Build a list of distributed virtual volumes]
*****
*****
ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Add distributed virtual volumes to distributed cg]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "add_virtual_volumes_to_cg": {
    "changed": true,
    "d_cg_details": {
      "auto_resume_at_loser": true,
      "detach_rule": {
        "type": "no_automatic_winner"
      },
      "name": "ansible_dr_cg",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "ok"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ],
      "virtual_volumes": [
        "/vplex/v2/distributed_storage/distributed_virtual_volumes/vir_vol_1",
        "/vplex/v2/distributed_storage/distributed_virtual_volumes/vir_vol_2"
      ],
      "visibility": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ]
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                        : ok=4    changed=1    unreachable=0    failed=0

```

## Remove distributed virtual volumes to a distributed consistency group

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
remove_vv_dist_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Distributed consistency group operations]
*****

```

```

*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Build a list of distributed virtual volumes]
*****
ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Remove distributed virtual volumes from distributed cg]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "remove_virtual_volumes_from_cg": {
    "changed": true,
    "d_cg_details": {
      "auto_resume_at_loser": true,
      "detach_rule": {
        "type": "no_automatic_winner"
      },
      "name": "ansible_dr_cg",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "ok"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ],
      "virtual_volumes": [],
      "visibility": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ]
    },
    "failed": false
  }
}

PLAY RECAP
*****
localhost                : ok=4    changed=1    unreachable=0    failed=0

```

## Update the detach rule

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook detach_rule.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

```

```

PLAY [Testing Distributed consistency group operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Update the detach rule of a distributed cg]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "detach_rule_dr_cg": {
    "changed": true,
    "d_cg_details": {
      "auto_resume_at_loser": true,
      "detach_rule": {
        "cluster": "/vplex/v2/clusters/cluster-1",
        "delay": 5,
        "type": "winner"
      },
      "name": "ansible_dr_cg",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "ok"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ],
      "virtual_volumes": [],
      "visibility": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ]
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Disable auto-resume-at-loser

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
disable_dist_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Distributed consistency group operations]

```



```

*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Disable auto-resume-at-loser on distributed cg]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "disable_dr_cg": {
    "changed": true,
    "d_cg_details": {
      "auto_resume_at_loser": false,
      "detach_rule": {
        "type": "no_automatic_winner"
      },
      "name": "ansible_dr_cg",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "ok"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ],
      "virtual_volumes": [],
      "visibility": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ]
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=4    changed=1    unreachable=0    failed=0

```

## Enable auto-resume-at-loser

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook enable_dist_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Distributed consistency group operations]
*****
*****

TASK [Gathering Facts]

```

```

*****
*****
ok: [localhost]

TASK [Enable auto-resume-at-loser on distributed cg]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "enable_dr_cg": {
    "changed": true,
    "d_cg_details": {
      "auto_resume_at_loser": true,
      "detach_rule": {
        "type": "no_automatic_winner"
      },
      "name": "ansible_dr_cg",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "ok"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ],
      "virtual_volumes": [],
      "visibility": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ]
    },
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=4    changed=1    unreachable=0    failed=0

```

## Rename a distributed consistency group

```

(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook rename_dist_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Distributed consistency group operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

```

```

TASK [Rename a distributed cg]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "rename_cg": {
    "changed": true,
    "d_cg_details": {
      "auto_resume_at_loser": true,
      "detach_rule": {
        "type": "no_automatic_winner"
      },
      "name": "ansible_dr_cg_name",
      "operational_status": [
        {
          "cluster": "cluster-1",
          "details": [],
          "summary": "ok"
        },
        {
          "cluster": "cluster-2",
          "details": [],
          "summary": "ok"
        }
      ],
      "read_only": false,
      "storage_at_clusters": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ],
      "virtual_volumes": [],
      "visibility": [
        "/vplex/v2/clusters/cluster-1",
        "/vplex/v2/clusters/cluster-2"
      ]
    },
    "failed": false
  }
}

PLAY RECAP
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Delete a distributed consistency group

```

(py3_ans2_7) [root@dsvej252 playbooks]# ansible-playbook delete_dist_cg.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Distributed consistency group operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Delete a distributed cg]
*****
changed: [localhost]

```

```

TASK [debug]
*****
*****
ok: [localhost] => {
  "delete_distributed_cg": {
    "changed": true,
    "d_cg_details": null,
    "failed": false
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Port module

The port module manages the FE ports in the VPLEX.

The manage ports module has the following functions:

- Get port
- Enable port
- Disable port

### Get port

To get the port details, run the appropriate playbook.

#### Prerequisite

Port should be present in the VPLEX.

The syntax of the task is as follows:

```

- name: Get port details
  dell EMC vplex_port:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    port_name: "P00000000046E0124B-A0-FC02"
    state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Get port - Success with "changed": False. Displays the corresponding port details.
2. If port\_name or cluster\_name is invalid - Failure with "changed": False. Exits with the failure message.

### Enable port

To enable the port details, run the appropriate playbook.

#### Prerequisite

Port should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Enable a port
  dellemc_vplex_port:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    port_name: "P0000000046E0124B-A0-FC02"
    state: "present"
    enabled: true
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Enable port - Success with "changed": True. Port is enabled as expected.
2. If the port is already enabled (Idempotency) - Success with "changed": False. No change to the port as it is enabled.
3. If port\_name or cluster\_name is invalid - Failure with "changed": False. Exits with the failure message.

## Disable port

To disable the port details, run the appropriate playbook.

#### Prerequisite

Port should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Disable a port
  dellemc_vplex_port:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    port_name: "P0000000046E0124B-A0-FC02"
    state: "present"
    enabled: false
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Disable port - Success with "changed": True. Port is disabled as expected.
2. If the port is already disabled (Idempotency) - Success with "changed": False. No change to the port as it is disabled.
3. If port\_name or cluster\_name is invalid - Failure with "changed": False. Exits with the failure message.

## Port module parameters

Parameters for the port module are listed.

**Table 12. Parameters for the port module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The user name to access the VPLEX server.

**Table 12. Parameters for the port module (continued)**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"> <li>• True - Verifies the SSL certificate</li> <li>• False - Specified that the SSL certificate should not be verified</li> </ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dell EMC ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	-	str	Mandatory	Name of the cluster.
port_name	-	str	Mandatory	Name of the port.
enabled	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> <li>• None</li> <li>• Default: None</li> </ul>	bool	Optional	The status of the port.
state	<ul style="list-style-type: none"> <li>• Present</li> <li>• Absent</li> </ul>	str	Mandatory	Presence of the port.

## Sample output

### Enable port

```
[root@centos76 playbooks]# ansible-playbook enable_port.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Ports]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Enable a Port]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "enable_port": {
    "changed": true,
    "failed": false,
    "port_details": {
      "director": "director-1-1-A",
      "director_id": "0x0000000046e0124b",
```

```

        "discovered_initiators": [],
        "enabled": true,
        "export_status": "suspended",
        "exports": [
            {
                "lun": "0",
                "status": "unknown",
                "view": "/vplex/v2/clusters/cluster-1/exports/storage_views/ansible-
storview",
                "volume": "/vplex/v2/distributed_storage/distributed_virtual_volumes/
ansible_dist_dev_vol"
            },
            {
                "lun": "1",
                "status": "unknown",
                "view": "/vplex/v2/clusters/cluster-1/exports/storage_views/ansible-
storview",
                "volume": "/vplex/v2/clusters/cluster-1/virtual_volumes/
ansible_vol_1"
            }
        ],
        "name": "P00000000046E0124B-A0-FC00",
        "node_wwn": "0x5000144046e0124b",
        "port_wwn": "0x5000144260124b00"
    }
}

PLAY RECAP
*****
*****
localhost          : ok=3    changed=1    unreachable=0    failed=0
skipped=0          rescued=0    ignored=0

```

## Get port

```

[root@centos76 playbooks]# ansible-playbook get_port.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Ports]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get port details]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
    "get_port": {
        "changed": false,
        "failed": false,
        "port_details": {
            "director": "director-1-1-A",
            "director_id": "0x0000000046e0124b",
            "discovered_initiators": [],
            "enabled": true,
            "export_status": "ok",
            "exports": [
                {
                    "lun": "0",
                    "status": "unknown",
                    "view": "/vplex/v2/clusters/cluster-1/exports/storage_views/ansible-

```

```

storview",
    "volume": "/vplex/v2/distributed_storage/distributed_virtual_volumes/
ansible_dist_dev_vol"
    },
    {
        "lun": "1",
        "status": "unknown",
        "view": "/vplex/v2/clusters/cluster-1/exports/storage_views/ansible-
storview",
        "volume": "/vplex/v2/clusters/cluster-1/virtual_volumes/
ansible_vol_1"
    }
    ],
    "name": "P0000000046E0124B-A0-FC00",
    "node_wwn": "0x5000144046e0124b",
    "port_wwn": "0x5000144260124b00"
}
}
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=0    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Disable port

```

[root@centos76 playbooks]# ansible-playbook disable_port.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Ports]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Disable a Port]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
    "disable_port": {
        "changed": true,
        "failed": false,
        "port_details": {
            "director": "director-1-1-A",
            "director_id": "0x0000000046e0124b",
            "discovered_initiators": [],
            "enabled": false,
            "export_status": "suspended",
            "exports": [
                {
                    "lun": "0",
                    "status": "unknown",
                    "view": "/vplex/v2/clusters/cluster-1/exports/storage_views/ansible-
storview",
                    "volume": "/vplex/v2/distributed_storage/distributed_virtual_volumes/
ansible_dist_dev_vol"
                },
                {
                    "lun": "1",
                    "status": "unknown",
                    "view": "/vplex/v2/clusters/cluster-1/exports/storage_views/ansible-

```



```

storview",
    "volume": "/vplex/v2/clusters/cluster-1/virtual_volumes/
ansible_vol_1"
    },
    ],
    "name": "P0000000046E0124B-A0-FC00",
    "node_wwn": "0x5000144046e0124b",
    "port_wwn": "0x5000144260124b00"
  }
}
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Initiator module

The initiator module manages the initiators available in VPLEX.

The initiator module has the following functionalities:

- Register an initiator (auto or manual) in cluster
- Get details of an initiator from a cluster
- Rename an initiator present in the cluster
- Unregister an initiator in cluster
- Rediscover Initiators from a cluster
- Rediscover Initiators from a cluster with timeout parameter

## Register an initiator

To register an initiator that is visible to VPLEX port (auto-register) and not visible to VPLEX port (manual register) using port\_wwn, run the appropriate playbook.

### Prerequisite

1. To register the initiator, the provided name should not be present in the VPLEX.
2. The **port\_wwn** should be present in the VPLEX, and it should be unregistered.
3. The initiator name should not have length more than 36 characters, and it should not have special characters other than - and \_.

The syntax of the task is as follows:

```

- name: Register Initiator with port_wwn
  dell EMC vplex_initiator:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    initiator_name: "ansible_init"
    port_wwn: "0x21000024ff30ae28"
    host_type: "hpux"
    registered: true
    state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

With the appropriate playbook syntax, on trying to run the playbook, the following are the expected output for the initiators:

1. If initiator is registered - Success with "changed": True. The initiator is registered.

2. If initiator is already registered (Idempotency) - Success with "changed": False. No change in the initiator as it is already created.
3. If port\_wwn or port\_ww-manual is invalid - Failure with "changed": False. Exits with failure message stating Could not register initiator.

## Get details of an initiator

To get the details of an initiator, run the appropriate playbook.

### Prerequisite

The initiator should be present in VPLEX.

The syntax of the task is as follows:

### Get details of an Initiator using the initiator name

```
- name: Get details of an Initiator
  dellemc_vplex_initiator:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    initiator_name: "ansible_init"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Get details of an Initiator using the port\_wwn

```
- name: Get details of an Initiator
  dellemc_vplex_initiator:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    port_wwn: "0x21000024ff30ae28"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the Initiator is present- Success with "changed": False. Displays the corresponding initiator details.
2. If the Initiator name or **port\_wwn** is invalid- Failure with "changed": False. Execution fails with error message "Could not get the initiator".

## Rename initiator

To rename the initiator, run the appropriate playbook.

### Prerequisite

1. The initiator should be present in the VPLEX.
2. The new name should not be present in the VPLEX and should not have special characters. Also, the length should not be greater than 36 characters.

The syntax of the task is as follows:

## Rename initiator using the initiator name

```
- name: Rename a registered Initiator name
dell EMC_vplex_initiator:
  vplexhost: "{{ vplexhost }}"
  vplexuser: "{{ vplexuser }}"
  vplexpassword: "{{ vplexpassword }}"
  verifycert: "{{ verifycert }}"
  cluster_name: "cluster-1"
  initiator_name: "ansible_init"
  new_initiator_name: "ansible_init_update_name"
  state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

## Rename initiator using the port\_wwn

```
- name: Rename a registered Initiator name
dell EMC_vplex_initiator:
  vplexhost: "{{ vplexhost }}"
  vplexuser: "{{ vplexuser }}"
  vplexpassword: "{{ vplexpassword }}"
  verifycert: "{{ verifycert }}"
  cluster_name: "cluster-1"
  port_wwn: "0x21000024ff30ae28"
  new_initiator_name: "ansible_init_update_wwn"
  state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If initiator is renamed - Success with "changed": True. The initiator is renamed.
2. If initiator that is already renamed (Idempotency) - Success with "changed": False. No change in the initiator as it is renamed.
3. If initiator name is invalid - Failure with "changed": False. Execution fails with error message "Could not rename the initiator".
4. If the new initiator name is in use- Failure with "changed": False. Execution fails with error message "Could not rename the initiator as initiator with the new name already exists" .

## Unregister an initiator

To unregister an initiator, run the appropriate playbook.

### Prerequisite

The initiator should be present in VPLEX.

The syntax of the task is as follows:

## Unregister an initiator using the initiator name

```
- name: Unregister Initiator
dell EMC_vplex_initiator:
  vplexhost: "{{ vplexhost }}"
  vplexuser: "{{ vplexuser }}"
  vplexpassword: "{{ vplexpassword }}"
  verifycert: "{{ verifycert }}"
  cluster_name: "cluster-1"
  initiator_name: "ansible_init"
```

```
registered: false
state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

## Unregister an initiator using the port\_wwn

```
- name: Unregister Initiator
  dellemc_vplex_initiator:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    port_wwn: "0x21000024ff30ae28"
    registered: false
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If initiator is unregistered - Success with "changed": True. The initiator is unregistered.
2. If initiator that is already unregistered (Idempotency) - Success with "changed": False with initiator details as none.
3. If port\_wwn is invalid - Failure with "changed": False. Exits with the failure message "Could not unregister".
4. If initiator name is invalid - Failure with "changed": False. Exits with the failure message "Could not get the initiator".

## Rediscover initiators

To rediscover the initiators, run the appropriate playbook.

### Prerequisite

The initiator on which the operation is to be performed is rediscovered (that is to discover all the initiators in particular cluster).


## Rediscover initiators without timeout

The syntax of the task is as follows:

```
- name: Rediscover Initiators
  dellemc_vplex_initiator:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

## Rediscover initiators with timeout value set

 **NOTE: rediscover\_timeout:** : The valid range for rediscovery timeout is 1 to 3600.

The syntax of the task is as follows:

```
- name: Rediscover Initiators with timeout value set
  dellemc_vplex_initiator:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    rediscover_timeout: "5"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If rediscover initiator - Success with "changed": False. All the initiators are rediscovered.
2. If cluster\_name is invalid -Failure with "changed": False. Exits with error message stating "Could not find resource".

## Initiator module parameters

The parameters for the initiator module are listed.

**Table 13. Parameters for the initiator module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameter	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The user name to access the VPLEX server.
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"><li>• True - Verifies the SSL certificate.</li><li>• False - Specified that the SSL certificate should not be verified.</li></ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"><li>• True</li><li>• False</li></ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dellemc_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	-	str	Mandatory	Name of the cluster.
initiator_name	-	str	Optional	The name of the initiator. <ul style="list-style-type: none"><li>• Do not use special characters other than '_' and not more than 36 character.</li></ul>
new_initiator_name	-	str	Optional	The name to be used while renaming the initiator. <ul style="list-style-type: none"><li>• Do not use special characters other than '_' and not more than 36 characters</li></ul>
host_type	<ul style="list-style-type: none"><li>• default</li><li>• hpux</li><li>• sun-vcs</li><li>• aix</li><li>• recoverpoint</li></ul>	str	Optional	Type of host associated with the initiator. For registering the initiator manually, host_type should be specified along with port_wwn or iscsi_name. The supported values are as follows: <ul style="list-style-type: none"><li>• default</li><li>• hpux</li></ul>

**Table 13. Parameters for the initiator module (continued)**

Parameter name	Choice or default	Type	Mandatory/Optional Parameter	Description
				<ul style="list-style-type: none"> <li>• sun-vcs</li> <li>• aix</li> <li>• recoverpoint</li> </ul> <p>The default value is 'default'.</p> <p><b>NOTE:</b> The host_type 'recoverpoint' is not supported in this release.</p>
port_wwn	-	str	Optional	WWN of the port to register. For registering the initiator as FC port, port_wwn should be specified. This parameter is optional for all the operations except the register initiator.
registered	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	<p>Defines whether the initiator state is to be registered or not. Valid values are True/False/None.</p> <ul style="list-style-type: none"> <li>• True - Register</li> <li>• False - Unregister</li> </ul> <p>The default value is None.</p>
rediscover_timeout	1	int	Optional	It is allowed time in seconds for the rediscovery process, and the default time is 1 s. The valid range for rediscovery timeout is 1 - 3600.
state	<ul style="list-style-type: none"> <li>• absent</li> <li>• present</li> </ul>	str	Mandatory	<p>Defines whether the initiator must be present in VPLEX.</p> <ul style="list-style-type: none"> <li>• absent - The initiator must not be present in VPLEX</li> <li>• present - The initiator must be present in VPLEX</li> </ul> <p>Valid values - Present (It is always assumed as initiators are visible in VPLEX).</p>

## Sample output

### Register initiator

```
[root@centos76 playbooks]# ansible-playbook register_initiator.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Initiators of VPLEX]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Register the initiator with port_wwn]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "reg_initiator": {
    "changed": true,
    "failed": false,
    "initiator_details": {
      "name": "ansible_init",
```

```

        "node_wwn": "0x21000024ff30aca6",
        "port_wwn": "0x21000024ff30aca6",
        "target_ports": [
            "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC01",
            "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC00",
            "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046F0124B-B0-FC01",
            "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046F0124B-B0-FC00"
        ],
        "type": "hpx"
    }
}
}

```

#### PLAY RECAP

```

*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

### Rediscover initiators with timeout

```

(py3_ans_10) [root@localhost init]# ansible-playbook rediscover.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

```

#### PLAY [Manage Initiators of VPLEX]

```

*****
*****

```

#### TASK [Gathering Facts]

```

*****
*****
ok: [localhost]

```

#### TASK [Rediscover Initiators with timeout value set]

```

*****
*****
changed: [localhost]

```

#### TASK [debug]

```

*****
*****
ok: [localhost] => {
  "rediscover_initiator": {
    "changed": true,
    "failed": false,
    "initiator_details": [
      {
        "name": "dsveg124_P1",
        "node_wwn": "0x20000000c9b42ce5",
        "port_wwn": "0x10000000c9b42ce5",
        "target_ports": [
          "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-FC00",
          "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-FC01",
          "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-FC00",
          "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-FC01"
        ],
        "type": "default"
      },
      {
        "name": "UNREGISTERED-0x2101001b32af8132",
        "node_wwn": "0x2001001b32af8132",
        "port_wwn": "0x2101001b32af8132",
        "target_ports": [
          "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-FC00",
          "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-FC01",

```

```

        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC01"
    ]
},
{
    "name": "UNREGISTERED-0x100000090fa181a6c",
    "node_wwn": "0x200000090fa181a6c",
    "port_wwn": "0x100000090fa181a6c",
    "target_ports": [
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC01",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC01"
    ]
},
{
    "name": "dsveg229_1",
    "node_wwn": "0x200000000c9988a69",
    "port_wwn": "0x100000000c9988a69",
    "target_ports": [
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC01",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC01"
    ],
    "type": "default"
},
{
    "name": "dsveg124_P0",
    "node_wwn": "0x200000000c9b42ce4",
    "port_wwn": "0x100000000c9b42ce4",
    "target_ports": [
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC01",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC01"
    ],
    "type": "default"
},
{
    "name": "UNREGISTERED-0x10000000c9b42dc4",
    "node_wwn": "0x20000000c9b42dc4",
    "port_wwn": "0x10000000c9b42dc4",
    "target_ports": [
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC01",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC01"
    ]
},
{
    "name": "UNREGISTERED-0x10000000c9b42dc5",
    "node_wwn": "0x20000000c9b42dc5",
    "port_wwn": "0x10000000c9b42dc5",
    "target_ports": [

```



```

        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC01",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC01"
    ]
},
{
    "name": "UNREGISTERED-0x10000090fa181a6d",
    "node_wwn": "0x20000090fa181a6d",
    "port_wwn": "0x10000090fa181a6d",
    "target_ports": [
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC01",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC01"
    ]
},
{
    "name": "dsveg229_0",
    "node_wwn": "0x20000000c9988a68",
    "port_wwn": "0x10000000c9988a68",
    "target_ports": [
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC01",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC01"
    ],
    "type": "default"
},
{
    "name": "UNREGISTERED-0x2100001b328f8132",
    "node_wwn": "0x2000001b328f8132",
    "port_wwn": "0x2100001b328f8132",
    "target_ports": [
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043F018B9-B0-
FC01",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC00",
        "/vplex/v2/clusters/Bangalore/exports/ports/P0000000043E018B9-A0-
FC01"
    ]
}
]
}
}
}

```

#### PLAY RECAP

```

*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

#### Get initiator

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook get_initiator.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

```

```
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
```

```
PLAY [Manage Initiators of VPLEX]
```

```
*****
*****
```

```
TASK [Gathering Facts]
```

```
*****
*****
```

```
ok: [localhost]
```

```
TASK [Get details of an Initiator with initiator name]
```

```
*****
*****
```

```
ok: [localhost]
```

```
TASK [debug]
```

```
*****
*****
```

```
ok: [localhost] => {
  "get_initiator": {
    "changed": false,
    "failed": false,
    "initiator_details": {
      "name": "init_1",
      "node_wnn": "0x20000024ff4fadd9",
      "port_wnn": "0x21000024ff4fadd9",
      "target_ports": [
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC01",
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC00",
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046F0124B-B0-FC01",
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046F0124B-B0-FC00"
      ],
      "type": "default"
    }
  }
}
```

```
PLAY RECAP
```

```
*****
*****
```

```
localhost : ok=3 changed=0 unreachable=0 failed=0
```

## Rename initiator

```
[root@centos76 playbooks]# ansible-playbook rename_initiator.yml
```

```
[WARNING]: No inventory was parsed, only implicit localhost is available
```

```
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
```

```
PLAY [Manage Initiators of VPLEX]
```

```
*****
*****
```

```
TASK [Gathering Facts]
```

```
*****
*****
```

```
ok: [localhost]
```

```
TASK [Rename the Initiator with port_wnn]
```

```
*****
*****
```

```
changed: [localhost]
```

```
TASK [debug]
```

```
*****
*****
```

```
ok: [localhost] => {
  "modify_initiator_wnn": {
    "changed": true,
    "failed": false,
```

```

    "initiator_details": {
      "name": "ansible_init_update_name",
      "node_wwn": "0x21000024ff30aca6",
      "port_wwn": "0x21000024ff30aca6",
      "target_ports": [
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC01",
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC00",
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046F0124B-B0-FC01",
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046F0124B-B0-FC00"
      ],
      "type": "hpx"
    }
  }
}

PLAY RECAP
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Unregister initiator

```

[root@centos76 playbooks]# ansible-playbook unregister_initiator.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Manage Initiators of VPLEX]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Unregister the initiator with port_wwn]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "unreg_initiator_wwn": {
    "changed": true,
    "failed": false,
    "initiator_details": null
  }
}

PLAY RECAP
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Storage View module

The storage view module manages the storage views available in VPLEX.

The storage view module has the following functionalities:

- Get details of a storage view
- Create a storage view
- Delete a storage view
- Rename a storage view

- Add ports to a storage view
- Remove ports from a storage view
- Add initiators to a storage view
- Remove initiators from a storage view
- Add virtual volumes to a storage view
- Remove virtual volumes from a storage view

## Get details of a storage view

To get the details of a storage view in VPLEX, run the appropriate playbook.

### Prerequisite

Storage View for which the operation is to be performed should exist in VPLEX.

The syntax of the task is as follows:

```
- name: Get storage view details
  dell EMC_vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view_name: "ansible_storview"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is present-Success with "changed": False. Display the storage view details about the connected ports, initiator ports, and virtual volumes to the user.
2. If the storage view is absent-Failure with "changed": False. Exits with the failure message stating as "Could not get the storage view <storage\_view\_name> from the specific cluster".

## Create a storage view

To create a storage view in VPLEX with ports, run the appropriate playbook.

### Prerequisite

No storage view with the user specified name should exist in VPLEX.

The syntax of the task is as follows:

```
- name: Create a storage view
  dell EMC_vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view_name: "ansible_storview"
    ports: ["P00000000046E0124B-A0-FC00", "P00000000046E0124B-A0-FC01"]
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is not present - Success with "changed": True. Creates the storage view with the VPLEX front-end port that is specified through the user.
2. If the storage view is already present with the same combination (name and port - Idempotency) - Success with "changed": False. No change happens to the existing storage view.
3. If the storage view is already present with different combination of ports - Failure with "changed": False. Exits with the failure message stating as "Could not create the storage view as it already exists with different ports".
4. If invalid storage view name is specified through the user - Failure with "changed": False. Exits with the failure message stating as "Invalid characters specified for the storage view name".
5. If length of the storage view name exceeds the maximum length - Failure with "changed": False. Exits with the failure message stating as "The length of the storage view name exceeds the maximum size".

## Delete a storage view

To delete a storage view in VPLEX, run the appropriate playbook.

### Prerequisite

Storage View for which the operation is to be performed should exist in VPLEX.

The syntax of the task is as follows:

```
- name: Delete a storage view
  dellemc_vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view_name: "ansible_storview"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is present - Success with "changed": True. Deletes the storage view from VPLEX.
2. If the storage view is absent in VPLEX ( Idempotency) - Success with "changed": False. No changes happen in the VPLEX as the storage view is not present.

## Rename a storage view

To rename a storage view in the VPLEX, run the appropriate playbook.

### Prerequisite

Storage View for which the rename operation is to be performed should exist in the VPLEX.

The syntax of the task is as follows:

```
- name: Rename a storage view
  dellemc_vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view_name: "ansible_storview"
    new_storage_view_name: "ansible_storview_new"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

## Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is present, and new storage view name is valid - Success with "changed": True. Renamed the storage view.
2. If the new name is same as the existing storage view name ( Idempotency ) - Success with "changed": False. No changes happen to the storage view as it is already visible with the same new name.
3. If the new name specified is part of another storage view - Failure with "changed": False. Exits with the failure message stating as "New name is already used by another storage view".
4. If any invalid characters or long naming characters are specified for the new name - Failure with "changed": False. Exits with the failure message stating as "Invalid characters or length of new name exceeds the maximum size".
5. If the storage view is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage view <storage\_view\_name> from the specific cluster".

## Add ports to a storage view

To add ports to a storage view in the VPLEX, run the appropriate playbook.

### Prerequisite

Storage View for which the operation is to be performed should exist in the VPLEX.

Ports to be added into the storage view should exist in the VPLEX cluster.

The syntax of the task is as follows:

```
- name: Add ports to a storage view
  dell EMC vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view name: "ansible_storview"
    ports: ["P00000000046E0124B-A0-FC00", "P00000000046E0124B-A0-FC01"]
    port_state: "present-in-view"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

## Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is present, and given ports are valid - Success with "changed": True. Added ports to the storage view.
2. If the storage view is already present with the same combination of ports - ( Idempotency ) - Success with "changed": False. No changes happen to the existing storage view.
3. If invalid or non-existent ports are specified through the user - Failure with "changed": False. Exits with the failure message stating as "Could not get port details <port\_name> from the specific cluster".
4. If the storage view is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage view <storage\_view\_name> from the specific cluster".

## Add initiators to a storage view

To add initiators to a storage view in the VPLEX, run the appropriate playbook.

### Prerequisite

Storage View for which the operation is to be performed should exist in the VPLEX.

Initiator ports to be added into the storage view should exist in the VPLEX cluster.

The syntax of the task is as follows:

```
- name: Add initiators to a storage view
  dell EMC_vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view_name: "ansible_storview"
    initiators: ["ansible_init_1", "ansible_init_2"]
    initiator_state: "present-in-view"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is present, and given initiators are valid - Success with "changed": True. Added initiator ports to the storage view.
2. If the storage view is already present with the same combination of initiator ports - (Idempotency) - Success with "changed": False. No changes happen to the existing storage view.
3. If invalid or non-existent initiator ports are specified through the user - Failure with "changed": False. Exits with the failure message stating as "Could not get initiator <initiator\_port\_name> details from the specific cluster".
4. If an unregistered initiator port is specified through the user - Failure with "changed": False. Exits with the failure message stating as "The initiator <initiator\_port\_name> is unregistered in specific cluster".
5. If the storage view is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage view <storage\_view\_name> from the specific cluster".

## Add virtual volumes to a storage view

To add virtual volumes (local and distributed) to a storage view in the VPLEX, run the appropriate playbook.

### Prerequisite

Storage View for which the operation is to be performed should exist in the VPLEX.

Virtual volumes to be added into the storage view should exist in the VPLEX cluster.

The syntax of the task is as follows:

```
- name: Add virtual volumes to a storage view
  dell EMC_vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view_name: "ansible_storview"
    virtual_volumes: ["ansible_vir_1", "ansible_vir_2"]
    virtual_volume_state: "present-in-view"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is present, and given virtual volumes exist in the VPLEX - Success with "changed": True. Added virtual volumes to the storage view.
2. If the storage view is already present with the same combination of virtual volumes - (Idempotency) - Success with "changed": False. No changes happen to the existing storage view.

3. If invalid or non-existent virtual volumes are specified through the user - Failure with "changed": False. Exits with the failure message stating as "Could not get virtual volume details <virtual\_volume\_name> from the specific cluster".
4. If the specified virtual volume is part of another storage view - Failure with "changed": False. Exits with the failure message stating as "Could not add the virtual volume <virtual\_volume\_name> as it is part of another storage view".
5. If the storage view is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage view <storage\_view\_name> from the specific cluster".

## Remove ports from a storage view

To remove ports from a storage view in VPLEX, run the appropriate playbook.

### Prerequisite

Storage View for which the operation is to be performed should exist in VPLEX.

Ports to be removed should be present in the corresponding storage view.

The syntax of the task is as follows:

```
- name: Remove ports from a storage view
  dellemc_vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view_name: "ansible_storview"
    ports: ["P0000000046E0124B-A0-FC00", "P0000000046E0124B-A0-FC01"]
    port_state: "absent-in-view"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is present - Success with "changed": True. Remove the user specified valid ports from the storage view.
2. If the storage view exists without the specified ports - (Idempotency) - Success with "changed": False. No changes happen to the existing storage view.
3. If the storage view is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage view <storage\_view\_name> from the specific cluster".

## Remove initiators from a storage view

To remove initiators from a storage view in VPLEX, run the appropriate playbook.

### Prerequisite

Storage View for which the operation is to be performed should exist in VPLEX.

Initiators to be removed should be present in the corresponding storage view.

The syntax of the task is as follows:

```
- name: Remove initiators from a storage view
  dellemc_vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view_name: "ansible_storview"
    initiators : ["ansible_init_1", "ansible_init_2"]
```



```
initiator_state: "absent-in-view"
state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is present - Success with "changed": True. Remove the user specified valid initiators from the storage view.
2. If the storage view exists without the specified initiator ports - (Idempotency) - Success with "changed": False. No changes happen to the existing storage view.
3. If the storage view is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage view <storage\_view\_name> from the specific cluster ".

## Remove virtual volumes from a storage view

To remove virtual volumes (local and distributed) from a storage view in VPLEX, run the appropriate playbook.

### Prerequisite

Storage View for which the operation is to be performed should exist in VPLEX.

Virtual volumes to be removed should be present in the corresponding storage view.

The syntax of the task is as follows:

```
- name: Remove virtual volumes from a storage view
  dellemc_vplex_storage_view:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    storage_view_name: "ansible_storview"
    virtual_volumes: ["ansible_vir_1", "ansible_vir_2"]
    virtual_volume_state: "absent-in-view"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage view is present - Success with "changed": True. Remove the user specified virtual volumes from the storage view.
2. If the storage view exists without the specified virtual volumes - (Idempotency) - Success with "changed": False. No changes happen to the existing storage view.
3. If the storage view is absent - Failure with "changed": False. Exits with the failure message stating as "Could not get the storage view <storage\_view\_name> from the specific cluster ".

## Storage view module parameters

The parameters for the storage view module are listed.

**Table 14. Parameters for the storage view module**

Parameter name	Choice or Default	Type	Mandatory/Optional Parameter	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The username to access the VPLEX server.

**Table 14. Parameters for the storage view module (continued)**

Parameter name	Choice or Default	Type	Mandatory/Optional Parameter	Description
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"> <li>• True - Verifies the SSL certificate</li> <li>• False - Specified that the SSL certificate should not be verified.</li> </ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dell EMC ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	-	str	Mandatory	Name of the cluster.
storage_view_name	-	str	Mandatory	Name of the storage view used by the CRUD operations. <ul style="list-style-type: none"> <li>• Do not use special characters other than '_' and not more than 36 characters</li> </ul>
new_storage_view_name	-	str	Optional	Name to be used for renaming the storage view. <ul style="list-style-type: none"> <li>• Do not use special characters other than '_' and not more than 36 characters</li> </ul>
ports	-	list	Optional	Ports list to add or remove to the storage view.
initiators	-	list	Optional	Initiators list to add or remove to storage view.

**Table 14. Parameters for the storage view module (continued)**

Parameter name	Choice or Default	Type	Mandatory/Optional Parameter	Description
virtual_volumes	-	list	Optional	Virtual volumes list to add or remove to storage view.
port_state	<ul style="list-style-type: none"> <li>• present-in-view</li> <li>• absent-in-view</li> </ul>	str	Optional	<p>Decides the presence of the ports in the storage view.</p> <ul style="list-style-type: none"> <li>• absent-in-view - The ports must not be present in the storage view.</li> <li>• present-in-view - The ports must be present in the storage view.</li> </ul>
Initiator_state	<ul style="list-style-type: none"> <li>• present-in-view</li> <li>• absent-in-view</li> </ul>	str	Optional	<p>Decides the presence of the initiators in the storage view.</p> <ul style="list-style-type: none"> <li>• absent-in-view - The initiators must not be present in the storage view</li> <li>• present-in-view - The initiators must be present in the storage view</li> </ul>
virtual_volume_state	<ul style="list-style-type: none"> <li>• present-in-view</li> <li>• absent-in-view</li> </ul>	str	Optional	<p>Decides the presence of the virtual volumes in the storage view.</p> <ul style="list-style-type: none"> <li>• absent-in-view - The virtual volumes must not be present in the storage view</li> <li>• present-in-view - The virtual volumes must be present in the storage view</li> </ul>
state	<ul style="list-style-type: none"> <li>• absent</li> <li>• present</li> </ul>	str	Mandatory	<p>Decides the presence of the storage view in VPLEX.</p> <ul style="list-style-type: none"> <li>• absent - The storage view must not be present in VPLEX</li> <li>• present - The storage view must be present in VPLEX</li> </ul>

## Sample output

### Create storage view

```
[root@centos76 playbooks]# ansible-playbook create_view.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing storage view operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Create a storage view]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "create_storage_view": {
    "changed": true,
    "failed": false,
    "storageview_details": {
      "initiators": [],
      "name": "ansible_storview",
      "operational_status": "stopped",
      "ports": [
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046F0124B-B0-FC00"
      ],
      "virtual_volumes": []
    }
  }
}

PLAY RECAP
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0
```

### Get storage view

```
(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
get_storage_view.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

[WARNING]: Found variable using reserved name: port

PLAY [Testing storage view operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]
```

```

TASK [Get a storage view]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "get_storage_view": {
    "changed": false,
    "failed": false,
    "storageview_details": {
      "initiators": [],
      "name": "ansible_storview",
      "operational_status": "stopped",
      "ports": [
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046F0124B-B0-FC00"
      ],
      "virtual_volumes": []
    }
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=0    unreachable=0    failed=0

```

## Rename storage view

```

[root@centos76 playbooks]# ansible-playbook rename_view.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing storage view operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Rename a storage view]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "rename_storage_view": {
    "changed": true,
    "failed": false,
    "storageview_details": {
      "initiators": [],
      "name": "ansible_storview_new",
      "operational_status": "stopped",
      "ports": [
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046F0124B-B0-FC00"
      ],
      "virtual_volumes": []
    }
  }
}

PLAY RECAP
*****
*****

```

```
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0
```

### Add initiators to a storage view

```
(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
add_init_st_view.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

[WARNING]: Found variable using reserved name: port

PLAY [Testing storage view operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Build a list of initiators]
*****
*****
ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Add initiators to the storage view]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "add_initiator_storage_view": {
    "changed": true,
    "failed": false,
    "storageview_details": {
      "initiators": [
        "/vplex/v2/clusters/cluster-1/exports/initiator_ports/init_1",
        "/vplex/v2/clusters/cluster-1/exports/initiator_ports/init_2"
      ],
      "name": "ansible_storview_new",
      "operational_status": "ok",
      "ports": [
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC00",
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC01"
      ],
      "virtual_volumes": []
    }
  }
}

PLAY RECAP
*****
*****
localhost      : ok=4    changed=1    unreachable=0    failed=0
```

### Add virtual volumes to a storage view

```
(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook add_vv_st_view.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available
```

```

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

[WARNING]: Found variable using reserved name: port

PLAY [Testing storage view operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Build a list of virtual volumes]
*****
*****
ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Add virtualvolume to the storage view]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "add_vv_storage_view": {
    "changed": true,
    "failed": false,
    "storageview_details": {
      "initiators": [
        "/vplex/v2/clusters/cluster-1/exports/initiator_ports/init_1",
        "/vplex/v2/clusters/cluster-1/exports/initiator_ports/init_2"
      ],
      "name": "ansible_storview_new",
      "operational_status": "ok",
      "ports": [
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC00",
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC01"
      ],
      "virtual_volumes": [
        {
          "capacity": 2148925440,
          "lun": 0,
          "uri": "/vplex/v2/clusters/cluster-1/virtual_volumes/vir_1",
          "vpd_id": "VPD83T3:6000144000000010f0124b3da38e28e1"
        },
        {
          "capacity": 2148925440,
          "lun": 1,
          "uri": "/vplex/v2/clusters/cluster-1/virtual_volumes/vir_2",
          "vpd_id": "VPD83T3:6000144000000010f0124b3da38e28e9"
        }
      ]
    }
  }
}

PLAY RECAP
*****
*****
localhost                : ok=5    changed=1    unreachable=0    failed=0

```

## Add ports to a storage view

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
add_port_st_view.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

```

```

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

[WARNING]: Found variable using reserved name: port

PLAY [Testing storage view operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Add ports to the storage view]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "add_port_storage_view": {
    "changed": true,
    "failed": false,
    "storageview_details": {
      "initiators": [],
      "name": "ansible_storview_new",
      "operational_status": "stopped",
      "ports": [
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC00",
        "/vplex/v2/clusters/cluster-1/exports/ports/P0000000046E0124B-A0-FC01"
      ],
      "virtual_volumes": []
    }
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0

```

## Remove initiators from a storage view

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
remove_init_st_view.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

[WARNING]: Found variable using reserved name: port

PLAY [Testing storage view operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Build a list of initiators]
*****
*****

```



```

ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Remove initiators from storage view]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "remove_initiator_storage_view": {
    "changed": true,
    "failed": false,
    "storageview_details": {
      "initiators": [],
      "name": "ansible_storview_new",
      "operational_status": "stopped",
      "ports": [],
      "virtual_volumes": [
        {
          "capacity": 2148925440,
          "lun": 0,
          "uri": "/vplex/v2/clusters/cluster-1/virtual_volumes/vir_1",
          "vpd_id": "VPD83T3:6000144000000010f0124b3da38e28e1"
        },
        {
          "capacity": 2148925440,
          "lun": 1,
          "uri": "/vplex/v2/clusters/cluster-1/virtual_volumes/vir_2",
          "vpd_id": "VPD83T3:6000144000000010f0124b3da38e28e9"
        }
      ]
    }
  }
}

PLAY RECAP
*****
localhost                : ok=5    changed=1    unreachable=0    failed=0

```

## Remove virtual volumes from a storage view

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
remove_vv_st_view.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

[WARNING]: Found variable using reserved name: port

PLAY [Testing storage view operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Build a list of virtual volumes]
*****
ok: [localhost] => (item=1)
ok: [localhost] => (item=2)

TASK [Remove virtualvolume from the storage view]

```

```

*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "remove_vv_storage_view": {
    "changed": true,
    "failed": false,
    "storageview_details": {
      "initiators": [],
      "name": "ansible_storview_new",
      "operational_status": "stopped",
      "ports": [],
      "virtual_volumes": []
    }
  }
}

PLAY RECAP
*****
*****
localhost                : ok=5    changed=1    unreachable=0    failed=0

```

## Remove ports from a storage view

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
remove_port_st_view.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

[WARNING]: Found variable using reserved name: port

PLAY [Testing storage view operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Remove ports from storage view]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "remove_port_storage_view": {
    "changed": true,
    "failed": false,
    "storageview_details": {
      "initiators": [
        "/vplex/v2/clusters/cluster-1/exports/initiator_ports/init_1",
        "/vplex/v2/clusters/cluster-1/exports/initiator_ports/init_2"
      ],
      "name": "ansible_storview_new",
      "operational_status": "ok",
      "ports": [],
      "virtual_volumes": [
        {
          "capacity": 2148925440,
          "lun": 0,

```

```

        "uri": "/vplex/v2/clusters/cluster-1/virtual_volumes/vir_1",
        "vpd_id": "VPD83T3:6000144000000010f0124b3da38e28e1"
    },
    {
        "capacity": 2148925440,
        "lun": 1,
        "uri": "/vplex/v2/clusters/cluster-1/virtual_volumes/vir_2",
        "vpd_id": "VPD83T3:6000144000000010f0124b3da38e28e9"
    }
]
}
}

PLAY RECAP
*****
*****
localhost                : ok=5    changed=1    unreachable=0    failed=0

```

## Delete storage view

```

[root@centos76 playbooks]# ansible-playbook delete_view.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing storage view operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Delete a storage view]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
    "delete_storage_view": {
        "changed": true,
        "failed": false,
        "storageview_details": {}
    }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0

```

# Data migration module

The data migration module manages the device and extent migration jobs in the VPLEX.

## Device migration

The data migration module manages the device migration jobs in the VPLEX.

The data migration module has the following functionalities:

- Create device migration job

- Pause device migration job
- Resume device migration job
- Cancel device migration job
- Commit device migration job
- Update transfer size of a device migration job
- Get device migration job
- Delete device migration job

## Create a device migration job

### Create device migration job (within cluster)

To create a device migration job within cluster, run the appropriate playbook.

#### Prerequisite

1. Device migration job name should not be present in the VPlex.
2. Source device and target device should be present in the VPlex and from same cluster.
3. The target device should not have the virtual volume, and source device should have virtual volume on top of it.
4. The size of the source device should be less than or equal to size of the target device.
5. The device migration job name should not have length more than 63 characters, and it should not have special characters other than - and \_.

The syntax of the task is as follows:

```
- name: Create device migration job - Within Cluster
  dell EMC vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "test_dev_mig"
    storage: "device"
    source_name: "test_dev_1"
    target_name: "test_dev_2"
    transfer_size: 131072
    cluster_name: "cluster-1"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If device migration job is created - Success with "changed": True. Device migration job is created.
2. If device migration job is already created (Idempotency) - Success with "changed": False. No change in the device migration job details.
3. If the source device or target device are used through another device migration job-Failure with "changed": False. Exits with the error message stating "Could not create device migration job as source device or target device is used by another device migration job".
4. If the size of the source device greater than size of the target device-Failure with "changed": False error message stating "Could not create device migration job as the size of source device is greater than size of target device".
5. If the source device has no virtual volume on top of it-Failure with "changed": False. Exits with the error message stating "Could not create device migration job as source device does not contain virtual volume".
6. If the target device has virtual volume top of it-Failure with "changed": False. Exits with the error message stating "Could not create device migration job as target device contains a virtual volume".

## Create device migration job (across cluster)

To create a device migration job across cluster, run the appropriate playbook.

### Prerequisite

1. The source device and target device should be present in the VPLEX and from different clusters.
2. Device migration job name should not be present in the VPLEX.
3. The target device should not have the virtual volume, and source device should have virtual volume on top of it.
4. The size of the source device should be less than or equal to size of the target device.
5. The device migration job name should not have length more than 63 characters, and it should not have special characters other than - and \_.

The syntax of the task is as follows:

```
- name: Create device migration job - across Clusters
  dellemc_vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "test_dev_mig"
    storage: "device"
    source_name: "test_dev_1"
    target_name: "test_dev_2"
    transfer_size: 131072
    cluster_name: "cluster-1"
    target_cluster: "cluster-2"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If device migration job is created - Success with "changed": True. Device migration job is created.
2. If device migration job is already created (Idempotency) - Success with "changed": False. No change in the device migration job details.
3. If the source device or target device are used through another device migration job-Failure with "changed": False. Exits with the error message stating "Could not create device migration job as source device or target device is used by another device migration job".
4. If the size of the source device greater than size of the target device-Failure with "changed": False error message stating "Could not create device migration job as the size of source device is greater than size of target device".
5. If the source device has no virtual volume on top of it-Failure with "changed": False. Exits with the error message stating "Could not create device migration job as source device does not contain virtual volume".
6. If the target device has virtual volume top of it-Failure with "changed": False. Exits with the error message stating "Could not create device migration job as target device contains a virtual volume".

## Pause a device migration job

To pause a device migration job, run the appropriate playbook.

### Prerequisite

1. The device migration should be present in the VPLEX.
2. If the status of device migration job is **in-progress**, then pause a job of device migration can be done.

The syntax of the task is as follows:

```
- name: Pause device migration job
  dellemc_vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
```

```

verifycert: "{{ verifycert }}"
migration_name: "test_dev_mig"
storage: "device"
status: "pause"
state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Device migration job is paused - Success with "changed": True. Device migration job is paused.
2. Device migration job is already paused (Idempotency) - Success with "changed": False. No change in the device migration job details.
3. Pausing the device migration job fails only if the device migration job status is other than "in-progress"-Failure with "changed": False. Exits with the error message stating "Could not update the status of device migration job".

## Resume a device migration job

To resume a device migration job, run the appropriate playbook.

### Prerequisite

1. The device migration should be present in the VPlex.
2. If the status of device migration job is **paused**, then resume a job of device migration can be done.

The syntax of the task is as follows:

```

- name: Resume device migration job
  dell EMC vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "test_dev_mig"
    storage: "device"
    status: "resume"
    state: "present"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Device migration job is resumed - Success with "changed": True. Device migration job is resumed.
2. Device migration job is already resumed (Idempotency) - Success with "changed": False. No change in the device migration job details.
3. Resume the device migration job fails only if device migration job status is other than "paused"-Failure with "changed": False. Exits with the error message stating "Could not update the status of device migration job".

## Cancel a device migration job

To cancel a device migration job, run the appropriate playbook.

### Prerequisite

1. The device migration should be present in the VPlex.
2. If the device migration job status is **in-progress**, or **paused**, or **commit-pending**, or **partially-committed**, or **partially-cancelled**, then cancel a job of device migration can be done.

The syntax of the task is as follows:

```
- name: Cancel device migration job
  dell EMC vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "test_dev_mig"
    storage: "device"
    status: "cancel"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Device migration job is cancelled - Success with "changed": True. Device migration job is cancelled.
2. Device migration job is already cancelled (Idempotency) - Success with "changed": False. No change in the device migration job details.
3. Cancel the device migration job fails only if device migration job status is other than in-progress, or paused, or, commit-pending, or partially-committed, or partially-cancelled - Failure and "changed": False. Exits with the error message stating "Could not update the status of device migration job".

## Commit a device migration job

To commit a device migration job, run the appropriate playbook.

### Prerequisite

1. The device migration should be present in the VPLEX.
2. If the device migration job status is **commit-pending**, or **partially-committed**, then commit a job of device migration can be done.

The syntax of the task is as follows:

```
- name: Commit device migration job
  dell EMC vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "test_dev_mig"
    storage: "device"
    status: "commit"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Device migration job is committed - Success with "changed": True. Device migration job is committed.
2. Device migration job is already committed (Idempotency) - Success with "changed": False. No change in the device migration job details.
3. Commit the device migration job fails only if device migration job status is other than commit-pending, or partially-committed- Failure and "changed": False. Exits with the error message stating "Could not update the status of device migration job".

## Update the transfer size of a device migration job

To update the transfer size of a device migration job, run the appropriate playbook.

## Prerequisite

1. The device migration should be present in the VPLEX.
2. Update a device migration job with the transfer size set to 40960. Transfer size can not be less than 40KB, or greater than 128MB, or not in multiples of 4K.

The syntax of the task is as follows:

```
- name: Update device migration job
  dellemc_vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "test_dev_mig"
    storage: "device"
    transfer_size: 40960
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

## Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Updating the transfer size of device migration job - Success with "changed": True. Device migration job is updated with transfer size.
2. Device migration job is updated with the same transfer size (Idempotency) - Success with "changed": False. No change in the device migration job details.
3. Update the transfer size of the device migration job fails if transfer size is not in multiples of 4K- Failure with "changed": False. Exits with the error message stating "The transfer size should be multiples of 4K".
4. Update the transfer size of the device migration job fails if size is high- Failure with "changed": False. Exits with the error message stating "transfer size can not be more than 134217728 bytes".
5. Update the transfer size of the device migration job fails if size is low- Failure with "changed": False. Exits with the error message stating "transfer size can not be less than 40960 bytes".

## Get a device migration job

To get the details of a device migration job, run the appropriate playbook.

### Prerequisite

The device migration job should be present in the VPLEX.

The syntax of the task is as follows:

```
- name: Get device migration job
  dellemc_vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "test_dev_mig"
    storage: "device"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

## Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the device migration job is present- Success with "changed": False. Displays the corresponding device migration job details.
2. If the device migration is not present- Failure with "changed": False. Execution fails with the error message "Could not get the device migration".



## Delete a device migration job

To delete a device migration job, run the appropriate playbook.

### Prerequisite

1. The device migration job should be present in the VPLEX.
2. If the device migration job status is **cancelled**, or **committed**, or **complete**, then delete or remove of device migration job can be done.

The syntax of the task is as follows:

```
- name: Delete a device migration job
  dellemc_vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "test_dev_mig"
    storage: "device"
    state: "absent"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Device migration job is removed or deleted- Success with "changed": True. Device migration job is deleted or removed.
2. Device migration job is already deleted or removed (Idempotency) - Success with "changed": False. Device migration job details are none.
3. Delete the device migration job fails only if device migration job status is other than cancelled, or committed, or complete- Failure with "changed": False. Exits the error message stating as "Could not delete the device migration job".

## Extent migration

The data migration module manages the extent migration jobs in the VPLEX.

The data migration module has the following functionalities:

- Create extent migration job
- Pause extent migration job
- Resume extent migration job
- Cancel extent migration job
- Commit extent migration job
- Update transfer size of a extent migration job
- Get extent migration job
- Delete extent migration job

## Create an extent migration job

### Create an extent migration job

To create an extent migration job within cluster, run the appropriate playbook.

### Prerequisite

1. The source extent and target extent should be present in the VPLEX and from same cluster.
2. Extent migration job name should not be present in the VPLEX.
3. The target extent should be in claimed state, and source extent should be in used state.
4. The size of the source extent should be less than or equal to size of the target extent.

5. The extent migration job name should not have length more than 63 characters, and it should not have special characters other than - and \_.

The syntax of the task is as follows:

```
- name: Create an extent migration job
dellemc_vplex_data_migration:
  vplexhost: "{{ vplexhost }}"
  vplexuser: "{{ vplexuser }}"
  vplexpassword: "{{ vplexpassword }}"
  verifycert: "{{ verifycert }}"
  cluster_name: "cluster-1"
  storage: "extent"
  source_name: "source_ext_1"
  target_name: "target_ext_1"
  migration_name: "mobility_job"
  state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent migration job is created - Success with "changed": True. Extent migration job is created.
2. If extent migration job is already created (Idempotency) - Success with "changed": False. No change in the extent migration job details.
3. If the source extent or target extent is used through another extent migration job-Fails with the error message stating "Could not create extent migration job as source extent or target extent is used by another extent migration job".
4. If the size of the source extent greater than size of the target extent- Failure with "changed": False. Exist with the error message stating "Could not create extent migration job as the size of source extent is greater than size of target extent".
5. If the source extent is under claimed state- Failure with "changed": False. Exits with the error message stating "Could not create extent migration job as source extent does not contains device".
6. If the target extent is under used state- Failure with "changed": False. Exits with the error message stating "Could not create extent migration job as target extent contains a device".

## Pause an extent migration job

To pause an extent migration job, run the appropriate playbook.

### Prerequisite

1. The extent migration should be present in the VPLEX.
2. If the status of extent migration job is **in-progress**, then pause a job of extent migration can be done.

The syntax of the task is as follows:

```
- name: Pause an extent migration job
dellemc_vplex_data_migration:
  vplexhost: "{{ vplexhost }}"
  vplexuser: "{{ vplexuser }}"
  vplexpassword: "{{ vplexpassword }}"
  verifycert: "{{ verifycert }}"
  migration_name: "mobility_job"
  storage: "extent"
  status: "pause"
  state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Extent migration job is paused - Success with "changed": True. Extent migration job is paused.

2. Extent migration job is already paused (Idempotency) - Success with "changed": False. No change in the Extent migration job details.
3. Pausing the extent migration job fails only if the extent migration job status is other than "in-progress"-Failure with "changed": False. Exits with the error message stating "Could not update the status of extent migration job".

## Resume an extent migration job

To resume an extent migration job, run the appropriate playbook.

### Prerequisite

1. The extent migration should be present in the VPlex.
2. If the status of extent migration job is **paused**, then resume a job of extent migration can be done.

The syntax of the task is as follows:

```
- name: Resume an extent migration job
  dell EMC_vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "mobility_job"
    storage: "extent"
    status: "resume"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Extent migration job is resumed - Success with "changed": True. Extent migration job is resumed.
2. Extent migration job is already resumed (Idempotency) - Success with "changed": False. No change in the extent migration job details.
3. Resume the extent migration job fails only if extent migration job status is other than "paused"-Failure with "changed": False. Exits with the error message stating "Could not update the status of extent migration job".

## Cancel an extent migration job

To cancel an extent migration job, run the appropriate playbook.

### Prerequisite

1. The extent migration should be present in the VPlex.
2. If the extent migration job status is **in-progress**, or **paused**, or **commit-pending**, or **partially-committed**, or **partially-cancelled**, then cancel a job of extent migration can be done.

The syntax of the task is as follows:

```
- name: Cancel an extent migration job
  dell EMC_vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "mobility_job"
    storage: "extent"
    status: "cancel"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Extent migration job is cancelled - Success with "changed": True. Extent migration job is cancelled.
2. Extent migration job is already cancelled (Idempotency) - Success with "changed": False. No change in the extent migration job details.
3. Cancel the extent migration job fails only if extent migration job status is other than in-progress, or paused, or, commit-pending, or partially-committed, or partially-cancelled - Failure and "changed": False. Exits with the error message stating "Could not update the status of extent migration job".

## Commit an extent migration job

To commit an extent migration job, run the appropriate playbook.

### Prerequisite

1. The extent migration should be present in the VPLEX.
2. If the extent migration job status is **commit-pending**, or **partially-committed**, then commit a job of extent migration can be done.

The syntax of the task is as follows:

```
- name: Commit an extent migration job
  dellemc_vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "mobility_job"
    storage: "extent"
    status: "commit"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Extent migration job is committed - Success with "changed": True. Extent migration job is committed.
2. Extent migration job is already committed (Idempotency) - Success with "changed": False. No change in the extent migration job details.
3. Commit the extent migration job fails only if extent migration job status is other than commit-pending, or partially-committed- Failure and "changed": False. Exits with the error message stating "Could not update the status of extent migration job".

## Update the transfer size of an extent migration job

To update the transfer size of an extent migration job, run the appropriate playbook.

### Prerequisite

1. The extent migration should be present in the VPLEX.
2. Update a extent migration job with the transfer size set to 40960. Transfer size can not be less than 40KB, or greater than 128MB, or not in multiples of 4K.

The syntax of the task is as follows:

```
- name: Update transfer size of an extent migration job
  dellemc_vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    migration_name: "mobility_job"
    storage: "extent"
```

```
transfer_size: 40960
state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Updating the transfer size of extent migration job - Success with "changed": True. Extent migration job is updated with transfer size.
2. Extent migration job is updated with the same transfer size (Idempotency) - Success with "changed": False. No change in the extent migration job details.
3. Update the transfer size of the extent migration job fails if transfer size is not in multiples of 4K- Failure with "changed": False. Exits with the error message stating "The transfer size should be multiples of 4K".
4. Update the transfer size of the extent migration job fails if size is high- Failure with "changed": False. Exits with the error message stating "transfer size can not be more than 134217728 bytes".
5. Update the transfer size of the extent migration job fails if size is low- Failure with "changed": False. Exits with the error message stating "transfer size can not be less than 40960 bytes".

## Get an extent migration job

To get the details of an extent migration job, run the appropriate playbook.

### Prerequisite

The extent migration job should be present in the VPlex.

The syntax of the task is as follows:

```
- name: Get an extent migration job
  dell EMC vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    storage: "extent"
    migration_name: "mobility_job"
    state: "present"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If the extent migration job is present- Success with "changed": False. Displays the corresponding extent migration job details.
2. If the extent migration is not present- Failure with "changed": False. Execution fails with the error message "Could not get the extent migration".

## Delete an extent migration job

To delete an extent migration job, run the appropriate playbook.

### Prerequisite

1. The extent migration job should be present in the VPlex.
2. If the extent migration job status is **cancelled**, or **committed**, or **complete**, then delete or remove of extent migration job can be done.

The syntax of the task is as follows:

```
- name: Delete an extent migration job
  dell EMC vplex_data_migration:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
```

```

vpsexpassword: "{{ vplexpassword }}"
verifycert: "{{ verifycert }}"
migration_name: "mobility_job"
storage: "extent"
state: "absent"

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. Extent migration job is removed or deleted- Success with "changed": True. Extent migration job is deleted or removed.
2. Extent migration job is already deleted or removed (Idempotency) - Success with "changed": False. Extent migration job details are none.
3. Delete the extent migration job fails only if extent migration job status is other than cancelled, or committed, or complete- Failure with "changed": False. Exits the error message stating as "Could not delete the extent migration job".

## Data migration module parameters

The parameters for the data migration module are listed

**Table 15. Parameters for the data migration module**

Parameter name	Choice or Default	Type	Mandatory/Optional Parameter	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The username to access the VPLEX server.
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"> <li>• True - Verifies the SSL certificate</li> <li>• False - Specified that the SSL certificate should not be verified.</li> </ul>
debug	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dellemc_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required

**Table 15. Parameters for the data migration module (continued)**

Parameter name	Choice or Default	Type	Mandatory/Optional Parameter	Description
				only when verifycert is set to "True".
migration_name	-	str	Mandatory	Name of a migration job.
source_name	-	str	Optional	Name of a source device. The source device should contain a virtual volume.
target_name	-	str	Optional	Name of a target device. The target device should not contain a virtual volume.
transfer_size	-	int	Optional	The amount of data that can be transferred during migration. The number should be in Byte and must be a multiple of 4K.  Range: 40KB -128M. Default: 128KB(131072).
status	<ul style="list-style-type: none"> <li>• Pause</li> <li>• Resume</li> <li>• Commit</li> <li>• Cancel</li> </ul>	str	Optional	Name of a operation to be performed on the data migration job.
storage	<ul style="list-style-type: none"> <li>• device</li> <li>• extent</li> </ul>	str	Mandatory	It specifies whether it is a device migration or an extent migration job.
cluster_name	-	str	Optional	Name of a source cluster.
target_cluster	-	str	Optional	Name of a target cluster.
state	<ul style="list-style-type: none"> <li>• Absent</li> <li>• Present</li> </ul>	str	Mandatory	Decides the presence of the storage view in VPLEX. <ul style="list-style-type: none"> <li>• absent - The storage view must not be present in VPLEX</li> <li>• present - The storage view must be present in VPLEX</li> </ul>

## Sample output

### Create device migration job (within cluster)

```
(py3_ans2_9) [root@dsvej252 data_migration]# ansible-playbook create.yml
[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Create a device migration job]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "create_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "test_dev_mig",
      "percentage_done": 0,
      "source": "/vplex/v2/clusters/cluster-1/devices/test_dev_1",
      "source_exported": false,
      "start_time": "Thu Dec 03 06:13:30 UTC 2020",
      "status": "in-progress",
      "target": "/vplex/v2/clusters/cluster-1/devices/test_dev_2",
      "target_exported": false,
      "to_cluster": "/vplex/v2/clusters/cluster-1",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

PLAY RECAP
*****
localhost : ok=3    changed=1    unreachable=0    failed=0
skipped=0   rescued=0   ignored=0
```

### Create device migration job (across cluster)

```
(py3_ans2_9) [root@dsvej252 data_migration]# ansible-playbook create.yml
[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****

TASK [Gathering Facts]
*****
```



```

ok: [localhost]

TASK [Create a device migration job]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "create_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "test_dev_mig",
      "percentage_done": 0,
      "source": "/vplex/v2/clusters/cluster-1/devices/test_dev_1",
      "source_exported": true,
      "start_time": "Thu Dec 03 06:21:02 UTC 2020",
      "status": "in-progress",
      "target": "/vplex/v2/clusters/cluster-2/devices/test_dev_2",
      "target_exported": true,
      "to_cluster": "/vplex/v2/clusters/cluster-2",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

PLAY RECAP
*****
localhost          : ok=3    changed=1    unreachable=0    failed=0
skipped=0          rescued=0    ignored=0

```

## Get a device migration job

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook
get_data_device.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Get a device migration job]
*****
ok: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "get_job": {
    "changed": false,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-2",
      "name": "mig_job_1",

```

```

        "percentage_done": 100,
        "source": "/vplex/v2/clusters/cluster-2/devices/hnk_123",
        "source_exported": false,
        "start_time": "Tue Dec 15 06:56:16 UTC 2020",
        "status": "complete",
        "target": "/vplex/v2/clusters/cluster-2/devices/vir_2_1",
        "target_exported": false,
        "to_cluster": "/vplex/v2/clusters/cluster-2",
        "transfer_size": 131072,
        "type": "full"
    }
}
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=0    unreachable=0    failed=0

```

## Pause and Resume device migration job

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook resume_pause.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Pause a device migration job]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "pause_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-2",
      "name": "mig_job_1",
      "percentage_done": 51,
      "source": "/vplex/v2/clusters/cluster-2/devices/hnk_123",
      "source_exported": false,
      "start_time": "Tue Dec 15 07:25:13 UTC 2020",
      "status": "paused",
      "target": "/vplex/v2/clusters/cluster-2/devices/vir_2_1",
      "target_exported": false,
      "to_cluster": "/vplex/v2/clusters/cluster-2",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

TASK [Resume a device migration job]
*****
*****
changed: [localhost]

```

```

TASK [debug]
*****
*****
ok: [localhost] => {
  "resume_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-2",
      "name": "mig_job_1",
      "percentage_done": 52,
      "source": "/vplex/v2/clusters/cluster-2/devices/hnk_123",
      "source_exported": false,
      "start_time": "Tue Dec 15 07:25:13 UTC 2020",
      "status": "in-progress",
      "target": "/vplex/v2/clusters/cluster-2/devices/vir_2_1",
      "target_exported": false,
      "to_cluster": "/vplex/v2/clusters/cluster-2",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

```

## Cancel device migration job

```

(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook cancel.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Cancel a device migration job]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "cancel_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-2",
      "name": "mig_job_1",
      "percentage_done": 100,
      "source": "/vplex/v2/clusters/cluster-2/devices/hnk_123",
      "source_exported": false,
      "start_time": "Tue Dec 15 06:56:16 UTC 2020",
      "status": "cancelled",
      "target": "/vplex/v2/clusters/cluster-2/devices/vir_2_1",
      "target_exported": false,
      "to_cluster": "/vplex/v2/clusters/cluster-2",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

```

```
PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0
```

## Commit device migration job

```
(py3_ans2_7) [root@localhost playbook_product_guide]# ansible-playbook commit.yml
[WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Commit a device migration job]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "commit_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-2",
      "name": "mig_job_1",
      "percentage_done": 100,
      "source": "/vplex/v2/clusters/cluster-2/devices/hnk_123",
      "source_exported": false,
      "start_time": "Tue Dec 15 07:08:55 UTC 2020",
      "status": "committed",
      "target": "/vplex/v2/clusters/cluster-2/devices/vir_2_1",
      "target_exported": false,
      "to_cluster": "/vplex/v2/clusters/cluster-2",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

PLAY RECAP
*****
*****
localhost                : ok=3    changed=1    unreachable=0    failed=0
```

## Update the transfer size of a device migration job

```
(py3_ans2_9) [root@dsvej252 data_migration]# ansible-playbook transfer_size.yml
[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****
*****
```

```

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Update transfer size of a device migration job]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "update_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "test_dev_mig",
      "percentage_done": 100,
      "source": "/vplex/v2/clusters/cluster-1/devices/test_dev_1",
      "source_exported": true,
      "start_time": "Thu Dec 03 06:21:02 UTC 2020",
      "status": "complete",
      "target": "/vplex/v2/clusters/cluster-2/devices/test_dev_2",
      "target_exported": true,
      "to_cluster": "/vplex/v2/clusters/cluster-2",
      "transfer_size": 40960,
      "type": "full"
    }
  }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Delete a device migration job

```

(py3_ans2_9) [root@dsvej252 data_migration]# ansible-playbook delete.yml
[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Delete device migration job]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "delete_job": {
    "changed": true,
    "failed": false,
    "job_details": null
  }
}

```

```
PLAY RECAP
*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0
```

## Create extent migration job

```
(py3_ans2_9) [root@localhost data_migration]# ansible-playbook create.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Create an extent migration job]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "create_ext_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "mig_ext_job_1",
      "percentage_done": 0,
      "source": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_1",
      "start_time": "Wed Jan 27 06:24:02 UTC 2021",
      "status": "in-progress",
      "target": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_2",
      "to_cluster": "/vplex/v2/clusters/cluster-1",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0
```

## Get an extent migration job

```
(py3_ans2_9) [root@localhost data_migration]# ansible-playbook get.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get an extent migration job]
*****
*****
```

```

*****
ok: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "get_ext_job": {
    "changed": false,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "mig_ext_job_1",
      "percentage_done": 100,
      "source": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_1",
      "start_time": "Wed Jan 27 06:24:02 UTC 2021",
      "status": "complete",
      "target": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_2",
      "to_cluster": "/vplex/v2/clusters/cluster-1",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

PLAY RECAP
*****
localhost      : ok=3    changed=0    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Pause and Resume extent migration job

```

(py3_ans2_9) [root@localhost data_migration]# ansible-playbook resume_pause.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Pause an extent migration job]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "update_ext_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "mig_ext_job_1",
      "percentage_done": 53,
      "source": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_1",
      "start_time": "Wed Jan 27 07:07:22 UTC 2021",
      "status": "paused",
      "target": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_2",
      "to_cluster": "/vplex/v2/clusters/cluster-1",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

```

```

TASK [Resume an extent migration job]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "resume_ext_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "mig_ext_job_1",
      "percentage_done": 53,
      "source": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_1",
      "start_time": "Wed Jan 27 07:07:22 UTC 2021",
      "status": "in-progress",
      "target": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_2",
      "to_cluster": "/vplex/v2/clusters/cluster-1",
      "transfer_size": 131072,
      "type": "full"
    }
  }
}

PLAY RECAP
*****
*****
localhost      : ok=5    changed=2    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Cancel extent migration job

```

(py3_ans2_9) [root@localhost data_migration]# ansible-playbook cancel.yml

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Cancel an extent migration job]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "cancel_ext_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "mig_ext_job_1",
      "status": "cancelled"
    }
  }
}

PLAY RECAP
*****
*****

```



```
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0
```

### Commit extent migration job

```
(py3_ans2_9) [root@localhost data_migration]# ansible-playbook commit.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Commit an extent migration job]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "commit_ext_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "mig_ext_job_1",
      "percentage_done": 100,
      "source": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_1",
      "start_time": "Wed Jan 27 07:07:22 UTC 2021",
      "status": "committed",
      "target": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_2",
      "to_cluster": "/vplex/v2/clusters/cluster-1",
      "transfer_size": 40960,
      "type": "full"
    }
  }
}

PLAY RECAP
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0
```

### Update the transfer size of extent migration job

```
(py3_ans2_9) [root@localhost data_migration]# ansible-playbook transfer_size.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Update transfer size of an extent migration job]
*****
changed: [localhost]

TASK [debug]
```

```

*****
*****
ok: [localhost] => {
  "update_ext_job": {
    "changed": true,
    "failed": false,
    "job_details": {
      "from_cluster": "/vplex/v2/clusters/cluster-1",
      "name": "mig_ext_job_1",
      "percentage_done": 100,
      "source": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_1",
      "start_time": "Wed Jan 27 07:07:22 UTC 2021",
      "status": "complete",
      "target": "/vplex/v2/clusters/cluster-1/extents/ans_mig_ext_2",
      "to_cluster": "/vplex/v2/clusters/cluster-1",
      "transfer_size": 40960,
      "type": "full"
    }
  }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

### Delete an extent migration job

```

(py3_ans2_9) [root@localhost data_migration]# ansible-playbook delete.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Testing Data Migration operations]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Delete an extent migration job]
*****
*****
changed: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "delete_ext_job": {
    "changed": true,
    "failed": false,
    "job_details": null
  }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=1    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Rediscover array module

The array module rediscovers the LUNs in the storage array.

The array module has the following functions.

- Rediscover Array
- Get Array

## Rediscover array

To rediscover the array, run the appropriate playbook.

### Prerequisite

To rediscover a storage array in given cluster, the storage array should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Rediscover the StorageArray
  dell EMC_vplex_array:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    array_name: "Dell EMC-PowerStore-4PFLBX2"
    rediscover: true
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If array is present- Success with "changed": True. Rediscovered the given array.
2. If array is not present- Failure with "changed": False. Exits with the failure message stating as "Resource not found".

## Get array

To get the array, run the appropriate playbook.

### Prerequisite

To get details of a storage array from a given cluster, the storage array should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Get StorageArray
  dell EMC_vplex_array:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    array_name: "Dell EMC-PowerStore-4PFLBX2"
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage array is present- Success with "changed": False. Displays the corresponding array details.
2. If storage array is not present- Failure with "changed": False. Exits with the failure message stating as "Resource not found".

## Rediscover array module parameters

The parameters for the rediscover array module are listed.

**Table 16. Parameters for the rediscover array module**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	The user name to access the VPLEX server.
vplexpassword	-	str	Mandatory	The password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Mandatory	To validate the SSL certificate. <ul style="list-style-type: none"> <li>• True - Verifies the SSL certificate</li> <li>• False - Specified that the SSL certificate should not be verified</li> </ul>
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dell EMC_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	-	str	Mandatory	Name of the cluster.
array_name	-	str	Mandatory	Name of the array.
rediscover	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	<ul style="list-style-type: none"> <li>• True - Rediscover Array.</li> <li>• False - Get array details.</li> </ul>

## Sample output

### Rediscover array

```
(py3_ans2_9) [root@dsvej252 arrays]# ansible-playbook rediscover.yml
[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Rediscover StorageArray Tests]
*****

TASK [Gathering Facts]
*****
ok: [localhost]

TASK [Rediscover StorageArray]
*****
changed: [localhost]

TASK [debug]
*****
ok: [localhost] => {
  "Rediscover_StorageArray_details": {
    "array_details": {
```

```

        "connectivity_status": "ok",
        "controllers": [
            "4PFLBX2"
        ],
        "logical_unit_count": 461,
        "name": "DellEMC-PowerStore-4PFLBX2",
        "ports": [
            "0x58ccf0904a6000f8",
            "0x58ccf0904a6100f8"
        ],
        "storage_array_family": "powerstore",
        "storage_groups": "/vplex/v2/clusters/cluster-1/storage_arrays/DellEMC-
PowerStore-4PFLBX2/storage_groups",
        "storage_pools": "/vplex/v2/clusters/cluster-1/storage_arrays/DellEMC-
PowerStore-4PFLBX2/storage_pools"
    },
    "changed": true,
    "failed": false
}
}

```

PLAY RECAP

```

*****
localhost           : ok=3    changed=1    unreachable=0    failed=0
skipped=0           rescued=0    ignored=0

```

## Get array

```

(py3_ans2_9) [root@dsvej252 arrays]# ansible-playbook get.yml
[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

```

PLAY [Rediscover StorageArray Tests]

```

*****
*****

```

TASK [Gathering Facts]

```

*****
*****

```

ok: [localhost]

TASK [Get StorageArray]

```

*****
*****

```

ok: [localhost]

TASK [debug]

```

*****
*****

```

```

ok: [localhost] => {
    "StorageArray_details": {
        "array_details": {
            "connectivity_status": "ok",
            "controllers": [
                "4PFLBX2"
            ],
            "logical_unit_count": 461,
            "name": "DellEMC-PowerStore-4PFLBX2",
            "ports": [
                "0x58ccf0904a6200f8",
                "0x58ccf0904a6300f8"
            ],
            "storage_array_family": "powerstore",
            "storage_groups": "/vplex/v2/clusters/cluster-2/storage_arrays/DellEMC-
PowerStore-4PFLBX2/storage_groups",
            "storage_pools": "/vplex/v2/clusters/cluster-2/storage_arrays/DellEMC-
PowerStore-4PFLBX2/storage_pools"
        },
        "changed": false,
    }
}

```

```

        "failed": false
    }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=0    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Maps module

The maps module shows the hierarchy of the provided storage entity.

The maps module has the following functions:

- Get map for given virtual volume
- Get map for given distributed virtual volume
- Get map for given device
- Get map for given distributed device
- Get map for given extent
- Get map for given storage volume

## Get map for given virtual volume

To get the map for given virtual volume, run the appropriate playbook.

### Prerequisite

The virtual volume should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```

- name: Get map for virtual volume
  dell EMC vplex_map:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    entity_type: 'virtual_volumes'
    entity_name: 'ansible_vv'

```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If virtual volume is present- Success with "changed": False. Display the map for virtual volume.
2. If virtual volume is invalid- Failure with "changed": False. Exits with failure message stating as "Could not get the map for virtual volume".

## Get map for given distributed virtual volume

To get the map for given distributed virtual volume, run the appropriate playbook.

### Prerequisite

The distributed virtual volume should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Get map for distributed virtual volume
  dellemc_vplex_map:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    entity_type: 'virtual_volumes'
    entity_name: 'ansible_dr_vv'
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed virtual volume is present- Success with "changed": False. Display the map for distributed virtual volume.
2. If distributed virtual volume is invalid- Failure with "changed": False. Exits with failure message stating as "Could not get the map for distributed virtual volume".

## Get map for given device

To get the map for given device, run the appropriate playbook.

#### Prerequisite

The device should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Get map for device
  dellemc_vplex_map:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    entity_type: 'devices'
    entity_name: 'ansible_dev'
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

#### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If device is present- Success with "changed": False. Display the map for device.
2. If device is invalid- Failure with "changed": False. Exits with failure message stating as "Could not get the map for device".

## Get map for given distributed device

To get the map for given distributed device, run the appropriate playbook.

#### Prerequisite

The distributed device should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Get map for distributed device
  dellemc_vplex_map:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
```

```
entity_type: 'devices'
entity_name: 'ansible_dev'
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If distributed device is present- Success with "changed": False. Display the map for distributed device.
2. If distributed device is invalid- Failure with "changed": False. Exits with failure message stating as "Could not get the map for distributed device".

## Get map for given extent

To get the map for given extent, run the appropriate playbook.

### Prerequisite

The extent should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Get map for extent
  dellemc_vplex_map:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    entity_type: 'extents'
    entity_name: 'ansible_extent'
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result

On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If extent is present- Success with "changed": False. Display the map for extent.
2. If extent is invalid- Failure with "changed": False. Exits with failure message stating as "Could not get the map for extent" .

## Get map for given storage volume

To get the map for given storage volume, run the appropriate playbook.

### Prerequisite

The storage volume should be present in the VPLEX setup.

The syntax of the task is shown as follows:

```
- name: Get map for storage volume
  dellemc_vplex_map:
    vplexhost: "{{ vplexhost }}"
    vplexuser: "{{ vplexuser }}"
    vplexpassword: "{{ vplexpassword }}"
    verifycert: "{{ verifycert }}"
    cluster_name: "cluster-1"
    entity_type: 'storage_volumes'
    entity_name: 'ansible_stor_vol'
```

The parameters must be set before the user runs the playbook. For more information about the parameters, see the [Parameters table](#).

### Expected result



On running the playbook with the appropriate playbook syntax, the following is the expected output:

1. If storage volume is present- Success with "changed": False. Display the map for storage volume.
2. If storage volume is invalid- Failure with "changed": False. Exits with failure message stating as "Could not get the map for storage volume".

## Maps module parameters

The maps module parameters are listed.

**Table 17. Maps module parameters**

Parameter name	Choice or default	Type	Mandatory/Optional Parameters	Description
vplexhost	-	str	Mandatory	IP or FQDN of the VPLEX host.
vplexuser	-	str	Mandatory	User name to access the VPLEX server.
vplexpassword	-	str	Mandatory	Password to access the VPLEX server.
verifycert	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Mandatory	To validate the SSL certificate. If it is True, it verifies the SSL certificate. If it is False, it does not verify the SSL certificate.
ssl_ca_cert	-	str	Optional	Path of SSL CA certificate file specified in .pem format. It is required only when verifycert is set to "True."
debug	<ul style="list-style-type: none"> <li>• True</li> <li>• False</li> </ul>	bool	Optional	It specifies log or does not log the debug statements in the Ansible module log file (dellemc_ansible_vplex.log).
vplex_timeout	Default: 30 sec	int	Optional	It specifies the network connectivity timeout value to connect to the VPLEX host in seconds.
cluster_name	-	str	Optional	Name of the cluster.
entity_type	<ul style="list-style-type: none"> <li>• virtual_volumes</li> <li>• devices</li> <li>• extents</li> <li>• storage_volumes</li> </ul>	str	Mandatory	Name of the entity type.  <b>Non supported entities</b> <ul style="list-style-type: none"> <li>• Distributed consistency group</li> <li>• Storage view</li> <li>• Initiators</li> <li>• Ports</li> <li>• Consistency group</li> <li>• Storage arrays</li> <li>• Array management providers</li> </ul>
entity_name	-	str	Mandatory	Name of the entity.

## Sample output

### Get map for given virtual volume

```
(py3_ans2_9) [root@localhost arrays]# ansible-playbook get_vv_map.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Maps module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]
```

```

TASK [Get Map - virtual volumes]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "virt_vol_map": {
    "changed": false,
    "failed": false,
    "map_details": [
      "( storage_views ): ansible_storage_view_new",
      "  (* virtual_volumes ): ansible_vir_1",
      "    ( devices ): ansible_dev_vir_cg_1",
      "    ( devices ): ansible_dev_1_1609994253713",
      "      ( extents ): ansible_dd_ext1_3",
      "      ( storage_volumes ): ansible_dd_stor_voll_3",
      "      ( storage_arrays ): EMC-SYMMETRIX-197200581",
      "    ( devices ): dev_123",
      "      ( extents ): ansible_dd_ext1_5",
      "      ( storage_volumes ): ansible_dd_stor_voll_5",
      "      ( storage_arrays ): EMC-SYMMETRIX-197200581",
      "    ( extents ): ansible_ext_cg_2",
      "      ( storage_volumes ): new_test_used",
      "      ( storage_arrays ): EMC-SYMMETRIX-197200581",
      "    ( extents ): ansible_ext_cg_1",
      "      ( storage_volumes ): Ansible_Test_Volume_4",
      "      ( storage_arrays ): EMC-SYMMETRIX-197200581"
    ]
  }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=0    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Get map for given distributed virtual volume

```

(py3_ans2_9) [root@localhost arrays]# ansible-playbook get_dvv_map.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Maps module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get Map - distributed virtual volume]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "dist_virt_vol_map": {
    "changed": false,
    "failed": false,
    "map_details": [
      "( storage_views ): view_C2_VATS_00000",
      "  (* distributed_virtual_volumes ): ansible_dr_vv_1",
      "    ( distributed_devices ): ansible_dd_5",
      "    ( devices ): device_test1_c2",

```

```

        "          ( extents ): extent_VPD83T3_60000970000197200581533030434430_1",
        "          ( storage_volumes ):
VPD83T3:60000970000197200581533030434430",
        "          ( storage_arrays ): EMC-SYMMETRIX-197200581",
        "          ( devices ): device_test1_c1",
        "          ( extents ): extent_VPD83T3_60000970000197200581533030353235_1",
        "          ( storage_volumes ):
VPD83T3:60000970000197200581533030353235",
        "          ( storage_arrays ): EMC-SYMMETRIX-197200581"
    ]
}
}

PLAY RECAP
*****
*****
localhost          : ok=3    changed=0    unreachable=0    failed=0
skipped=0          rescued=0    ignored=0

```

## Get map for given device

```

(py3_ans2_9) [root@localhost arrays]# ansible-playbook get_device_map.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Maps module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get Map - devices]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "dev_map": {
    "changed": false,
    "failed": false,
    "map_details": [
      "( virtual_volumes ): ansible_vir_2",
      "  (* devices ): ansible_dev_vir_cg_2",
      "    ( extents ): extent_1",
      "      ( storage_volumes ): vi_ex_test_4",
      "      ( storage_arrays ): EMC-SYMMETRIX-197200581"
    ]
  }
}

PLAY RECAP
*****
*****
localhost          : ok=3    changed=0    unreachable=0    failed=0
skipped=0          rescued=0    ignored=0

```

## Get map for given distributed device

```

(py3_ans2_9) [root@localhost arrays]# ansible-playbook get_dist_device_map.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Maps module operations on VPLEX]
*****
*****

```

```

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get Map - distributed devices]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "dist_dev_map": {
    "changed": false,
    "failed": false,
    "map_details": [
      "( storage_views ): view_C2_VATS_00000",
      "( storage_views ): ansible_storage_view_new",
      "  ( distributed_consistency_groups ): ansible_dr_cg_vol",
      "    ( distributed_virtual_volumes ): ansible_vv_dd_vol",
      "      (* distributed_devices ): ansible_vv_dd",
      "        ( devices ): device_ansible_vv_c2",
      "          ( extents ):
extent_VPD83T3_60000970000197200581533030434136_1",
      "            ( storage_volumes ):
VPD83T3:60000970000197200581533030434136",
      "              ( storage_arrays ): EMC-SYMMETRIX-197200581",
      "                ( devices ): device_ansible_vv_c1",
      "                  ( extents ): extent__strg_vol_new_name_1",
      "                    ( storage_volumes ): _strg_vol_new_name",
      "                      ( storage_arrays ): EMC-SYMMETRIX-197200581"
    ]
  }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=0    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Get map for given extent

```

(py3_ans2_9) [root@localhost arrays]# ansible-playbook get_extent_map.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Maps module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get Map - extents]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "ext_map": {
    "changed": false,
    "failed": false,
    "map_details": [
      "( * extents ): ansible_ext_dd_cg_3",
      "  ( storage_volumes ): Symm0581_051A",

```

```

        "      ( storage_arrays ): EMC-SYMMETRIX-197200581"
    ]
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=0    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Get map for storage volume

```

(py3_ans2_9) [root@localhost arrays]# ansible-playbook get_stor_vol_map.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Perform Maps module operations on VPLEX]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [localhost]

TASK [Get Map - storage volumes]
*****
*****
ok: [localhost]

TASK [debug]
*****
*****
ok: [localhost] => {
  "strg_vol_map": {
    "changed": false,
    "failed": false,
    "map_details": [
      "( * storage_volumes ): Symm0581_0556",
      "      ( storage_arrays ): EMC-SYMMETRIX-197200581"
    ]
  }
}

PLAY RECAP
*****
*****
localhost      : ok=3    changed=0    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

## Sample playbooks

Sample playbooks illustrate the proper usage and some advance capabilities of the existing modules.

**NOTE:** If the user does not use modules and playbooks as collections, then the roles folder must be placed in the samples folder.

```
[root@vmrhel-94 ansible_vplex]# ls
CHANGELOG.md  dellemc-vplex-1.2.0.tar.gz  docs      meta      requirements.txt
sanity_logs
CHANGELOG.rst  dellemc-vplex-1.2.2.tar.gz  galaxy.yml  plugins  requirements.yml  tools
changelogs    dellemc-vplex-1.3.0.tar.gz  LICENSE    README.md  roles

[root@vmrhel-94 ansible_vplex]:~/GIT/ansible_vplex$ mv roles/ docs/samples/
[root@vmrhel-94 ansible_vplex]:~/GIT/ansible_vplex$ ls

CHANGELOG.md  dellemc-vplex-1.2.0.tar.gz  docs      meta      requirements.txt
sanity_logs
CHANGELOG.rst  dellemc-vplex-1.2.2.tar.gz  galaxy.yml  plugins  requirements.yml  tools
changelogs    dellemc-vplex-1.3.0.tar.gz  LICENSE    README.md  roles
[root@vmrhel-94 ansible_vplex]:~/GIT/ansible_vplex$ tree
-- CHANGELOG.md
-- CHANGELOG.rst
-- changelogs
-- | changelog.yaml
-- | config.yaml
-- dellemc-vplex-1.2.0.tar.gz
-- dellemc-vplex-1.2.2.tar.gz
-- dellemc-vplex-1.3.0.tar.gz
-- docs
-- | Ansible modules for Dell EMC VPLEX v1.2 Product Guide.pdf
-- | Ansible modules for Dell EMC VPLEX v1.2 Release Notes.pdf
-- | Ansible modules for Dell EMC VPLEX v1.3 Product Guide.pdf
-- | Ansible modules for Dell EMC VPLEX v1.3 Release Notes.pdf
-- | playbooks
-- | | dellemc_vplex_array_tests.yml
-- | | dellemc_vplex_consistency_group_tests.yml
-- | | dellemc_vplex_data_migration_tests.yml
-- | | dellemc_vplex_device_tests.yml
-- | | dellemc_vplex_distributed_cg_tests.yml
-- | | dellemc_vplex_distributed_device_tests.yml
-- | | dellemc_vplex_distributed_vv_tests.yml
-- | | dellemc_vplex_extent_tests.yml
-- | | dellemc_vplex_gatherfacts_tests.yml
-- | | dellemc_vplex_initiator_tests.yml
-- | | dellemc_vplex_maps_tests.yml
-- | | dellemc_vplex_port_tests.yml
-- | | dellemc_vplex_storage_view_tests.yml
-- | | dellemc_vplex_storage_volume_tests.yml
-- | | dellemc_vplex_virtual_volume_tests.yml
-- | | playbooks_manual_testing
-- | | | dellemc_vplex_array_tests.yml
-- | | | dellemc_vplex_consistency_group_tests.yml
-- | | | dellemc_vplex_data_migration_tests.yml
-- | | | dellemc_vplex_device_tests.yml
-- | | | dellemc_vplex_distributed_cg_tests.yml
-- | | | dellemc_vplex_distributed_device_tests.yml
-- | | | dellemc_vplex_distributed_vv_tests.yml
-- | | | dellemc_vplex_extent_tests.yml
-- | | | dellemc_vplex_gatherfacts_tests.yml
-- | | | dellemc_vplex_initiator_tests.yml
-- | | | dellemc_vplex_maps_tests.yml
-- | | | dellemc_vplex_port_tests.yml
-- | | | dellemc_vplex_storage_view_tests.yml
```

```

-- dellemc_vplex_storage_volume_tests.yml
-- dellemc_vplex_virtual_volume_tests.yml
-- samples
--   add_volumes_to_views.yml
--   get_unclaimed_volumes.yml
--   get_volumes_given_size.yml
--   vars
--     connection.yml
--     credentials.yml
--     input_details.yml
--     input_provisioning.yml
--     input_teardown.yml
--     input.yml
--     view_details.yml
--     volume_name.yml
--   vplex_cache_invalidate_volumes.yml
--   vplex_local_metro_teardown.yml
--   vplex_provisioning_end_to_end.yml
--   vplex_provisioning_local.yml
--   vplex_provisioning_metro.yml
-- galaxy.yml
-- LICENSE
-- meta

```

**Table 18. Sample playbooks**

Playbook names	Operations
get_unclaimed_volumes.yml	List of storage volumes that are unclaimed in a specified cluster.
get_volumes_given_size.yml	Provides a list of storage volumes whose capacity is equal to or greater than the specified size in a cluster.
add_volumes_to_views.yml	It creates multiple virtual volumes and adds it to different storage views equally.
vplex_cache_invalidate_volumes.yml	It performs the cache-invalidate operation on virtual volumes.
vplex_provisioning_local.yml	It performs end-to-end provisioning on a local cluster with claiming of two storage volumes until creation of a virtual volume and storage view and finally adding the virtual volume into the created storage view.
vplex_provisioning_metro.yml	It creates end-to-end provisioning on metro VPLEX setup starting from claiming of storage volumes followed by adding the virtual volumes into the created storage view .
vplex_local_metro_teardown_local.yml	It performs tear down operation of the virtual volume or distributed virtual volume through specifying the storage view name.
vplex_provisioning_end_to_end.yml	It performs end to end provisioning of local or metro from claiming of storage volumes until creation of the virtual volume or distributed virtual volume, and adding them to the storage view.