Dell Storage PowerShell SDK Cookbook

Dell EMC Engineering
April 2017

A Dell EMC Technical White Paper

Revisions

Date	Description
December 2015	Initial release
April 2016	Added samples for replication, replication QoS, Live Volume, and Copy/Mirror/Migrate (CMM)
October 2016	Added samples for FluidFS
April 2017	Added samples for disaster recovery (DR)

Acknowledgements

Authors: Mike Matthews, Mordi Shushan

Table of contents

Executive summary

Dell EMC has provided a PowerShell interface for SC Series arrays for many years. The Dell™ Storage Center PowerShell Command Set provides cmdlets for many storage tasks, allowing a single script to automate processes that work with storage as well as other components in the data center. For example, the same script can create volumes on an SC Series array, present the volumes to a server, format the disks in Microsoft® Windows Server®, and then create Microsoft SQL Server databases on the new disks.

The Dell Storage PowerShell SDK command set is the next-generation PowerShell interface that provides more functionality than the legacy command set. It is designed to work with Dell EMC™ Storage Manager (DSM). Any task that can be performed using the DSM client can be automated in a PowerShell script using the PowerShell SDK.

This document is designed to help IT professionals understand how to use the Dell Storage PowerShell SDK to automate storage-related tasks. Instead of providing detailed information about how each individual PowerShell command works, this cookbook provides sample scripts (recipes) to demonstrate how to use the commands together to perform common storage tasks. The information provided through the sample scripts should allow readers to more quickly develop PowerShell scripts to automate tasks in their data centers.

The initial focus of this document was to help users transition from the legacy command set to the Dell Storage PowerShell SDK command set. The document now covers functionality originally provided by the legacy command set as well as new functionality provided by the Dell Storage PowerShell SDK. Subsequent releases of this document will include additional functionality until the entire PowerShell SDK is covered.

1 Getting started

Before a script can take advantage of the cmdlets in the Dell Storage PowerShell SDK, the command set must first be installed on the machine that is running the script. Once installed, a script will be able to import the PowerShell module containing the cmdlets.

One major difference between the legacy Storage Center PowerShell Command Set and the Dell Storage PowerShell SDK is the connection required to use the cmdlets. The legacy command set connects directly to an SC Series array, while the PowerShell SDK connects to a DSM Data Collector. Once a connection is made to the DSM Data Collector, the PowerShell SDK cmdlets can be used to manage SC Series arrays registered with the Data Collector.

1.1 Install the command set

The Dell Storage PowerShell SDK must be installed on any machine that will run scripts using the SDK cmdlets. The Dell Storage PowerShell SDK can be downloaded from Dell Support or from the Knowledge Center located on the SC Series Portal (login required). To install the PowerShell SDK, choose an install directory and unzip the contents of the downloaded .zip file into that directory. Make a note of the install directory as it is referenced when importing the module containing the PowerShell SDK cmdlets.

1.2 Import the module

Instead of using a snap-in like the legacy command set, the Dell Storage PowerShell SDK is implemented using a module. Before a script can use any of the cmdlets in the PowerShell SDK, the module containing those cmdlets must be imported. The following sample script will import the module installed in **C:\PS_SDK**.

```
# Import the module for the Dell Storage PowerShell SDK
Import-Module "C:\PS_SDK\DellStorage.ApiCommandSet.psd1"
```

The **Get-DellStorageApiCommandSetVersion** cmdlet can be used to retrieve the version of the PowerShell SDK once the module is imported. The following sample script will get the Dell Storage PowerShell SDK version.

```
# Get the Dell Storage PowerShell SDK version
Get-DellStorageApiCommandSetVersion
```

1.3 Dell Storage PowerShell SDK help

In the install directory, there is a zip file named **PowerShellApiHtml.zip** that contains the HTML help files for the PowerShell SDK. To use the HTML help files, unzip the help file and open **start.html**.

The PowerShell SDK also includes help for individual cmdlets that can be accessed from the command line using the **Get-Help** cmdlet. The following sample script will get the full help for the **Get-DellScVolume** cmdlet.

```
# Get the help for Get-DellScVolume

Get-Help Get-DellScVolume –Full
```

The **Get-Command** cmdlet can be used to get a list of cmdlets in the module. The following sample script will return the names of the cmdlets in the PowerShell SDK.

```
# Get the names of the cmdlets in the Dell Storage PowerShell SDK

Get-Command -Module "DellStorage.ApiCommandSet" | Select-Object Name
```

The **Name** parameter of the **Get-Command** cmdlet accepts wildcards. This can reduce the number of cmdlets returned by **Get-Command**, making it easier to find the desired cmdlet. The following sample script will return the cmdlets that work with SC Series snapshots (also known as Replays).

```
# Get the names of the cmdlets that work with snapshots (Replays)

Get-Command -Module "DellStorage.ApiCommandSet" -Name "*dellsc*Replay*" `
| Select-Object Name
```

1.4 Enumerations

The Dell Storage PowerShell SDK includes a set of enumerations that are used as parameters for many cmdlets. It is important to pass in a valid value defined as an enumeration, as opposed to a string or number, when using those parameters.

Each enumeration has a **GetList()** method that can be used from within PowerShell to get a list of valid names and values. The following sample script will return the valid names and values for the **CmmPriorityEnum** enumeration.

```
# Get a list of values for CmmPriorityEnum

[DellStorage.Api.Enums.CmmPriorityEnum]::GetList()
```

The HTML help files also include help for each enumeration. The **Enumerations** directory, located in the root directory for the PowerShell SDK HTML help files, contains an HTML help file for each enumeration. For example, the **CmmPriorityEnum.html** file contains the help for the **CmmPriorityEnum** enumeration.

To assign an enumeration value to a variable, you can cast either the name or the value using the full specification of the enumeration. For example, the following script will set the **$Priority** variable to use **Medium**, which is a valid name for the CmmPriorityEnum enumeration. The **$Priority** variable can then be used for a parameter expecting a **CmmPriorityEnum** enumeration value.

```
# Set $Priority to Medium
$Priority = [DellStorage.Api.Enums.CmmPriorityEnum] "Medium"
```

1.5 A word of caution

The cmdlets included with the Dell Storage PowerShell SDK are very powerful. Using them incorrectly can have disastrous consequences including data loss. It is extremely important to thoroughly test any scripts before running them in a production environment.

The sample scripts in this document do not contain any error handling. While this makes it easier to see how the cmdlets work, there may be undesirable side effects if an error occurs. Error handling should be added to all scripts, especially if they will be used in a production capacity.

Note: The sample scripts in this document are provided as an educational resource to show how the Dell Storage PowerShell SDK can be used. All sample scripts in this document are provided as-is, without warranty of any kind. The entire risk of the use or the results from the use of the samples scripts remain with the user. All scripts, regardless of the source, should be thoroughly tested before running in a production environment.

2 Connecting to a DSM Data Collector

In order to execute cmdlets from the Dell Storage PowerShell SDK, a connection has to be established to the DSM Data Collector. Cmdlets that require a connection to a DSM Data Collector will have the parameters **Connection** and **ConnectionName**. The **Connection** parameter takes a connection object. The **ConnectionName** parameter takes the name of a saved connection. If neither of these parameters are specified, the default saved connection is used. If a default saved connection is not defined, the cmdlet will fail.

2.1 Create a connection

The **Connect-DellApiConnection** cmdlet can be used to define a connection to a DSM Data Collector. The output of this cmdlet can be stored in a variable and passed to a cmdlet using the **Connection** parameter. This sample script will create a connection to the DSM Data Collector at **dsm-02.techsol.local** using the credentials for **DSMUser**. The script also shows an example of using the connection after it has been created.

```
# Assign variables

$DsmHostName = "dsm-02.techsol.local"
$DsmUserName = "DSMUser"

# Prompt for the password

$DsmPassword = Read-Host -AsSecureString `
                         -Prompt "Please enter the password for $DsmUserName"

# Create the connection

$Connection = Connect-DellApiConnection -HostName $DsmHostName `
                                        -User     $DsmUserName `
                                        -Password $DsmPassword

# Get a list of Storage Centers using the connection

Get-DellStorageCenter -Connection $Connection `
| Select-Object @{ Name="Type"; Expression={"StorageCenter"} },
                @{ Name="Name"; Expression={$_.Name}}

# Get a list of FluidFS Clusters using the connection

Get-DellFluidFSCluster -Connection $Connection `
| Select-Object @{ Name="Type"; Expression={ "FluidFS Cluster" } },
                @{ Name="Name"; Expression={ $_.InstanceName } }
```

2.2 Create a saved connection

The **Connect-DellApiConnection** cmdlet can be used to create a saved connection to a DSM Data Collector. Once it is defined, the name of the saved connection can be passed to a cmdlet using the **ConnectionName** parameter. This sample script will create a saved connection named **DSMDC** using the credentials for **DSMUser**. The script also shows an example of using the named connection.

```
# Assign variables

$DsmHostName = "dsm-02.techsol.local"
$DsmUserName = "DSMUser"
$ConnName    = "DSMDC"

# Prompt for the password

$DsmPassword = Read-Host -AsSecureString `
                         -Prompt "Please enter the password for $DsmUserName"

# Create a saved connection

Connect-DellApiConnection -HostName $DsmHostName `
                          -User     $DsmUserName `
                          -Password $DsmPassword `
                          -Save     $ConnName | Out-Null

# Get a list of Storage Centers using the named connection

Get-DellStorageCenter -ConnectionName $ConnName | Select-Object Name
```

2.3 Get a list of saved connections

The **Get-DellSavedApiConnection** cmdlet can be used to get a list of saved connections. This sample script will get a list of the saved connection that have already been created.

```
# Get a list of saved connections

Get-DellSavedApiConnection
```

2.4 Remove a saved connection

The **Remove-DellSavedApiConnection** cmdlet can be used to remove a saved connection. The saved connection can be the default saved connection or a non-default saved connection. This sample script will remove the **DSMDC** saved connection.

```
# Assign variables

$ConnName = "DSMDC"

# Remove the saved connection

Remove-DellSavedApiConnection -Name $ConnName
```

2.5 Create a default saved connection

The **Connect-DellApiConnection** cmdlet can be used to create a default saved connection. This sample script will create a default saved connection named **DSMDefault** using the credentials for **DSMUser**. This script also shows an example of using the default saved connection.

```
# Assign variables

$DsmHostName = "dsm-02.techsol.local"
$DsmUserName = "DSMUser"
$ConnName    = "DSMDefault"

# Prompt for the password

$DsmPassword = Read-Host -AsSecureString `
                         -Prompt "Please enter the password for $DsmUserName"


# Create the default saved connection

Connect-DellApiConnection -HostName $DsmHostName `
                          -User     $DsmUserName `
                          -Password $DsmPassword `
                          -Save     $ConnName `
                          -Default | Out-Null

# Get a list of Storage Centers using the default saved connection

Get-DellStorageCenter | Select-Object Name
```

2.6 Modify the default saved connection

The default saved connection can be modified so that it is no longer the default connection. To do that, a new default saved connection has to be created. This sample script will create a new default saved connection named **Temp** using the credentials for **DSMUser**. After it is created, the **DSMDefault** connection will no longer be the default saved connection. The new default connection will then be deleted. After the script is run, there will be not be a default saved connection.

```
# Assign variables

$DsmHostName  = "dsm-02.techsol.local"
$DsmUserName  = "DSMUser"
$ConnName     = "DSMDefault"
$TempConnName = "Temp"

# Prompt for the password

$DsmPassword  = Read-Host -AsSecureString `
                          -Prompt "Please enter the password for $DsmUserName"

# Get info about the default connection

Get-DellSavedApiConnection -Name $ConnName | Select-Object Name, DefaultConnection

# Create a new default saved connection

Connect-DellApiConnection -HostName $DsmHostName `
                          -User     $DsmUserName `
                          -Password $DsmPassword `
                          -Save     $TempConnName `
                          -Default | Out-Null
```

```powershell
# Remove the new default saved connection
Remove-DellSavedApiConnection -Name $TempConnName
# Get info about the old default connection
Get-DellSavedApiConnection -Name $ConnName | Select-Object Name, DefaultConnection
```

3 Working with SC Series storage

The Dell Storage PowerShell SDK cmdlets that interact directly with SC Series storage will have the prefix **DellSc** or **DellStorageCenter** in the noun portion of the cmdlet. For example, the cmdlet that will return the SC Series server objects is called **Get-DellScServer**. The following sample script will return the cmdlets that interact with SC Series storage.

```
# Get the cmdlets that interact with Storage Center

Get-Command -Module "DellStorage.ApiCommandSet" `
| Where-Object { $_.Name -like "*-DellSc*" `
                 -or
                 $_.Name -like "*-DellStorageCenter*" } `
| Sort-Object Name
| Select-Object Name
```

DSM can manage multiple SC Series arrays from the same Data Collector, so it is important to specify the SC Series array when executing the PowerShell SDK cmdlets.

SC Series arrays do not require volume names and server names to be unique, even within the same folder. Folder names are not required to be unique either. When using a PowerShell cmdlet to get a volume, server, or folder by name, it is highly recommended to add additional logic to verify that only one item is returned. To make the sample scripts easier to read, that logic is not included.

The sample scripts in this section assume the Dell Storage PowerShell SDK module has already been imported and a saved connection named **DSMDC** has already been created.

3.1 Server folders

The Dell Storage PowerShell SDK provides cmdlets to manage server folders on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 1 shows the server folder cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 1 Server folder cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCServerFolder	Get-DellScServerFolder
New-SCServerFolder	New-DellScServerFolder
Remove-SCServerFolder	Remove-DellScServerFolder
Set-SCServerFolder	Set-DellScServerFolder

3.1.1 Create a server folder

The **New-DellScServerFolder** cmdlet can be used to create a server folder. This sample script creates a folder in the **Production** server folder called **Database**. The full path of the new folder will be **Production/Database**.

```
# Assign variables

$ConnName         = "DSMDC"
$ScName           = "SC 13"
$ParentFolderPath = ""
$ParentFolderName = "Production"
$FolderName       = "Database"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -Name $ScName

# Get the parent server folder

$ParentServerFolder = Get-DellScServerFolder -ConnectionName $ConnName `
                                             -StorageCenter  $StorageCenter `
                                             -FolderPath     $ParentFolderPath `
                                             -Name           $ParentFolderName

# Create the server folder

$ServerFolder = New-DellScServerFolder -ConnectionName $ConnName `
                                       -StorageCenter  $StorageCenter `
                                       -Parent         $ParentServerFolder `
                                       -Name           $FolderName
```

3.1.2 Rename a server folder

The **Set-DellScServerFolder** cmdlet can be used to rename a server folder. This sample script will rename the **Database** server folder within the **Production** folder to **SQLDatabase**. The full path of the folder will be **Production/SQLDatabase** after it is renamed.

```
# Assign variables

$ConnName      = "DSMDC"
$ScName        = "SC 13"
$FolderPath    = "Production/"
$FolderName    = "Database"
$NewFolderName = "SQLDatabase"

# Get the server folder

$ServerFolder = Get-DellScServerFolder -ConnectionName $ConnName `
                                       -ScName         $ScName `
                                       -FolderPath     $FolderPath `
                                       -Name           $FolderName

# Rename the server folder

$ServerFolder = Set-DellScServerFolder -ConnectionName $ConnName `
                                       -Instance       $ServerFolder `
                                       -Name           $NewFolderName
```

3.1.3 Move a server folder

The **Set-DellScServerFolder** cmdlet can be used to move a server folder. This sample script moves the **SQLDatabase** server folder from the **Production** folder to the **Development** folder. The full path of the folder will be **Development/SQLDatabase** after it is moved.

```
# Assign variables

$ConnName             = "DSMDC"
$ScName               = "SC 13"
$FolderPath           = "Production/"
$FolderName           = "SQLDatabase"
$NewParentFolderPath  = ""
$NewParentFolderName  = "Development"

# Get the server folder

$ServerFolder = Get-DellScServerFolder -ConnectionName $ConnName `
                                       -ScName         $ScName `
                                       -FolderPath     $FolderPath `
                                       -Name           $FolderName

# Get the new parent folder

$NewParentFolder = Get-DellScServerFolder -ConnectionName $ConnName `
                                          -ScName         $ScName `
                                          -FolderPath     $NewParentFolderPath `
                                          -Name           $NewParentFolderName

# Move the server folder

$ServerFolder = Set-DellScServerFolder -ConnectionName $ConnName `
                                       -Instance       $ServerFolder `
                                       -Parent         $NewParentFolder
```

3.1.4 Remove a server folder

The **Remove-DellScServerFolder** cmdlet can be used to remove a server folder. This sample script removes the **SQLDatabase** server folder from the **Development** folder. In order to remove a server folder, the folder must be empty. This script assumes the folder is already empty.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 13"
$FolderPath = "Development/"
$FolderName = "SQLDatabase"

# Get the server folder

$ServerFolder = Get-DellScServerFolder -ConnectionName $ConnName `
                                       -ScName         $ScName `
                                       -FolderPath     $FolderPath `
                                       -Name           $FolderName

# Remove the server folder

Remove-DellScServerFolder -ConnectionName $ConnName `
                          -Instance       $ServerFolder `
                          -Confirm:$false
```

3.2 Servers

The Dell Storage PowerShell SDK provides cmdlets to manage server objects on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 2 shows the server cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 2 Server cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Add-SCServerPort	Add-DellScPhysicalServerHba Add-DellScVirtualServerHba
Get-SCOSType	Get-DellScServerOperatingSystem
Get-SCServer	Get-DellScServer Get-DellScPhysicalServer
New-SCServer	New-DellScServerCluster New-DellScPhysicalServer New-DellScVirtualServer
Remove-SCServer	Remove-DellScServer
Remove-SCServerPort	Remove-DellScVirtualServerHba Remove-DellScPhysicalServerHba
Set-SCServer	Set-DellScPhysicalServer Set-DellScVirtualServer Set-DellScServerCluster

3.2.1 Create a Fibre Channel server

The **New-DellScPhysicalServer** cmdlet can the used to create a server. The **Add-DellScPhysicalServerHba** cmdlet can be used to add Fibre Channel connectivity to the server. This sample script will create the **SQLProd** server in the **Production/SQLDatabase** folder with two ports of Fibre Channel connectivity. The server will be defined as a Windows 2012 server with MPIO.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 13"
$ServerName = "SQLProd"
$FolderPath = "Production/"
$FolderName = "SQLDatabase"
$OSName     = "Windows 2012 MPIO"
$PortType   = [DellStorage.Api.Enums.FrontEndTransportTypeEnum] "FibreChannel"
$WWNList    = ( "21000024FF46AD70", "21000024FF46AD71" )

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -Name $ScName
```

```
# Get the server folder

$ServerFolder = Get-DellScServerFolder -ConnectionName $ConnName `
                                       -StorageCenter  $StorageCenter `
                                       -FolderPath     $FolderPath `
                                       -Name           $FolderName

# Get the server operating system

$ServerOS = Get-DellScServerOperatingSystem -ConnectionName $ConnName `
                                            -StorageCenter  $StorageCenter `
                                            -Name           $OSName

# Create the server

$Server = New-DellScPhysicalServer -ConnectionName  $ConnName `
                                   -StorageCenter   $StorageCenter `
                                   -Name            $ServerName `
                                   -OperatingSystem $ServerOS `
                                   -ServerFolder    $ServerFolder

# Add the FC WWNs to the server

ForEach( $WWN in $WWNList )
{
    Add-DellScPhysicalServerHba -ConnectionName $ConnName `
                                -Instance       $Server `
                                -HbaPortType    $PortType `
                                -WwnOrIscsiName $WWN `
                                -Confirm:$false
}
```

3.2.2 Create an iSCSI server

The **New-DellScPhysicalServer** cmdlet can be used to create new server. The **Add-DellScPhysicalServerHba** cmdlet can be used to add iSCSI connectivity to a server. This sample script will create the **SQLProd1** server in the **Production/SQLDatabase** folder with iSCSI connectivity. The server will be defined as a Windows 2012 server with MPIO.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 13"
$ServerName = "SQLProd1"
$FolderPath = "Production/"
$FolderName = "SQLDatabase"
$OSName     = "Windows 2012 MPIO"
$PortType   = [DellStorage.Api.Enums.FrontEndTransportTypeEnum] "Iscsi"
$iScsiIqn   = "iqn.1991-05.com.microsoft:sqlprod1.techsol.local"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -Name $ScName

# Get the server folder

$ServerFolder = Get-DellScServerFolder -ConnectionName $ConnName `
                                       -StorageCenter  $StorageCenter `
                                       -FolderPath     $FolderPath `
                                       -Name           $FolderName
```

```
# Get the server OS

$ServerOS = Get-DellScServerOperatingSystem -ConnectionName $ConnName `
                                            -StorageCenter  $StorageCenter `
                                            -Name           $OSName

# Create the server

$Server = New-DellScPhysicalServer -ConnectionName  $ConnName `
                                   -StorageCenter   $StorageCenter `
                                   -Name            $ServerName `
                                   -OperatingSystem $ServerOS `
                                   -ServerFolder    $ServerFolder

# Add the iSCSI IQN to the server

Add-DellScPhysicalServerHba -ConnectionName $ConnName `
                            -Instance       $Server `
                            -HbaPortType    $PortType `
                            -WwnOrIscsiName $iScsiIqn `
                            -Confirm:$false
```

3.2.3 Create a cluster server

The **New-DellScServerCluster** cmdlet can be used to create a cluster server and the **Add-DellScPhysicalServerToCluster** cmdlet can be used to add servers to the server cluster. This sample script will create a cluster server called **SQLProdCluster** and add the **SQLProd1** and **SQLProd2** servers to the cluster. The cluster server will be created in the **Production/SQLDatabase** folder and defined as a Windows 2012 cluster with MPIO.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$ServerClusterName = "SQLProdCluster"
$FolderPath        = "Production/"
$FolderName        = "SQLDatabase"
$OSName            = "Windows 2012 MPIO"
$ServerNameList    = ( "SQLProd1", "SQLProd2" )

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -Name $ScName

# Get the server folder

$ServerFolder = Get-DellScServerFolder -ConnectionName $ConnName `
                                       -StorageCenter  $StorageCenter `
                                       -FolderPath     $FolderPath `
                                       -Name           $FolderName

# Get the server operating system

$ServerOS = Get-DellScServerOperatingSystem -ConnectionName $ConnName `
                                            -StorageCenter  $StorageCenter `
                                            -Name           $OSName

# Create the server cluster

$ServerCluster = New-DellScServerCluster -ConnectionName  $ConnName `
                                         -StorageCenter   $StorageCenter `
                                         -Name            $ServerClusterName `
                                         -OperatingSystem $ServerOS `
                                         -ServerFolder    $ServerFolder
```

```
# Add servers to the cluster

ForEach ( $ServerName in $ServerNameList )
{
    $Server = Get-DellScPhysicalServer -ConnectionName $ConnName `
                                       -StorageCenter  $StorageCenter `
                                       -Name           $ServerName

    Add-DellScPhysicalServerToCluster -ConnectionName $ConnName `
                                      -Instance       $Server `
                                      -Parent         $ServerCluster `
                                      -Confirm:$false
}
```

3.2.4 Add an HBA port to a server

The **Add-DellScPhysicalServerHba** cmdlet can be used to add an HBA port to a server. This sample script will add a Fibre Channel port to the **SQLProd** server in the **Production/SQLDatabase** folder. Multiple HBA ports will be added if the **$WWNList** variable contains multiple WWNs.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$ServerName        = "SQLProd"
$ServerFolderPath  = "Production/SQLDatabase/"
$PortType          = [DellStorage.Api.Enums.FrontEndTransportTypeEnum] "FibreChannel"
$WWNList           = @( "21000024FF46AD70" )

# Get the server

$Server = Get-DellScPhysicalServer -ConnectionName   $ConnName `
                                   -ScName           $ScName `
                                   -ServerFolderPath $ServerFolderPath `
                                   -Name             $ServerName `

# Loop through the list of WWNs and add them to the server

ForEach ( $WWN in $WWNList )
{
    Add-DellScPhysicalServerHba -ConnectionName $ConnName `
                                -Instance       $Server `
                                -HbaPortType    $PortType `
                                -WwnOrIscsiName $WWN `
                                -Confirm:$false
}
```

3.2.5 Remove an HBA port from a server

The **Remove-DellScPhysicalServerHba** cmdlet can be used to remove an HBA port from a server. This sample script will remove a Fibre Channel port from the **SQLProd** server in the **Production/SQLDatabase** folder. Multiple HBA ports will be removed if the **$WWNList** variable contains multiple WWNs.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$ServerName        = "SQLProd"
$ServerFolderPath  = "Production/SQLDatabase/"
$PortType          = [DellStorage.Api.Enums.FrontEndTransportTypeEnum] "FibreChannel"
$WWNList           = @( "21000024FF46AD70" )
```

```
# Get the server

$Server = Get-DellScPhysicalServer -ConnectionName   $ConnName `
                                   -ScName           $ScName `
                                   -ServerFolderPath $ServerFolderPath `
                                   -Name             $ServerName `

# Loop through the list of WWNs and remove them from the server

ForEach ( $WWN in $WWNList )
{
    # Get the HBA port

    $ServerHBA = Get-DellScServerHba -ConnectionName $ConnName `
                                     -ScName         $ScName `
                                     -Server         $Server `
                                     -PortType       $PortType `
                                     -PortWwnList    $WWN

    # Remove the HBA port

    Remove-DellScPhysicalServerHba -ConnectionName $ConnName `
                                   -Instance       $Server `
                                   -ServerHba      $ServerHBA `
                                   -Confirm:$false
}
```

3.2.6 Remove a cluster server

The **Remove-DellScServer** cmdlet can be used to remove a cluster server. This sample script will remove all servers from the **SQLProdCluster** server cluster in the **Production/SQLDatabase** folder and then remove the server cluster.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$ServerClusterName = "SQLProdCluster"
$ServerFolderPath  = "Production/SQLDatabase/"

# Get the server cluster

$ServerCluster = Get-DellScServerCluster -ConnectionName   $ConnName `
                                         -ScName           $ScName `
                                         -ServerFolderPath $ServerFolderPath `
                                         -Name             $ServerClusterName

# Remove the servers from the server cluster

$ServerList = @( Get-DellScPhysicalServer -ConnectionName $ConnName `
                                          -ScName         $ScName `
                                          -Parent         $ServerCluster )

ForEach( $Server in $ServerList )
{
    Remove-DellScPhysicalServerFromCluster -ConnectionName $ConnName `
                                           -Instance       $Server `
                                           -Confirm:$false
}

# Remove the server cluster

Remove-DellScServer -ConnectionName $ConnName `
                    -Instance       $ServerCluster `
                    -Confirm:$false
```

3.2.7 Remove a server

The **Remove-DellScServer** cmdlet can be used to remove a server. This sample script will remove the **SQLProd1** server in the **Production/SQLDatabase** folder. In order to remove a server, all volumes must be unmapped from the server. This script assumes that the server does not have any volume mappings.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$ServerName        = "SQLProd1"
$ServerFolderPath  = "Production/SQLDatabase/"

# Get the server

$Server = Get-DellScServer -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -ServerFolderPath $ServerFolderPath `
                           -Name             $ServerName

# Remove the server

Remove-DellScServer -ConnectionName $ConnName `
                    -Instance       $Server `
                    -Confirm:$false
```

3.2.8 Get the volumes mapped to a server

The **Get-DellScMappingProfile** and **Get-DellScVolume** cmdlets can be used to list the volumes that are mapped to a server. This sample script will get the volumes mapped to the server **SQLProd** in the **Production/SQLDatabase** server folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 12"
$ServerName        = "SQLProd"
$ServerFolderPath  = "Production/SQLDatabase/"

# Get the server

$Server = Get-DellScServer -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -ServerFolderPath $ServerFolderPath `
                           -Name             $ServerName

# Get the volume mappings

$MappingList = @( Get-DellScMappingProfile -ConnectionName $ConnName `
                                           -Server         $Server )

# Get the servers

$VolumeList = @()

ForEach( $Mapping in $MappingList )
{
    $VolumeList += @( Get-DellScVolume -ConnectionName $ConnName `
                                       -Instance       $Mapping.Volume )
}
```

```
# Display the server and volume name

$VolumeList | Sort-Object Name `
            | Select-Object @{ Name="Server Name"; Expression={ $Server.Name } },
                            @{ Name="Volume Name"; Expression={ $_.Name } }
```

3.3 Volume folders

The Dell Storage PowerShell SDK provides cmdlets to manage volume folders on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 3 shows the volume folder cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 3 Volume folder cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCVolumeFolder	Get-DellScVolumeFolder
New-SCVolumeFolder	New-DellScVolumeFolder
Remove-SCVolumeFolder	Remove-DellScVolumeFolder
Set-SCVolumeFolder	Set-DellScVolumeFolder

3.3.1 Create a volume folder

The **New-DellScVolumeFolder** cmdlet can be used to create volume folder. This sample script will create the **SQLProd** volume folder in the **Production/Database** folder. The full path of the new folder will be **Production/Database/SQLProd**.

```
# Assign variables

$ConnName         = "DSMDC"
$ScName           = "SC 13"
$ParentFolderPath = "Production/"
$ParentFolderName = "Database"
$FolderName       = "SQLProd"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -Name $ScName

# Get the parent volume folder

$ParentVolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                             -StorageCenter  $StorageCenter `
                                             -FolderPath     $ParentFolderPath `
                                             -Name           $ParentFolderName

# Create the volume folder

$VolumeFolder = New-DellScVolumeFolder -ConnectionName $ConnName `
                                       -StorageCenter  $StorageCenter `
                                       -Parent         $ParentVolumeFolder `
                                       -Name           $FolderName
```

3.3.2 Rename a volume folder

The **Set-DellScVolumeFolder** cmdlet can be used to rename a volume folder. This sample script will rename the **Database** volume folder in the **Production** folder to **SQLDatabase**. The full path of the folder will be **Production/SQLDatabase** after it is renamed.

```
# Assign variables

$ConnName       = "DSMDC"
$ScName         = "SC 13"
$FolderPath     = "Production/"
$FolderName     = "Database"
$NewFolderName = "SQLDatabase"

# Get the volume folder

$VolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                       -ScName         $ScName `
                                       -FolderPath     $FolderPath `
                                       -Name           $FolderName

# Rename the volume folder

$VolumeFolder = Set-DellScVolumeFolder -ConnectionName $ConnName `
                                       -Instance       $VolumeFolder `
                                       -Name           $NewFolderName
```

3.3.3 Move a volume folder

The **Set-DellScVolumeFolder** cmdlet can be used to move a volume folder. This sample script will move the **SQLDatabase** volume folder and all subfolders from the **Production** folder to the **Development** folder. The full path of the folder will be **Development/SQLDatabase** after it is moved.

```
# Assign variables

$ConnName            = "DSMDC"
$ScName              = "SC 13"
$FolderPath          = "Production/"
$FolderName          = "SQLDatabase"
$NewParentFolderPath = ""
$NewParentFolderName = "Development"

# Get the volume folder

$VolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                       -ScName         $ScName `
                                       -FolderPath     $FolderPath `
                                       -Name           $FolderName

# Get the new parent folder

$NewParentFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                          -ScName         $ScName `
                                          -FolderPath     $NewParentFolderPath `
                                          -Name           $NewParentFolderName

# Move the volume folder

$VolumeFolder = Set-DellScVolumeFolder -ConnectionName $ConnName `
                                       -Instance       $VolumeFolder `
                                       -Parent         $NewParentFolder
```

3.3.4 Remove a volume folder

The **Remove-DellScVolumeFolder** cmdlet can be used to remove a volume folder. This sample script will remove the **SQLDatabase** volume folder from the **Development** folder. In order to remove a volume folder, the folder must be empty. This script assumes the folder is already empty.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 13"
$FolderPath = "Development/"
$FolderName = "SQLDatabase"

# Get the volume folder

$VolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                       -ScName         $ScName `
                                       -FolderPath     $FolderPath `
                                       -Name           $FolderName

# Remove the volume folder

Remove-DellScVolumeFolder -ConnectionName $ConnName `
                          -Instance       $VolumeFolder `
                          -Confirm:$false
```

3.4 Volumes

The Dell Storage PowerShell SDK provides cmdlets to manage volumes on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 4 shows the volume cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 4 Volume cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Expand-SCVolume	Expand-DellScVolume Expand-DellScVolumeToSize
Get-SCStorageType	Get-DellScStorageType
Get-SCVolume	Get-DellScVolume
Get-SCVolumeMap	Get-DellScMapping Get-DellScMappingProfile
New-SCVolume	New-DellScVolume
New-SCVolumeMap	Add-DellScVolumeToServerMap
Preallocate-SCVolume	Start-DellScVolumePreallocateStorage
Remove-SCVolume	Start-DellScVolumeRecycle
Remove-SCVolume -SkipRecycleBin	Remove-DellScVolume
Set-SCVolume	Set-DellScVolume Set-DellScVolumeConfiguration

3.4.1 Create a volume

The **New-DellScVolume** cmdlet can be used to create a new volume. This sample script will create a 110 GB volume named **SQLData** in the **Production/SQLDatabase/SQLProd** volume folder. The volume will use the default disk folder and the default storage profile.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeSize        = [DellStorage.Api.Types.StorageSize] "110GB"
$VolumeFolderPath  = "Production/SQLDatabase/"
$VolumeFolderName  = "SQLProd"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -Name $ScName
```

```
# Get the volume folder

$VolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                       -StorageCenter  $StorageCenter `
                                       -FolderPath     $VolumeFolderPath `
                                       -Name           $VolumeFolderName

# Create the volume

$Volume = New-DellScVolume -ConnectionName $ConnName `
                           -StorageCenter  $StorageCenter `
                           -Name           $VolumeName `
                           -Size           $VolumeSize `
                           -VolumeFolder   $VolumeFolder
```

3.4.2 Map a volume to a server

The **Add-DellScVolumeToServerMap** cmdlet can be used to map a volume to a server. This sample script will map the **SQLData** volume in the **Production/SQLDatabase/SQLProd** volume folder to the **SQLProd** server in the **Production/SQLDatabase** server folder.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$ServerName        = "SQLProd"
$ServerFolderPath  = "Production/SQLDatabase/"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Get the server

$Server = Get-DellScServer -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -ServerFolderPath $ServerFolderPath `
                           -Name             $ServerName

# Map the volume to the server

$VolumeMap = Add-DellScVolumeToServerMap -ConnectionName $ConnName `
                                         -Server         $Server `
                                         -Instance       $Volume
```

3.4.3 Expand a volume by adding space

The **Expand-DellScVolume** cmdlet can be used to increase the size of a volume. This sample script shows how to expand the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder by adding an additional 20 GB of space.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$SpaceToAdd        = [DellStorage.Api.Types.StorageSize] "20GB"
```

```
# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Expand the volume

$Volume = Expand-DellScVolume -ConnectionName $ConnName `
                              -Instance       $Volume `
                              -ExpandAmount   $SpaceToAdd `
                              -Confirm:$false
```

3.4.4 Expand a volume to a new size

The **Expand-DellScVolumeToSize** cmdlet can be used to increase the size of a volume. This sample script shows how to expand the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder to 175 GB.

```
# Assign variables

$ConnName         = "DSMDC"
$ScName           = "SC 13"
$VolumeName       = "SQLData"
$VolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$NewVolumeSize    = [DellStorage.Api.Types.StorageSize] "175GB"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Expand the volume

$Volume = Expand-DellScVolumeToSize -ConnectionName $ConnName `
                                    -Instance       $Volume `
                                    -NewSize        $NewVolumeSize `
                                    -Confirm:$false
```

3.4.5 Preallocate a volume

The **Start-DellScVolumePreallocateStorage** cmdlet can be used to pre-allocate a volume. This sample script shows how to pre-allocate the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder. In order to pre-allocate a volume, it must be mapped to a server. In addition, a volume cannot be pre-allocated once a snapshot has been taken. This script assumes that the volume is mapped to a server and no snapshots have been taken on the volume.

```
# Assign variables

$ConnName         = "DSMDC"
$ScName           = "SC 13"
$VolumeName       = "SQLData"
$VolumeFolderPath = "Production/SQLDatabase/SQLProd/"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -Name             $VolumeName `
                           -VolumeFolderPath $VolumeFolderPath
```

```
# Preallocate the volume

Start-DellScVolumePreallocateStorage -ConnectionName $ConnName `
                                     -Instance       $Volume `
                                     -Confirm:$false
```

3.4.6 Rename a volume

The **Set-DellScVolume** cmdlet can be used to rename a volume. This sample script will rename the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder to **SQLBackup**.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$NewVolumeName     = "SQLBackup"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Change the name of the volume

$Volume = Set-DellScVolume -ConnectionName $ConnName `
                           -Instance       $Volume `
                           -Name           $NewVolumeName
```

3.4.7 Modify volume settings

The **Set-DellScVolumeConfiguration** cmdlet can be used to change various configuration settings on a volume. This sample script shows how to use this cmdlet to change some of the configuration settings on the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder. Multiple settings can be changed at the same time by specifying multiple parameters when executing the cmdlet. For a complete list of settings that can be changed on a volume, see the help for this cmdlet.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -Name             $VolumeName `
                           -VolumeFolderPath $VolumeFolderPath

# Get the volume configuration

$VolumeConfig = Get-DellScVolumeConfiguration -ConnectionName $ConnName `
                                              -Instance       $Volume.InstanceId
```

```
# Turn off read and write cache

$VolumeConfig = Set-DellScVolumeConfiguration -ConnectionName $ConnName `
                                              -Instance       $VolumeConfig `
                                              -ReadCacheEnabled:$false `
                                              -WriteCacheEnabled:$false

# Turn on read and write cache

$VolumeConfig = Set-DellScVolumeConfiguration -ConnectionName $ConnName `
                                              -Instance       $VolumeConfig `
                                              -ReadCacheEnabled:$true `
                                              -WriteCacheEnabled:$true

# Enable compression

$VolumeConfig = Set-DellScVolumeConfiguration -ConnectionName $ConnName `
                                              -Instance       $VolumeConfig `
                                              -CompressionEnabled:$true
```

3.4.8 Move a volume to another folder

The **Set-DellScVolume** cmdlet can be used to move a volume to another folder. This sample script will move the **SQLData** volume from the **Production/SQLDatabase/SQLProd** folder to the **Development/SQLDatabase/SQLDev** folder.

```
# Assign variables

$ConnName         = "DSMDC"
$ScName           = "SC 13"
$VolumeName       = "SQLData"
$VolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$NewFolderPath    = "Development/SQLDatabase/"
$NewFolderName    = "SQLDev"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Get the new volume folder

$NewVolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                          -ScName         $ScName `
                                          -FolderPath     $NewFolderPath `
                                          -Name           $NewFolderName

# Move the volume

$Volume = Set-DellScVolume -ConnectionName $ConnName `
                           -Instance       $Volume `
                           -VolumeFolder   $NewVolumeFolder
```

3.4.9 Unmap a volume

The **Remove-DellScVolumeToServerMap** cmdlet can be used to unmap a volume from a server. This sample script will remove the server mappings from the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Unmap the volume from the server

Remove-DellScVolumeToServerMap -ConnectionName $ConnName `
                               -Instance       $Volume `
                               -Confirm:$false
```

3.4.10 Remove a volume

The **Start-DellScVolumeRecycle** cmdlet can be used to remove a volume and put it in the recycle bin. This sample script will remove the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder and place it in the recycle bin. When removing volumes in a PowerShell script, it is highly recommended to remove a volume by first placing it in the recycle bin and then removing the volume from the recycle bin.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Remove the volume, sending it to the recycle bin

Start-DellScVolumeRecycle -ConnectionName $ConnName `
                          -Instance       $Volume `
                          -Confirm:$false
```

3.4.11 Permanently remove a volume

The **Remove-DellScVolume** cmdlet can be used to permanently remove a volume. The volume is not placed in the recycle bin, which means it cannot be recovered. This sample script will permanently remove the **SQLData** volume from the **Production/SQLDatabase/SQLProd** folder. When removing volumes in a PowerShell script, it is highly recommended to remove a volume by first placing it in the recycle bin and then removing the volume from the recycle bin.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath = "Production/SQLDatabase/SQLProd/"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Remove the volume, skipping the recycle bin

Remove-DellScVolume -ConnectionName $ConnName `
                    -Instance       $Volume `
                    -Confirm:$false
```

3.4.12 Get the server mappings for a volume

The **Get-DellScMappingProfile** and **Get-DellScServer** cmdlets can be used to list the servers to which a volume is mapped. This sample script will get the server mappings for the **SQLData** volume in the **Production/SQLDatabase/SQLProd** volume folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 12"
$VolumeName        = "SQLData"
$VolumeFolderPath = "Production/SQLDatabase/SQLProd/"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Get the server mappings

$MappingList = @( Get-DellScMappingProfile -ConnectionName $ConnName `
                                           -Volume         $Volume )

# Get the servers

$ServerList = @()

ForEach ( $Mapping in $MappingList )
{
    $ServerList += @( Get-DellScServer -ConnectionName $ConnName `
                                       -Instance       $Mapping.Server )
}
```

```
# Display the server and volume name

$ServerList | Sort-Object Name `
            | Select-Object @{ Name="Server Name"; Expression={ $_.Name } },
                            @{ Name="Volume Name"; Expression={ $Volume.Name } }
```

3.5 Storage profiles

The Dell Storage PowerShell SDK provides cmdlets to manage storage profiles on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 5 shows the storage profile cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 5 Storage profile cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCStorageProfile	Get-DellScStorageProfile
New-SCStorageProfile	New-DellScStorageProfile
Remove-SCStorageProfile	Remove-DellScStorageProfile
Set-SCStorageProfile	Set-DellScStorageProfile
Set-SCVolumeStorageProfile	Set-DellScVolumeConfiguration

3.5.1 Create a storage profile

The **New-DellScStorageProfile** cmdlet can be used to create a storage profile. This sample script shows how to create the **RAID 5 Tier 1 Only** storage profile. Volumes using this storage profile will store all data using RAID 5 on tier 1.

```
# Assign variables

$ConnName    = "DSMDC"
$ScName      = "SC 13"
$ProfileName = "RAID 5 Tier 1 Only"
$RaidType    = [DellStorage.Api.Enums.StorageProfileRaidTypeEnum] "RaidFiveorSixOnly"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -ScName $ScName

# Create the profile

$StorageProfile = New-DellScStorageProfile -ConnectionName $ConnName `
                                           -StorageCenter  $StorageCenter `
                                           -Name           $ProfileName `
                                           -RaidTypeUsed   $RaidType `
                                           -UseTier1Storage:$true `
                                           -UseTier2Storage:$false `
                                           -UseTier3Storage:$false
```

3.5.2 Modify a storage profile

The **Set-DellScStorageProfile** cmdlet can be used to change a custom storage profile. Standard storage profiles included with SC Series arrays cannot be modified. This sample script will modify the **RAID 5 Tier 1 Only** storage profile to only use RAID 10 on Tier 3 and change the name to **RAID 10 Tier 3 Only**.

```
# Assign variables

$ConnName        = "DSMDC"
$ScName          = "SC 13"
$ProfileName     = "RAID 5 Tier 1 Only"
$NewProfileName = "RAID 10 Tier 3 Only"
$NewRaidType     = [DellStorage.Api.Enums.StorageProfileRaidTypeEnum] "RAIDTenOnly"

# Get the storage profile

$StorageProfile = Get-DellScStorageProfile -ConnectionName $ConnName `
                                           -ScName         $ScName `
                                           -Name           $ProfileName

# Modify the profile

$NewStorageProfile = Set-DellScStorageProfile -ConnectionName $ConnName `
                                              -Instance       $StorageProfile `
                                              -Name           $NewProfileName `
                                              -RaidTypeUsed   $NewRaidType `
                                              -UseTier1Storage:$false `
                                              -UseTier2Storage:$false `
                                              -UseTier3Storage:$true
```

3.5.3 Change the storage profile on a volume

The **Set-DellScVolumeConfiguration** cmdlet can be used to change the storage profile on a volume. This sample script changes the storage profile on the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder to **High Priority (Tier 1)**.

```
# Assign variables

$ConnName              = "DSMDC"
$ScName                = "SC 13"
$VolumeName            = "SQLData"
$VolumeFolderPath      = "Production/SQLDatabase/SQLProd/"
$NewStorageProfileName = "High Priority (Tier 1)"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -Name             $VolumeName `
                           -VolumeFolderPath $VolumeFolderPath

# Get the volume configuration

$VolumeConfig = Get-DellScVolumeConfiguration -ConnectionName $ConnName `
                                              -Instance       $Volume.InstanceId

# Get the new storage profile

$NewSCStorageProfile = Get-DellScStorageProfile -ConnectionName $ConnName `
                                                -ScName         $ScName `
                                                -Name           $NewStorageProfileName
```

```
# Change the storage profile

$VolumeConfig = Set-DellScVolumeConfiguration -ConnectionName $ConnName `
                                              -Instance       $VolumeConfig `
                                              -StorageProfile $NewSCStorageProfile
```

3.5.4 Remove a storage profile

The **Remove-DellScStorageProfile** cmdlet can be used to remove a custom storage profile. Standard storage profiles included with SC Series arrays cannot be removed. This sample script removes the **RAID 10 Tier 3 Only** storage profile. A storage profile cannot be removed if it used by a volume. This script assumes that the profile is not assigned to any volumes.

```
# Assign variables

$ConnName           = "DSMDC"
$ScName             = "SC 13"
$StorageProfileName = "RAID 10 Tier 3 Only"

# Get the storage profile

$StorageProfile = Get-DellScStorageProfile -ConnectionName $ConnName `
                                           -ScName         $ScName `
                                           -Name           $StorageProfileName

# Remove the profile

Remove-DellScStorageProfile -ConnectionName $ConnName `
                            -Instance       $StorageProfile `
                            -Confirm:$false
```

3.6 Snapshots (Replays)

The Dell Storage PowerShell SDK provides cmdlets to manage snapshots (Replays) on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 6 shows the snapshot cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 6 Snapshot (Replay) cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCReplay	Get-DellScReplay
New-SCReplay	New-DellScVolumeReplay
Remove-SCReplay	Start-DellScReplayExpire
Set-SCReplay	Set-DellScReplay

3.6.1 Create a snapshot

The **New-DellScVolumeReplay** cmdlet can be used to create a snapshot on a volume. This sample script creates two snapshots on the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder. Both snapshots will have a description of **Created by PowerShell SDK**. One snapshot is set to never expire and the other is set to expire in 3 days.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$Description       = "Created by PowerShell SDK"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Create a snapshot (Replay) that will never expire

$Replay = New-DellScVolumeReplay -ConnectionName $ConnName `
                                 -Instance       $Volume `
                                 -Description    $Description `
                                 -ExpireTime     0

# Create a snapshot (Replay) that will expire in 3 days

$ThreeDaysInMinutes = 3 * 24 * 60  # 3 days * 24 hours/day * 60 minutes/hour

$Replay = New-DellScVolumeReplay -ConnectionName $ConnName `
                                 -Instance       $Volume `
                                 -Description    $Description `
                                 -ExpireTime     $ThreeDaysInMinutes
```

3.6.2 Create a view volume on a snapshot

The **New-DellScReplayView** cmdlet can be used to create a view volume on a snapshot. This sample script will create a view volume using the latest snapshot on the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder. The view volume will be created in the same folder as the **SQLData** volume. The freeze time is used to determine the latest snapshot. If multiple snapshots have the most recent freeze time, the snapshot with the largest index will be considered the latest snapshot.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$ViewVolumeName    = $VolumeName + " View"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName
```

```
# Get the latest frozen snapshot (Replay) on the volume

$Replay = Get-DellScReplay -ConnectionName $ConnName `
                           -ScName         $ScName `
                           -CreateVolume   $Volume `
                           -Active:$false `
          | Sort-Object { [datetime]$_.FreezeTime }, `
                        { [UInt32] ( $_.GlobalIndex.Split( "-" ) )[-1] } `
          | Select-Object -Last 1

# Create the view volume in the same folder as the volume

$ViewVolume = New-DellScReplayView -ConnectionName $ConnName `
                                   -Instance       $Replay `
                                   -Name           $ViewVolumeName `
                                   -VolumeFolder   $Volume.VolumeFolder
```

3.6.3 Change the expiration date on a snapshot

The **Set-DellScReplay** cmdlet can be used to change the expiration date. Using the freeze time, this sample script will get a snapshot on the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder and change the expiration date to 6 days from the current date.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$FreezeTime        = [datetime]"11/19/2015 04:35:22PM"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Get the snapshot (Replay) on the volume

$Replay = Get-DellScReplay -ConnectionName $ConnName `
                           -ScName         $ScName `
                           -CreateVolume   $Volume `
                           -FreezeTime     $FreezeTime `
                           -Active:$false

# Change the expiration date on the snapshot (Replay)
# Set the latest snapshot to expire in 6 days

$NewExpireTime = ( Get-Date ).AddDays( 6 )

$Replay = Set-DellScReplay -ConnectionName $ConnName `
                           -Instance       $Replay `
                           -ExpireTime     $NewExpireTime `
                           -Confirm:$false
```

3.6.4 Set a snapshot to never expire

Currently, the PowerShell SDK does not provide a way to set a snapshot to never expire. However, the **Set-DellScReplay** cmdlet can be used to set a snapshot to expire many years in the future, which effectively prevents it from expiring. Using the freeze time, this sample script will get a snapshot on the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder and set it to expire on January 1, 2030.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$FreezeTime        = [datetime] "11/20/2015 09:38:25AM"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName

# Get the snapshot (Replay) on the volume

$Replay = Get-DellScReplay -ConnectionName $ConnName `
                           -ScName         $ScName `
                           -CreateVolume   $Volume `
                           -FreezeTime     $FreezeTime `
                           -Active:$false

# Set the snapshot (Replay) to expire far in the future

$NewExpireTime = [datetime] "1/1/2030 00:00:00"

$Replay = Set-DellScReplay -ConnectionName $ConnName `
                           -Instance       $Replay `
                           -ExpireTime     $NewExpireTime `
                           -Confirm:$false
```

3.6.5 Remove a snapshot

The **Start-DellScReplayExpire** cmdlet can be used to remove a snapshot by setting it to expire. This sample script will expire the oldest snapshot on the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder. The freeze time is used to determine the oldest snapshot. If multiple snapshots have the oldest freeze time, the snapshot with the smallest index will be considered the oldest snapshot.

```
# Assign variables

$ConnName          = "DSMDC"
$ScName            = "SC 13"
$VolumeName        = "SQLData"
$VolumeFolderPath  = "Production/SQLDatabase/SQLProd/"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $VolumeFolderPath `
                           -Name             $VolumeName
```

```
# Get the oldest frozen snapshot (Replay) on the volume

$Replay = Get-DellScReplay -ConnectionName $ConnName `
                           -ScName         $ScName `
                           -CreateVolume   $volume `
                           -Active:$false `
          | Sort-Object { [datetime]$_.FreezeTime }, `
                        { [UInt32] ( $_.GlobalIndex.Split( "-" ) )[-1] } `
          | Select-Object -First 1

# Expire the snapshot (Replay)

Start-DellScReplayExpire -ConnectionName $ConnName `
                         -Instance       $Replay `
                         -Confirm:$false
```

3.7 Snapshot profiles

The Dell Storage PowerShell SDK provides cmdlets to manage snapshot profiles on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 7 shows the snapshot profile cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 7 Snapshot profile cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Add-SCReplayProfileRule	New-DellScReplayProfileRule
Get-SCReplayProfile	Get-DellScReplayProfile
Get-SCReplaySchedule	Get-DellScReplayProfileRule
New-SCReplayProfile	New-DellScReplayProfile
New-SCReplaySchedule	New-DellScReplayProfileRule
Remove-SCReplayProfile	Remove-DellScReplayProfile
Remove-SCReplaySchedule	Remove-DellScReplayProfileRule
Remove-SCVolumeReplayProfile	Set-DellScVolumeConfiguration
Set-SCReplayProfile	Set-DellScReplayProfile
Set-SCVolumeReplayProfile	Set-DellScVolumeConfiguration

3.7.1 Create a snapshot profile

The **New-DellScReplayProfileRule** cmdlet can be used to create a snapshot profile. This sample script will create a serial snapshot profile called **Dev Database**. The **New-DellScReplayProfileRule** cmdlet is used to add a schedule rule to take a snapshot every hour on a daily basis. Snapshots created by this rule will expire in one day.

```
# Assign variables

$ConnName         = "DSMDC"
$ScName           = "SC 13"
$ProfileName      = "Dev Database"
$ProfileType      = [DellStorage.Api.Enums.ReplayProfileTypeEnum] "Standard"
$RuleName         = "Daily every 1 hour"
$ExpireDays       = 1
$ScheduleType     = [DellStorage.Api.Enums.ScheduleTypeEnum] "Daily"
$ScheduleInterval = 60

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -Name           $ScName

# Create the snapshot (Replay) profile

$ReplayProfile = New-DellScReplayProfile -ConnectionName $ConnName `
                                         -StorageCenter  $StorageCenter `
                                         -Name           $ProfileName `
                                         -Type           $ProfileType

# Create the schedule for the snapshot profile rule

$Schedule = New-DellSchedule -ScheduleType $ScheduleType `
                             -Interval     $ScheduleInterval

# Create the snapshot (Replay) profile rule

$ExpireMinutes = $ExpireDays * 24 * 60

$ProfileRule = New-DellScReplayProfileRule -ConnectionName $ConnName `
                                           -ReplayProfile  $ReplayProfile `
                                           -Name           $RuleName `
                                           -Schedule       $Schedule `
                                           -Expiration     $ExpireMinutes `
                                           -Confirm:$false
```

3.7.2 Modify a snapshot profile

The **Set-DellScReplayProfile** cmdlet can be used to modify a snapshot profile. Standard snapshot profiles included with SC Series arrays cannot be modified. This sample script will modify the **Dev Database** snapshot profile to take snapshots in a consistency group. The name of the profile is also changed to **Dev Database Consistent**.

```
# Assign variables

$ConnName       = "DSMDC"
$ScName         = "SC 13"
$ProfileName    = "Dev Database"
$NewProfileName = "Dev Database Consistent"
$NewProfileType = [DellStorage.Api.Enums.ReplayProfileTypeEnum] "Consistent"
```

```
# Get the snapshot (Replay) profile

$ReplayProfile = Get-DellScReplayProfile -ConnectionName $ConnName `
                                         -ScName         $ScName `
                                         -Name           $ProfileName

# Change the snapshot (Replay) profile to create consistency groups

$ReplayProfile = Set-DellScReplayProfile -ConnectionName $ConnName `
                                         -Instance       $ReplayProfile `
                                         -Name           $NewProfileName `
                                         -Type           $NewProfileType
```

3.7.3 Add a snapshot profile schedule rule

The **New-DellScReplayProfileRule** cmdlet can be used to add a schedule rule to a snapshot profile. This sample script will add the **Weekly on Monday at 12:15 AM** schedule rule to the **Dev Database** snapshot profile. After adding the rule, volumes using the snapshot profile will create a snapshot every Monday at 12:15 a.m. that will expire in 7 days. This will be in addition to snapshots created by other rules in the profile.

```
# Assign variables

$ConnName           = "DSMDC"
$ScName             = "SC 13"
$ProfileName        = "Dev Database"
$RuleName           = "Weekly on Monday at 12:15 AM"
$ExpireDays         = 7
$ScheduleType       = [DellStorage.Api.Enums.ScheduleTypeEnum] "Daily"
$ScheduleDayOfWeek  = [DellStorage.Api.Enums.DayOfWeekEnum] "Monday"
$ScheduleStartTime  = [DellStorage.Api.Types.Time] "12:15 AM"

# Get the snapshot (Replay) profile

$ReplayProfile = Get-DellScReplayProfile -ConnectionName $ConnName `
                                         -ScName         $ScName `
                                         -Name           $ProfileName

# Create the schedule for the snapshot profile rule

$Schedule = New-DellSchedule -ScheduleType $ScheduleType `
                             -DayOfWeek    $ScheduleDayOfWeek `
                             -StartTime    $ScheduleStartTime

# Create the snapshot (Replay) profile rule

$ExpireMinutes = $ExpireDays * 24 * 60

$ProfileRule = New-DellScReplayProfileRule -ConnectionName $ConnName `
                                           -ReplayProfile  $ReplayProfile `
                                           -Name           $RuleName `
                                           -Schedule       $Schedule `
                                           -Expiration     $ExpireMinutes `
                                           -Confirm:$false
```

3.7.4 Modify a snapshot profile schedule rule

The **Set-DellScReplayProfileRule** cmdlet can be used to modify a snapshot profile schedule rule. This sample script will modify the **Weekly on Monday at 12:15 AM** schedule rule to take snapshots every Sunday and Monday at 4:00 a.m. The name of the rule will also be changed to **Weekly on Sunday and Monday at 4:00 AM**.

```
# Assign variables

$ConnName           = "DSMDC"
$ScName             = "SC 13"
$ProfileName        = "Dev Database"
$RuleName           = "Weekly on Monday at 12:15 AM"
$NewRuleName        = "Weekly on Sunday and Monday at 4:00 AM"
$ExpireDays         = 5
$ScheduleType       = [DellStorage.Api.Enums.ScheduleTypeEnum] "Weekly"
$ScheduleDayOfWeek  = [DellStorage.Api.Enums.DayOfWeekEnum[]] ( "Sunday", "Monday" )
$ScheduleStartTime  = [DellStorage.Api.Types.Time] "4:00 AM"

# Get the snapshot (Replay) profile

$ReplayProfile = Get-DellScReplayProfile -ConnectionName $ConnName `
                                         -ScName         $ScName `
                                         -Name           $ProfileName

# Get the snapshot (Replay) profile rule

$ProfileRule = Get-DellScReplayProfileRule -ConnectionName $ConnName `
                                           -ScName         $ScName `
                                           -ReplayProfile  $ReplayProfile `
                                           -Name           $RuleName

# Create the new schedule for the snapshot profile rule

$Schedule = New-DellSchedule -ScheduleType $ScheduleType `
                             -DayOfWeek    $ScheduleDayOfWeek `
                             -StartTime    $ScheduleStartTime

# Modify the snapshot (Replay) profile rule

$ExpireMinutes = $ExpireDays * 24 * 60

$ProfileRule = Set-DellScReplayProfileRule -ConnectionName $ConnName `
                                           -Instance       $ProfileRule `
                                           -Name           $NewRuleName `
                                           -Schedule       $Schedule `
                                           -Expiration     $ExpireMinutes `
                                           -Confirm:$false
```

3.7.5 Remove a snapshot profile schedule rule

The **Remove-DellScReplayProfileRule** cmdlet can be used to remove a snapshot profile schedule rule. This sample script will remove the **Weekly on Sunday and Monday at 4:00 AM** rule.

```
# Assign variables

$ConnName    = "DSMDC"
$ScName      = "SC 13"
$ProfileName = "Dev Database"
$RuleName    = "Weekly on Sunday and Monday at 4:00 AM"

# Get the snapshot (Replay) profile

$ReplayProfile = Get-DellScReplayProfile -ConnectionName $ConnName `
                                         -ScName         $ScName `
                                         -Name           $ProfileName

# Get the snapshot (Replay) profile rule

$ProfileRule = Get-DellScReplayProfileRule -ConnectionName $ConnName `
                                           -ReplayProfile  $ReplayProfile `
                                           -Name           $RuleName

# Remove the snapshot (Replay) profile rule

Remove-DellScReplayProfileRule -ConnectionName $ConnName `
                               -Instance       $ProfileRule `
                               -Confirm:$false
```

3.7.6 Modify snapshot profiles on a volume

The **Set-DellScVolumeConfiguration** cmdlet can be used to change the snapshot profile on a volume. This sample script shows several examples of how to change the snapshot profile on the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder.

```
# Assign variables

$ConnName         = "DSMDC"
$ScName           = "SC 13"
$VolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$VolumeName       = "SQLData"

# Get the volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -Name             $VolumeName `
                           -VolumeFolderPath $VolumeFolderPath

# Get the volume configuration

$VolumeConfig = Get-DellScVolumeConfiguration -ConnectionName $ConnName `
                                              -Instance       $Volume.InstanceId
```

```
# Add the "Hourly" snapshot (Replay) profile

$ProfileName  = "Hourly"

$ProfileList  = $VolumeConfig.ReplayProfileList
$ProfileList += Get-DellScReplayProfile -ConnectionName $ConnName `
                                        -ScName         $ScName `
                                        -Name           $ProfileName

$VolumeConfig = Set-DellScVolumeConfiguration -ConnectionName    $ConnName `
                                              -Instance          $VolumeConfig `
                                              -ReplayProfileList $ProfileList

# Remove the "Hourly" snapshot (Replay) profile

$ProfileName = "Hourly"

$ProfileList = @( $VolumeConfig.ReplayProfileList | `
                    Where-Object { $_.InstanceName -ne $ProfileName } )
$VolumeConfig = Set-DellScVolumeConfiguration -ConnectionName    $ConnName `
                                              -Instance          $VolumeConfig `
                                              -ReplayProfileList $ProfileList

# Replace existing snapshot (Replay) profiles with the "Sample" snapshot profile

$ProfileName = "Sample"

$ProfileList = @( Get-DellScReplayProfile -ConnectionName $ConnName `
                                          -ScName         $ScName `
                                          -Name           $ProfileName )

$VolumeConfig = Set-DellScVolumeConfiguration -ConnectionName    $ConnName `
                                              -Instance          $VolumeConfig `
                                              -ReplayProfileList $ProfileList

# Remove all snapshot (Replay) profiles

$VolumeConfig = Set-DellScVolumeConfiguration -ConnectionName    $ConnName `
                                              -Instance          $VolumeConfig `
                                              -ReplayProfileList $null
```

3.7.7 Remove a snapshot profile

The **Remove-DellScReplayProfile** cmdlet can be used to remove a snapshot profile. This sample script will remove the **Dev Database** snapshot profile.

```
# Assign variables

$ConnName    = "DSMDC"
$ScName      = "SC 13"
$ProfileName = "Dev Database"

# Get the snapshot (Replay) profile

$ReplayProfile = Get-DellScReplayProfile -ConnectionName $ConnName `
                                         -ScName         $ScName `
                                         -Name           $ProfileName

# Remove the snapshot (Replay) profile

Remove-DellScReplayProfile -ConnectionName $ConnName `
                           -Instance       $ReplayProfile `
                           -Confirm:$false
```

3.8 Recycle bin

The Dell Storage PowerShell SDK provides cmdlets to manage the recycle bin on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 8 shows the recycle bin cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 8 Recycle bin cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCRecycleBinVolume	Get-DellScVolume -InRecycleBin:$true
Remove-SCRecycleBinVolume	Remove-DellScVolume
Restore-SCRecycleBinVolume	Restore-DellScVolume

3.8.1 Recover a volume from the recycle bin

The **Restore-DellScVolume** cmdlet can be used to recover a volume from the recycle bin. This sample script will recover the **SQLData** volume from the recycle bin. The recovered volume will be placed in the volume folder from which it was deleted.

```
# Assign variables

$ConnName    = "DSMDC"
$ScName      = "SC 13"
$VolumeName = "SQLData"

# Get the volume

$RecycleBinVolume = Get-DellScVolume -ConnectionName $ConnName `
                                     -ScName         $ScName `
                                     -Name           $VolumeName `
                                     -InRecycleBin:$true

# Restore the volume from the recycle bin

Restore-DellScVolume -ConnectionName $ConnName `
                     -Instance       $RecycleBinVolume `
                     -Confirm:$false
```

3.8.2 Remove a volume from the recycle bin

The **Remove-DellScVolume** cmdlet can be used to remove a volume from the recycle bin. This sample script will permanently delete the **SQLData** volume by removing it from the recycle bin.

```
# Assign variables

$ConnName    = "DSMDC"
$ScName      = "SC 13"
$VolumeName = "SQLData"

# Get the volume from the recycle bin

$RecycleBinVolume = Get-DellScVolume -ConnectionName $ConnName `
                                     -ScName         $ScName `
                                     -Name           $VolumeName `
                                     -InRecycleBin:$true
```

```
# Remove the volume from the recycle bin

Remove-DellScVolume -ConnectionName $ConnName `
                    -Instance       $RecycleBinVolume `
                    -Confirm:$false
```

3.9 Copy/Mirror/Migrate

The Dell Storage PowerShell SDK provides cmdlets to manage Copy/Mirror/Migrate (CMM) operations on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 9 shows the CMM cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 9 CMM cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCCmm	Get-DellScCopyMirrorMigrate
New-SCCmmCopy	New-DellScCopyMirrorMigrateViaCopy
New-SCCmmMigrate	New-DellScCopyMirrorMigrateViaMigrate
New-SCCmmMirror	New-DellScCopyMirrorMigrateViaMirror
Pause-SCCmm	Suspend-DellScCopyMirrorMigrate
Remove-SCCmm	Remove-DellScCopyMirrorMigrate
Resume-SCCmm	Resume-DellScCopyMirrorMigrate
Set-SCCmm	Set-DellScCopyMirrorMigrate

3.9.1 Copy a volume using CMM

The **New-DellScCopyMirrorMigrateViaCopy** cmdlet can be used to copy a volume using CMM. This sample script will copy the **SQLData02** volume in the **Production/SQLDatabase/SQLProd** volume folder on Storage Center **SC 2**. The volume copy will be named **Copy of SQLData02** and placed in the **Production/SQLDatabase/SQLDev** volume folder. All data, including snapshots, will be copied.

```
# Assign variables

$ConnName             = "DSMDC"
$ScName               = "SC 2"
$SrcVolumeName        = "SQLData02"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstVolumeNamePrefix  = "Copy of "
$DstVolumeFolderPath  = "Production/SQLDatabase/"
$DstVolumeFolderName  = "SQLDev"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -ScName         $ScName
```

```
# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $StorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination volume folder

$DstVolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                          -StorageCenter  $StorageCenter `
                                          -FolderPath     $DstVolumeFolderPath `
                                          -Name           $DstVolumeFolderName

# Create the destination volume with the same size as the source volume

$DstVolumeName = $DstVolumeNamePrefix + $SrcVolume.Name

$DstVolume = New-DellScVolume -ConnectionName $ConnName `
                              -StorageCenter  $StorageCenter `
                              -Name           $DstVolumeName `
                              -Size           $SrcVolume.ConfiguredSize `
                              -VolumeFolder   $DstVolumeFolder

# Start the CMM copy

$CMM = New-DellScCopyMirrorMigrateViaCopy -ConnectionName    $ConnName `
                                          -StorageCenter     $StorageCenter `
                                          -DestinationVolume $DstVolume `
                                          -SourceVolume      $SrcVolume `
                                          -CopyReplays:$true
```

3.9.2 Migrate a volume using CMM

The **New-DellScCopyMirrorMigrateViaMigrate** cmdlet can be used to migrate a volume using CMM. This sample script will migrate the **SQLData02** volume in the **Production/SQLDatabase/SQLProd** volume folder on Storage Center **SC 2** to the disk pool **DiskPool2**. The migrated volume will be named **Copy of SQLData02** and placed in the same volume folder as the original volume. All data, including snapshots, will be migrated.

```
# Assign variables

$ConnName            = "DSMDC"
$ScName              = "SC 2"
$SrcVolumeName       = "SQLData02"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstVolumeNamePrefix = "Copy of "
$DstStorageTypeName  = "DiskPool2 - Redundant - 2 MB"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -ScName         $ScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $StorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName
```

```
# Get the destination storage type

$DstStorageType = Get-DellScStorageType -ConnectionName $ConnName `
                                        -StorageCenter  $StorageCenter `
                                        -Name           $DstStorageTypeName
# Create the destination volume

$DstVolumeName = $DstVolumeNamePrefix + $SrcVolume.Name

$DstVolume = New-DellScVolume -ConnectionName $ConnName `
                              -StorageCenter  $StorageCenter `
                              -Name           $DstVolumeName `
                              -Size           $SrcVolume.ConfiguredSize `
                              -StorageType    $DstStorageType
                              -VolumeFolder   $SrcVolume.VolumeFolder

# Start the CMM migration

$CMM = New-DellScCopyMirrorMigrateViaMigrate -ConnectionName    $ConnName `
                                             -StorageCenter     $StorageCenter `
                                             -DestinationVolume $DstVolume `
                                             -SourceVolume      $SrcVolume `
                                             -DeleteSource:$true `
                                             -CopyReplays:$true
```

3.9.3 Mirror a volume using CMM

The **New-DellScCopyMirrorMigrateViaMirror** cmdlet can be used to mirror a volume using CMM. This sample script will mirror the **SQLData02** volume in the **Production/SQLDatabase/SQLProd** volume folder on Storage Center **SC 2** to volume in the disk pool **DiskPool2**. The mirrored volume will be named **Mirror of SQLData02** and placed in the same volume folder as the original volume. All data, including snapshots, will be mirrored.

```
# Assign variables

$ConnName            = "DSMDC"
$ScName              = "SC 2"
$SrcVolumeName       = "SQLData02"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstVolumeNamePrefix = "Mirror of "
$DstStorageTypeName  = "DiskPool2 - Redundant - 2 MB"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -ScName         $ScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $StorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination storage type

$DstStorageType = Get-DellScStorageType -ConnectionName $ConnName `
                                        -StorageCenter  $StorageCenter `
                                        -Name           $DstStorageTypeName
```

```
# Create the destination volume

$DstVolumeName = $DstVolumeNamePrefix + $SrcVolume.Name

$DstVolume = New-DellScVolume -ConnectionName $ConnName `
                              -StorageCenter  $StorageCenter `
                              -Name           $DstVolumeName `
                              -Size           $SrcVolume.ConfiguredSize `
                              -StorageType    $DstStorageType
                              -VolumeFolder   $SrcVolume.VolumeFolder

# Start the CMM migrate

$CMM = New-DellScCopyMirrorMigrateViaMirror -ConnectionName    $ConnName `
                                            -StorageCenter     $StorageCenter `
                                            -DestinationVolume $DstVolume `
                                            -SourceVolume      $SrcVolume `
                                            -CopyReplays:$true
```

3.9.4 Change the priority of a CMM

The **Set-DellScCopyMirrorMigrate** cmdlet can be used to change the priority of a CMM. This sample script will show how to change the priority for the CMM mirror between the **SQLData02** and **Mirror of SQLData02** volumes in the **Production/SQLDatabase/SQLProd** volume folder on Storage Center **SC 2**.

```
# Assign variables

$ConnName            = "DSMDC"
$ScName              = "SC 2"
$SrcVolumeName       = "SQLData02"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstVolumeName       = "Mirror of SQLData02"
$DstVolumeFolderPath = "Production/SQLDatabase/SQLProd/"

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $ScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination volume

$DstVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $ScName `
                              -VolumeFolderPath $DstVolumeFolderPath `
                              -Name             $DstVolumeName

# Get the CMM

$CMM = Get-DellScCopyMirrorMigrate -ConnectionName    $ConnName `
                                   -ScName            $ScName `
                                   -SourceVolume      $SrcVolume `
                                   -DestinationVolume $DstVolume

# Change the priority to Low

$NewPriority = [DellStorage.Api.Enums.CmmPriorityEnum]"Low"

$CMM = Set-DellScCopyMirrorMigrate -ConnectionName $ConnName `
                                   -Instance       $CMM `
                                   -Priority       $NewPriority
```

```
# Change the priority to Medium

$NewPriority = [DellStorage.Api.Enums.CmmPriorityEnum]"Medium"

$CMM = Set-DellScCopyMirrorMigrate -ConnectionName $ConnName `
                                   -Instance       $CMM `
                                   -Priority       $NewPriority

# Change the priority to High

$NewPriority = [DellStorage.Api.Enums.CmmPriorityEnum]"High"

$CMM = Set-DellScCopyMirrorMigrate -ConnectionName $ConnName `
                                   -Instance       $CMM `
                                   -Priority       $NewPriority
```

3.9.5 Pause a CMM

The **Suspend-DellScCopyMirrorMigrate** cmdlet can be used to pause a CMM. This sample script will pause the CMM mirror between the **SQLData02** and **Mirror of SQLData02** volumes in the **Production/SQLDatabase/SQLProd** volume folder on Storage Center **SC 2**.

```
# Assign variables

$ConnName             = "DSMDC"
$ScName               = "SC 2"
$SrcVolumeName        = "SQLData02"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstVolumeName        = "Mirror of SQLData02"
$DstVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $ScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination volume

$DstVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $ScName `
                              -VolumeFolderPath $DstVolumeFolderPath `
                              -Name             $DstVolumeName

# Get the CMM

$CMM = Get-DellScCopyMirrorMigrate -ConnectionName    $ConnName `
                                   -ScName            $ScName `
                                   -SourceVolume      $SrcVolume `
                                   -DestinationVolume $DstVolume

# Pause the CMM

If ( $CMM.State -ne "Paused" )
{
    Suspend-DellScCopyMirrorMigrate -ConnectionName $ConnName `
                                    -Instance       $CMM `
                                    -Confirm:$false
}
```

3.9.6 Resume a paused CMM

The **Resume-DellScCopyMirrorMigrate** cmdlet can be used to resume a CMM that has been paused. This sample script will resume the CMM mirror between the **SQLData02** and **Mirror of SQLData02** volumes in the **Production/SQLDatabase/SQLProd** volume folder on Storage Center **SC 2**.

```
# Assign variables

$ConnName             = "DSMDC"
$ScName               = "SC 2"
$SrcVolumeName        = "SQLData02"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstVolumeName        = "Mirror of SQLData02"
$DstVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $ScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination volume

$DstVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $ScName `
                              -VolumeFolderPath $DstVolumeFolderPath `
                              -Name             $DstVolumeName

# Get the CMM

$CMM = Get-DellScCopyMirrorMigrate -ConnectionName    $ConnName `
                                   -ScName            $ScName `
                                   -SourceVolume      $SrcVolume `
                                   -DestinationVolume $DstVolume

# Resume the paused CMM

If ( $CMM.State -eq "Paused" )
{
    Resume-DellScCopyMirrorMigrate -ConnectionName $ConnName `
                                   -Instance       $CMM `
                                   -Confirm:$false
}
```

3.9.7 Remove a CMM

The **Remove-DellScCopyMirrorMigrate** cmdlet can be used to remove a CMM. This sample script will remove the CMM mirror between the **SQLData02** and **Mirror of SQLData02** volumes in the **Production/SQLDatabase/SQLProd** volume folder on Storage Center **SC 2**.

```
# Assign variables

$ConnName             = "DSMDC"
$ScName               = "SC 2"
$SrcVolumeName        = "SQLData02"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstVolumeName        = "Mirror of SQLData02"
$DstVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
```

```
# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $ScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination volume

$DstVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $ScName `
                              -VolumeFolderPath $DstVolumeFolderPath `
                              -Name             $DstVolumeName

# Get the CMM

$CMM = Get-DellScCopyMirrorMigrate -ConnectionName    $ConnName `
                                   -ScName            $ScName `
                                   -SourceVolume      $SrcVolume `
                                   -DestinationVolume $DstVolume
# Remove the CMM

Remove-DellScCopyMirrorMigrate -ConnectionName $ConnName `
                               -Instance       $CMM `
                               -Confirm:$false
```

3.10 Volume replication

The Dell Storage PowerShell SDK provides cmdlets to manage volume replication on SC Series arrays. A subset of this functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 10 shows the user cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 10 Replication cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCAsyncReplication	Get-DellScReplication
New-SCAsyncReplication	New-DellScReplication
Pause-SCAsyncReplication	Suspend-DellScReplication
Remove-SCAsyncReplication	Remove-DellScReplication
Resume-SCAsyncReplication	Resume-DellScReplication
Set-SCAsyncReplication	Set-DellScReplication

3.10.1 Create an asynchronous replication

The **New-DellScReplication** cmdlet can be used to create a new asynchronous volume replication. This sample script will create a new asynchronous replication between Storage Centers **SC 12** and **SC 13** for the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName                   = "DSMDC"
$SrcScName                  = "SC 12"
$SrcVolumeName              = "SQLData"
$SrcVolumeFolderPath        = "Production/SQLDatabase/SQLProd/"
$TransportTypes             = [DellStorage.Api.Enums.FrontEndTransportTypeEnum[]] `
                              ( "FibreChannel", "Iscsi" )
$ReplType                   = [DellStorage.Api.Enums.ReplicationTypeEnum] "Asynchronous"
$QosNodeName                = "4Gbps_SC12"
$DstScName                  = "SC 13"
$DstVolumeNamePrefix        = "Repl of "
$DstDataReductionProfileName = "None"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName

# Get the QoS Node

$QosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                        -ScName         $SrcScName `
                                        -Name           $QosNodeName

# Get the data reduction profile for the destination volume

$DstDataReductionProfile = Get-DellScDataReductionProfile -ConnectionName "DSMDC" `
                                                   -StorageCenter $DstStorageCenter `
                                                   -InstanceName  $DstDataReductionProfileName

# Define the destination volume attributes

$DstVolumeName = $DstVolumeNamePrefix + $SrcVolumeName

$DstVolumeAttrib = New-DellScVolumeCreateAttributes -Name      $DstVolumeName `
                                        -DataReductionProfile $DstDataReductionProfile `
                                        -ReadCache:$true `
                                        -WriteCache:$true `
                                        -CreateSourceVolumeFolderPath:$true
```

```
# Create the replication

$Replication = New-DellScReplication -ConnectionName             $ConnName `
                                     -StorageCenter              $SrcStorageCenter `
                                     -SourceVolume               $SrcVolume `
                                     -TransportTypes             $TransportTypes `
                                     -Type                       $ReplType `
                                     -QosNode                    $QosNode `
                                     -DestinationStorageCenter   $DstStorageCenter `
                                     -DestinationVolumeAttributes $DstVolumeAttrib `
                                     -ReplicateStorageToLowestTier:$true `
                                     -ReplicateActiveReplay:$false `
                                     -Dedup:$false
```

3.10.2 Create a synchronous replication

The **New-DellScReplication** cmdlet can be used to create a new synchronous volume replication. This sample script will create a new synchronous replication in high availability mode between Storage Centers **SC 12** and **SC 13** for the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLData"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$TransportTypes       = [DellStorage.Api.Enums.FrontEndTransportTypeEnum[]] "FibreChannel"
$ReplType             = [DellStorage.Api.Enums.ReplicationTypeEnum] "Synchronous"
$SyncMode             = [DellStorage.Api.Enums.SyncReplicationModeEnum] "HighAvailability"
$QosNodeName          = "4Gbps_SC12"
$DstScName            = "SC 13"
$DstVolumeNamePrefix  = "Repl of "
$DstDataReductionProfileName = "None"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName

# Get the QoS Node

$QosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                        -ScName         $SrcScName `
                                        -Name           $QosNodeName

# Get the data reduction profile for the destination volume

$DstDataReductionProfile = Get-DellScDataReductionProfile -ConnectionName "DSMDC" `
                                                          -StorageCenter $DstStorageCenter `
                                                          -InstanceName  $DstDataReductionProfileName
```

```
# Define the destination volume attributes

$DstVolumeName = $DstVolumeNamePrefix + $SrcVolumeName

$DstVolumeAttrib = New-DellScVolumeCreateAttributes -Name      $DstVolumeName `
                                                    -DataReductionProfile $DstDataReductionProfile `
                                                    -ReadCache:$true `
                                                    -WriteCache:$true `
                                                    -CreateSourceVolumeFolderPath:$true

# Create the replication

$Replication = New-DellScReplication -ConnectionName              $ConnName `
                                     -StorageCenter               $SrcStorageCenter `
                                     -SourceVolume                $SrcVolume `
                                     -TransportTypes              $TransportTypes `
                                     -Type                        $ReplType `
                                     -SyncMode                    $SyncMode `
                                     -QosNode                     $QosNode `
                                     -DestinationStorageCenter    $DstStorageCenter `
                                     -DestinationVolumeAttributes $DstVolumeAttrib `
                                     -ReplicateStorageToLowestTier:$false `
                                     -ReplicateActiveReplay:$true `
                                     -Dedup:$false
```

3.10.3 Modify the replication type

The **Set-DellScReplication** cmdlet can be used to modify the replication type. This sample script will modify the replication between Storage Centers **SC 12** and **SC 13** for the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder on Storage Center **SC 12**. This sample script shows how to change the replication type to asynchronous, synchronous in high-availability mode, and synchronous in high-consistency mode.

```
# Assign variables

$ConnName            = "DSMDC"
$SrcScName           = "SC 12"
$SrcVolumeName       = "SQLData"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstScName           = "SC 13"

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $SrcScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName

# Get the replication

$Replication = Get-DellScReplication -ConnectionName           $ConnName `
                                     -SourceVolume             $SrcVolume `
                                     -DestinationStorageCenter $DstStorageCenter
```

```
# Change the replication type to asynchronous

$NewReplType = [DellStorage.Api.Enums.ReplicationTypeEnum] "Asynchronous"

$Replication = Set-DellScReplication -ConnectionName $ConnName `
                                     -Instance       $Replication `
                                     -Type           $NewReplType

# Change the replication type to synchronous, high-availability mode

$NewReplType = [DellStorage.Api.Enums.ReplicationTypeEnum] "Synchronous"
$NewSyncMode = [DellStorage.Api.Enums.SyncReplicationModeEnum] "HighAvailability"

$Replication = Set-DellScReplication -ConnectionName $ConnName `
                                     -Instance       $Replication `
                                     -Type           $NewReplType `
                                     -SyncMode       $NewSyncMode `
                                     -ReplicateStorageToLowestTier:$false `
                                     -ReplicateActiveReplay:$true

# Change the replication type to synchronous, high-consistency mode

$NewReplType = [DellStorage.Api.Enums.ReplicationTypeEnum] "Synchronous"
$NewSyncMode = [DellStorage.Api.Enums.SyncReplicationModeEnum] "HighConsistency"

$Replication = Set-DellScReplication -ConnectionName $ConnName `
                                     -Instance       $Replication `
                                     -Type           $NewReplType `
                                     -SyncMode       $NewSyncMode `
                                     -ReplicateStorageToLowestTier:$false `
                                     -ReplicateActiveReplay:$true
```

3.10.4 Modify replication attributes

The **Set-DellScReplication** cmdlet can be used to modify replication attributes. This sample script will modify the replication between Storage Centers **SC 12** and **SC 13** for the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder on Storage Center **SC 12**. This sample script shows how to change the QoS node as well as how to change the settings for replicating the active snapshot (Replay), deduplication, and replicating to the lowest storage tier.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLData"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstScName            = "SC 13"

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $SrcScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName
```

```
# Get the replication

$Replication = Get-DellScReplication -ConnectionName           $ConnName `
                                     -SourceVolume             $SrcVolume `
                                     -DestinationStorageCenter $DstStorageCenter

# Change the QoS Node to "QOS 1Gb"

$NewQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $SrcScName `
                                           -Name           "QOS 1Gb"

$Replication = Set-DellScReplication -ConnectionName $ConnName `
                                     -Instance       $Replication `
                                     -QosNode        $NewQosNode

# Enable "Replicate Active Snapshot"

$Replication = Set-DellScReplication -ConnectionName $ConnName `
                                     -Instance       $Replication `
                                     -ReplicateActiveReplay:$true

# Enable "Deduplication"

$Replication = Set-DellScReplication -ConnectionName $ConnName `
                                     -Instance       $Replication `
                                     -Dedup:$true

# Enable "Replicate Storage to Lowest Tier"

$Replication = Set-DellScReplication -ConnectionName $ConnName `
                                     -Instance       $Replication `
                                     -ReplicateStorageToLowestTier:$true
```

3.10.5 Pause a replication

The **Suspend-DellScReplication** cmdlet can be used to pause a volume replication. This sample script will pause the replication between Storage Centers **SC 12** and **SC 13** for the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName            = "DSMDC"
$SrcScName           = "SC 12"
$SrcVolumeName       = "SQLData"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstScName           = "SC 13"

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $SrcScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName
```

```
# Get the replication

$Replication = Get-DellScReplication -ConnectionName           $ConnName `
                                     -SourceVolume             $SrcVolume `
                                     -DestinationStorageCenter $DstStorageCenter

# Pause the replication

If ( $Replication.PauseAllowed )
{
    Suspend-DellScReplication -ConnectionName $ConnName `
                              -Instance       $Replication `
                              -Confirm:$false
}
```

3.10.6 Resume a paused replication

The **Resume-DellScReplication** cmdlet can be used to resume a volume replication that has been paused. This sample script will resume the replication between Storage Centers **SC 12** and **SC 13** for the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLData"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstScName            = "SC 13"

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $SrcScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName

# Get the replication

$Replication = Get-DellScReplication -ConnectionName           $ConnName `
                                     -SourceVolume             $SrcVolume `
                                     -DestinationStorageCenter $DstStorageCenter

# Resume the replication

If ( $Replication.State.Name -eq "Paused" )
{
    Resume-DellScReplication -ConnectionName $ConnName `
                             -Instance       $Replication
}
```

3.10.7 Get summary information for a replication

The **Get-DellScReplication** and **Get-DellScReplicationProgress** cmdlets can be used to get summary information for a volume replication. This sample script will display information for the replication between Storage Centers **SC 12** and **SC 13** for the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLData"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstScName            = "SC 13"

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $SrcScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName

# Get the replication

$Replication = Get-DellScReplication -ConnectionName           $ConnName `
                                     -SourceVolume             $SrcVolume `
                                     -DestinationStorageCenter $DstStorageCenter

# Display the source and target volumes

$Replication `
| Select-Object @{ Name       = "SourceStorageCenter"; `
                   Expression = { $_.SourceStorageCenter.InstanceName } }, `
                @{ Name       = "SourceVolume"; `
                   Expression = { $_.SourceVolume.InstanceName } }, `
                @{ Name       = "DestinationStorageCenter"; `
                   Expression = { $_.DestinationStorageCenter.InstanceName } }, `
                @{ Name       = "DestinationVolume"; `
                   Expression = { $_.DestinationVolume.InstanceName } } `
| Format-List

# Display replication summary

$Replication `
| Select-Object State, `
                @{ Name = "QosNode"; Expression = { $_.QosNode.InstanceName } }, `
                Type, `
                TransportTypes, `
                ManagedByLiveVolume, `
                Simulation, `
                @{ Name = "Deduplication"; Expression = { $_.Dedup } }, `
                ReplicateActiveReplay, `
                ReplicateStorageToLowestTier `
| Format-List

# Get the replication progress

$ReplicationProgress = Get-DellScReplicationProgress -ConnectionName $ConnName `
                                                     -Replication    $Replication
```

```
# Display replication progress

$ReplicationProgress `
| Select-Object PercentComplete, `
                AmountRemaining, `
                AsyncBehind `
| Format-List
```

3.10.8 Remove a replication

The **Remove-DellScReplication** cmdlet can be used to remove a volume replication. This sample script will remove the replication between Storage Centers **SC 12** and **SC 13** for the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder on Storage Center **SC 12.** This sample script will also remove the destination volume on Storage Center **SC 13**.

```
# Assign variables

$ConnName            = "DSMDC"
$SrcScName           = "SC 12"
$SrcVolumeName       = "SQLData"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstScName           = "SC 13"

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -ScName           $SrcScName `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName

# Get the replication

$Replication = Get-DellScReplication -ConnectionName           $ConnName `
                                     -SourceVolume             $SrcVolume `
                                     -DestinationStorageCenter $DstStorageCenter

# Remove the replication

Remove-DellScReplication -ConnectionName $ConnName `
                         -Instance       $Replication `
                         -RecycleDestinationVolume:$true `
                         -DeleteDestinationVolume:$true `
                         -DeleteRestorePoint:$true `
                         -Confirm:$false
```

3.11 Replication QoS

The Dell Storage PowerShell SDK provides cmdlets to manage replication QoS nodes on SC Series arrays. A subset of this functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 11 shows the user cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 11 Replication QoS node cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCReplicationQos	Get-DellScReplicationQosNode
New-SCReplicationQos	New-DellScReplicationQosNode New-DellScReplicationQosAdvancedSettings
Remove-SCReplicationQos	Remove-DellScReplicationQosNode
Set-SCReplicationQos	Set-DellScReplicationQosNode

3.11.1 Create a replication QoS node

The **New-DellScReplicationQosNode** cmdlet can be used to create a replication QoS node. This sample script will create a QoS node named **8Gbps QoS** on Storage Center **SC 12**, with a link speed of 8 Gbps that is not bandwidth limited.

```
# Assign variables

$ConnName = "DSMDC"
$ScName   = "SC 12"
$QosName  = "8Gbps QoS"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -ScName         $ScName

# Define the link speed (8Gbps) in Kbps

$LinkSpeedKbps = [DellStorage.Api.Types.NetworkLinkSpeed] ( 8GB / 1KB )

# Create the QoSNode

$QosNode = New-DellScReplicationQosNode -ConnectionName $ConnName `
                                        -Name           $QosName `
                                        -StorageCenter  $StorageCenter `
                                        -LinkSpeed      $LinkSpeedKbps `
                                        -BandwidthLimited:$false
```

3.11.2 Modify a replication QoS node

The **Set-DellScReplicationQosNode** cmdlet can be used to modify a replication QoS node. This sample script will modify the QoS node **8Gbps QoS** on Storage Center **SC 12**. This sample script shows how to change the QoS node name, link speed and limit bandwidth setting.

```
# Assign variables

$ConnName     = "DSMDC"
$ScName       = "SC 12"
$QosNodeName = "8Gbps QoS"

# Get the QoS Node

$QosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                        -ScName         $ScName `
                                        -Name           $QosNodeName

# Change the link speed to 4 Gbps and change the name

$NewQosNodeName   = "4Gbps QoS"
$NewLinkSpeedKbps = [DellStorage.Api.Types.NetworkLinkSpeed] ( 4GB / 1KB )

$QosNode = Set-DellScReplicationQosNode -ConnectionName $ConnName `
                                        -Instance       $QosNode `
                                        -LinkSpeed      $NewLinkSpeedKbps `
                                        -Name           $NewQosNodeName

# Set the QoSNode to limit bandwidth

$QosNode = Set-DellScReplicationQosNode -ConnectionName $ConnName `
                                        -Instance       $QosNode `
                                        -BandwidthLimited:$true

# Set the QoSNode to not limit bandwidth

$QosNode = Set-DellScReplicationQosNode -ConnectionName $ConnName `
                                        -Instance       $QosNode `
                                        -BandwidthLimited:$false
```

3.11.3 Modify a QoS node schedule

The **Set-DellScReplicationQosNodeBandwidthLimitingSchedule** cmdlet can be used to modify a replication QoS node schedule. This sample script will modify the schedule for the QoS node **4Gbps QoS** on Storage Center **SC 12**. After running the script, the QoS node will limit bandwidth to 30% on Monday, Tuesday, and Wednesday between 3:00 p.m. and 6:59 p.m.

```
# Assign variables

$ConnName           = "DSMDC"
$ScName             = "SC 12"
$QosNodeName        = "4Gbps_SC13"
$DaysToChange       = @( "Monday", "Tuesday", "Wednesday" )
$HoursToChange      = @( 15, 16, 17, 18 )
$NewBandwidthLimit = [DellStorage.Api.Enums.QosBandwidthLimitEnum] "Percent30"

# Get the QoS Node

$QosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                        -ScName         $ScName `
                                        -Name           $QosNodeName
```

```
# Get the bandwidth limiting schedule for the QoS node

$QosNodeSchedule = @( Get-DellScReplicationQosNodeScheduleListAssociation `
                                                    -ConnectionName $ConnName `
                                                    -Instance       $QosNode )

# If an entry for the new bandwidth limit doesn't exist in the schedule, add it

$NewSchedule = $QosNodeSchedule `
               | Where-Object { $_.BandwidthLimit -eq $NewBandwidthLimit }

If ( !( $NewSchedule ) )
{
    $NewSchedule = New-DellScReplicationQosBandwidthLimitSchedule `
                                                        -BandwidthLimit $NewBandwidthLimit

    $QosNodeSchedule += $NewSchedule
}

# Change the schedule for each hour

ForEach( $Hour in $HoursToChange )
{
    # Get the HourOfDay enumeration

    $HourOfDay = [DellStorage.Api.Enums.HourOfDayEnum] [math]::Pow( 2, $Hour )

    # Loop through each day and move the hour to the new bandwidth limit schedule

    ForEach( $Day in $DaysToChange )
    {
        # Get the name of the attribute for the day's schedule

        $ScheduleName = $Day + "Schedule"

        # Get the schedule that currently contains the limit for the hour and day

        $OldSchedule = $QosNodeSchedule `
                       | Where-Object { $_.$ScheduleName -contains $HourOfDay }

        # Remove the hour from it's existing bandwidth limit

        $OldSchedule.$ScheduleName = $OldSchedule.$ScheduleName -ne $HourOfDay

        # Add the hour to the new bandwidth limit

        $NewSchedule.$ScheduleName += $HourOfDay
    }
}

# Modify the schedule in the QoS node

Set-DellScReplicationQosNodeBandwidthLimitingSchedule -ConnectionName  $ConnName `
                                                      -Instance        $QosNode `
                                                      -BandwidthLimitSchedules $QosNodeSchedule `
                                                      -Confirm:$false
```

3.11.4 Remove a replication QoS node

The **Remove-DellScReplicationQosNode** cmdlet can be used to remove a replication QoS node. This sample script will remove a QoS node named **4Gbps QoS** on Storage Center **SC 12**.

```
# Assign variables

$ConnName     = "DSMDC"
$ScName       = "SC 12"
$QosNodeName = "4Gbps QoS"

# Get the QoS node

$QosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                        -ScName         $ScName `
                                        -Name           $QosNodeName

# Remove the QoS node

Remove-DellScReplicationQosNode -ConnectionName $ConnName `
                                -Instance       $QosNode `
                                -Confirm:$false
```

3.12 Live Volume

The Dell Storage PowerShell SDK provides cmdlets to manage Live Volume on SC Series arrays. Some of this functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 12 shows the user cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 12 Live Volume cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
CancelSwapRole-SCLiveVolume	Stop-DellScLiveVolumeSwapRole
ConvertTo-SCAsyncReplication	Start-DellScLiveVolumeToReplicationRevert
ConvertTo-SCLiveVolume	Convert-DellScReplicationToLiveVolume
Get-SCLiveVolume	Get-DellScLiveVolume
New-SCLiveVolume	New-DellScLiveVolume
Remove-SCLiveVolume	Remove-DellScLiveVolume
Set-SCLiveVolume	Set-DellScLiveVolume
SwapRole-SCLiveVolume	Start-DellScLiveVolumeSwapRole

3.12.1 Create an asynchronous Live Volume

The **New-DellScLiveVolume** cmdlet can be used to create a new asynchronous Live Volume. This sample script will create a new asynchronous Live Volume for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**. The primary volume will be on Storage Center **SC 12** and the secondary volume will be on Storage Center **SC 13**.

```
# Assign variables

$ConnName             = "DSMDC"
$PriScName            = "SC 12"
$PriVolumeName        = "Backup"
$PriVolumeFolderPath  = "Production/FileServer/SalesFS01/"
$PriQosNodeName       = "4Gbps_SC12"
$TransportTypes       = [DellStorage.Api.Enums.FrontEndTransportTypeEnum[]] "FibreChannel"
$ReplType             = [DellStorage.Api.Enums.ReplicationTypeEnum] "Asynchronous"
$SecScName            = "SC 13"
$SecVolumeNamePrefix  = "LV of "
$SecQosNodeName       = "4Gbps_SC13"
$SecServerName        = "SalesFS01"
$SecServerFolderPath  = "Production/FileServer/"
$SecDataReductionProfileName = "None"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the primary QoS Node

$PriQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $PriScName `
                                           -Name           $PriQosNodeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the secondary QoS Node

$SecQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $SecScName `
                                           -Name           $SecQosNodeName

# Get the data reduction profile for the secondary volume

$SecDataReductionProfile = Get-DellScDataReductionProfile -ConnectionName "DSMDC" `
                                                          -StorageCenter $SecStorageCenter `
                                                          -InstanceName  $SecDataReductionProfileName
```

```powershell
# Define the secondary volume attributes

$SecVolumeName = $SecVolumeNamePrefix + $PriVolumeName

$SecVolumeAttrib = New-DellScVolumeCreateAttributes -Name      $SecVolumeName `
                                                    -DataReductionProfile $SecDataReductionProfile `
                                                    -ReadCache:$true `
                                                    -WriteCache:$true `
                                                    -CreateSourceVolumeFolderPath:$true

# Get the server on the secondary Storage Center

$SecServer = Get-DellScServer -ConnectionName   $ConnName `
                              -StorageCenter    $SecStorageCenter `
                              -Name             $SecServerName `
                              -ServerFolderPath $SecServerFolderPath

# Define the advanced mapping attributes for the secondary Storage Center

$SecAdvMapping = New-DellScAdvancedMapping -UseSameLunAsSource:$true

# Create the Live Volume

$LiveVolume = New-DellScLiveVolume -ConnectionName            $ConnName `
                                   -StorageCenter             $PriStorageCenter `
                                   -PrimaryVolume             $PriVolume `
                                   -TransportTypes            $TransportTypes `
                                   -Type                      $ReplType `
                                   -PrimaryQosNode            $PriQosNode `
                                   -SecondaryStorageCenter    $SecStorageCenter `
                                   -SecondaryVolumeAttributes $SecVolumeAttrib `
                                   -SecondaryQosNode          $SecQosNode `
                                   -SecondaryServer           $SecServer `
                                   -SecondaryAdvancedMapping  $SecAdvMapping `
                                   -SwapRolesAutomaticallyEnabled:$true `
                                   -Dedup:$false
```

3.12.2 Create a synchronous Live Volume

The **New-DellScLiveVolume** cmdlet can be used to create a new synchronous Live Volume. This sample script will create a new synchronous Live Volume in high availability mode for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**. The primary volume will be on Storage Center **SC 12** and the secondary volume will be on Storage Center **SC 13**.

```
# Assign variables

$ConnName             = "DSMDC"
$PriScName            = "SC 12"
$PriVolumeName        = "Backup"
$PriVolumeFolderPath  = "Production/FileServer/SalesFS01/"
$PriQosNodeName       = "4Gbps_SC12"
$TransportTypes       = [DellStorage.Api.Enums.FrontEndTransportTypeEnum[]] "FibreChannel"
$ReplType             = [DellStorage.Api.Enums.ReplicationTypeEnum] "Synchronous"
$SyncMode             = [DellStorage.Api.Enums.SyncReplicationModeEnum] "HighAvailability"
$SecScName            = "SC 13"
$SecVolumeNamePrefix  = "LV of "
$SecQosNodeName       = "4Gbps_SC13"
$SecServerName        = "SalesFS01"
$SecServerFolderPath  = "Production/FileServer/"
$SecDataReductionProfileName = "None"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the primary QoS Node

$PriQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $PriScName `
                                           -Name           $PriQosNodeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the secondary QoS Node

$SecQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $SecScName `
                                           -Name           $SecQosNodeName

# Get the data reduction profile for the secondary volume

$SecDataReductionProfile = Get-DellScDataReductionProfile -ConnectionName "DSMDC" `
                                                          -StorageCenter $SecStorageCenter `
                                                          -InstanceName  $SecDataReductionProfileName
```

```
# Define the secondary volume attributes

$SecVolumeName = $SecVolumeNamePrefix + $PriVolumeName

$SecVolumeAttrib = New-DellScVolumeCreateAttributes -Name      $SecVolumeName `
                                                    -DataReductionProfile $SecDataReductionProfile `
                                                    -ReadCache:$true `
                                                    -WriteCache:$true `
                                                    -CreateSourceVolumeFolderPath:$true

# Get the server on the secondary Storage Center

$SecServer = Get-DellScServer -ConnectionName   $ConnName `
                              -StorageCenter    $SecStorageCenter `
                              -Name             $SecServerName `
                              -ServerFolderPath $SecServerFolderPath

# Define the advanced mapping attributes for the secondary Storage Center

$SecAdvMapping = New-DellScAdvancedMapping -UseSameLunAsSource:$true

# Create the Live Volume

$LiveVolume = New-DellScLiveVolume -ConnectionName            $ConnName `
                                   -StorageCenter             $PriStorageCenter `
                                   -PrimaryVolume             $PriVolume `
                                   -TransportTypes            $TransportTypes `
                                   -Type                      $ReplType `
                                   -SyncMode                  $SyncMode `
                                   -PrimaryQosNode            $PriQosNode `
                                   -SecondaryStorageCenter    $SecStorageCenter `
                                   -SecondaryVolumeAttributes $SecVolumeAttrib `
                                   -SecondaryQosNode          $SecQosNode `
                                   -SecondaryServer           $SecServer `
                                   -SecondaryAdvancedMapping  $SecAdvMapping `
                                   -SwapRolesAutomaticallyEnabled:$true
```

3.12.3 Create a synchronous Live Volume with auto failover

The **New-DellScLiveVolume** cmdlet can be used to create a new synchronous Live Volume with auto failover. This sample script will create a new synchronous Live Volume with auto failover for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**. The primary volume will be on Storage Center **SC 12** and the secondary volume will be on Storage Center **SC 13**.

```
# Assign variables

$ConnName             = "DSMDC"
$PriScName            = "SC 12"
$PriVolumeName        = "Backup"
$PriVolumeFolderPath  = "Production/FileServer/SalesFS01/"
$PriQosNodeName       = "4Gbps_SC12"
$TransportTypes       = [DellStorage.Api.Enums.FrontEndTransportTypeEnum[]] "FibreChannel"
$ReplType             = [DellStorage.Api.Enums.ReplicationTypeEnum] "Synchronous"
$SyncMode             = [DellStorage.Api.Enums.SyncReplicationModeEnum] "HighAvailability"
$SecScName            = "SC 13"
$SecVolumeNamePrefix  = "LV of "
$SecQosNodeName       = "4Gbps_SC13"
$SecServerName        = "SalesFS01"
$SecServerFolderPath  = "Production/FileServer/"
$SecDataReductionProfileName = "None"
```

```powershell
# Get the primary Storage Center
$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume
$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the primary QoS Node
$PriQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $PriScName `
                                           -Name           $PriQosNodeName

# Get the secondary Storage Center
$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the secondary QoS Node
$SecQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $SecScName `
                                           -Name           $SecQosNodeName

# Get the data reduction profile for the secondary volume
$SecDataReductionProfile = Get-DellScDataReductionProfile -ConnectionName "DSMDC" `
                                                          -StorageCenter $SecStorageCenter `
                                                          -InstanceName  $SecDataReductionProfileName

# Define the secondary volume attributes
$SecVolumeName = $SecVolumeNamePrefix + $PriVolumeName

$SecVolumeAttrib = New-DellScVolumeCreateAttributes -Name      $SecVolumeName `
                                                    -DataReductionProfile $SecDataReductionProfile `
                                                    -ReadCache:$true `
                                                    -WriteCache:$true `
                                                    -CreateSourceVolumeFolderPath:$true

# Get the server on the secondary Storage Center
$SecServer = Get-DellScServer -ConnectionName   $ConnName `
                              -StorageCenter    $SecStorageCenter `
                              -Name             $SecServerName `
                              -ServerFolderPath $SecServerFolderPath

# Define the advanced mapping attributes for the secondary Storage Center
$SecAdvMapping = New-DellScAdvancedMapping -UseSameLunAsSource:$true
```

```
# Create the Live Volume

$LiveVolume = New-DellScLiveVolume -ConnectionName            $ConnName `
                                   -StorageCenter             $PriStorageCenter `
                                   -PrimaryVolume             $PriVolume `
                                   -TransportTypes            $TransportTypes `
                                   -Type                      $ReplType `
                                   -SyncMode                  $SyncMode `
                                   -PrimaryQosNode            $PriQosNode `
                                   -SecondaryStorageCenter    $SecStorageCenter `
                                   -SecondaryVolumeAttributes $SecVolumeAttrib `
                                   -SecondaryQosNode          $SecQosNode `
                                   -SecondaryServer           $SecServer `
                                   -SecondaryAdvancedMapping  $SecAdvMapping `
                                   -SwapRolesAutomaticallyEnabled:$true `
                                   -FailoverAutomaticallyEnabled:$true `
                                   -RestoreAutomaticallyEnabled:$true
```

3.12.4 Convert a Live Volume to a replication

The **Start-DellScLiveVolumeToReplicationRevert** cmdlet can be used to convert a Live Volume to a replication. This sample script will convert the Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary) for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**. After the script is run, the volume will be replicated from Storage Center **SC 12** to Storage Center **SC 13**.

```
# Assign variables

$ConnName            = "DSMDC"
$PriScName           = "SC 12"
$PriVolumeName       = "Backup"
$PriVolumeFolderPath = "Production/FileServer/SalesFS01/"
$SecScName           = "SC 13"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the Live Volume

$LiveVolume = Get-DellScLiveVolume -ConnectionName         $ConnName `
                                   -PrimaryVolume          $PriVolume `
                                   -PrimaryStorageCenter   $PriStorageCenter `
                                   -SecondaryStorageCenter $SecStorageCenter

# Convert the Live Volume to a replication

$Replication = Start-DellScLiveVolumeToReplicationRevert -ConnectionName $ConnName `
                                                         -Instance       $LiveVolume `
                                                         -Confirm:$false
```

3.12.5 Convert a replication to a Live Volume

The **Convert-DellScReplicationToLiveVolume** cmdlet can be used to convert a replication to a Live Volume. This sample script will convert the replication to Storage Center **SC 13** for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**. After the script is run, the volume will be a Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary).

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "Backup"
$SrcVolumeFolderPath  = "Production/FileServer/SalesFS01/"
$DstScName            = "SC 13"
$DstQosNodeName       = "4Gbps_SC13"
$DstServerName        = "SalesFS01"
$DstServerFolderPath  = "Production/FileServer/"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName

# Get the replication

$Replication = Get-DellScReplication -ConnectionName           $ConnName `
                                     -SourceStorageCenter      $SrcStorageCenter `
                                     -SourceVolume             $SrcVolume `
                                     -DestinationStorageCenter $DstStorageCenter


# Get the QoS Node for the destination Storage Center

$DstQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $DstScName `
                                           -Name           $DstQosNodeName

# Get the server on the destination Storage Center

$DstServer = Get-DellScServer -ConnectionName   $ConnName `
                              -StorageCenter    $DstStorageCenter `
                              -Name             $DstServerName `
                              -ServerFolderPath $DstServerFolderPath

# Define the advanced mapping attributes for the destination Storage Center

$DstAdvMapping = New-DellScAdvancedMapping -UseSameLunAsSource:$true
```

```
# Convert the replication to a Live Volume

$LiveVolume = Convert-DellScReplicationToLiveVolume -ConnectionName      $ConnName `
                                                    -Instance            $Replication `
                                                    -SecondaryQosNode    $DstQosNode `
                                                    -SecondaryServer     $DstServer `
                                                    -SecondaryAdvancedMapping $DstAdvMapping `
                                                    -SwapRolesAutomaticallyEnabled:$true
```

3.12.6 Modify the replication type for a Live Volume

The **Set-DellScLiveVolume** cmdlet can be used to modify the replication type for a Live Volume. This sample script will modify the Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary) for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**. This sample script shows how to change the replication type to asynchronous, synchronous in high-consistency mode, or synchronous in high-availability mode.

```
# Assign variables

$ConnName            = "DSMDC"
$PriScName           = "SC 12"
$PriVolumeName       = "Backup"
$PriVolumeFolderPath = "Production/FileServer/SalesFS01/"
$SecScName           = "SC 13"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the Live Volume

$LiveVolume = Get-DellScLiveVolume -ConnectionName         $ConnName `
                                   -PrimaryVolume          $PriVolume `
                                   -PrimaryStorageCenter   $PriStorageCenter `
                                   -SecondaryStorageCenter $SecStorageCenter

# Change the replication type to asynchronous

$NewReplType = [DellStorage.Api.Enums.ReplicationTypeEnum] "Asynchronous"

$LiveVolume = Set-DellScLiveVolume -ConnectionName $ConnName `
                                   -Instance       $LiveVolume `
                                   -Type           $NewReplType
```

```
# Change the replication type to synchronous, high-consistency mode

$NewReplType = [DellStorage.Api.Enums.ReplicationTypeEnum] "Synchronous"
$NewSyncMode = [DellStorage.Api.Enums.SyncReplicationModeEnum] "HighConsistency"

$LiveVolume = Set-DellScLiveVolume -ConnectionName $ConnName `
                                   -Instance       $LiveVolume `
                                   -Type           $NewReplType `
                                   -SyncMode       $NewSyncMode `
                                   -ReplicateStorageToLowestTier:$false `
                                   -ReplicateActiveReplay:$true

# Change the replication type to synchronous, high-availability mode

$NewReplType = [DellStorage.Api.Enums.ReplicationTypeEnum] "Synchronous"
$NewSyncMode = [DellStorage.Api.Enums.SyncReplicationModeEnum] "HighAvailability"

$LiveVolume = Set-DellScLiveVolume -ConnectionName $ConnName `
                                   -Instance       $LiveVolume `
                                   -Type           $NewReplType `
                                   -SyncMode       $NewSyncMode `
                                   -ReplicateStorageToLowestTier:$false `
                                   -ReplicateActiveReplay:$true
```

3.12.7 Modify Live Volume attributes

The **Set-DellScLiveVolume** cmdlet can be used to modify Live Volume attributes. This sample script will modify the Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary) for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**. This sample script shows how to change several different attributes for a Live Volume.

```
# Assign variables

$ConnName            = "DSMDC"
$PriScName           = "SC 12"
$PriVolumeName       = "Backup"
$PriVolumeFolderPath = "Production/FileServer/SalesFS01/"
$SecScName           = "SC 13"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the Live Volume

$LiveVolume = Get-DellScLiveVolume -ConnectionName         $ConnName `
                                   -PrimaryVolume          $PriVolume `
                                   -PrimaryStorageCenter   $PriStorageCenter `
                                   -SecondaryStorageCenter $SecStorageCenter
```

```
# Enable swap roles automatically

$LiveVolume = Set-DellScLiveVolume -ConnectionName                $ConnName `
                                   -Instance                      $LiveVolume `
                                   -MinAmountBeforeSwap           "1 MB" `
                                   -MinSecondaryPercentBeforeSwap 60 `
                                   -SwapRolesAutomaticallyEnabled:$true

# Disable swap roles automatically

$LiveVolume = Set-DellScLiveVolume -ConnectionName $ConnName `
                                   -Instance       $LiveVolume `
                                   -SwapRolesAutomaticallyEnabled:$false

# Enable Failover Automatically and Restore Automatically

$LiveVolume = Set-DellScLiveVolume -ConnectionName $ConnName `
                                   -Instance       $LiveVolume `
                                   -FailoverAutomaticallyEnabled:$true `
                                   -RestoreAutomaticallyEnabled:$true

# Disable Restore Automatically

$LiveVolume = Set-DellScLiveVolume -ConnectionName $ConnName `
                                   -Instance       $LiveVolume `
                                   -RestoreAutomaticallyEnabled:$false

# Disable Failover Automatically

$LiveVolume = Set-DellScLiveVolume -ConnectionName $ConnName `
                                   -Instance       $LiveVolume `
                                   -FailoverAutomaticallyEnabled:$false

# Change the primary QoS Node to "QOS 1Gb"

$NewQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $PriScName `
                                           -Name           "QOS 1Gb"

$LiveVolume = Set-DellScLiveVolume -ConnectionName $ConnName `
                                   -Instance       $LiveVolume `
                                   -PrimaryQosNode $NewQosNode
```

3.12.8 Initiate a Live Volume swap role

The **Start-DellScLiveVolumeSwapRole** cmdlet can be used to swap the primary SC Series array for a Live Volume. This sample script will swap the primary SC Series array for the Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary) for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**. After the script is run, the primary SC Series array will be **SC 13**.

```
# Assign variables

$ConnName            = "DSMDC"
$PriScName           = "SC 12"
$PriVolumeName       = "Backup"
$PriVolumeFolderPath = "Production/FileServer/SalesFS01/"
$SecScName           = "SC 13"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName
```

```
# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the Live Volume

$LiveVolume = Get-DellScLiveVolume -ConnectionName         $ConnName `
                                   -PrimaryVolume          $PriVolume `
                                   -PrimaryStorageCenter   $PriStorageCenter `
                                   -SecondaryStorageCenter $SecStorageCenter

# Initiate a swap of the primary Storage Center

Start-DellScLiveVolumeSwapRole -ConnectionName $ConnName `
                               -Instance       $LiveVolume `
                               -Confirm:$false
```

3.12.9 Pause a Live Volume

The **Suspend-DellScLiveVolume** cmdlet can be used to pause a Live Volume. This sample script will pause the Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary) for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName            = "DSMDC"
$PriScName           = "SC 12"
$PriVolumeName       = "Backup"
$PriVolumeFolderPath = "Production/FileServer/SalesFS01/"
$SecScName           = "SC 13"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the Live Volume

$LiveVolume = Get-DellScLiveVolume -ConnectionName         $ConnName `
                                   -PrimaryVolume          $PriVolume `
                                   -PrimaryStorageCenter   $PriStorageCenter `
                                   -SecondaryStorageCenter $SecStorageCenter
```

```
# Pause the Live Volume

If ( $LiveVolume.PauseAllowed )
{
    Suspend-DellScLiveVolume -ConnectionName $ConnName `
                             -Instance       $LiveVolume `
                             -Confirm:$false
}
```

3.12.10 Resume a paused Live Volume

The **Resume-DellScLiveVolume** cmdlet can be used to resume a Live Volume that has been paused. This sample script will resume the Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary) for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName            = "DSMDC"
$PriScName           = "SC 12"
$PriVolumeName       = "Backup"
$PriVolumeFolderPath = "Production/FileServer/SalesFS01/"
$SecScName           = "SC 13"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the Live Volume

$LiveVolume = Get-DellScLiveVolume -ConnectionName         $ConnName `
                                   -PrimaryVolume          $PriVolume `
                                   -PrimaryStorageCenter   $PriStorageCenter `
                                   -SecondaryStorageCenter $SecStorageCenter

# Resume the Live Volume

If ( $LiveVolume.ReplicationState.Name -eq "Paused" )
{
    Resume-DellScLiveVolume -ConnectionName $ConnName `
                            -Instance       $LiveVolume `
                            -Confirm:$false
}
```

3.12.11 Add a managed replication to a Live Volume

The **New-DellScReplication** cmdlet can be used to create a replication managed by a Live Volume. This sample script will create an asynchronous replication to Storage Center **SC 2** managed by the Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary) for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName             = "DSMDC"
$PriScName            = "SC 12"
$PriVolumeName        = "Backup"
$PriVolumeFolderPath  = "Production/FileServer/SalesFS01/"
$PriQosNodeName       = "4Gbps_SC12"
$SecScName            = "SC 13"
$SecQosNodeName       = "4Gbps_SC13"
$MRReplType           = [DellStorage.Api.Enums.ReplicationTypeEnum] "Asynchronous"
$PriTransportTypes    = [DellStorage.Api.Enums.FrontEndTransportTypeEnum[]] "FibreChannel"
$SecTransportTypes    = [DellStorage.Api.Enums.FrontEndTransportTypeEnum[]] "FibreChannel"
$DstScName            = "SC 2"
$DstVolumeNamePrefix  = "Repl of "
$DstDataReductionProfileName = "None"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary QoS Node

$PriQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $PriScName `
                                           -Name           $PriQosNodeName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the secondary QoS Node

$SecQosNode = Get-DellScReplicationQosNode -ConnectionName $ConnName `
                                           -ScName         $SecScName `
                                           -Name           $SecQosNodeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName

# Get the data reduction profile for the destination volume

$DstDataReductionProfile = Get-DellScDataReductionProfile -ConnectionName "DSMDC" `
                                                          -StorageCenter $DstStorageCenter `
                                                          -InstanceName  $DstDataReductionProfileName
```

```
# Define the destination volume attributes

$DstVolumeName = $DstVolumeNamePrefix + $PriVolumeName

$DstVolumeAttrib = New-DellScVolumeCreateAttributes -Name      $DstVolumeName `
                                                    -DataReductionProfile $DstDataReductionProfile `
                                                    -ReadCache:$true `
                                                    -WriteCache:$true `
                                                    -CreateSourceVolumeFolderPath:$true

# Create the managed replication

$Replication = New-DellScReplication -ConnectionName              $ConnName `
                                     -Type                        $MRReplType `
                                     -StorageCenter               $PriStorageCenter `
                                     -SourceVolume                $PriVolume `
                                     -TransportTypes              $PriTransportTypes `
                                     -QosNode                     $PriQosNode `
                                     -SecondaryTransportTypes     $SecTransportTypes `
                                     -SecondaryQosNode            $SecQosNode `
                                     -DestinationStorageCenter    $DstStorageCenter `
                                     -DestinationVolumeAttributes $DstVolumeAttrib `
                                     -ReplicateStorageToLowestTier:$true `
                                     -ReplicateActiveReplay:$false `
                                     -ManagedByLiveVolume:$true `
                                     -Dedup:$false
```

3.12.12 Remove a managed replication from a Live Volume

The **Remove-DellScReplication** cmdlet can be used to remove a replication managed by a Live Volume. This sample script will remove the replication to Storage Center **SC 2** that is managed by the Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary) for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName            = "DSMDC"
$PriScName           = "SC 12"
$PriVolumeName       = "Backup"
$PriVolumeFolderPath = "Production/FileServer/SalesFS01/"
$SecScName           = "SC 13"
$DstScName           = "SC 2"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $DstScName
```

```
# Get the Live Volume

$LiveVolume = Get-DellScLiveVolume -ConnectionName          $ConnName `
                                   -PrimaryVolume           $PriVolume `
                                   -PrimaryStorageCenter    $PriStorageCenter `
                                   -SecondaryStorageCenter $SecStorageCenter

# Get the managed replication

$Replication = Get-DellScReplication -ConnectionName           $ConnName `
                                     -ManagingLiveVolume       $LiveVolume `
                                     -DestinationStorageCenter $DstStorageCenter

# Remove the managed replication

Remove-DellScReplication -ConnectionName $ConnName `
                         -Instance       $Replication `
                         -RecycleDestinationVolume:$true `
                         -DeleteDestinationVolume:$true `
                         -DeleteRestorePoint:$true `
                         -Confirm:$false
```

3.12.13 Remove a Live Volume

The **Remove-DellScLiveVolume** cmdlet can be used to remove a Live Volume. This sample script will remove the Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary) for the **Backup** volume in the **Production/FileServer/SalesFS01** folder on Storage Center **SC 12**.

```
# Assign variables

$ConnName             = "DSMDC"
$PriScName            = "SC 12"
$PriVolumeName        = "Backup"
$PriVolumeFolderPath = "Production/FileServer/SalesFS01/"
$SecScName            = "SC 13"

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $PriScName

# Get the primary volume

$PriVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $PriStorageCenter `
                              -VolumeFolderPath $PriVolumeFolderPath `
                              -Name             $PriVolumeName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -ScName         $SecScName

# Get the Live Volume

$LiveVolume = Get-DellScLiveVolume -ConnectionName          $ConnName `
                                   -PrimaryVolume           $PriVolume `
                                   -PrimaryStorageCenter    $PriStorageCenter `
                                   -SecondaryStorageCenter $SecStorageCenter
```

```
# Remove the Live Volume

Remove-DellScLiveVolume -ConnectionName $ConnName `
                        -Instance       $LiveVolume `
                        -ConvertToReplication:$false `
                        -RecycleSecondaryVolume:$true `
                        -DeleteSecondaryVolume:$true `
                        -DeleteRestorePoint:$true `
                        -Confirm:$false
```

3.13 Users

The Dell Storage PowerShell SDK provides cmdlets to manage users on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 13 shows the user cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 13 User cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCUser	Get-DellScUser
New-SCUser	New-DellScUser
Remove-SCUser	Remove-DellScUser
Set-SCUser	Set-DellScUser

3.13.1 Create a user

The **New-DellScUser** cmdlet can be used to create a user on an SC Series array. This sample script will create a **Reporter** user for **John Doe** with a user name of **JohnD**. The script will prompt for the account's password.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 13"
$ScUserName = "JohnD"
$FullName   = "John Doe"
$Privilege  = "Reporter"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -Name           $ScName

# Get the password

$ScPassword = Read-Host -AsSecureString -Prompt "Enter the password for $ScUserName"

# Create the user

$ScUser = New-DellScUser -ConnectionName $ConnName `
                         -StorageCenter  $StorageCenter `
                         -Name           $ScUserName `
                         -Password       $ScPassword `
                         -Privilege      $Privilege `
                         -RealName       $FullName
```

3.13.2 Modify a user

The **Set-DellScUser** cmdlet can be used to modify a user. This sample script will grant **Administrator** privileges to the **JohnD** user.

```
# Assign variables

$ConnName     = "DSMDC"
$ScName       = "SC 13"
$ScUserName   = "JohnD"
$NewPrivilege = "Admin"

# Get the user

$ScUser = Get-DellScUser -ConnectionName $ConnName `
                         -ScName         $ScName `
                         -Name           $ScUserName

# Change the user privilege to Administrator

$ScUser = Set-DellScUser -ConnectionName $ConnName `
                         -Instance       $ScUser `
                         -Privilege      $NewPrivilege
```

3.13.3 Remove a user

The **Remove-DellScUser** cmdlet can be used to remove a user. This sample script will remove the **JohnD** user.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 13"
$ScUserName = "JohnD"

# Get the user

$ScUser = Get-DellScUser -ConnectionName $ConnName `
                         -ScName         $ScName `
                         -Name           $ScUserName

# Remove the user

Remove-DellScUser -ConnectionName $ConnName `
                  -Instance       $ScUser `
                  -Confirm:$false
```

3.14 Disks

The Dell Storage PowerShell SDK provides cmdlets to manage the disks on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 14 shows the disk cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 14 Disk cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Get-SCDisk	Get-DellScDisk
Get-SCDiskFolder	Get-DellScDiskFolder
Set-SCDisk	Set-DellScDiskRevertToManaged

3.14.1 Get all disks in a disk folder

The **Get-DellScDisk** cmdlet can be used the get a list of disks in a disk folder. This sample script will get the disks in the **Assigned** disk folder.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 2"
$FolderName = "Assigned"

# Get the disk folder

$DiskFolder = Get-DellScDiskFolder -ConnectionName $ConnName `
                                   -ScName         $ScName `
                                   -Name           $FolderName

# Get the disks in the disk folder

$DiskList = Get-DellScDisk -ConnectionName $ConnName `
                           -ScName         $ScName `
                           -DiskFolder     $DiskFolder

# Display the name and size of each disk, sorted by name

$DiskList | Select-Object Name, Size, DiskClass | Sort-Object Name
```

3.14.2 Get all disks in a tier

The **Get-DellScDiskFolderTierDiskListAssociation** cmdlet can be used to get a list of disks in a tier. This sample script will get the disks in **Tier 1**, in the **Assigned** disk folder.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 2"
$FolderName = "Assigned"
$TierName   = "Tier1"

# Get the disk folder

$DiskFolder = Get-DellScDiskFolder -ConnectionName $ConnName `
                                   -ScName         $ScName `
                                   -Name           $FolderName
# Get the disk tier

$DiskTier = Get-DellScDiskFolderTier -ConnectionName $ConnName `
                                     -ScName         $ScName `
                                     -DiskFolder     $DiskFolder `
                                     -DiskTier       $TierName

# Get the disks in the tier

$DiskList = Get-DellScDiskFolderTierDiskListAssociation -ConnectionName $ConnName `
                                                        -Instance       $DiskTier

# Display the name and size of each disk, sorted by name

$DiskList | Select-Object Name, Size, DiskClass | Sort-Object Name
```

3.14.3 Get all unmanaged disks

The **Get-DellScDisk** cmdlet can be used to get a list of the unmanaged disks. This sample script will get all of the unmanaged disks. The unmanaged disks will be in the **Unassigned** disk folder.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 2"
$FolderName = "Unassigned"

# Get the unassigned disk folder

$DiskFolder = Get-DellScDiskFolder -ConnectionName $ConnName `
                                   -ScName         $ScName `
                                   -Name           $FolderName

# Get the disks in the unassigned disk folder

$DiskList = Get-DellScDisk -ConnectionName $ConnName `
                           -ScName         $ScName `
                           -DiskFolder     $DiskFolder

# Display the name and size of each disk, sorted by name

$DiskList | Select-Object Name, Size, DiskClass | Sort-Object Name
```

3.14.4 Add unmanaged disks to a disk folder

The **Add-DellScDiskFolderDisk** cmdlet can be used to add unmanaged disks to a disk folder. This sample script add all of the unmanaged disks to the **Assigned** disk folder.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 2"
$FolderName = "Assigned"

# Get the disk folder

$DiskFolder = Get-DellScDiskFolder -ConnectionName $ConnName `
                                   -ScName         $ScName `
                                   -Name           $FolderName
# Get all unmanaged disks

$DiskList = Get-DellScDisk -ConnectionName $ConnName `
                           -ScName         $ScName `
                           -ControlType    "Unmanaged"

# Add the disks to the disk folder

Add-DellScDiskFolderDisk -ConnectionName $ConnName `
                         -Instance       $DiskFolder `
                         -Disks          $DiskList `
                         -Confirm:$false
```

3.14.5 Release disks from a disk folder

The **Set-DellScDiskRelease** cmdlet can be used to release disks from a disk folder. This sample script will release disks 12 through 17 in enclosure 5 from the **Assigned** disk folder.

```
# Assign variables

$ConnName     = "DSMDC"
$ScName       = "SC 2"
$FolderName   = "Assigned"
$DiskNameList = ( "05-12","05-13","05-14","05-15","05-16","05-17" )

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -Name           $ScName

# Get the disk folder

$DiskFolder = Get-DellScDiskFolder -ConnectionName $ConnName `
                                   -ScName         $ScName `
                                   -Name           $FolderName

# Get the disk

$DiskList = Get-DellScDisk -ConnectionName $ConnName `
                           -ScName         $ScName `
                           -DiskFolder     $DiskFolder `
            | Where-Object { $DiskNameList -contains $_.Name }

# Release the disks

Set-DellScDiskRelease -ConnectionName $ConnName `
                      -StorageCenter  $StorageCenter `
                      -ReleasedDisks  $DiskList `
                      -Confirm:$false
```

3.14.6 Running RAID rebalance

The **Start-DellScDiskFolderRaidRebalance** cmdlet can be used to initiate a RAID rebalance. The **Get-DellScDiskFolderGenerateRaidRebalanceStatus** cmdlet can be used to determine if a RAID rebalance is needed or if it is running. This sample script will start a RAID rebalance if it is needed. It also shows an example of checking to see if a RAID rebalance is running.

```
# Assign variables

$ConnName = "DSMDC"
$ScName   = "SC 2"

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -Name           $ScName

# See if RAID rebalance is running

$Status = Get-DellScDiskFolderGenerateRaidRebalanceStatus -ConnectionName $ConnName `
                                                          -StorageCenter  $StorageCenter

If ( $Status.Status -like "InProgress*" )
{
    Write-Host "RAID rebalance is in progress" -ForegroundColor Green
}
Else
{
    Write-Host "RAID rebalance is not running" -ForegroundColor Green
}

# If RAID rebalance is needed, start it

$Status = Get-DellScDiskFolderGenerateRaidRebalanceStatus -ConnectionName $ConnName `
                                                          -StorageCenter  $StorageCenter

If ( $Status.Status -eq "RebalanceNeeded" )
{
    Start-DellScDiskFolderRaidRebalance -ConnectionName $ConnName `
                                        -StorageCenter  $StorageCenter `
                                        -Confirm:$false
}
Else
{
    Write-Host "RAID rebalance is not needed" -ForegroundColor Green
}
```

3.15 Alerts

The Dell Storage PowerShell SDK provides cmdlets to manage alerts on SC Series arrays. This functionality is also provided in the legacy Storage Center PowerShell Command Set. Table 15 shows the disk cmdlets in the legacy command set, along with the PowerShell SDK cmdlets that provide similar functionality.

Table 15 Alert cmdlets

Legacy cmdlet	PowerShell SDK cmdlet
Acknowledge-SCAlert	Set-DellScAlertAcknowledged
Get-SCAlert	Get-DellScAlert

3.15.1 Get active alerts

The **Get-DellScAlert** cmdlet can be used to get active alerts. This sample script will get the active alerts from Storage Center **SC 13**.

```
# Assign variables

$ConnName  = "DSMDC"
$ScName    = "SC 13"
$AlertType = [DellStorage.Api.Enums.AlertTypeEnum] "Alert"

# Get active alerts

$ScAlertList = @( Get-DellScAlert -ConnectionName $ConnName `
                                  -ScName         $ScName `
                                  -Type           $AlertType `
                                  -Maintenance:$false `
                                  -Active:$true `
                                   )

# Display the alerts, sorted by CreateTime

$ScAlertList | Select-Object CreateTime, Acknowledged, AlertStatus, Message `
             | Sort-Object CreateTime
```

3.15.2 Get maintenance alerts

The **Get-DellScAlert** cmdlet can be used to get maintenance alerts. This sample script will get the maintenance alerts from Storage Center **SC 13**.

```
# Assign variables

$ConnName  = "DSMDC"
$ScName    = "SC 13"
$AlertType = [DellStorage.Api.Enums.AlertTypeEnum] "Alert"

# Get active alerts

$ScAlertList = @( Get-DellScAlert -ConnectionName $ConnName `
                                  -ScName         $ScName `
                                  -Type           $AlertType `
                                  -Maintenance:$true `
                                  -Active:$true `
                                   )

# Display the alerts, sorted by CreateTime

$ScAlertList | Select-Object CreateTime, Acknowledged, AlertStatus, Message `
             | Sort-Object CreateTime
```

3.15.3 Get indications

The **Get-DellScAlert** cmdlet can be used to get indication alerts. This sample script will get the indication alerts from Storage Center **SC 13**.

```
# Assign variables

$ConnName  = "DSMDC"
$ScName    = "SC 13"
$AlertType = [DellStorage.Api.Enums.AlertTypeEnum] "Indication"
```

```
# Get active alerts

$ScAlertList = @( Get-DellScAlert -ConnectionName $ConnName `
                                  -ScName         $ScName `
                                  -Type           $AlertType )

# Display the alerts, sorted by CreateTime

$ScAlertList | Select-Object CreateTime, Type, AlertStatus, Message `
             | Sort-Object CreateTime
```

3.15.4 Get alert history

The **Get-DellScAlert** cmdlet can be used to get alert history. This sample script will get the alert history from Storage Center **SC 13**.

```
# Assign variables

$ConnName  = "DSMDC"
$ScName    = "SC 13"
$AlertType = [DellStorage.Api.Enums.AlertTypeEnum] "Alert"

# Get alert history

$ScAlertList = @( Get-DellScAlert -ConnectionName $ConnName `
                                  -ScName         $ScName `
                                  -Type           $AlertType `
                                  -Active:$false )

# Display the alerts, sorted by CreateTime

$ScAlertList | Select-Object CreateTime, Acknowledged, AlertStatus, Message `
             | Sort-Object CreateTime
```

3.15.5 Acknowledge active alerts

The **Set-DellScAlertAcknowledged** cmdlet can be used to acknowledge active alerts. This sample script will acknowledge active alerts on Storage Center **SC 13**.

```
# Assign variables

$ConnName  = "DSMDC"
$ScName    = "SC 14"
$AlertType = [DellStorage.Api.Enums.AlertTypeEnum] "Alert"

# Get active alerts that have not been acknowledged

$ScAlertList = @( Get-DellScAlert -ConnectionName $ConnName `
                                  -ScName         $ScName `
                                  -Maintenance:$false `
                                  -Active:$true `
                  | Where-Object { $_.Acknowledged -eq $false -and `
                                   $_.AlertType    -eq $AlertType } )

# Loop through the list of alerts and acknowledge them

ForEach ( $ScAlert in $ScAlertList )
{
    Set-DellScAlertAcknowledged -ConnectionName $ConnName `
                                -Instance       $ScAlert `
                                -Confirm:$false
}
```

3.15.6 Acknowledge maintenance alerts

The **Set-DellScAlertAcknowledged** cmdlet can be used to acknowledge maintenance alerts. This sample script will acknowledge maintenance alerts on Storage Center **SC 13**.

```
# Assign variables

$ConnName  = "DSMDC"
$ScName    = "SC 13"
$AlertType = [DellStorage.Api.Enums.AlertTypeEnum] "Alert"

# Get maintenance alerts that have not been acknowledged

$ScAlertList = @( Get-DellScAlert -ConnectionName $ConnName `
                                  -ScName         $ScName `
                                  -Maintenance:$true `
                                  -Active:$true `
                  | Where-Object { $_.Acknowledged -eq $false -and `
                                   $_.AlertType    -eq $AlertType } )

# Loop through the list of alerts and acknowledge them

ForEach ( $ScAlert in $ScAlertList )
{
    Set-DellScAlertAcknowledged -ConnectionName $ConnName `
                                -Instance       $ScAlert `
                                -Confirm:$false
}
```

4 Working with disaster recovery

The Dell Storage PowerShell SDK cmdlets that can be used to automate disaster recovery (DR) will have the prefix **DellDr** in the noun portion of the cmdlet. For example, the cmdlet that will retrieve replication restore points is called **Get-DellDrScReplicationRestorePoint**. The following sample script will return the DR cmdlets.

```
# Get the DR cmdlets

Get-Command -Module "DellStorage.ApiCommandSet" `
| Where-Object { $_.Name -like "*-DellDr*" } `
| Sort-Object Name `
| Select-Object Name
```

The sample scripts in this section assume the Dell Storage PowerShell SDK module has already been imported and a saved connection named **DSMDC** has already been created.

4.1 Restore points

The Dell Storage PowerShell SDK includes cmdlets that can be used to manage restore points associated with volume replications and Live Volumes.

4.1.1 Save restore points for an SC Series array

The **Save-DellStorageCenterRestorePoint** cmdlet can be used to save the restore points for an SC Series array. This sample script will save all of the restore points for Storage Center **SC 12**.

```
# Assign variables

$ConnName = "DSMDC"
$ScName   = "SC 12"

# Get the source Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -Name           $ScName

# Save restore points

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $StorageCenter `
                                   -Confirm:$false
```

4.1.2 Save a single restore point

The **Save-DellDrScReplicationRestorePointSingle** cmdlet can be used to save a single restore point. This sample script will save the restore point for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLBackup2"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstScName            = "SC 13"
```

```
# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Get the restore point for the replicated volume

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Save the restore point

Save-DellDrScReplicationRestorePointSingle -ConnectionName $ConnName `
                                           -Instance       $RestorePoint
```

4.1.3 Delete a single restore point

The **Remove-DellDrScReplicationRestorePoint** cmdlet can be used to remove a restore point. This sample script will delete the restore point for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLBackup2"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstScName            = "SC 13"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName
```

```
# Get the restore point for the replicated volume

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Save the restore point

Remove-DellDrScReplicationRestorePoint -ConnectionName $ConnName `
                                       -Instance       $RestorePoint `
                                       -Confirm:$false
```

4.1.4 Validate all restore points

The **Test-DellStorageCenterRestorePoint** cmdlet can be used to validate all restore points. This sample script will validate all restore points.

```
# Assign variables

$ConnName = "DSMDC"

# Validate all restore points

Test-DellDrScReplicationRestorePointAll -ConnectionName $ConnName `
                                        -Confirm:$false
```

4.1.5 Validate restore points for an SC Series array

The **Test-DellStorageCenterRestorePoint** cmdlet can be used to validate all of the restore points for an SC Series array. This sample script will validate all of the restore points for Storage Center **SC 12**.

```
# Assign variables

$ConnName = "DSMDC"
$ScName   = "SC 12"

# Get the source Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                       -Name           $ScName

# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $StorageCenter `
                                   -Confirm:$false
```

4.1.6 Validate a single restore point

The **Test-DellDrScReplicationRestorePointSingle** cmdlet can be used to validate a single restore point. This sample script will validate the restore point for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13**.

```
# Assign variables

$ConnName            = "DSMDC"
$SrcScName           = "SC 12"
$SrcVolumeName       = "SQLBackup2"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstScName           = "SC 13"
```

```
# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Get the restore point for the replicated volume

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Validate restore points

Test-DellDrScReplicationRestorePointSingle -ConnectionName $ConnName `
                                           -Instance       $RestorePoint `
                                           -Confirm:$false
```

4.2 Predefined DR Plans

The PowerShell SDK includes cmdlets that can be used to manage predefined DR plans for replicated volumes.

4.2.1 Create a predefined DR plan

The **Set-DellDrScReplicationRestorePointPredefine** cmdlet can be used to predefine a DR plan for a replicated volume. This sample script will prefine the DR plan for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13**. The predefined DR plan will recover the volume from the active snapshot and map it to server **SQLDR** as a writable volume. The recovered volume will be mapped using the next available LUN.

```
# Assign variables

$ConnName            = "DSMDC"
$SrcScName           = "SC 12"
$SrcVolumeName       = "SQLBackup2"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstScName           = "SC 13"
$DRServerName        = "SQLDR"
$DRVolumeName        = "DR of " + $SrcVolumeName

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName
```

```
# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Save restore points

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Get the restore point for the replicated volume

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Get the DR server

$DRServer = Get-DellScServer -ConnectionName $ConnName `
                             -StorageCenter  $DstStorageCenter `
                             -Name           $DRServerName

# Define the advanced mapping options

$AdvancedMapping = New-DellScAdvancedMapping -ReadOnly:$false `
                                             -UseNextAvailable:$true

# Define the activation settings

$DrActivateSettings = New-DellDrActivateSettings -Name             $DRVolumeName `
                                                 -ServerList       $DRServer `
                                                 -AdvancedMapping  $AdvancedMapping `
                                                 -UseActiveReplay:$true `
                                                 -UseOriginalVolumesFolder:$true

# Predefine disaster recovery for the restore point

Set-DellDrScReplicationRestorePointPredefine -ConnectionName     $ConnName `
                                             -Instance           $RestorePoint `
                                             -DrActivateSettings $DrActivateSettings
```

4.2.2 Modify a predefined DR plan

The **Set-DellDrScReplicationRestorePointPredefine** cmdlet can be used to modify a predefined DR plan for a replicated volume. This sample script will modify the predefined DR plan for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13**. The predefined DR plan will be modified to recover the volume from the last frozen snapshot and map it to server **SQLProd** as a read-only volume.

```
# Assign variables

$ConnName            = "DSMDC"
$SrcScName           = "SC 12"
$SrcVolumeName       = "SQLBackup2"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstScName           = "SC 13"
$NewDRServerName     = "SQLProd"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Save restore points

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Get the new DR server

$NewDRServer = Get-DellScServer -ConnectionName $ConnName `
                                -StorageCenter  $DstStorageCenter `
                                -Name           $NewDRServerName

# Get the existing DR activation settings

$DrActivateSettings = Get-DellDrScReplicationRestorePointDrActivateSettingAssociation `
                                                                -ConnectionName $ConnName `
                                                                -Instance       $RestorePoint
```

```
# Modify the DR server
$DrActivateSettings.ServerList = $NewDRServer
# Modify to use the last frozen snapshot (Replay)
$DrActivateSettings.UseActiveReplay = $false
# Modify the advanced mapping options to map the volume as read-only
$AdvancedMapping = $DrActivateSettings.AdvancedMapping
$AdvancedMapping.ReadOnly = $true
$DrActivateSettings.AdvancedMapping = $AdvancedMapping
# Predefine disaster recovery for the restore point
Set-DellDrScReplicationRestorePointPredefine -ConnectionName     $ConnName `
                                             -Instance           $RestorePoint `
                                             -DrActivateSettings $DrActivateSettings
```

4.2.3 Remove a predefined DR plan

There is not a cmdlet to remove a predefined DR plan. However, if you save and validate restore points after deleting the restore point, a new restore point will be generated without a predefined DR plan. This effectively removes the predefined DR plan. This sample script will remove the predefined DR plan for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13**.

```
# Assign variables
$ConnName            = "DSMDC"
$SrcScName           = "SC 12"
$SrcVolumeName       = "SQLBackup2"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstScName           = "SC 13"
# Get the source Storage Center
$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName
# Get the source volume
$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName
# Get the destination Storage Center
$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName
# Save restore points
Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false
# Validate restore points
Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false
```

```
# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Remove the restore point

Remove-DellDrScReplicationRestorePoint -ConnectionName $ConnName `
                                       -Instance       $RestorePoint `
                                       -Confirm:$false

# Save restore points, which will recreate the restore point without a predefined DR plan

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false
```

4.3 Test DR activations

The PowerShell SDK can be used to test the DR activation of replicated volumes. The **Start-DellDrScReplicationRestorePointActivate** cmdlet is used to initiate a test DR activation of a replicated volume. When testing a DR activation, it is critical to use the **-Test** parameter to prevent a real DR activation. When performing a test DR activation, the **-Test** parameter should be set to true.

4.3.1 Perform a test DR activation using the predefined DR plan

This sample script will test the DR activation for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13** using the predefined DR plan.

```
# Assign variables

$ConnName            = "DSMDC"
$SrcScName           = "SC 12"
$SrcVolumeName       = "SQLBackup2"
$SrcVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$DstScName           = "SC 13"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName
```

```powershell
# Save restore points

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Start the test DR activation

$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName $ConnName `
                                                           -Instance       $RestorePoint `
                                                           -Test:$true `
                                                           -Confirm:$false

# Wait for the test DR activation task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.3.2 Perform a test DR activation using the active snapshot

This sample script will test the DR activation for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13** without using the predefined DR plan. This strategy can be used when a DR plan has not be predefined. The test DR volume will be recovered from the active snapshot and mapped to server **SQLDR** as a writable volume using the next available LUN. The test DR volume will be named **Test DR of SQLBackup2**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLBackup2"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstScName            = "SC 13"
$DRServerName         = "SQLDR"
$DRVolumeName         = "Test DR of " + $SrcVolumeName

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Save restore points

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Get the DR server

$DRServer = Get-DellScServer -ConnectionName $ConnName `
                             -StorageCenter  $DstStorageCenter `
                             -Name           $DRServerName

# Define the advanced mapping options

$AdvancedMapping = New-DellScAdvancedMapping -ReadOnly:$false `
                                             -UseNextAvailable:$true
```

```
# Define the activation settings

$DRActivateSettings = New-DellDrActivateSettings -Name              $DRVolumeName `
                                                 -ServerList        $DRServer `
                                                 -AdvancedMapping   $AdvancedMapping `
                                                 -UseActiveReplay:$true `
                                                 -UseOriginalVolumesFolder:$true

# Start the test DR activation

$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName $ConnName `
                                                           -Instance           $RestorePoint `
                                                           -DrActivateSettings $DrActivateSettings `
                                                           -Test:$true `
                                                           -Confirm:$false

# Wait for the test DR activation task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.3.3 Perform a test DR activation using the latest frozen snapshot

This sample script will test the DR activation for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13** without using the predefined DR plan. This strategy can be used when a DR plan has not been predefined. The test DR volume will be recovered from the latest frozen snapshot and mapped to server **SQLDR** as a writable volume using the next available LUN. The test DR volume will be named **Test DR of SQLBackup2**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLBackup2"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstScName            = "SC 13"
$DRServerName         = "SQLDR"
$DRVolumeName         = "Test DR of " + $SrcVolumeName

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName
```

```
# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Save restore points

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Get the latest frozen snapshot (Replay)

$Replay = Get-DellScVolumeLastFrozenReplayAssociation -ConnectionName $ConnName `
                                                      -Instance $RestorePoint.DestinationVolume

# Get the DR server

$DRServer = Get-DellScServer -ConnectionName $ConnName `
                             -StorageCenter  $DstStorageCenter `
                             -Name           $DRServerName

# Define the advanced mapping options

$AdvancedMapping = New-DellScAdvancedMapping -ReadOnly:$false `
                                             -UseNextAvailable:$true

# Define the activation settings

$DrActivateSettings = New-DellDrActivateSettings -Name            $DRVolumeName `
                                                 -ServerList      $DRServer `
                                                 -AdvancedMapping $AdvancedMapping `
                                                 -UseActiveReplay:$false `
                                                 -Replay          $Replay `
                                                 -UseOriginalVolumesFolder:$true

# Start the test DR activation

$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName    $ConnName `
                                                           -Instance          $RestorePoint `
                                                           -DrActivateSettings $DrActivateSettings `
                                                           -Test:$true `
                                                           -Confirm:$false
```

```
# Wait for the test DR activation task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.3.4 Perform a test DR activation using a specific frozen snapshot

This sample script will test the DR activation for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13** without using the predefined DR plan. This strategy can be used when a DR plan has not been predefined. The test DR volume will be recovered from the frozen snapshot with a freeze time of **04/18/17 7:42:44AM** and mapped to server **SQLDR** as a writable volume using the next available LUN. The test DR volume will be named **Test DR of SQLBackup2**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLBackup2"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstScName            = "SC 13"
$DRServerName         = "SQLDR"
$DRVolumeName         = "Test DR of " + $SrcVolumeName
$ReplayFreezeTime     = [datetime]"04/18/17 7:42:44AM"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Save restore points

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false
```

```powershell
# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $SrcStorageCenter `
                                   -Confirm:$false

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -SourceVolume             $SrcVolume `
                                                    -DestinationStorageCenter $DstStorageCenter

# Get the snapshot (Replay)

$Replay = Get-DellScVolumeReplayListAssociation -ConnectionName $ConnName `
                                                -Instance $RestorePoint.DestinationVolume `
          | Where-Object { $_.FreezeTime -eq $ReplayFreezeTime -and $_.Active -eq $false
}

# Get the DR server

$DRServer = Get-DellScServer -ConnectionName $ConnName `
                             -StorageCenter  $DstStorageCenter `
                             -Name           $DRServerName

# Define the advanced mapping options

$AdvancedMapping = New-DellScAdvancedMapping -ReadOnly:$false `
                                             -UseNextAvailable:$true

# Define the activation settings

$DRActivateSettings = New-DellDrActivateSettings -Name            $DRVolumeName `
                                                 -ServerList      $DRServer `
                                                 -AdvancedMapping $AdvancedMapping `
                                                 -UseActiveReplay:$false `
                                                 -Replay          $Replay `
                                                 -UseOriginalVolumesFolder:$true

# Start the test DR activation

$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName $ConnName `
                                                           -Instance           $RestorePoint `
                                                           -DrActivateSettings $DrActivateSettings `
                                                           -Test:$true `
                                                           -Confirm:$false

# Wait for the test DR activation task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.3.5 Delete a test DR volume

The **Remove-DellDrTestActivatedScVolume** cmdlet can be used to delete the volume created by a test DR activation. This sample script will remove volume **Test DR of SQLBackup2** which was created by a test DR activation for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13**.

```
# Assign variables

$ConnName            = "DSMDC"
$SrcScName           = "SC 12"
$DstScName           = "SC 13"
$DstVolumeName       = "Test DR of SQLBackup2"
$DstVolumeFolderPath = "Production/SQLDatabase/SQLProd/"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Get the DR test volume

$DstVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $DstStorageCenter `
                              -VolumeFolderPath $DstVolumeFolderPath `
                              -Name             $DstVolumeName

# Get the test activated volume object

$TestActivatedVolume = Get-DellDrTestActivatedScVolume -ConnectionName $ConnName `
                                                       -SourceStorageCenter      $SrcStorageCenter `
                                                       -DestinationStorageCenter $DstStorageCenter `
                                                       -TestDrActivatedVolume    $DstVolume

# Remove the test DR volume

Remove-DellDrTestActivatedScVolume -ConnectionName $ConnName `
                                   -Instance       $TestActivatedVolume `
                                   -Confirm:$false
```

4.4 DR activations

The PowerShell SDK can be used to automate the DR activation of replicated volumes. The **Start-DellDrScReplicationRestorePointActivate** cmdlet is used to initiate a DR activation of a replicated volume. When performing a real DR activation, instead of a test DR activation, the **-Test** parameter should be set to false.

4.4.1 Perform a DR activation using the predefined DR plan

This sample script will activate DR for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13** using the predefined DR plan.

```
# Assign variables

$ConnName      = "DSMDC"
$SrcScName     = "SC 12"
$SrcVolumeName = "SQLBackup2"
$DstScName     = "SC 13"
```

```
# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Validate restore points

Test-DellDrScReplicationRestorePointAll -ConnectionName $ConnName `
                                        -Confirm:$false

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -DestinationStorageCenter $DstStorageCenter `
                | Where-Object { $_.SourceVolume.InstanceName -eq $SrcVolumeName }

# Start the DR activation

$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName $ConnName `
                                                           -Instance      $RestorePoint `
                                                           -Test:$false `
                                                           -Confirm:$false

# Wait for the DR activation task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.4.2 Perform a DR activation using the active snapshot

This sample script will activate DR for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13** without using the predefined DR plan. This strategy can be used when a DR plan has not be predefined. The DR volume will be recovered from the active snapshot and mapped to server **SQLDR** as a writable volume using the next available LUN. The DR volume will be named **DR of SQLBackup2**.

```
# Assign variables

$ConnName      = "DSMDC"
$SrcScName     = "SC 12"
$SrcVolumeName = "SQLBackup2"
$DstScName     = "SC 13"
$DRServerName  = "SQLDR"
$DRVolumeName  = "DR of " + $SrcVolumeName
```

```powershell
# Get the source Storage Center
$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the destination Storage Center
$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Validate restore points
Test-DellDrScReplicationRestorePointAll -ConnectionName $ConnName `
                                        -Confirm:$false

# Get the restore point
$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -DestinationStorageCenter $DstStorageCenter `
                | Where-Object { $_.SourceVolume.InstanceName -eq $SrcVolumeName }

# Get the DR server
$DRServer = Get-DellScServer -ConnectionName $ConnName `
                             -StorageCenter  $DstStorageCenter `
                             -Name           $DRServerName

# Define the advanced mapping options
$AdvancedMapping = New-DellScAdvancedMapping -ReadOnly:$false `
                                             -UseNextAvailable:$true

# Define the activation settings
$DRActivateSettings = New-DellDrActivateSettings -Name            $DRVolumeName `
                                                 -ServerList      $DRServer `
                                                 -AdvancedMapping $AdvancedMapping `
                                                 -UseActiveReplay:$true `
                                                 -UseOriginalVolumesFolder:$true

# Start the DR activation
$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName $ConnName `
                                                           -Instance           $RestorePoint `
                                                           -DrActivateSettings $DrActivateSettings `
                                                           -Test:$false `
                                                           -Confirm:$false

# Wait for the DR activation task to finish
$LoopCount = 0;
While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++
    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"
    Start-Sleep -Seconds 2
    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.4.3 Perform a DR activation using the latest frozen snapshot

This sample script will activate DR for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13** without using the predefined DR plan. This strategy can be used when a DR plan has not be predefined. The DR volume will be recovered from the latest frozen snapshot and mapped to server **SQLDR** as a writable volume using the next available LUN. The DR volume will be named **DR of SQLBackup2**.

```
# Assign variables

$ConnName       = "DSMDC"
$SrcScName      = "SC 12"
$SrcVolumeName  = "SQLBackup2"
$DstScName      = "SC 13"
$DRServerName   = "SQLDR"
$DRVolumeName   = "DR of " + $SrcVolumeName

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Validate restore points

Test-DellDrScReplicationRestorePointAll -ConnectionName $ConnName `
                                        -Confirm:$false

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -DestinationStorageCenter $DstStorageCenter `
                | Where-Object { $_.SourceVolume.InstanceName -eq $SrcVolumeName }

# Get the latest frozen snapshot (Replay)

$Replay = Get-DellScVolumeLastFrozenReplayAssociation -ConnectionName $ConnName `
                                                      -Instance $RestorePoint.DestinationVolume

# Get the DR server

$DRServer = Get-DellScServer -ConnectionName $ConnName `
                             -StorageCenter  $DstStorageCenter `
                             -Name           $DRServerName

# Define the advanced mapping options

$AdvancedMapping = New-DellScAdvancedMapping -ReadOnly:$false `
                                             -UseNextAvailable:$true

# Define the activation settings

$DRActivateSettings = New-DellDrActivateSettings -Name             $DRVolumeName `
                                                 -ServerList       $DRServer `
                                                 -AdvancedMapping  $AdvancedMapping `
                                                 -UseActiveReplay:$false `
                                                 -Replay           $Replay `
                                                 -UseOriginalVolumesFolder:$true
```

```
# Start the DR activation

$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName $ConnName `
                                                           -Instance           $RestorePoint `
                                                           -DrActivateSettings $DrActivateSettings `
                                                           -Test:$false `
                                                           -Confirm:$false

# Wait for the DR activation task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.4.4 Perform a DR activation using a specific frozen snapshot

This sample script will activate DR for the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13** without using the predefined DR plan. This strategy can be used when a DR plan has not be predefined. The DR volume will be recovered from the frozen snapshot with a freeze time of **04/18/17 7:42:44AM** and mapped to server **SQLDR** as a writable volume using the next available LUN. The DR volume will be named **DR of SQLBackup2**.

```
# Assign variables

$ConnName          = "DSMDC"
$SrcScName         = "SC 12"
$SrcVolumeName     = "SQLBackup2"
$DstScName         = "SC 13"
$DRServerName      = "SQLDR"
$DRVolumeName      = "DR of " + $SrcVolumeName
$ReplayFreezeTime = [datetime]"04/18/17 7:42:44AM"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Validate restore points

Test-DellDrScReplicationRestorePointAll -ConnectionName $ConnName `
                                        -Confirm:$false
```

```
# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -DestinationStorageCenter $DstStorageCenter `
                | Where-Object { $_.SourceVolume.InstanceName -eq $SrcVolumeName }

# Get the frozen snapshot (Replay)

$Replay = Get-DellScVolumeReplayListAssociation -ConnectionName $ConnName `
                                                -Instance $RestorePoint.DestinationVolume `
          | Where-Object { $_.FreezeTime -eq $ReplayFreezeTime -and $_.Active -eq $false
}

# Get the DR server

$DRServer = Get-DellScServer -ConnectionName $ConnName `
                             -StorageCenter  $DstStorageCenter `
                             -Name           $DRServerName

# Define the advanced mapping options

$AdvancedMapping = New-DellScAdvancedMapping -ReadOnly:$false `
                                             -UseNextAvailable:$true

# Define the activation settings

$DRActivateSettings = New-DellDrActivateSettings -Name            $DRVolumeName `
                                                 -ServerList      $DRServer `
                                                 -AdvancedMapping $AdvancedMapping `
                                                 -UseActiveReplay:$false `
                                                 -Replay          $Replay `
                                                 -UseOriginalVolumesFolder:$true

# Start the DR activation

$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName $ConnName `
                                                           -Instance           $RestorePoint `
                                                           -DrActivateSettings $DrActivateSettings `
                                                           -Test:$false `
                                                           -Confirm:$false

# Wait for the DR activation task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.4.5 Replicate back to the source volume after a DR activation

The **Restore-DellDrScReplicationRestorePoint** cmdlet can be used to reverse replication after a DR activation. This sample script will reverse the replication of volume **SQLBackup2** between Storage Centers **SC 12** and **SC 13** after it has been DR activated. After the script has run, the DR volume on Storage Center **SC 13** will be replicating to the original source volume on Storage Center **SC 12**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "SQLBackup2"
$SrcVolumeFolderPath  = "Production/SQLDatabase/SQLProd/"
$DstScName            = "SC 13"
$RestorePointState    = [DellStorage.Api.Enums.ReplicationRestorePointStateEnum] `
                        "DrActivated"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceVolume     $SrcVolume `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -DestinationStorageCenter $DstStorageCenter `
                                                    -State                    $RestorePointState

# Define the restore settings

$RestoreSettings = New-DellDrRestoreSettings -MirrorBackOnly:$true

# Replicate back to the original source volume

$AsyncTask = Restore-DellDrScReplicationRestorePoint -ConnectionName  $ConnName `
                                                     -Instance        $RestorePoint `
                                                     -RestoreSettings $RestoreSettings `
                                                     -Confirm:$false
```

```
# Wait for the restore task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}

# Save restore points

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $DstStorageCenter `
                                   -Confirm:$false

# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $DstStorageCenter `
                                   -Confirm:$false
```

4.5 Live Volumes

The Dell Storage PowerShell SDK can be used to automate the DR activation of Live Volume in the event the primary volume is unavailable. The **Start-DellDrScReplicationRestorePointActivate** cmdlet is used to initiate a DR activation of a Live Volume.

4.5.1 Recover a Live Volume using the active snapshot

This sample script will activate DR for the volume **SQLBackup2** that is configured as a Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary). The Live Volume will be recovered from the active snapshot, keeping the Live Volume intact. The recovered volume will be named **DR of SQLBackup2**.

```
# Assign variables

$ConnName      = "DSMDC"
$PriScName     = "SC 12"
$PriVolumeName = "Backup"
$SecScName     = "SC 13"
$DRVolumeName  = "DR of " + $PriVolumeName

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $PriScName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SecScName
```

```powershell
# Validate restore points

Test-DellDrScReplicationRestorePointAll -ConnectionName $ConnName `
                                        -Confirm:$false

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $PriStorageCenter `
                                                    -DestinationStorageCenter $SecStorageCenter `
                                                    -LiveVolume:$true `
                | Where-Object { $_.SourceVolume.InstanceName -eq $PriVolumeName }

# Define the activation settings

$DRActivateSettings = New-DellDrActivateSettings -Name $DRVolumeName `
                                                 -UseActiveReplay:$true `
                                                 -PreserveLiveVolume:$true

# Start the DR activation

$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName $ConnName `
                                                           -Instance           $RestorePoint `
                                                           -DrActivateSettings $DrActivateSettings `
                                                           -Test:$false `
                                                           -Confirm:$false

# Wait for the DR activation task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.5.2 Recover a Live Volume using the latest frozen snapshot

This sample script will activate DR for the volume **SQLBackup2** that is configured as a Live Volume between Storage Centers **SC 12** (primary) and **SC 13** (secondary). The Live Volume will be recovered from the latest frozen snapshot, keeping the Live Volume intact. The recovered volume will be named **DR of SQLBackup2**.

```
# Assign variables

$ConnName       = "DSMDC"
$PriScName      = "SC 12"
$PriVolumeName = "Backup"
$SecScName      = "SC 13"
$DRVolumeName   = "DR of " + $PriVolumeName

# Get the primary Storage Center

$PriStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $PriScName

# Get the secondary Storage Center

$SecStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SecScName

# Validate restore points

Test-DellDrScReplicationRestorePointAll -ConnectionName $ConnName `
                                        -Confirm:$false

# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceStorageCenter      $PriStorageCenter `
                                                    -DestinationStorageCenter $SecStorageCenter `
                                                    -LiveVolume:$true `
                | Where-Object { $_.SourceVolume.InstanceName -eq $PriVolumeName }

# Define the activation settings

$DRActivateSettings = New-DellDrActivateSettings -Name $DRVolumeName `
                                                 -UseActiveReplay:$false `
                                                 -PreserveLiveVolume:$true

# Start the DR activation

$AsyncTask = Start-DellDrScReplicationRestorePointActivate -ConnectionName $ConnName `
                                                           -Instance           $RestorePoint `
                                                           -DrActivateSettings $DrActivateSettings `
                                                           -Test:$false `
                                                           -Confirm:$false
```

```
# Wait for the DR activation task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}
```

4.5.3 Restore a Live Volume after DR activation

The **Restore-DellDrScReplicationRestorePoint** cmdlet can be used to restore a Live Volume after a DR activation. This sample script will restore the Live Volume for the volume **SQLBackup2** after it has been DR activated. This volume was configured as a Live Volume between Storage Centers **SC 12** and **SC 13**. After the script has run, the Live Volume will be restored and data will be replicating between Storage Centers **SC 12** and **SC 13**.

```
# Assign variables

$ConnName             = "DSMDC"
$SrcScName            = "SC 12"
$SrcVolumeName        = "Backup"
$SrcVolumeFolderPath  = "Production/FileServer/SalesFS01/"
$DstScName            = "SC 13"
$RestorePointState    = [DellStorage.Api.Enums.ReplicationRestorePointStateEnum] `
                        "DrActivated"

# Get the source Storage Center

$SrcStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $SrcScName

# Get the source volume

$SrcVolume = Get-DellScVolume -ConnectionName   $ConnName `
                              -StorageCenter    $SrcStorageCenter `
                              -VolumeFolderPath $SrcVolumeFolderPath `
                              -Name             $SrcVolumeName

# Get the destination Storage Center

$DstStorageCenter = Get-DellStorageCenter -ConnectionName $ConnName `
                                          -Name           $DstScName

# Validate restore points

Test-DellDrScReplicationRestorePointAll -ConnectionName $ConnName `
                                        -Confirm:$false
```

```powershell
# Get the restore point

$RestorePoint = Get-DellDrScReplicationRestorePoint -ConnectionName   $ConnName `
                                                    -SourceVolume     $SrcVolume `
                                                    -SourceStorageCenter      $SrcStorageCenter `
                                                    -DestinationStorageCenter $DstStorageCenter `
                                                    -State                    $RestorePointState

# Define the restore settings

$RestoreSettings = New-DellDrRestoreSettings -RecoverLiveVolume:$true

# Restore the Live Volume

$AsyncTask = Restore-DellDrScReplicationRestorePoint -ConnectionName  $ConnName `
                                                     -Instance        $RestorePoint `
                                                     -RestoreSettings $RestoreSettings `
                                                     -Confirm:$false

# Wait for the restore task to finish

$LoopCount = 0;

While ( ( "Finished", "Error", "Canceled", "BeingCanceled" ) `
        -notcontains `
        $AsyncTask.State.Name )
{
    $LoopCount++

    Write-Host "Waiting for activation to complete$( '.' * $LoopCount )"

    Start-Sleep -Seconds 2

    $AsyncTask = Get-DellDrReplicationAsyncTask -ConnectionName $ConnName `
                                                -Instance       $AsyncTask
}

# Save restore points

Save-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $DstStorageCenter `
                                   -Confirm:$false

# Validate restore points

Test-DellStorageCenterRestorePoint -ConnectionName $ConnName `
                                   -Instance       $DstStorageCenter `
                                   -Confirm:$false
```

5 Working with FluidFS

The Dell Storage PowerShell SDK cmdlets that interact directly with FluidFS will have the prefix **DellFluidFS** in the noun portion of the cmdlet. For example, the cmdlet that returns FluidFS NAS volumes is called **Get-DellFluidFsNasVolume**. The following sample script will return the cmdlets that interact with FluidFS.

```
# Get the cmdlets that interact with FluidFS

Get-Command -Module "DellStorage.ApiCommandSet" `
| Where-Object { $_.name -like "*DellFluidFS*" } `
| Sort-Object Name `
| Select-Object Name
```

DSM can manage multiple FluidFS clusters from the same Data Collector, so it is important to specify the Fluid FS cluster when executing the PowerShell SDK cmdlets.

NAS volume folder names are not required to be unique. When using a PowerShell cmdlet to get a NAS volume folder by name, it is highly recommended to add additional logic to verify that only one item is returned. To make the sample scripts easier to read, that logic is not included.

The sample scripts in this section assume the Dell Storage PowerShell SDK module has already been imported and a saved connection named **DSMDC** has already been created.

5.1 NAS volume folders

The Dell Storage PowerShell SDK provides cmdlets to manage FluidFS NAS volume folders. Table 1 shows the NAS volume folder cmdlets available in the Dell Storage PowerShell SDK.

Table 16 NAS volume folder cmdlets

PowerShell SDK cmdlet
Get-DellFluidFsNasVolumeFolder
New-DellFluidFsNasVolumeFolder
Remove-DellFluidFsNasVolumeFolder
Set-DellFluidFsNasVolumeFolder

5.1.1 Create a NAS volume folder

The **New-DellFluidFsNasVolumeFolder** cmdlet can be used to create a volume folder. This sample script creates a NAS volume folder named **FolderX** in the root folder on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName       = "DSMDC"
$FsClusterName  = "fluidfs2D"
$ParentFolderID = 0
$FolderName     = "FolderX"

# Get the FluidFS Cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Create the volume folder

$VolumeFolder = New-DellFluidFsNasVolumeFolder -ConnectionName $ConnName `
                                               -ClusterId      $FsCluster.InstanceId `
                                               -ParentId       $ParentFolderID `
                                               -Name           $FolderName
```

5.1.2 Get a NAS volume folder

The **Get-DellFluidFsNasVolumeFolder** cmdlet can be used to get a NAS volume folder. This sample script shows how to get the NAS volume folder **FolderX** in the root folder and how to get all NAS volume folders on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName       = "DSMDC"
$FsClusterName  = "fluidfs2D"
$ParentFolderID = 0
$FolderName     = "FolderX"

# Get the FluidFS Cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume folder

$Folder = Get-DellFluidFsNasVolumeFolder -ConnectionName $ConnName `
                                         -ClusterId      $FsCluster.InstanceId `
                                         -ParentId       $ParentFolderID `
                                         -Name           $FolderName

# Show information about all NAS volume folders

$Folder | Select-Object Name

# Get all NAS volume folders

$FolderList = @( Get-DellFluidFsNasVolumeFolder -ConnectionName $ConnName `
                                                -ClusterId      $FsCluster.InstanceId )

# Show information about the NAS volume folders

$FolderList | Select-Object Name
```

5.1.3 Rename a NAS volume folder

The **Set-DellFluidFsNasVolumeFolder** cmdlet can be used to rename a NAS volume folder. This sample script will rename the NAS volume folder **FolderX** in the root folder on the FluidFS cluster **fluidfs2D** to **FolderX1**.

```
# Assign variables

$ConnName       = "DSMDC"
$FsClusterName  = "fluidfs2D"
$ParentFolderId = 0
$FolderName     = "FolderX"
$NewFolderName  = "FolderX1"

# Get the FluidFS Cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS Volume folder

$Folder = Get-DellFluidFsNasVolumeFolder -ConnectionName $ConnName `
                                         -ClusterId      $FsCluster.InstanceId `
                                         -ParentId       $ParentFolderID `
                                         -Name           $FolderName

# Rename the NAS volume folder

$Folder = Set-DellFluidFsNasVolumeFolder -ConnectionName $ConnName `
                                         -Instance       $Folder `
                                         -Name           $NewFolderName
```

5.1.4 Remove a NAS volume folder

The **Remove-DellFluidFsNasVolumeFolder** cmdlet can be used to remove a NAS volume folder. This sample script removes the NAS volume folder **FolderX1** from the root folder on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName       = "DSMDC"
$FsClusterName  = "fluidfs2D"
$ParentFolderId = 0
$FolderName     = "FolderX1"

# Get the FluidFS Cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume folder

$Folder = Get-DellFluidFsNasVolumeFolder -ConnectionName $ConnName `
                                         -ClusterId      $FsCluster.InstanceId `
                                         -ParentId       $ParentFolderId `
                                         -Name           $FolderName

# Remove the NAS volume folder

Remove-DellFluidFsNasVolumeFolder -ConnectionName  $ConnName `
                                  -Instance        $Folder `
                                  -Confirm:$false
```

5.2 NAS volumes

The Dell Storage PowerShell SDK provides cmdlets to manage FluidFS NAS volumes. Table 4 shows the NAS volume cmdlets available in the Dell Storage PowerShell SDK.

Table 17 NAS volume cmdlets

PowerShell SDK cmdlet
Get-DellFluidFsNasVolume
New-DellFluidFsNasVolume
Set-DellFluidFsNasVolume
Remove-DellFluidFsNasVolume

5.2.1 Create a NAS volume

The **New-DellFluidFsNasVolume** cmdlet can be used to create a new NAS volume. This sample script will create a 100GB NAS volume named **newvol** in the NAS volume folder **production** on the FluidFS cluster **fluidfs2D**. The volume will use the default settings.

```
# Assign variables

$ConnName       = "DSMDC"
$FsClusterName  = "fluidfs2D"
$FolderName     = "production"
$NasVolumeName  = "newvol"
$NasVolumeSize  = "100GB"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume folder

$Folder = Get-DellFluidFsNasVolumeFolder -ConnectionName $ConnName `
                                         -ClusterId      $FsCluster.InstanceId `
                                         -Name           $FolderName

# Create the NAS Volume

$NasVolume = New-DellFluidFsNasVolume -ConnectionName    $ConnName `
                                      -ClusterId         $FsCluster.InstanceId`
                                      -NasVolumeFolderId $Folder.FolderId `
                                      -Name              $NasVolumeName `
                                      -Size              $NasVolumeSize
```

5.2.2 Get a NAS volume

The **Get-DellFluidFsNasVolume** cmdlet can be used to get a NAS volume. This sample script will get the NAS volume **newvol** on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "newvol"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Show information about the NAS volume

$NasVolume | Select-Object Name
```

5.2.3 Modify a NAS volume

The **Set-DellFluidFsNasVolume** cmdlet can be used to modify a NAS volume. This sample script shows how to modify the NAS volume **newvol** on the FluidFS cluster **fluidfs2D**, setting the interoperability policy to **Unix** and changing the default Unix permissions to full control (777).

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "newvol"
$InteropPolicy = "Unix"
$DefaultPerm   = "777"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Modify the NAS Volume

$NasVolume = Set-DellFluidFsNasVolume -ConnectionName             $ConnName `
                                      -Instance                   $NasVolume `
                                      -InteroperabilityPolicy     $InteropPolicy `
                                      -DefaultUnixFilePermissions $DefaultPerm
```

5.2.4 Rename a NAS volume

The **Set-DellFluidFsNasVolume** cmdlet can be used to rename a NAS volume. This sample script will rename the NAS volume **newvol** on the FluidFS cluster **fluidfs2D** to **sales**.

```
# Assign variables

$ConnName          = "DSMDC"
$FsClusterName     = "fluidfs2D"
$NasVolumeName     = "newvol"
$NewNasVolumeName  = "salesvol"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Rename the NAS Volume

$NasVolume = Set-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -Instance       $NasVolume `
                                      -Name           $NewNasVolumeName
```

5.2.5 Remove a NAS volume

The **Remove-DellFluidFsNasVolume** cmdlet can be used to remove a NAS volume. This sample script will remove the **salesvol2** volume on the FluidFS cluster **fluidfs2D**. Before removing NAS volumes, make sure that the volume does not include any CIFS/NFS shares and no active policies are configured (replication/snapshots).

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "salesvol"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Remove the NAS Volume

Remove-DellFluidFsNasVolume -ConnectionName $ConnName `
                            -Instance       $NasVolume `
                            -Confirm:$false
```

5.3 NFS exports

The Dell Storage PowerShell SDK provides cmdlets to manage NFS exports on Fluid FS NAS volumes. Table 18 shows the NFS export cmdlets available in the Dell Storage PowerShell SDK.

Table 18 NFS export cmdlets

PowerShell SDK cmdlet
Get-DellFluidFsNfsExport
New-DellFluidFsNfsExport New-DellFluidFsNfsExportAccessDetails
Set-DellFluidFsNfsExport
Remove-DellFluidFsNfsExport

5.3.1 Create an NFS export

The **New-DellFluidFsNfsExport** cmdlet can be used to create a new NFS export on a NAS volume. This sample script will create an NFS export for the **/engtest** directory on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. The export directory must exist before running the script.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"
$FolderPath    = "/engtest"

# Get FluidFS the cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId`
                                      -Name           $NasVolumeName

# Create the NAS Volume NFS export

$NfsExport = New-DellFluidFsNfsExport -ConnectionName  $ConnName `
                                      -ClusterId       $FsCluster.InstanceId `
                                      -NasVolumeId     $NasVolume.NasVolumeId `
                                      -FolderPath      $FolderPath
```

5.3.2 Grant access to an NFS export

The **New-DellFluidFsNfsExportAccessDetails** cmdlet can be used to control access to an NFS export. This sample script will create an NFS export for the **/engtest** directory on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. Root access will be allowed for clients connecting from the IP address 172.16.30.25.

```
# Assign variables

$ConnName        = "DSMDC"
$FsClusterName   = "fluidfs2D"
$NasVolumeName   = "engvol"
$FolderPath      = "/engtest"
$ExportTo        = [DellStorage.Api.Enums.FluidFsExportToEnum] "OneClient"
$ExportToClients = "172.16.30.25"
$ReadWrite       = $true
$TrustUsers      = [DellStorage.Api.Enums.FluidFsTrustUsersEnum] "NoRootSquash"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Create NFS access for the NFS export

$AccessDetails = New-DellFluidFsNfsExportAccessDetails -ExportTo       $ExportTo `
                                                       -ExportToClients $ExportToClients `
                                                       -ReadWrite       $ReadWrite `
                                                       -TrustUsers      $TrustUsers

# Create NAS Volume NFS export

$NfsExport = New-DellFluidFsNfsExport -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -NasVolumeId    $NasVolume.NasVolumeId `
                                      -FolderPath     $FolderPath `
                                      -AccessDetails  $AccessDetails
```

5.3.3 Get an NFS export

The **Get-DellFluidFSNfsExport** cmdlet can be used to get an NFS export. This sample script gets all NFS exports on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get all NFS exports

$NfsExportList = @( Get-DellFluidFsNfsExport -ConnectionName $ConnName `
                                             -ClusterId      $FsCluster.InstanceId )
```

```
# Show information about all NFS exports

$NfsExportList | Select-Object InstanceName
```

5.3.4 Remove an NFS export

The **Remove-DellFluidFsNfsExport** cmdlet can be used to remove a NFS export. This sample script removes the NFS export **/engtest** on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NfsExportName = "/engtest"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NFS export

$NfsExport = Get-DellFluidFsNfsExport -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -InstanceName   $NfsExportName

# Remove NFS export

Remove-DellFluidFsNfsExport -ConnectionName $ConnName `
                            -Instance       $NfsExport `
                            -Confirm:$false
```

5.4 SMB shares

The Dell Storage PowerShell SDK provides cmdlets to manage SMB shares on Fluid FS NAS volumes. Table 19 shows the SMB share cmdlets available in the Dell Storage PowerShell SDK.

Table 19 SMB share cmdlets

PowerShell SDK cmdlet
Get-DellFluidFsSmbShare Get-DellFluidFsSmbHomeShare
New-DellFluidFsSmbShare New-DellFluidFsSmbShareExtensionsFilterExcludeGroup New-DellFluidFsSmbHomeShareExtensionsFilterExcludeGroup
Set-DellFluidFsSmbShare Set-DellFluidFsSmbHomeShare
Remove-DellFluidFsSmbShare

5.4.1 Create an SMB share

The **New-DellFluidFsSmbShare** cmdlet can be used to create a SMB share on a NAS volume. This sample script creates the SMB share **SMBshare1** on the **/share1** directory, on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"
$FolderPath    = "/share1"
$SmbShareName  = "SMBShare1"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Create an SMB share on the NAS Volume

$SmbShare = New-DellFluidFsSmbShare -ConnectionName $ConnName `
                                    -ClusterId      $FsCluster.InstanceId `
                                    -NasVolumeId    $NasVolume.NasVolumeId `
                                    -Path           $FolderPath `
                                    -ShareName      $SmbShareName
```

5.4.2 Get an SMB share

The **Get-DellFluidFsSmbShare** cmdlet can be used to get an SMB share on a NAS volume on a FluidFS cluster. This sample script shows how to get the the SMB share **SMBShare1** on the FluidFS cluster **fluidfs2D**. This sample script also shows how to get all SMB shares on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$SmbShareName  = "SMBShare1"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the SMB share

$SmbShare = Get-DellFluidFsSmbShare -ConnectionName  $ConnName `
                                    -ClusterId       $FsCluster.InstanceId `
                                    -ShareName       $SmbShareName

# Show information about the SMB share

$SmbShare | Select-Object ShareName

# Get all SMB shares

$SmbShareList = @( Get-DellFluidFsSmbShare -ConnectionName $ConnName `
                                           -ClusterId      $FsCluster.InstanceId )
```

```
# Show information about all SMB shares

$SmbShareList | Select-Object ShareName
```

5.4.3 Modify an SMB share

The **Set-DellFluidFsSmbShare** cmdlet can be used to modify an SMB share. This sample script enables Microsoft access based enumeration (ABE) on the SMB share **SMBshare1**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$SmbShareName  = "SMBShare1"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the SMB Share

$SmbShare = Get-DellFluidFsSmbShare -ConnectionName  $ConnName `
                                    -ClusterId       $FsCluster.InstanceId `
                                    -ShareName       $SmbShareName

# Enable Microsoft ABE on the SMB share

$SmbShare = Set-DellFluidFsSmbShare -ConnectionName         $ConnName `
                                    -Instance               $SmbShare `
                                    -AccessBasedEnumeration:$true
```

5.4.4 Remove an SMB share

The **Remove-DellFluidFsSmbShare** cmdlet can be used to remove a SMB share. This sample script removes the **SMBshare1** share from the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$SmbShareName  = "SMBShare1"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the SMB share

$SmbShare = Get-DellFluidFsSmbShare -ConnectionName  $ConnName `
                                    -ClusterId       $FsCluster.InstanceId `
                                    -ShareName       $SmbShareName

# Remove the SMB share

Remove-DellFluidFsSmbShare -ConnectionName $ConnName `
                           -Instance       $SmbShare `
                           -Confirm:$false
```

5.5 NAS volume snapshots

The Dell Storage PowerShell SDK provides cmdlets to manage FluidFS NAS volume snapshots. Table 20 shows the NAS volume snapshots cmdlets available in the Dell Storage PowerShell SDK.

Table 20 NAS volume snapshot cmdlets

PowerShell SDK cmdlet
Get-DellFluidFsSnapshot Get-DellFluidFsSnapshotSchedule Get-DellFluidFsNasVolumeSnapshotListAssociation Get-DellFluidFsNasVolumeSnapshotScheduleListAssociation
New-DellFluidFsSnapshot New-DellFluidFsSnapshotSchedule
Set-DellFluidFsSnapshot Set-DellFluidFsSnapshotSchedule
Remove-DellFluidFsSnapshot Remove-DellFluidFsSnapshotSchedule

5.5.1 Create a snapshot

The **New-DellFluidFsSnapshot** cmdlet can be used to create a snapshot for a NAS volume . This sample script will create a snapshot called **basesnap** on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. The new snapshot will have an expiration date of 8/26/2017 at 10:00:25AM.

```
# Assign variables

$ConnName       = "DSMDC"
$FsClusterName  = "fluidfs2D"
$NasVolumeName  = "engvol"
$SnapshotName   = "basesnap"
$ExpirationDate = [datetime] "08/26/2017 10:00:25AM"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Create the snapshot

$Snapshot = New-DellFluidFsSnapshot -ConnectionName    $ConnName `
                                    -ClusterId         $FsCluster.InstanceId `
                                    -NasVolumeId       $NasVolume.NasVolumeId `
                                    -Name              $SnapshotName `
                                    -Expiration        $ExpirationDate `
                                    -ExpirationEnabled:$true
```

5.5.2 Create a snapshot schedule

The **New-DellFluidFsSnapshotSchedule** cmdlet can be used to create a snapshot schedule for a NAS volume . This sample script will create a snapshot schedule named **daily1** on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. The new snapshot schedule will create a daily snapshot with a retention period of 7 days.

```
# Assign variables

$ConnName        = "DSMDC"
$FsClusterName   = "fluidfs2D"
$NasVolumeName   = "engvol"
$ScheduleName    = "daily1"
$ScheduleType    = [DellStorage.Api.Enums.FluidFsTimerScheduleEnum] "Periodic"
$PeriodInDays    = 1
$RetentionInDays = 7

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId`
                                      -Name           $NasVolumeName

# Calculate the period:  # days * 24 hours/day * 60 minutes/hour

$PeriodInMinutes = $PeriodInDays * 24 * 60

# Calculate the retention:  # days * 24 hours/day * 60 minutes/hour * 60 seconds/minute

$RetentionInSeconds = $RetentionInDays * 24 * 60 * 60

# Create the snapshot schedule on the NAS volume

$Schedule = New-DellFluidFsSnapshotSchedule -ConnectionName   $ConnName `
                                            -ClusterId        $FsCluster.InstanceId `
                                            -NasVolumeId      $NasVolume.NasVolumeId `
                                            -Name             $ScheduleName `
                                            -ScheduleType     $ScheduleType `
                                            -Period           $PeriodInMinutes `
                                            -RetentionPeriod  $RetentionInSeconds `
                                            -RetentionEnabled:$true
```

5.5.3 Get a snapshot

The **Get-DellFluidFsSnapshot** cmdlet can be used to get a snapshot on a NAS volume. This sample script shows how to get the snapshot named **basesnap** on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. This script also shows how to get all snaphshots on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"
$SnapshotName  = "basesnap"
```

```
# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get the snapshot on the NAS volume

$Snapshot = Get-DellFluidFsSnapshot -ConnectionName $ConnName `
                                    -ClusterId      $FsCluster.InstanceId `
                                    -NasVolumeId    $NasVolume.NasVolumeId `
                                    -Name           $SnapshotName

# Show information about the snapshot

$Snapshot | Select-Object Name

# Get all snapshots on the NAS volume

$SnapshotList = @( Get-DellFluidFsSnapshot -ConnectionName $ConnName `
                                           -ClusterId      $FsCluster.InstanceId `
                                           -NasVolumeId    $NasVolume.NasVolumeId )

# Show information about all snapshots

$SnapshotList | Select-Object Name
```

5.5.4 Get a snapshot schedule

The **Get-DellFluidFsSnapshotSchedule** cmdlet can be used to get a snapshot schedule for a NAS volume. This sample script shows how to get the snapshot schedule **daily** on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. This sample script also shows how to get all snapshot schedules on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"
$ScheduleName  = "daily"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get snapshot schedule for the NAS volume

$Schedule = Get-DellFluidFsSnapshotSchedule -ConnectionName $ConnName `
                                            -ClusterId      $FsCluster.InstanceId `
                                            -NasVolumeId    $NasVolume.NasVolumeId `
                                            -Name           $ScheduleName
```

```
# Show information about the snapshot schedule

$Schedule | Select-Object Name

# Get all snapshot schedules for the NAS volume

$ScheduleList = @( Get-DellFluidFsSnapshotSchedule -ConnectionName $ConnName `
                                                   -ClusterId      $FsCluster.InstanceId `
                                                   -NasVolumeId    $NasVolume.NasVolumeId )

# Show information about all snapshot schedules

$ScheduleList | Select-Object Name
```

5.5.5 Modify a snapshot

The **Set-DellFluidFsSnapshot** cmdlet can be used to modify a snapshot on a NAS volume. This sample script will set the expiration date on the snapshot **basesnap**, on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D** to **8/26/2018** at **10:00:25AM**.

```
# Assign variables

$ConnName           = "DSMDC"
$FsClusterName      = "fluidfs2D"
$NasVolumeName      = "engvol"
$SnapshotName       = "basesnap"
$NewExpirationDate = [datetime] "08/26/2018 10:00:25AM"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get the snapshot on the NAS volume

$Snapshot = Get-DellFluidFsSnapshot -ConnectionName $ConnName `
                                    -ClusterId      $FsCluster.InstanceId `
                                    -NasVolumeId    $NasVolume.NasVolumeId `
                                    -Name           $SnapshotName

# Modify the snapshot

$Snapshot = Set-DellFluidFsSnapshot -ConnectionName $ConnName `
                                    -Instance       $Snapshot `
                                    -Expiration     $NewExpirationDate
```

5.5.6 Modify a snapshot schedule

The **Set-DellFluidFsSnapshotSchedule** cmdlet can be used to modify a snapshot schedule. This sample script will modify the snapshot schedule **daily1**, on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**, to create a snapshot every 2 days.

```
# Assign variables

$ConnName        = "DSMDC"
$FsClusterName   = "fluidfs2D"
$NasVolumeName   = "engvol"
$ScheduleName    = "daily1"
$NewPeriodInDays = 2

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get the snapshot schedule on the NAS volume

$Schedule = Get-DellFluidFsSnapshotSchedule -ConnectionName $ConnName `
                                            -ClusterId      $FsCluster.InstanceId `
                                            -NasVolumeId    $NasVolume.NasVolumeId `
                                            -Name           $ScheduleName

# Calculate the new period:  # days * 24 hours/day * 60 minutes/hour

$NewPeriodInMinutes = $NewPeriodInDays * 24 * 60

# Modify the snapshot schedule

$Schedule = Set-DellFluidFsSnapshotSchedule -ConnectionName $ConnName `
                                            -Instance       $Schedule `
                                            -Period         $NewPeriodInMinutes
```

5.5.7 Remove a snapshot

The **Remove-DellFluidFsSnapshot** cmdlet can be used to remove a snapshot on a NAS volume. This sample script will remove the snapshot **basesnap**, on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"
$SnapshotName  = "basesnap"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName
```

```
# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get snapshot on the NAS volume

$Snapshot = Get-DellFluidFsSnapshot -ConnectionName $ConnName `
                                    -ClusterId      $FsCluster.InstanceId `
                                    -NasVolumeId    $NasVolume.NasVolumeId `
                                    -Name           $SnapshotName

# Remove the snapshot

Remove-DellFluidFsSnapshot -ConnectionName $ConnName `
                           -Instance       $Snapshot `
                           -Confirm:$false
```

5.5.8 Remove a snapshot schedule

The **Remove-DellFluidFsSnapshotSchedule** cmdlet can be used to remove asnapshot schedule from a NAS volume. This sample script will remove the snapshot schedule **daily1** from the NAS volume **engvol** on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"
$ScheduleName  = "daily1"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get the snapshot schedule on the NAS volume

$Schedule = Get-DellFluidFsSnapshotSchedule -ConnectionName $ConnName `
                                            -ClusterId      $FsCluster.InstanceId `
                                            -NasVolumeId    $NasVolume.NasVolumeId `
                                            -Name           $ScheduleName

# Remove the snapshot schedule

Remove-DellFluidFsSnapshotSchedule -ConnectionName $ConnName `
                                   -Instance       $Schedule `
                                   -Confirm:$false
```

5.6 NAS volume quotas

The Dell Storage PowerShell SDK provides cmdlets to manage FluidFS NAS volume quotas. Table 21 shows the NAS volume quota cmdlets available in the Dell Storage PowerShell SDK.

Table 21 NAS volume quota cmdlets

PowerShell SDK cmdlet
Get-DellFluidFsDirectoryQuotaRule Get-DellFluidFsGroupQuotaRule Get-DellFluidFsGroupQuotaUsage Get-DellFluidFsUserQuotaRule Get-DellFluidFsUserQuotaUsage
New-DellFluidFsDirectoryQuotaRule New-DellFluidFsGroupQuotaRule New-DellFluidFsUserQuotaRule
Set-DellFluidFsDirectoryQuotaRule Set-DellFluidFsGroupQuotaRule Set-DellFluidFsUserQuotaRule
Remove-DellFluidFsDirectoryQuotaRule Remove-DellFluidFsGroupQuotaRule Remove-DellFluidFsUserQuotaRule

5.6.1 Create a user quota rule

The **New-DellFluidFsUserQuotaRule** cmdlet can be used to create a quota on a NAS volume for a user. This sample script will create a quota rule for the user **mordi** in the **FluidFS** domain on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. The user quota rule will have a 30 GB hard quota and 20 GB soft quota.

```
# Assign variables

$ConnName       = "DSMDC"
$FsClusterName  = "fluidfs2D"
$NasVolumeName  = "engvol"
$UserDomainName = "FLUIDFS"
$UserName       = "mordi"
$HardQuota      = 30GB
$SoftQuota      = 20GB

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName
```

```
# Create a user quota rule on the NAS volume

$QuotaRule = New-DellFluidFsUserQuotaRule -ConnectionName $ConnName `
                                          -ClusterId      $FsCluster.InstanceId `
                                          -NasVolumeId    $NasVolume.NasVolumeId `
                                          -UserDomainName $UserDomainName `
                                          -UserName       $UserName `
                                          -HardQuota      $HardQuota `
                                          -SoftQuota      $SoftQuota `
                                          -QuotaLimited:$true `
                                          -AlertRequired:$true
```

5.6.2 Create a group quota rule

The **New-DellFluidFsGroupQuotaRule** cmdlet can be used to create a quota on a NAS volume for a user group. This sample script will create a quota rule for the user group **Domain Users** in the **FluidFS** domain on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. The group quota rule will have a 30GB hard quota and a 20GB soft quota.

```
# Assign variables

$ConnName        = "DSMDC"
$FsClusterName   = "fluidfs2D"
$NasVolumeName   = "engvol"
$GroupDomainName = "FLUIDFS"
$GroupName       = "Domain Users"
$HardQuota       = 30GB
$SoftQuota       = 20GB
$QuotaType       = [DellStorage.Api.Enums.FluidFsQuotaTypesEnum] "AnyUserInGroup"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Create a group quota rule on the NAS volume

$QuotaRule = New-DellFluidFsGroupQuotaRule -ConnectionName  $ConnName `
                                           -ClusterId       $FsCluster.InstanceId `
                                           -NasVolumeId     $NasVolume.NasVolumeId `
                                           -GroupDomainName $GroupDomainName `
                                           -GroupName       $GroupName `
                                           -HardQuota       $HardQuota `
                                           -SoftQuota       $SoftQuota `
                                           -QuotaType       $QuotaType `
                                           -AlertRequired:$true `
                                           -QuotaLimited:$true
```

5.6.3 Create a directory quota rule

The **New-DellFluidFsDirectoryQuotaRule** cmdlet can be used to create a quota on a NAS volume for a directory. This sample script will create a directory quota rule for the directory **/eng1** on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. The directory quota rule will have a 30GB hard quota and a 20GB soft quota.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"
$DirectoryPath = "/eng1"
$HardQuota     = 30GB
$SoftQuota     = 20GB

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Create a directory quota on the NAS volume

$QuotaRule = New-DellFluidFsDirectoryQuotaRule -ConnectionName $ConnName `
                                               -ClusterId     $FsCluster.InstanceId `
                                               -NasVolumeId   $NasVolume.NasVolumeId `
                                               -HardQuota     $HardQuota `
                                               -SoftQuota     $SoftQuota `
                                               -Path          $DirectoryPath `
                                               -QuotaLimited:$true
```

5.6.4 Get user quota rules

The **Get-DellFluidFsUserQuotaRule** cmdlet can be used to get the user quota rules on a NAS volume. This sample script will all user quota rules for the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"

# Get FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get all user quota rules on the NAS volume

$QuotaRuleList = @( Get-DellFluidFsUserQuotaRule -ConnectionName $ConnName `
                                                 -ClusterId     $FsCluster.InstanceId `
                                                 -NasVolumeId   $NasVolume.NasVolumeId )
```

```
# Show information about the user quota rules

$QuotaRuleList | Select-Object UserDomainName, UserName, SoftQuota, HardQuota
```

5.6.5 Get group quota rules

The **Get-DellFluidFsGroupQuotaRule** cmdlet can be used get the group quota rules on a NAS volume. This sample script will get all group quota rules for the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get all group quota rules on the NAS volume

$QuotaRuleList = @( Get-DellFluidFsGroupQuotaRule -ConnectionName $ConnName `
                                                  -ClusterId     $FsCluster.InstanceId `
                                                  -NasVolumeId   $NasVolume.NasVolumeId )

# Show information about the group quota rules

$QuotaRuleList | Select-Object GroupDomainName, GroupName, SoftQuota, HardQuota
```

5.6.6 Get directory quota rules

The **Get-DellFluidFsDirectoryQuotaRule** cmdlet can be used to get directory quota rules on a NAS volume. This sample script will get all directory quota rules for the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName
```

```
# Get the directory quota rules for the NAS volume

$QuotaRuleList = @( Get-DellFluidFsDirectoryQuotaRule -ConnectionName $ConnName `
                                                      -ClusterId    $FsCluster.InstanceId `
                                                      -NasVolumeId $NasVolume.NasVolumeId )

# Show information about the directory quota rules

$QuotaRuleList | Select-Object Path, SoftQuota, HardQuota
```

5.6.7 Modify a user quota rule

The **Set-DellFluidFsUserQuotaRule** cmdlet can be used to modify a user quota rule on a NAS volume. This sample script will modify the quota rule for the user **mordi** in the **FLUIDFS** domain on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. After the script has run, the user will have a new hard quota of 40GB.

```
# Assign variables

$ConnName       = "DSMDC"
$FsClusterName  = "fluidfs2D"
$NasVolumeName  = "engvol"
$UserDomainName = "FLUIDFS"
$UserName       = "mordi"
$NewHardQuota   = 40GB

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get the user quota rule on the NAS volume

$QuotaRule = Get-DellFluidFsUserQuotaRule  -ConnectionName $ConnName `
                                           -ClusterId      $FsCluster.InstanceId `
                                           -NasVolumeId    $NasVolume.NasVolumeId `
                                           -UserDomainName $UserDomainName `
                                           -UserName       $UserName

# Modify the user quota rule

$QuotaRule = Set-DellFluidFsUserQuotaRule -ConnectionName $ConnName `
                                          -Instance       $QuotaRule `
                                          -HardQuota      $NewHardQuota
```

5.6.8 Modify a group quota rule

The **Set-DellFluidFsGroupQuotaRule** cmdlet can be used to modify a group quota rule on a NAS volume. This sample script will modify the quota rule for the group **Domain Users** in the **FLUIDFS** domain on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. After the script has run, the group will have a new soft quota of 10GB.

```
# Assign variables

$ConnName        = "DSMDC"
$FsClusterName   = "fluidfs2D"
$NasVolumeName   = "engvol"
$GroupDomainName = "FLUIDFS"
$GroupName       = "Domain Users"
$NewSoftQuota    = 10GB

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get the group quota rule on the NAS volume

$QuotaRule = Get-DellFluidFsGroupQuotaRule -ConnectionName  $ConnName `
                                           -ClusterId       $FsCluster.InstanceId `
                                           -NasVolumeId     $NasVolume.NasVolumeId `
                                           -GroupDomainName $GroupDomainName `
                                           -GroupName       $GroupName

# Modify the group quota rule

$QuotaRule = Set-DellFluidFsGroupQuotaRule -ConnectionName $ConnName `
                                           -Instance       $QuotaRule `
                                           -SoftQuota      $NewSoftQuota
```

5.6.9 Modify a directory quota rule

The **Set-DellFluidFsGroupQuotaRule** cmdlet can be used to modify a directory quota rule on a NAS volume. This sample script will modify the quota rule for the ***/eng1*** directory on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**. After the script has run, the group will have a new soft quota of 11GB.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"
$DirectoryPath = "/eng1"
$NewSoftQuota  = 11GB

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName
```

```
# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get the directory quota rule for the NAS volume

$QuotaRule = Get-DellFluidFsDirectoryQuotaRule -ConnectionName $ConnName `
                                               -ClusterId      $FsCluster.InstanceId `
                                               -NasVolumeId    $NasVolume.NasVolumeId `
                                               -Path           $DirectoryPath

# Modify the directory quota rule

$QuotaRule = Set-DellFluidFsDirectoryQuotaRule -ConnectionName $ConnName `
                                               -Instance       $QuotaRule `
                                               -SoftQuota      $NewSoftQuota
```

5.6.10 Remove a user quota rule

The **Remove-DellFluidFsUserQuotaRule** cmdlet can be used to remove a user quota rule. This sample script will remove the user quota rule for the user **mordi** in the **FLUIDFS** domain on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName       = "DSMDC"
$FsClusterName  = "fluidfs2D"
$NasVolumeName  = "engvol"
$UserDomainName = "FLUIDFS"
$UserName       = "mordi"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get the user quota rule on the NAS volume

$QuotaRule = Get-DellFluidFsUserQuotaRule  -ConnectionName $ConnName `
                                           -ClusterId      $FsCluster.InstanceId `
                                           -NasVolumeId    $NasVolume.NasVolumeId `
                                           -UserDomainName $UserDomainName `
                                           -UserName       $UserName

# Remove the user quota rule

Remove-DellFluidFsUserQuotaRule -ConnectionName $ConnName `
                                -Instance       $QuotaRule `
                                -Confirm:$false
```

5.6.11 Remove a group quota rule

The **Remove-DellFluidFsGroupQuotaRule** cmdlet can be used to remove a group quota rule. This sample script will remove the group quota rule for the group **Domain Users** in the **FLUIDFS** domain on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName        = "DSMDC"
$FsClusterName   = "fluidfs2D"
$NasVolumeName   = "engvol"
$GroupDomainName = "FLUIDFS"
$GroupName       = "Domain Users"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName

# Get the group quota rule on the NAS volume

$QuotaRule = Get-DellFluidFsGroupQuotaRule -ConnectionName  $ConnName `
                                           -ClusterId       $FsCluster.InstanceId `
                                           -NasVolumeId     $NasVolume.NasVolumeId `
                                           -GroupDomainName $GroupDomainName `
                                           -GroupName       $GroupName

# Remove the group quota rule

Remove-DellFluidFsGroupQuotaRule -ConnectionName $ConnName `
                                 -Instance       $QuotaRule `
                                 -Confirm:$false
```

5.6.12 Remove a directory quota rule

The **Remove-DellFluidFsDirectoryQuotaRule** cmdlet can be used to remove a directory quota rule. This sample script will remove the directory quota rule for the directory ***/eng1*** on the NAS volume **engvol**, on the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"
$NasVolumeName = "engvol"
$directoryPath = "/eng1"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the NAS volume

$NasVolume = Get-DellFluidFsNasVolume -ConnectionName $ConnName `
                                      -ClusterId      $FsCluster.InstanceId `
                                      -Name           $NasVolumeName
```

```
# Get the directory quota rule for the NAS volume

$QuotaRule = Get-DellFluidFsDirectoryQuotaRule -ConnectionName $ConnName `
                                               -ClusterId      $FsCluster.InstanceId `
                                               -NasVolumeId    $NasVolume.NasVolumeId `
                                               -Path           $DirectoryPath

# Remove the directory quota rule

Remove-DellFluidFsDirectoryQuotaRule -ConnectionName  $ConnName `
                                     -Instance        $QuotaRule `
                                     -Confirm:$false
```

5.7 Events

The Dell Storage PowerShell SDK provides cmdlets to view FluidFS events. Table 22 shows the event cmdlets available in the Dell Storage PowerShell SDK.

Table 22 Event cmdlets

PowerShell SDK cmdlet
Get-DellFluidFsPlatformEvent
Get-DellFluidFsCollapsedPlatformEvent

5.7.1 Get platform events

The **Get-DellFluidFsPlatformEvent** cmdlet can be used to get FluidFS platfom events . This sample script will get all FluidFS platform events for FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the platform events

$EventList = @( Get-DellFluidFsPlatformEvent -ConnectionName $ConnName `
                                             -ClusterId      $FsCluster.InstanceId )

# Show the platform events

$EventList | Select-Object text
```

5.7.2 Get collapsed platform events

The **Get-DellFluidFsCollapsedPlatformEvent** cmdlet can be used to get the collapsed FluidFS platfom events. This sample script will get all collapsed FluidFS platform events for the FluidFS cluster **fluidfs2D**.

```
# Assign variables

$ConnName      = "DSMDC"
$FsClusterName = "fluidfs2D"

# Get the FluidFS cluster

$FsCluster = Get-DellFluidFsCluster -ConnectionName $ConnName `
                                    -InstanceName   $FsClusterName

# Get the collapsed platform events

$EventList = @( Get-DellFluidFsCollapsedPlatformEvent -ConnectionName $ConnName `
                                                      -ClusterId  $FsCluster.InstanceId )

# Show the collapsed platform events

$EventList | Select-Object text
```

6 Working with Windows

Microsoft introduced storage cmdlets with Windows Server 2012 that can be used to administer Windows disks and volumes. Using these cmdlets with the Dell Storage PowerShell SDK allow the same script to perform both SC Series tasks and Windows storage tasks. The sample scripts in this section use both sets of cmdlets together, demonstrating the connection between SC Series volumes and Windows disks and volumes.

6.1 Disks and volumes

When working with disks and volumes on standalone Windows servers, the disk serial number can be used to link a Windows disk to a volume on an SC Series array. This works with both master boot record (MBR) and with GUID partition table (GPT) disks.

The sample scripts in this section assume the Dell Storage PowerShell SDK module has already been imported.

6.1.1 Get the SC Series volume for a Windows volume

This sample script will get the SC Series volume used by the **E:** drive. This script is designed to run on the server hosting the **E:** drive.

```
# Assign variables

$ConnName    = "DSMDC"
$ScName      = "SC 13"
$DriveLetter = "E"

# Get the Windows volume

$WinVolume = Get-Volume -DriveLetter $DriveLetter

# Get the Windows partition

$WinPartition = Get-Partition -Volume $WinVolume

# Get the Windows disk

$WinDisk = Get-Disk -Partition $WinPartition

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -SerialNumber   $WinDisk.SerialNumber

# Display the SC volume folder and volume name

$ScVolume | Select-Object VolumeFolderPath, Name
```

6.1.2 Get the Windows disk for an SC Series volume

This sample script will get the Windows volume used by the SC Series volume **SQLData** in the **Production/SQLDatabase/SQLProd** folder. This script is designed to run from the Windows server on which the SC Series volume is mounted.

```
# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 13"
$VolumeName = "SQLData"
$FolderPath = "Production/SQLDatabase/SQLProd/"

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName   $ConnName `
                             -ScName           $ScName `
                             -Name             $VolumeName `
                             -VolumeFolderPath $FolderPath

# Get the Windows disk

$WinDisk = Get-Disk | Where-Object { $_.SerialNumber -eq $ScVolume.SerialNumber }

# Get the partitions on the disk

$WinPartitions = Get-Partition -DiskId $WinDisk.ObjectId

# Get the volumes on the partitions

$WinVolumes = $WinPartitions | Get-Volume

# Display information about the Windows volume

$WinVolumes | Select-Object DriveLetter,
                            FileSystemLabel,
                            FileSystem,
                            @{ Name       = "SizeGB";
                               Expression = { [math]::Round( $_.Size / 1GB ) } }
```

6.2 Failover clusters

When working with Windows failover clusters, an additional command set is need to manage cluster resources. In Windows Server 2008 R2, Microsoft introduced cmdlets to administer Windows failover clusters. The module containing the failover cluster cmdlets must be imported before the cmdlets can be used in a PowerShell script. The following sample script will import the failover cluster module.

```
# Import the Failover Cluster module

Import-Module FailoverClusters
```

The identifier used to find the Windows disk for a given cluster disk is different, depending on the Windows disk type. For MBR disks, the disk signature is used. For GPT disks, the disk GUID is used. Linking a Windows disk to an SC Series volume will use the disk serial number for both disk types.

The sample scripts in this section assume both the Dell Storage PowerShell SDK module and the failover cluster module have already been imported.

6.2.1 Get the SC Series volume for a clustered MBR disk

This sample script will get the SC Series volume for a clustered MBR disk by using the name of the disk displayed in Failover Cluster Manager. The SC Series volume used by the MBR clustered disk **Disk E: - SQLData** will be displayed. This script is designed to run from a node in the cluster.

```
# Assign variables

$ConnName        = "DSMDC"
$ScName          = "SC 12"
$ClusterName     = "ProdCluster"
$ClusterDiskName = "Disk E: - SQLData"

# Get the cluster

$Cluster = Get-Cluster -Name $ClusterName

# Get the cluster disk resource

$ClusterDiskResource = Get-ClusterResource -Cluster $Cluster -Name $ClusterDiskName `
                       | Where-Object { $_.ResourceType -eq "Physical Disk" }

# Get the disk signature

$DiskSignature = ( $ClusterDiskResource `
                   | Get-ClusterParameter `
                   | Where-Object { $_.Name -eq "DiskSignature" } ).Value

# Get the Windows disk

$WinDisk = Get-Disk `
           | Where-Object { $_.PartitionStyle -eq "MBR" `
                            -and `
                            $_.Signature      -eq [UInt32]$DiskSignature }

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -SerialNumber   $WinDisk.SerialNumber

# Display the SC volume folder and volume name

$ScVolume | Select-Object VolumeFolderPath, Name
```

6.2.2 Get the SC Series volume for a clustered GPT disk

This sample script will get the SC Series volume for a clustered GPT disk by using the name of the disk displayed in Failover Cluster Manager. The SC Series volume used by the MBR clustered disk **Disk K: - TestVolume** will be displayed. This script is designed to run from a node in the cluster.

```
# Assign variables

$ConnName        = "DSMDC"
$ScName          = "SC 12"
$ClusterName     = "ProdCluster"
$ClusterDiskName = "Disk K: - SQLData03"

# Get the cluster

$Cluster = Get-Cluster -Name $ClusterName
```

```powershell
# Get the cluster disk resource

$ClusterDiskResource = Get-ClusterResource -Cluster $Cluster -Name $ClusterDiskName `
                       | Where-Object { $_.ResourceType -eq "Physical Disk" }

# Get the disk GUID

$DiskGuid = ( $ClusterDiskResource `
              | Get-ClusterParameter `
              | Where-Object { $_.Name -eq "DiskIdGuid" } ).Value

# Get the Windows disk

$WinDisk = Get-Disk `
           | Where-Object { $_.PartitionStyle -eq "GPT" `
                            -and `
                            $_.Guid           -eq $DiskGuid }

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -SerialNumber   $WinDisk.SerialNumber

# Display the SC volume folder and volume name

$ScVolume | Select-Object VolumeFolderPath, Name
```

7 Working with VMware

VMware® vSphere® PowerCLI provides a PowerShell interface to perform virtualization tasks in vSphere. The PowerCLI cmdlets, combined with the Dell Storage PowerShell SDK and the Window storage cmdlets, allow the same script to perform SC Series, vSphere, and Windows storage tasks. The sample scripts in this section use all three sets of cmdlets, showing how to link between SC Series volumes, VMware datastores, VMware physical RDMs, and Windows disks in VMware guests.

The device ID of an SC Series volume can be used to find a datastore or physical RDM by adding **naa.** to the device ID, which will match the canonical name of the datastore. Inside of a Windows guest, the disk serial number can be used to find the VMDK in vSphere by inserting dashes into the disk serial number, which will match the UUID of the VMDK.

Note: The sample scripts in this section are provided as an educational resource to show how the Dell Storage PowerShell SDK can be used with the PowerShell interface provided by VMware. The PowerCLI PowerShell interface may change at any time, without notice, causing the sample scripts to stop working. Thoroughly test the sample scripts before running in a production environment.

7.1 Getting started

The vSphere PowerCLI can be downloaded from the VMware downloads site (login required). It will need to be installed on any machine running a PowerShell script that uses PowerCLI cmdlets.

Before the PowerCLI cmdlets can be used, the PowerCLI snap-in has to be loaded and a connection has to be made to VMware vCenter™. The following sample script will load the PowerCLI snap-in and connect to the vCenter at **vcsite2.techsol.local**.

```
# Assign variables

$vcName       = "vcsite2.techsol.local"
$vcUserName   = "Domain\VMwareAdmin"
$PowerCliName = "VMware.VimAutomation.Core"

# Load PowerCLI. Try loading the snap-in first. If it does not exist, load the module.

If ( Get-PSSnapin -Registered | Where-Object { $_.Name -eq $PowerCliName } )
{
    Add-PSSnapin $PowerCliName
}
ElseIf ( Get-Module -ListAvailable | Where-Object { $_.Name -eq $PowerCliName } )
{
    Import-Module $PowerCliName
}

# Get the password

$vcPassword = Read-Host -AsSecureString `
                        -Prompt "Please enter the password for $vcUserName"

# Build a credential object to connect to vCenter

$vcPSCred = New-Object System.Management.Automation.PSCredential( $vcUserName, `
                                                                  $vcPassword )
```

```
# Connect to vCenter

$vcConnection = Connect-VIServer -Server      $vcName `
                                 -Credential $vcPSCred `
                                 -Force `
                                 -WarningAction SilentlyContinue
```

7.2 Datastores and virtual hard disks

The sample scripts in this section show how to link an SC Series volume to the VMware datastore or VMDK that uses that volume. These scripts assume that the Dell Storage PowerShell SDK module has been imported, the PowerCLI snap-in as been loaded, and a connection to vCenter has been established.

7.2.1 Get the datastore for an SC Series volume

This sample script will get the VMware datastore used by the **vmboot** volume in the **Microsoft/Volumes/VCSite2** folder on Storage Center **SC 13**.

```
# Assign variables

$ConnName       = "DSMDC"
$ScName         = "SC 13"
$VolumeName     = "vmboot"
$FolderPath     = "Microsoft/Volumes/VCSite2/"
$DatacenterName = "Datacenter"

# Get the SC volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $FolderPath `
                           -Name             $VolumeName

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Get the canonical name of the datastore
# Add "naa." to the SC volume device ID

$CanonicalName = "naa." + $Volume.DeviceID

# Get the datastore

$Datastore = Get-Datastore -Location $Datacenter `
             | Where-Object { $_.ExtensionData.Info.Vmfs.Extent[0].DiskName `
                              -eq $CanonicalName }

# Display information about the datastore

$Datastore | Select-Object Datacenter, Name
```

7.2.2 Get the SC Series volume for a datastore

This sample script will get the SC Series volume used by the **vmboot** datastore.

```
# Assign variables

$ConnName       = "DSMDC"
$ScName         = "SC 13"
$DatacenterName = "Datacenter"
$DatastoreName  = "vmboot"

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Get the datastore

$Datastore = Get-Datastore -Location $Datacenter -Name $DatastoreName

# Get the device ID for the SC volume
# Remove "naa." from the canonical name

$DeviceId = $Datastore.ExtensionData.Info.Vmfs.Extent[0].DiskName.Substring(4)

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -DeviceId       $DeviceId

# Display the SC volume folder and volume name

$ScVolume | Select-Object ScName, VolumeFolderPath, Name
```

7.2.3 Get the SC Series volume for a virtual hard disk (VMDK)

This sample script gets the SC Series volume used by **Hard Disk 3** on the **esx-dev-01** virtual machine.

```
# Assign variables

$ConnName       = "DSMDC"
$ScName         = "SC 13"
$DatacenterName = "Datacenter"
$VMName         = "esx-dev-01"
$HardDiskName   = "Hard disk 3"

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Get the virtual machine

$VM = Get-VM -Location $Datacenter -Name $VMName

# Get the virtual hard disk (VMDK)

$Vmdk = Get-HardDisk -VM       $VM `
                     -Name     $HardDiskName `
                     -DiskType "Flat"

# Get the datastore

$Datastore = Get-Datastore -Id $Vmdk.ExtensionData.Backing.Datastore
```

```
# Get the device ID for the SC volume
# Remove "naa." from the canonical name

$DeviceId = $Datastore.ExtensionData.Info.Vmfs.Extent[0].DiskName.Substring(4)

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -DeviceId       $DeviceId

# Display the SC volume folder and volume name

$ScVolume | Select-Object ScName, VolumeFolderPath, Name
```

7.2.4 Get the SC Series volume for a Windows volume in a guest

This sample script will return the SC Series volume used by the **F:** drive inside of the **esx-dev-01** virtual machine. This script is designed to run from inside the VMware guest. Both PowerCLI and the Dell Storage PowerShell SDK need to be installed inside of the guest. The VMware guest must be running Windows Server 2012 or higher.

```
# Assign variables

$ConnName       = "DSMDC"
$ScName         = "SC 13"
$DriveLetter    = "F"
$DatacenterName = "Datacenter"
$VMName         = "esx-dev-01"

# Get the Windows partition

$WinPartition = Get-Partition -DriveLetter $DriveLetter

# Get the Windows disk

$WinDisk = Get-Disk -Partition $WinPartition

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Get the virtual machine

$VM = Get-VM -Location $Datacenter -Name $VMName

# Get the uuid for the vmdk
# Insert dashes into the disk serial number

$uuid = $WinDisk.SerialNumber.insert(20, "-" )
$uuid = $uuid.insert(16, "-" )
$uuid = $uuid.insert(12, "-" )
$uuid = $uuid.insert(8, "-" )

# Get the virtual hard disk (vmdk)

$Vmdk = Get-HardDisk -VM $VM -DiskType "Flat" `
        | Where-Object {  $_.ExtensionData.Backing.Uuid -eq $uuid }

# Get the datastore

$Datastore = Get-Datastore -Id $Vmdk.ExtensionData.Backing.Datastore
```

```
# Get the device ID for the SC volume
# Remove "naa." from the canonical name

$DeviceId = $Datastore.ExtensionData.Info.Vmfs.Extent[0].DiskName.Substring(4)

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -DeviceId       $DeviceId

# Display the SC volume folder and volume name

$ScVolume | Select-Object ScName, VolumeFolderPath, Name
```

7.3 Raw device mappings (RDMs)

The sample scripts in this section show how to link an SC Series volume to the physical RDM that uses that volume. These scripts assume that the Dell Storage PowerShell SDK module has been imported, the PowerCLI snap-in has been loaded, and a connection to vCenter has been established.

7.3.1 Get the physical RDM for an SC Series volume

This sample script will get the physical RDM that uses the **ESX12-01 - SQLData** volume in the **Microsoft/Volumes/ESX12-01** folder on Storage Center **SC 13**.

```
# Assign variables

$ConnName       = "DSMDC"
$ScName         = "SC 13"
$VolumeName     = "ESX12-01 - SQLData"
$FolderPath     = "Microsoft/Volumes/ESX12-01/"
$DatacenterName = "Datacenter"

# Get the SC volume

$Volume = Get-DellScVolume -ConnectionName   $ConnName `
                           -ScName           $ScName `
                           -VolumeFolderPath $FolderPath `
                           -Name             $VolumeName

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Create the canonical name
# Add "naa." to the volume DeviceID

$CanonicalName = "naa." + $Volume.DeviceID

# Get the physical RDM

$pRDM = Get-VM -Location $Datacenter `
        | Get-HardDisk -DiskType "rawPhysical" `
        | where-Object { $_.ScsiCanonicalName -eq $CanonicalName }

# Display information about the physical RDM

$pRDM | Select-Object Parent, Name
```

7.3.2 Get the SC Series volume for a physical RDM

This sample script will get the SC Series volume for the physical RDM **Hard disk 3** on the virtual machine **esx12-01**.

```
# Assign variables

$ConnName       = "DSMDC"
$ScName         = "SC 13"
$DatacenterName = "Datacenter"
$VMName         = "esx12-01"
$HardDiskName   = "Hard disk 3"

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Get the virtual machine

$VM = Get-VM -Location $Datacenter -Name $VMName

# Get the physical RDM

$pRDM = Get-HardDisk -VM       $VM `
                     -Name     $HardDiskName `
                     -DiskType "RawPhysical"

# Get the device ID for the SC volume
# Remove "naa." from the canonical name

$DeviceId = $pRDM.ScsiCanonicalName.Substring(4)

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -DeviceId       $DeviceId

# Display the SC volume folder and volume name

$ScVolume | Select-Object ScName, VolumeFolderPath, Name
```

7.3.3 Get the physical RDM for a Windows volume in a guest

This sample script will get the physical RDM used by the **E:** drive inside of the **esx12-01** virtual machine. This script is designed to run from inside the VMware guest. PowerCLI needs to be installed inside of the guest. The VMware guest must be running Windows Server 2012 or higher.

```
# Assign variables

$ScName      = "SC 13"
$DriveLetter = "E"
$VMName      = "esx12-01"

# Get the Windows partition

$WinPartition = Get-Partition -DriveLetter $DriveLetter

# Get the Windows disk

$WinDisk = Get-Disk -Partition $WinPartition
```

```
# Get the virtual machine

$VM = Get-VM -Name $VMName

# Get the canonical name
# Add "naa." to the unique ID for the Windows disk

$CanonicalName = "naa." + $WinDisk.UniqueId

# Get the physical RDM

$pRDM = Get-HardDisk -VM $VM -DiskType "rawPhysical" `
        | Where-Object { $_.ScsiCanonicalName -eq $CanonicalName }

# Display information about the physical RDM

$pRDM | Select-Object Parent, Name
```

7.3.4 Get the SC Series volume for a Windows volume in a guest

This sample script will get the SC Series volume used by the **E:** drive in the **esx12-01** virtual machine. This script is designed to run from inside the VMware guest. Both PowerCLI and the Dell Storage PowerShell SDK need to be installed inside of the guest. The VMware guest must be running Windows Server 2012 or higher.

```
# Assign variables

$ConnName    = "DSMDC"
$ScName      = "SC 13"
$DriveLetter = "E"

# Get the Windows partition

$WinPartition = Get-Partition -DriveLetter $DriveLetter

# Get the Windows disk

$WinDisk = Get-Disk -Partition $WinPartition

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -SerialNumber   $WinDisk.SerialNumber

# Display the SC volume folder and volume name

$ScVolume | Select-Object ScName, VolumeFolderPath, Name
```

8 Automating common tasks

The sample scripts in this section combine techniques demonstrated in previous sections to perform larger tasks. All scripts assume that a saved connection named **DSMDC** has already been created.

8.1 Create a new volume for a Windows server

This sample script will create a 110 GB volume on the SC Series array, map it to a Windows server, and present the volume as the **E:** drive in Windows. This script is designed to run on the Windows server. Since the script uses the Windows storage cmdlets, the server must be running Windows Server 2012 or higher.

```
# Set error handling preference

$ErrorActionPreference = "Stop"

# Import the Dell Storage PowerShell SDK module

Import-Module "C:\PS_SDK\DellStorage.ApiCommandSet.psd1"

# Assign variables

$ConnName           = "DSMDC"
$ScName             = "SC 13"

$ScVolumeName       = "SQLData"
$ScVolumeSize       = [DellStorage.Api.Types.StorageSize] "110GB"
$ScVolumeFolderPath = "Production/SQLDatabase/"
$ScVolumeFolderName = "SQLProd"

$ScServerName       = "SQLProd"
$ScServerFolderPath = "Production/SQLDatabase/"

$PartitionStyle     = "MBR"
$Offset             = 1048576
$AccessPath         = "E:\"
$FileSystem         = "NTFS"
$AllocationUnit     = 65536

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -Name $ScName

# Get the SC volume folder

$ScVolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                         -StorageCenter  $StorageCenter `
                                         -FolderPath     $ScVolumeFolderPath `
                                         -Name           $ScVolumeFolderName

# Get the SC server

$ScServer = Get-DellScServer -ConnectionName   $ConnName `
                             -StorageCenter    $StorageCenter `
                             -ServerFolderPath $ScServerFolderPath `
                             -Name             $ScServerName

# Create the SC volume

$ScVolume = New-DellScVolume -ConnectionName $ConnName `
                             -StorageCenter  $StorageCenter `
                             -Name           $ScVolumeName `
                             -Size           $ScVolumeSize `
                             -VolumeFolder   $ScVolumeFolder
```

```powershell
# Map the SC volume to the server
$ScVolumeMap = Add-DellScVolumeToServerMap -ConnectionName $ConnName `
                                           -Server         $ScServer `
                                           -Instance       $ScVolume

# Rescan disks in Windows
Update-HostStorageCache

# Get the Windows disk
$WinDisk = Get-Disk | Where-Object { $_.SerialNumber -eq $ScVolume.SerialNumber }

# Initialize the Windows disk
Initialize-Disk -InputObject $WinDisk -PartitionStyle $PartitionStyle

# Create a partition on the disk
$WinPartition = New-Partition -InputObject $WinDisk `
                              -Offset      $Offset `
                              -UseMaximumSize

# Add the access path to the partition
Add-PartitionAccessPath -DiskNumber      $WinPartition.DiskNumber `
                        -PartitionNumber $WinPartition.PartitionNumber `
                        -AccessPath      $AccessPath

# Format the Windows volume on the partition
$WinVolume = Format-Volume -Partition          $WinPartition `
                           -FileSystem         $FileSystem `
                           -AllocationUnitSize $AllocationUnit `
                           -NewFileSystemLabel $ScVolumeName `
                           -Confirm:$false
```

8.2 Create a new volume from a snapshot for a Windows server

This sample script will create a view volume using the latest snapshot on the **SQLData** volume in the **Production/SQLDatabase/SQLProd** folder. The view volume will be created in the same folder as the **SQLData** volume and will be presented to Windows as the **G:** drive. This script is designed to run on the Windows server. Since the script uses the Windows storage cmdlets, the server must be running Windows Server 2012 or higher.

```
# Set error handling preference

$ErrorActionPreference = "Stop"

# Import the Dell Storage PowerShell SDK module

Import-Module "C:\PS_SDK\DellStorage.ApiCommandSet.psd1"

# Assign variables

$ConnName           = "DSMDC"
$ScName             = "SC 13"

$ScVolumeName       = "SQLData"
$ScVolumeFolderPath = "Production/SQLDatabase/SQLProd/"
$ScViewVolumeName   = $ScVolumeName + " View"

$ScServerName       = "SQLProd"
$ScServerFolderPath = "Production/SQLDatabase/"

$AccessPath         = "G:\"

# Get the SC server

$ScServer = Get-DellScServer -ConnectionName   $ConnName `
                             -ScName           $ScName `
                             -ServerFolderPath $ScServerFolderPath `
                             -Name             $ScServerName

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName   $ConnName `
                             -ScName           $ScName `
                             -VolumeFolderPath $ScVolumeFolderPath `
                             -Name             $ScVolumeName

# Get the latest frozen snapshot (Replay) on the volume

$ScReplay = Get-DellScReplay -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -CreateVolume   $ScVolume `
                             -Active:$false `
          | Sort-Object { [datetime]$_.FreezeTime }, `
                        { [UInt32] ( $_.GlobalIndex.Split( "-" ) )[-1] } `
          | Select-Object -Last 1

# Create the view volume in the same folder as the volume

$ScViewVolume = New-DellScReplayView -ConnectionName $ConnName `
                                     -Instance       $ScReplay `
                                     -Name           $ScViewVolumeName `
                                     -VolumeFolder   $ScVolume.VolumeFolder
```

```powershell
# Map the SC volume to the server

$ScVolumeMap = Add-DellScVolumeToServerMap -ConnectionName $ConnName `
                                           -Server         $ScServer `
                                           -Instance       $ScViewVolume

# Rescan disks in Windows

Update-HostStorageCache

# Get the Windows Disk

$WinDisk = Get-Disk | Where-Object { $_.SerialNumber -eq $ScViewVolume.SerialNumber }

# Make the disk writable and bring it online

Set-Disk -InputObject $WinDisk -IsReadOnly:$False
Set-Disk -InputObject $WinDisk -IsOffline:$False

# Get the partition

$WinPartition = Get-Partition -Disk $WinDisk

# If the partition was assigned a path, remove it
# This can happen if automount is enabled

$ExistingPath = $WinPartition.AccessPaths `
                | Where-Object { $_ -notlike "\\?\Volume*" }

If ( $ExistingPath )
{
    Remove-PartitionAccessPath -InputObject $WinPartition `
                               -AccessPath  $ExistingPath
}

# Add the accesspath

Add-PartitionAccessPath -InputObject $WinPartition -AccessPath $AccessPath
```

8.3 Create a new volume for a VMware VMFS datastore

This sample script will create a 200 GB SC Series volume named **sqldata-ds1** in the **Production/SQLDatabase/SQLProd** folder. A VMFS datastore named **sqldata-ds1** will then be created using that volume. This script is designed to run on a machine from which ESXi is administered.

```
# Set error handling preference

$ErrorActionPreference = "Stop"

# Import the Dell Storage PowerShell SDK module

Import-Module "C:\PS_SDK\DellStorage.ApiCommandSet.psd1"

# Assign variables

$ConnName            = "DSMDC"
$ScName              = "SC 13"

$ScVolumeName        = "sqldata-ds1"
$ScVolumeSize        = [DellStorage.Api.Types.StorageSize] "200GB"
$ScVolumeFolderPath  = "Production/SQLDatabase/"
$ScVolumeFolderName  = "SQLProd"

$ScServerName        = "VCSite2-Cluster1"
$ScServerFolderPath  = "VMware/Demo/"

$PowerCliName        = "VMware.VimAutomation.Core"
$vcName              = "vcsite2.techsol.local"
$vcUserName          = "Domain\VMwareAdmin"
$DatacenterName      = "Datacenter"
$VMClusterName       = "Prod Cluster"
$DatastoreHostName   = "esxi10.techsol.local"

# Load PowerCLI. Try loading the snap-in first. If it does not exist, load the module.

If ( Get-PSSnapin -Registered | Where-Object { $_.Name -eq $PowerCliName } )
{
    Add-PSSnapin $PowerCliName
}
ElseIf ( Get-Module -ListAvailable | Where-Object { $_.Name -eq $PowerCliName } )
{
    Import-Module $PowerCliName
}

# Get the vCenter password

$vcPassword = Read-Host -AsSecureString `
                        -Prompt "Please enter the password for $vcUserName"

# Build a credential object to connect to vCenter

$vcPSCred = New-Object System.Management.Automation.PSCredential( $vcUserName, `
                                                                  $vcPassword )

# Connect to vCenter

$vcConnection = Connect-VIServer -Server     $vcName `
                                 -Credential $vcPSCred `
                                 -Force `
                                 -WarningAction SilentlyContinue

# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -Name $ScName
```

```
# Get the volume folder

$ScVolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                         -StorageCenter  $StorageCenter `
                                         -FolderPath     $ScVolumeFolderPath `
                                         -Name           $ScVolumeFolderName

# Create the SC volume

$ScVolume = New-DellScVolume -ConnectionName $ConnName `
                             -StorageCenter  $StorageCenter `
                             -Name           $ScVolumeName `
                             -Size           $ScVolumeSize `
                             -VolumeFolder   $ScVolumeFolder

# Get the server object for vSphere

$ScServer = Get-DellScServer -ConnectionName   $ConnName `
                             -StorageCenter    $StorageCenter `
                             -ServerFolderPath $ScServerFolderPath `
                             -Name             $ScServerName

# Map the volume to the server

$ScVolumeMap = Add-DellScVolumeToServerMap -ConnectionName $ConnName `
                                           -Server         $ScServer `
                                           -Instance       $ScVolume

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Get the vSphere cluster

$VMCluster = Get-Cluster -Location $Datacenter -Name $VMClusterName

# Rescan HBAs on each host

ForEach ( $VMHost in ( Get-VMHost -Location $VMCluster ) )
{
    Get-VMHostStorage -VMHost $VMHost -RescanAllHba -RescanVmfs
}

# Get the host for the new datastore

$DsHost = Get-VMHost -Location $VMCluster -Name $DatastoreHostName

# Get the SCSI Lun for the volume

$ScsiLun = Get-ScsiLun -VmHost $DsHost `
           | Where-Object { $_.CanonicalName -eq $( "naa." + $ScVolume.DeviceID ) }

# Create the datastore

$Datastore = New-Datastore -VMHost $DsHost `
                           -Name   $ScVolumeName `
                           -Path   $ScsiLun.CanonicalName `
                           -Vmfs
```

8.4 Create a new volume for a VMware physical RDM

This sample script will create a 200 GB SC Series volume named **sqldata-rdm** in the **Production/SQLDatabase/SQLProd** folder. A physical RDM will then be created using that volume and presented to the **esx-dev-01** virtual machine. This script is designed to run on a machine from which ESXi is administered.

```
# Set error handling preference

$ErrorActionPreference = "Stop"

# Import the Dell Storage PowerShell SDK module

Import-Module "C:\PS_SDK\DellStorage.ApiCommandSet.psd1"

# Assign variables

$ConnName            = "DSMDC"
$ScName              = "SC 13"

$ScVolumeName        = "sqldata-rdm"
$ScVolumeSize        = [DellStorage.Api.Types.StorageSize] "200GB"
$ScVolumeFolderPath  = "Production/SQLDatabase/"
$ScVolumeFolderName  = "SQLProd"

$ScServerName        = "VCSite2-Cluster1"
$ScServerFolderPath  = "VMware/Demo/"

$PowerCliName        = "VMware.VimAutomation.Core"
$vcName              = "vcsite2.techsol.local"
$vcUserName          = "Domain\VMwareAdmin"
$DatacenterName      = "Datacenter"
$VMClusterName       = "Prod Cluster"
$VMName              = "esx-dev-01"

# Load PowerCLI. Try loading the snap-in first. If it does not exist, load the module.

If ( Get-PSSnapin -Registered | Where-Object { $_.Name -eq $PowerCliName } )
{
    Add-PSSnapin $PowerCliName
}
ElseIf ( Get-Module -ListAvailable | Where-Object { $_.Name -eq $PowerCliName } )
{
    Import-Module $PowerCliName
}

# Get the vCenter password

$vcPassword = Read-Host -AsSecureString `
                        -Prompt "Please enter the password for $vcUserName"

# Build a credential object to connect to vCenter

$vcPSCred = New-Object System.Management.Automation.PSCredential( $vcUserName, `
                                                                  $vcPassword )

# Connect to vCenter

$vcConnection = Connect-VIServer -Server     $vcName `
                                 -Credential $vcPSCred `
                                 -Force `
                                 -WarningAction SilentlyContinue
```

```
# Get the Storage Center

$StorageCenter = Get-DellStorageCenter -ConnectionName $ConnName -Name $ScName

# Get the volume folder

$ScVolumeFolder = Get-DellScVolumeFolder -ConnectionName $ConnName `
                                         -StorageCenter  $StorageCenter `
                                         -FolderPath     $ScVolumeFolderPath `
                                         -Name           $ScVolumeFolderName

# Create the SC volume

$ScVolume = New-DellScVolume -ConnectionName $ConnName `
                             -StorageCenter  $StorageCenter `
                             -Name           $ScVolumeName `
                             -Size           $ScVolumeSize `
                             -VolumeFolder   $ScVolumeFolder

# Get the server object for vSphere

$ScServer = Get-DellScServer -ConnectionName   $ConnName `
                             -StorageCenter    $StorageCenter `
                             -ServerFolderPath $ScServerFolderPath `
                             -Name             $ScServerName

# Map the volume to the server

$ScVolumeMap = Add-DellScVolumeToServerMap -ConnectionName $ConnName `
                                           -Server         $ScServer `
                                           -Instance       $ScVolume

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Get the vSphere cluster

$VMCluster = Get-Cluster -Location $Datacenter -Name $VMClusterName

# Rescan HBAs on each host

ForEach ( $VMHost in ( Get-VMHost -Location $VMCluster ) )
{
    Get-VMHostStorage -VMHost $VMHost -RescanAllHba -RescanVmfs
}

# Get the VM

$VM = Get-VM -Location $VMCluster -Name $VMName

# Get the VM host

$VMHost = Get-VMHost -VM $VM

# Get the SCSI Lun for the volume

$ScsiLun = Get-ScsiLun -VmHost $VMHost `
           | Where-Object { $_.CanonicalName -eq $( "naa." + $ScVolume.DeviceID ) }

# Add the volumes as a physical RDM to the VM

$pRDM = New-HardDisk -VM         $VM `
                     -DiskType   "rawPhysical" `
                     -DeviceName $( $ScsiLun.ConsoleDeviceName )
```

8.5 Remove a volume from a Windows server

This sample script will remove the **G:** drive from a Windows server. In Windows, the drive letter is removed and the disk is taken offline. The corresponding volume on the SC Series array is then removed. This script is designed to run on the Windows server. Since the script uses the Windows storage cmdlets, the server must be running Windows Server 2012 or higher.

```
# Set error handling preference

$ErrorActionPreference = "Stop"

# Import the Dell Storage PowerShell SDK module

Import-Module "C:\PS_SDK\DellStorage.ApiCommandSet.psd1"

# Assign variables

$ConnName   = "DSMDC"
$ScName     = "SC 13"
$AccessPath = "G:\"

# Get the Windows partition

$WinPartition = Get-Partition | Where-Object { $_.AccessPaths[0] -eq $AccessPath }

# Get the Windows disk

$WinDisk = Get-Disk -Partition $WinPartition

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -SerialNumber   $WinDisk.SerialNumber

# Remove the access path from the partition

Remove-PartitionAccessPath -InputObject $WinPartition `
                           -AccessPath  $AccessPath

# Take the disk offline

Set-Disk -InputObject $WinDisk -IsOffline:$true

# Remove the volume, sending it to the recycle bin

Start-DellScVolumeRecycle -ConnectionName $ConnName `
                          -Instance       $ScVolume `
                          -Confirm:$false

# Get the volume from the recycle bin

$RecycleBinVolume = Get-DellScVolume -ConnectionName $ConnName `
                                     -ScName         $ScName `
                                     -InstanceID     $ScVolume.InstanceId `
                                     -InRecycleBin:$true

# Remove the volume from the recycle bin

Remove-DellScVolume -ConnectionName $ConnName `
                    -Instance       $RecycleBinVolume `
                    -Confirm:$false
```

8.6 Remove a VMware VMFS datastore

This sample script will remove the **sqldata-ds1** datastore. In ESXi, the datastore is unmounted and the SCSI LUN is detached. The corresponding volume on the SC Series array is then removed. This script assumes that there are no vmdks on the datastore that are being used. All VMDKs should be removed from their respective virtual machines before running this script. This script is designed to run on a machine from which ESXi is administered.

```
# Set error handling preference

$ErrorActionPreference = "Stop"

# Import the Dell Storage PowerShell SDK module

Import-Module "C:\PS_SDK\DellStorage.ApiCommandSet.psd1"

# Assign variables

$ConnName           = "DSMDC"
$ScName             = "SC 13"

$PowerCliName       = "VMware.VimAutomation.Core"
$vcName             = "vcsite2.techsol.local"
$vcUserName         = "Domain\VMwareAdmin"
$DatacenterName     = "Datacenter"
$VMClusterName      = "Prod Cluster"
$DatastoreName      = "sqldata-ds1"

# Load PowerCLI. Try loading the snap-in first. If it does not exist, load the module.

If ( Get-PSSnapin -Registered | Where-Object { $_.Name -eq $PowerCliName } )
{
    Add-PSSnapin $PowerCliName
}
ElseIf ( Get-Module -ListAvailable | Where-Object { $_.Name -eq $PowerCliName } )
{
    Import-Module $PowerCliName
}

# Get the vCenter password

$vcPassword = Read-Host -AsSecureString `
                        -Prompt "Please enter the password for $vcUserName"

# Build a credential object to connect to vCenter

$vcPSCred = New-Object System.Management.Automation.PSCredential( $vcUserName, `
                                                                  $vcPassword )

# Connect to vCenter

$vcConnection = Connect-VIServer -Server     $vcName `
                                 -Credential $vcPSCred `
                                 -Force `
                                 -WarningAction SilentlyContinue

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Get the vSphere cluster

$VMCluster = Get-Cluster -Location $Datacenter -Name $VMClusterName
```

```powershell
# Get the datastore

$Datastore = Get-Datastore -Location $Datacenter -Name $DatastoreName

# Get the datastore canonical name

$DsCanonicalName = $Datastore.ExtensionData.Info.Vmfs.Extent[0].DiskName

# Get the datastore uuid

$DsUuid = $Datastore.ExtensionData.Info.Vmfs.Uuid

# Get the device ID for the SC volume
# Remove "naa." from the datastore canonical name

$DeviceId = $DsCanonicalName.Substring(4)

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -DeviceId       $DeviceId

# Remove the datastore from each host

ForEach ( $VMHost in ( Get-VMHost -Location $VMCluster ) )
{
    # Get the Storage System view

    $StorageSystem = Get-View $VMHost.ExtensionData.ConfigManager.StorageSystem

    # Unmount the datastore

    $StorageSystem.UnmountVmfsVolume( $DsUuid )

    # Get the SCSI Lun

    $ScsiLun = Get-ScsiLun -VmHost $VMHost `
               | Where-Object { $_.CanonicalName -eq $DsCanonicalName }

    # Detach the SCSI Lun

    $StorageSystem.DetachScsiLun( $ScsiLun.ExtensionData.Uuid )

    # Remove the detached device from the host configuration

    $EsxCli = Get-EsxCli -VMHost $VMHost

    If ( $EsxCli.storage.core.device.detached.list( $DsCanonicalName ) )
    {
        $EsxCli.storage.core.device.detached.remove( $null, $DsCanonicalName )
    }
}

# Remove the volume, sending it to the recycle bin

Start-DellScVolumeRecycle -ConnectionName $ConnName `
                          -Instance       $ScVolume `
                          -Confirm:$false

# Rescan HBAs on each host

ForEach ( $VMHost in ( Get-VMHost -Location $VMCluster ) )
{
    Get-VMHostStorage -VMHost $VMHost -RescanAllHba -RescanVmfs
}
```

```
# Get the volume from the recycle bin

$RecycleBinVolume = Get-DellScVolume -ConnectionName $ConnName `
                                     -ScName         $ScName `
                                     -InstanceID     $ScVolume.InstanceId `
                                     -InRecycleBin:$true

# Remove the volume from the recycle bin

Remove-DellScVolume -ConnectionName $ConnName `
                    -Instance       $RecycleBinVolume `
                    -Confirm:$false
```

8.7 Remove a VMware physical RDM

This sample script will remove the physical RDM named **Hard disk 9** from the **esx-dev-01** virtual machine. In ESXi, the RDM is removed from the virtual machine and the SCSI LUN is detached. The corresponding volume on the SC Series array is then removed. This script is designed to run on a machine from which ESXi is administered.

```
# Set error handling preference

$ErrorActionPreference = "Stop"

# Import the Dell Storage PowerShell SDK module

Import-Module "C:\PS_SDK\DellStorage.ApiCommandSet.psd1"

# Assign variables

$ConnName           = "DSMDC"
$ScName             = "SC 13"

$PowerCliName       = "VMware.VimAutomation.Core"
$vcName             = "vcsite2.techsol.local"
$vcUserName         = "Domain\VMwareAdmin"
$DatacenterName     = "Datacenter"
$VMClusterName      = "Prod Cluster"
$VMName             = "esx-dev-01"
$HardDiskName       = "Hard disk 9"

# Load PowerCLI. Try loading the snap-in first. If it does not exist, load the module.

If ( Get-PSSnapin -Registered | Where-Object { $_.Name -eq $PowerCliName } )
{
    Add-PSSnapin $PowerCliName
}
ElseIf ( Get-Module -ListAvailable | Where-Object { $_.Name -eq $PowerCliName } )
{
    Import-Module $PowerCliName
}

# Get the vCenter password

$vcPassword = Read-Host -AsSecureString `
                        -Prompt "Please enter the password for $vcUserName"

# Build a credential object to connect to vCenter

$vcPSCred = New-Object System.Management.Automation.PSCredential( $vcUserName, `
                                                                  $vcPassword )
```

```powershell
# Connect to vCenter

$vcConnection = Connect-VIServer -Server     $vcName `
                                 -Credential $vcPSCred `
                                 -Force `
                                 -WarningAction SilentlyContinue

# Get the datacenter

$Datacenter = Get-Datacenter -Name $DatacenterName

# Get the vSphere cluster

$VMCluster = Get-Cluster -Location $Datacenter -Name $VMClusterName

# Get the VM

$VM = Get-VM -Location $VMCluster -Name $VMName

# Get the physical RDM

$pRDM = Get-HardDisk -VM       $VM `
                     -DiskType "rawPhysical" `
                     -Name     $HardDiskName

# Get the device ID for the SC volume
# Remove "naa." from the canonical name

$DeviceId = $pRDM.ScsiCanonicalName.Substring(4)

# Get the SC volume

$ScVolume = Get-DellScVolume -ConnectionName $ConnName `
                             -ScName         $ScName `
                             -DeviceId       $DeviceId

# Remove the pRDM from the VM

Remove-HardDisk -HardDisk $pRDM -DeletePermanently -Confirm:$False

# Detach the LUN from each host

ForEach ( $VMHost in ( Get-VMHost -Location $VMCluster ) )
{
    # Get the Storage System view

    $StorageSystem = Get-View $VMHost.ExtensionData.ConfigManager.StorageSystem

    # Get the SCSI Lun

    $ScsiLun = Get-ScsiLun -VmHost $VMHost `
               | Where-Object { $_.CanonicalName -eq $pRDM.ScsiCanonicalName }

    # Detach the SCSI Lun

    $StorageSystem.DetachScsiLun( $ScsiLun.ExtensionData.Uuid )

    # Remove the detached device from the host configuration

    $EsxCli = Get-EsxCli -VMHost $VMHost

    If ( $EsxCli.storage.core.device.detached.list( $pRDM.ScsiCanonicalName ) )
    {
        $EsxCli.storage.core.device.detached.remove( $null, $pRDM.ScsiCanonicalName )
    }
}
```

```powershell
# Remove the volume, sending it to the recycle bin

Start-DellScVolumeRecycle -ConnectionName $ConnName `
                          -Instance       $ScVolume `
                          -Confirm:$false

# Rescan HBAs on each host

ForEach ( $VMHost in ( Get-VMHost -Location $VMCluster ) )
{
    Get-VMHostStorage -VMHost $VMHost -RescanAllHba -RescanVmfs
}

# Get the volume from the recycle bin

$RecycleBinVolume = Get-DellScVolume -ConnectionName $ConnName `
                                     -ScName         $ScName `
                                     -InstanceID     $ScVolume.InstanceId `
                                     -InRecycleBin:$true

# Remove the volume from the recycle bin

Remove-DellScVolume -ConnectionName $ConnName `
                    -Instance       $RecycleBinVolume `
                    -Confirm:$false
```

A Additional resources

A.1 Technical support

Dell.com/support is focused on meeting customer needs with proven services and support.

Dell TechCenter is an online technical community where IT professionals have access to numerous resources for Dell software, hardware, and services.

Storage Solutions Technical Documents on Dell TechCenter provide expertise that helps to ensure customer success on Dell EMC storage platforms.

A.2 Referenced or recommended resources

Referenced or recommended publications and resources:

- Scripting with Windows PowerShell: https://technet.microsoft.com/en-us/library/bb978526.aspx
- Dell Storage PowerShell SDK: Download from Dell Support or from the Knowledge Center on the SC Series Portal (login required)
- vSphere PowerCLI: Download from the VMware downloads site (login required)