

The All-Flash Array Built for the Next Generation Data Center

SolidFire PowerShell Tools User Guide

Version: 1.0

9/30/15

SolidFire Legal Notices

The software described in this user guide is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Copyright Notice

Copyright © 2015 SolidFire, Inc.

All Rights Reserved.

Any technical documentation that is made available by SolidFire is the copyrighted work of SolidFire, Inc. and is owned by SolidFire, Inc.

NO WARRANTY: The technical documentation that is made available by SolidFire, Inc. is delivered AS-IS, and SolidFire, Inc. makes no warranty as to its accuracy or use.

Trademarks

SolidFire is a U.S. registered trademark.

SolidFire, Element, SolidFire Helix and the helix design are U.S trademarks of SolidFire, Inc.

Patent Pending U.S. Patent and Trademark Office

TABLE OF CONTENTS

SolidFire Legal Notices	ii
Introduction	1
Software Prerequisites	1
Installing the SolidFire PowerShell Tools	1
Upgrading the SolidFire PowerShell Tools	6
How to Use SolidFire PowerShell Tools	7
Listing Available Functions	7
Accessing Embedded Help	8
Parameter Sets	8
Managing Connections to a SolidFire Cluster	9
Connecting to a SolidFire Cluster	9
Connecting to a SolidFire Node	9
Disconnecting from a SolidFire Cluster or Node	9
Changing API Versions	10
Global Variables for All Functions	11
Using SolidFire PowerShell Tools with Other Modules and Snap-ins	12
Appendix A — Available Cmdlets	13
Common Parameters	13
Add-SFDrive	13
Add-SFInitiatorToVolumeAccessGroup	13
Add-SFNode	14
Add-SFSnmpNetwork	14
Add-SFSnmpTrapRecipient	14
Add-SFSnmpUsmUser	14
Add-SFVolumeToVolumeAccessGroup	15
Connect-SFCluster	15
Disconnect-SFCluster	15
Enable-SFSnmp	16

Get-SFAccount	16
Get-SFAccountEfficiency	16
Get-SFActiveNode	16
Get-SFAsyncResult	17
Get-SFBootstrapConfig	17
Get-SFBulkVolumeJob	17
Get-SFClusterAdmin	18
Get-SFClusterCapacity	18
Get-SFClusterConfig	18
Get-SFClusterFault	18
Get-SFClusterFullThreshold	18
Get-SFClusterInfo	19
Get-SFClusterMasterNodeID	19
Get-SFClusterStats	19
Get-SFClusterVersionInfo	20
Get-SFConfig	20
Get-SFDefaultQoS	20
Get-SFDeletedVolume	20
Get-SFDrive	21
Get-SFDriveConfig	21
Get-SFDriveHardwareInfo	21
Get-SFDriveStats	21
Get-SFEvent	21
Get-SFGroupSnapshot	22
Get-SFIscsiSession	22
Get-SFNetworkConfig	22
Get-SFNtpInfo	22
Get-SFPendingNode	23
Get-SFSnapshot	23
Get-SFSnmpAcl	23

Get-SFSnmpInfo	23
Get-SFSnmpState	24
Get-SFSnmpTrapInfo	24
Get-SFSyncJob	24
Get-SFVolume	24
Get-SFVolumeAccessGroup	25
Get-SFVolumeAccessGroupEfficiency	25
Get-SFVolumeEfficiency	25
Get-SFVolumeStats	26
Invoke-SFSecureEraseDrive	26
New-SFAccount	26
New-SFClone	27
New-SFCloneMultiple	27
New-SFCluster	27
New-SFClusterAdmin	28
New-SFGroupSnapshot	28
New-SFSnapshot	28
New-SFVolume	29
New-SFVolumeAccessGroup	29
Remove-SFAccount	30
Remove-SFClusterAdmin	30
Remove-SFDeletedVolume	30
Remove-SFDrive	30
Remove-SFGroupSnapshot	31
Remove-SFInitiatorFromVolumeAccessGroup	31
Remove-SFNode	31
Remove-SFSnapshot	32
Remove-SFVolume	32
Remove-SFVolumeAccessGroup	32
Remove-SFVolumeFromVolumeAccessGroup	32

Set-SFAccount	33
Set-SFClusterConfig	33
Set-SFNetworkConfig	33
Set-SFNtpInfo	34
Set-SFSnmpAcl	34
Set-SFSnmpInfo	34
Set-SFSnmpTrapInfo	34
Set-SFVolume	35
Set-SFVolumeAccessGroup	35
Appendix B — Return Object Descriptions	36
Contacting SolidFire PowerShell Tools Support	51

Introduction

The SolidFire PowerShell Tools is a collection of Microsoft® Windows® PowerShell functions that use SolidFire API to control a SolidFire storage system. These functions allow administrators to query for information, make changes to objects in a storage system, and develop complex scripts on a single platform. Users can implement this module with other modules and snap-ins, such as VMware® PowerCLI and Cisco® UCS PowerTool, to extend capabilities throughout the infrastructure.

Any user with a SolidFire storage system and Windows PowerShell can take advantage of the SolidFire PowerShell Tools. Users of the SolidFire PowerShell Tools should have an understanding of Windows PowerShell functions. The SolidFire PowerShell Tools module can be obtained through the SolidFire Support [FTP](#) site.

Software Prerequisites

Component	Application	Description
PowerShell	PowerShell 4.0 or 5.0	Version 4.0* is the minimum recommended version to use with SolidFire PowerShell Tools. Functionality may vary on earlier versions. It is also recommended to additionally enable PowerShell 2.0 on your system. PowerShell 2.0 is a prerequisite for other PowerShell snap-ins and modules, such as PowerCLI and UCS PowerTool.
Operating system options**	Microsoft® Windows® 8.1	PowerShell is installed by default. Install the KB2883200 update.
	Microsoft® Windows® 7 SP1	PowerShell is supported but not installed.
	Windows® Server 2012 R2 64-bit	PowerShell is installed by default.
	Windows® Server 2008 R2 with SP1	PowerShell is supported but not installed. Install the PowerShell ISE role prior to installing Windows Management Framework (WMF) 4.0.
.NET framework		4.5.1 or later
SolidFire OS		Element versions 6, 7, and 8

*Additional components might be required in order to take full advantage of PowerShell 4.0 and the SolidFire PowerShell Tools. These components include WS-Management 3.0 and Windows Management Instrumentation (WMI) 3.0.

**The installer for SolidFire PowerShell Tools requires a 64-bit operating system to successfully complete installation.

Installing the SolidFire PowerShell Tools

SolidFire PowerShell Tools is a module that is imported into your PowerShell modules when you install it. When you open a PowerShell window after installation, SolidFire cmdlets will be available and can be invoked in the same way as other existing cmdlets.

Install the SolidFire PowerShell Tools according to the following procedure.

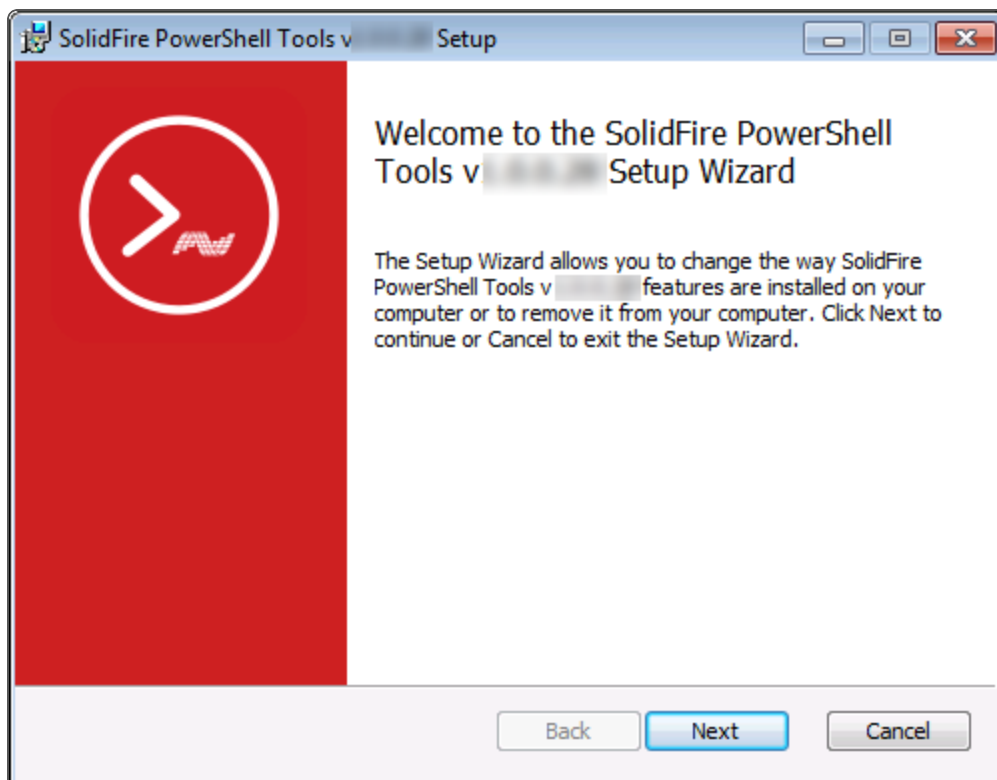
Prerequisites

- Administrative privileges to be able to complete the installation.
- 64-bit operating system needed to run the installer.
- Review [Software Prerequisites](#).

Procedure

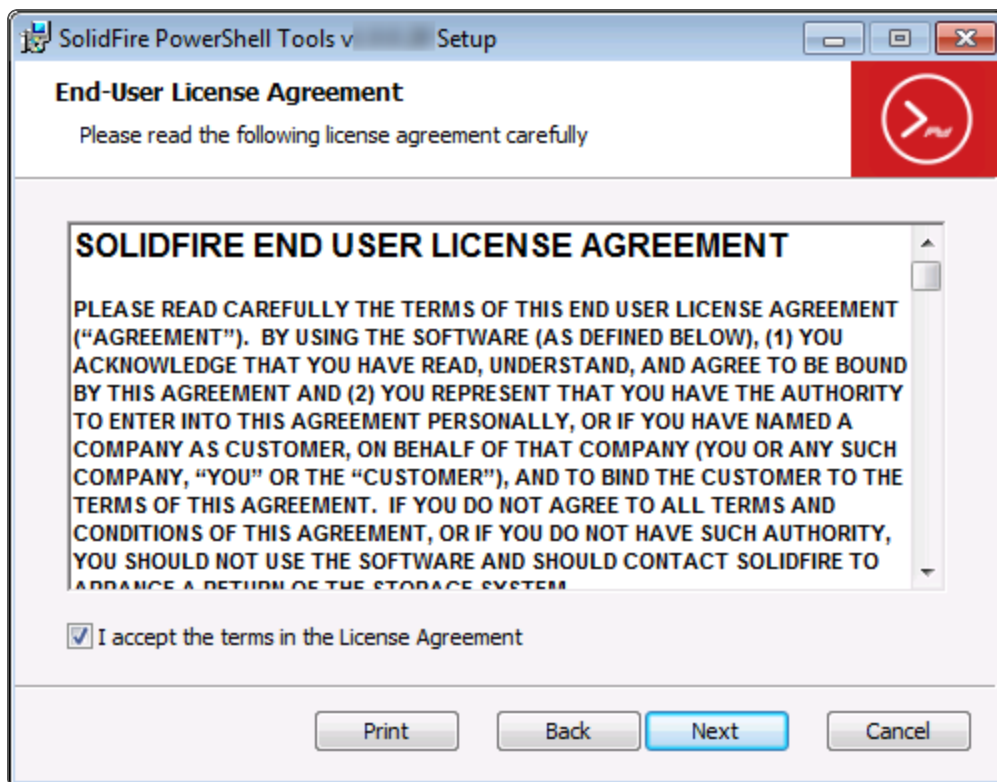
1. Download the installer from SolidFire [BrickFTP](#) to a local or network directory that is accessible to the Windows system from which you will be running PowerShell commands.
2. Double-click the `SolidFire_PowerShell_<version number>-install.msi` installer.

The *Welcome* window appears.



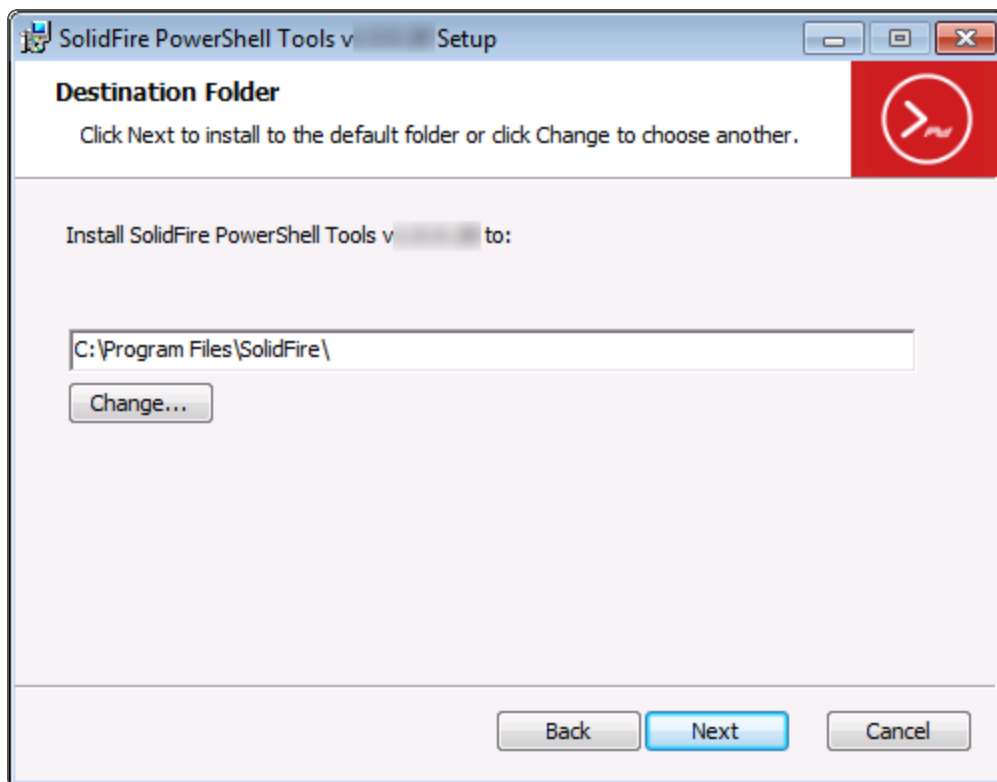
3. Click **Next**.

The *SolidFire End User License Agreement* appears.



4. Read the license agreement and select the check box to accept the terms of the agreement.
5. Click **Next**.

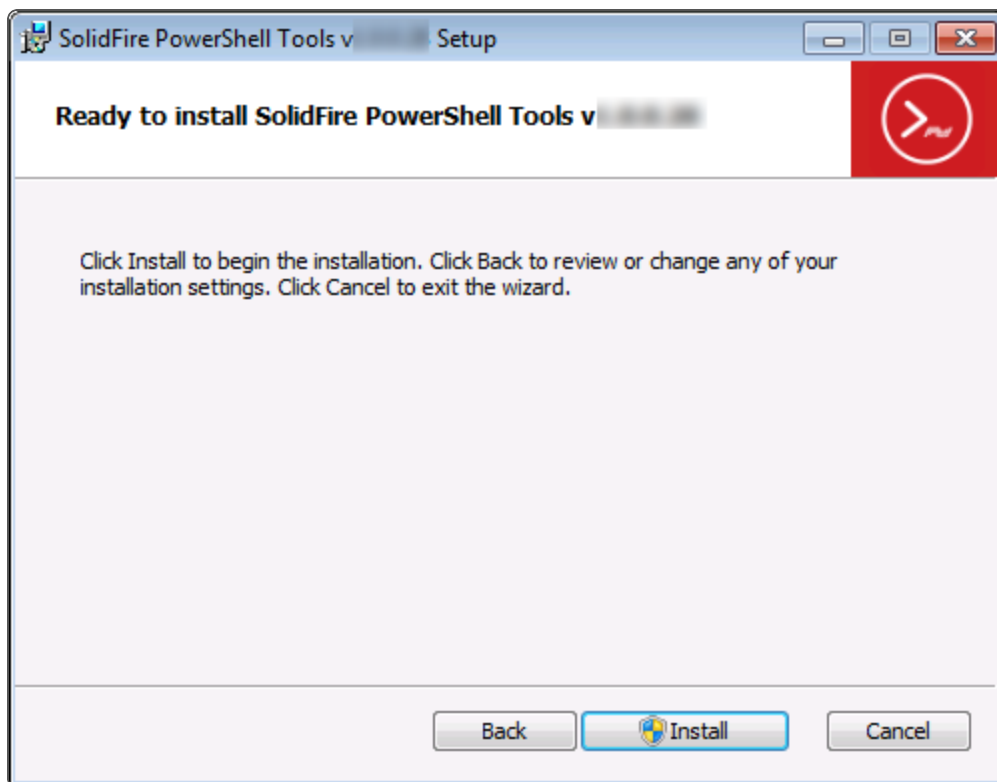
The *Destination Folder* window appears.



NOTE: By default, SolidFire PowerShell Tools installs to **C:\Program Files\SolidFire**. To change the installation location, click **Change** and provide the new location.

6. Click **Next**.

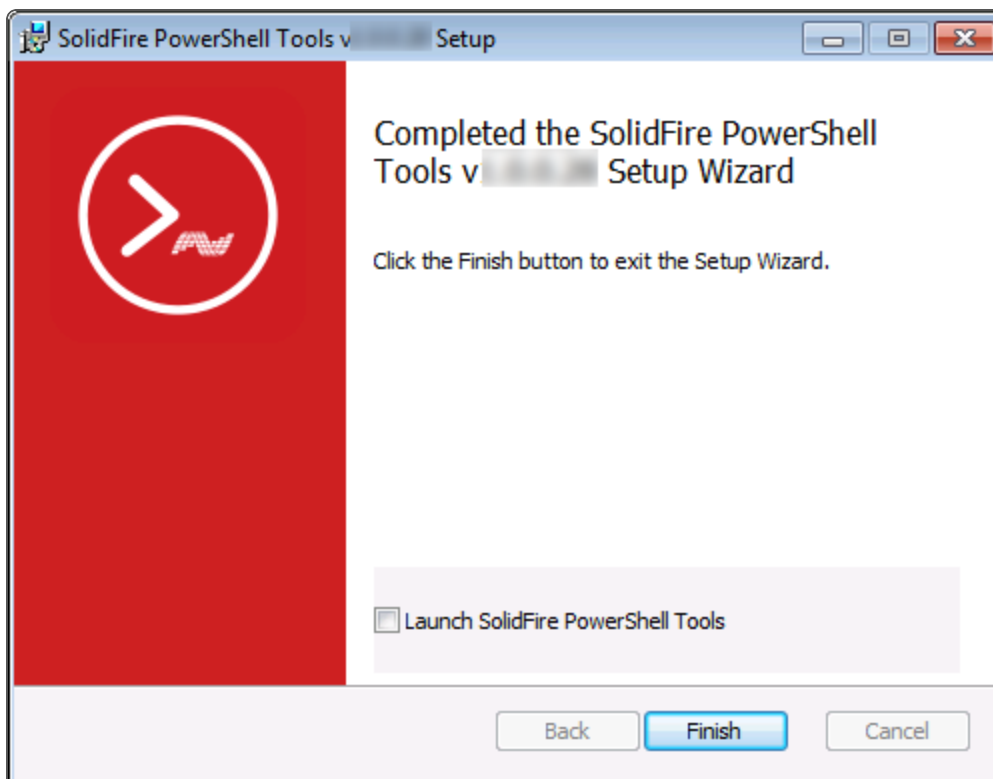
The *Ready to install* window appears.



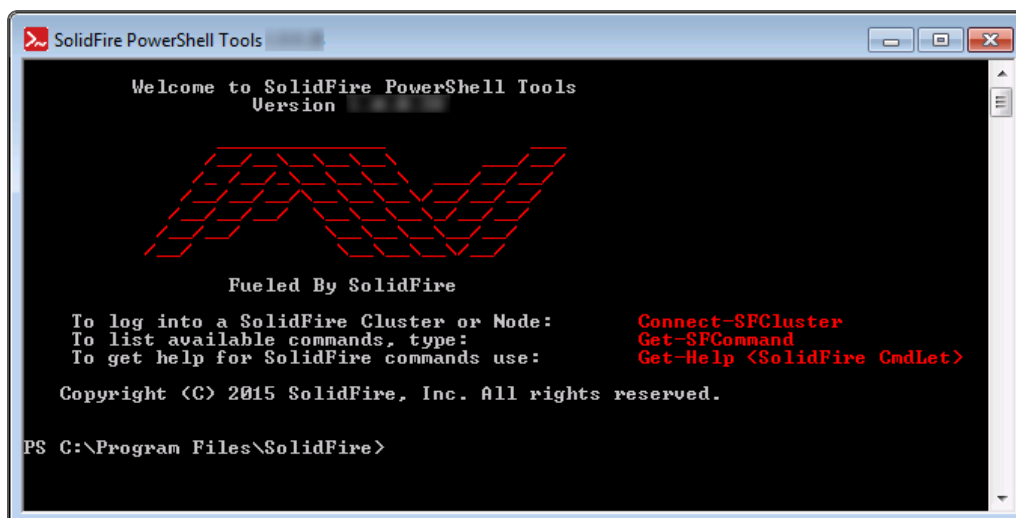
7. Click **Install**.

NOTE: You must have administrative privileges to complete the process.

The *Completed the SolidFire PowerShell Tools Setup Wizard* window appears after the installation has completed successfully.



8. Click **Launch SolidFire PowerShell Tools**.
9. Click **Finish**.
10. After successful installation, SolidFire PowerShell Tools opens in a customized PowerShell window.



Upgrading the SolidFire PowerShell Tools

To upgrade the SolidFire PowerShell Tools, download the latest MSI release from the SolidFire public [GitHub](#) repository or from [BrickFTP](#). Once the MSI is downloaded and brought into your existing PowerShell environment, double-click the MSI file and follow the installation prompts. For a description of the installation process, see [Installing the SolidFire PowerShell Tools](#).

How to Use SolidFire PowerShell Tools

The following topics describe the available functions for SolidFire PowerShell Tools, including accessing the cmdlet library, accessing embedded help, and managing connections to a SolidFire node.

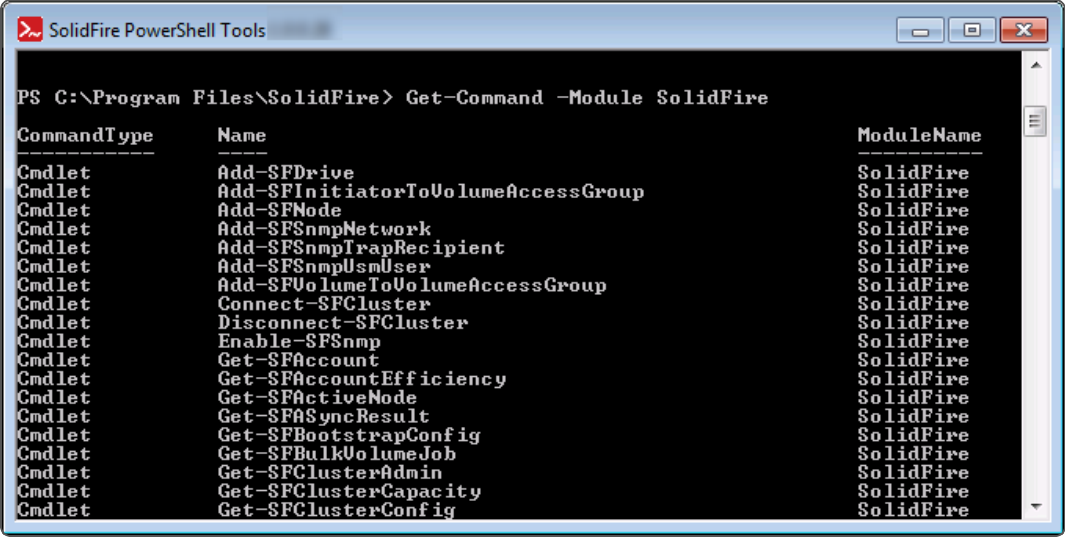
Listing Available Functions

The available functions for SolidFire PowerShell Tools can be explored using the native `Get-Command` PowerShell Tools cmdlet.

Procedure

1. In the command line interface, type `Get-Command -Module SolidFire`.

The list of available commands appears.

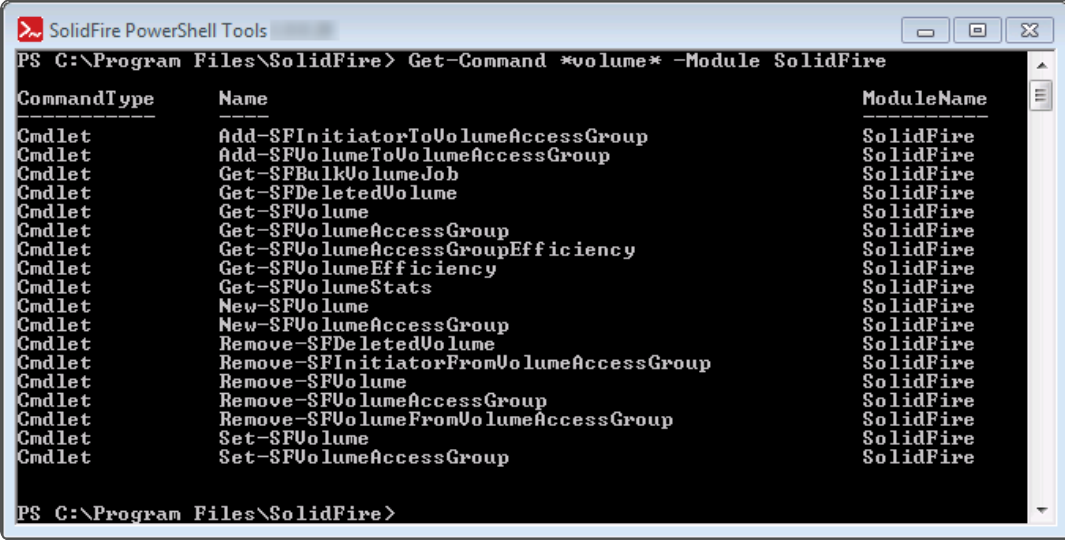


```
PS C:\Program Files\SolidFire> Get-Command -Module SolidFire
```

CommandType	Name	ModuleName
Cmdlet	Add-SFDrive	SolidFire
Cmdlet	Add-SFInitiatorToVolumeAccessGroup	SolidFire
Cmdlet	Add-SFNode	SolidFire
Cmdlet	Add-SFSnmpNetwork	SolidFire
Cmdlet	Add-SFSnmpTrapRecipient	SolidFire
Cmdlet	Add-SFSnmpUsmUser	SolidFire
Cmdlet	Add-SFVolumeToVolumeAccessGroup	SolidFire
Cmdlet	Connect-SFCluster	SolidFire
Cmdlet	Disconnect-SFCluster	SolidFire
Cmdlet	Enable-SFSnmp	SolidFire
Cmdlet	Get-SFAccount	SolidFire
Cmdlet	Get-SFAccountEfficiency	SolidFire
Cmdlet	Get-SFActiveNode	SolidFire
Cmdlet	Get-SFASyncResult	SolidFire
Cmdlet	Get-SFBootstrapConfig	SolidFire
Cmdlet	Get-SFBulkVolumeJob	SolidFire
Cmdlet	Get-SFClusterAdmin	SolidFire
Cmdlet	Get-SFClusterCapacity	SolidFire
Cmdlet	Get-SFClusterConfig	SolidFire

2. Type a search term with an asterisk before and after the term to filter the command list: `Get-Command *volume* -Module SolidFire`.

The filtered list of available commands appears.



```
PS C:\Program Files\SolidFire> Get-Command *volume* -Module SolidFire
```

CommandType	Name	ModuleName
Cmdlet	Add-SFInitiatorToVolumeAccessGroup	SolidFire
Cmdlet	Add-SFVolumeToVolumeAccessGroup	SolidFire
Cmdlet	Get-SFBulkVolumeJob	SolidFire
Cmdlet	Get-SFDeletedVolume	SolidFire
Cmdlet	Get-SFVolume	SolidFire
Cmdlet	Get-SFVolumeAccessGroup	SolidFire
Cmdlet	Get-SFVolumeAccessGroupEfficiency	SolidFire
Cmdlet	Get-SFVolumeEfficiency	SolidFire
Cmdlet	Get-SFVolumeStats	SolidFire
Cmdlet	New-SFVolume	SolidFire
Cmdlet	New-SFVolumeAccessGroup	SolidFire
Cmdlet	Remove-SFDeletedVolume	SolidFire
Cmdlet	Remove-SFInitiatorFromVolumeAccessGroup	SolidFire
Cmdlet	Remove-SFVolume	SolidFire
Cmdlet	Remove-SFVolumeAccessGroup	SolidFire
Cmdlet	Remove-SFVolumeFromVolumeAccessGroup	SolidFire
Cmdlet	Set-SFVolume	SolidFire
Cmdlet	Set-SFVolumeAccessGroup	SolidFire

```
PS C:\Program Files\SolidFire>
```

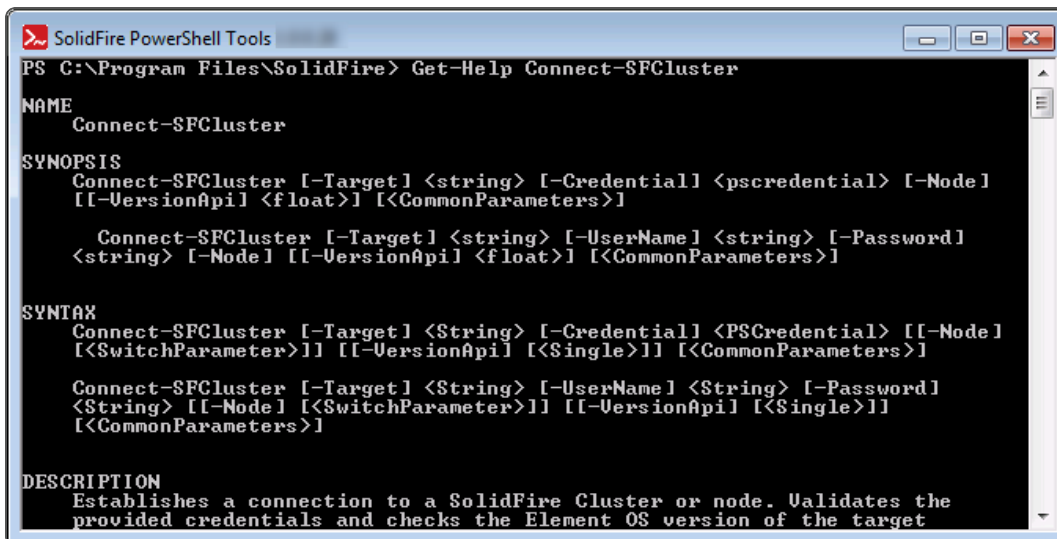
Accessing Embedded Help

SolidFire PowerShell Tools contains help examples that are accessible through the command line. Help content includes details about each command and examples of each function in use.

Procedure

1. In the command line interface, type `Get-Help <cmdlet>`.

The cmdlet description from embedded help appears.



```
PS C:\Program Files\SolidFire> Get-Help Connect-SFCluster

NAME
    Connect-SFCluster

SYNOPSIS
    Connect-SFCluster [-Target] <string> [-Credential] <pscredential> [-Node]
    [[-VersionApi] <float>] [<CommonParameters>]

    Connect-SFCluster [-Target] <string> [-UserName] <string> [-Password]
    <string> [-Node] [[-VersionApi] <float>] [<CommonParameters>]

SYNTAX
    Connect-SFCluster [-Target] <String> [-Credential] <PSCredential> [[-Node]
    [<SwitchParameter>]] [[-VersionApi] [<Single>]] [<CommonParameters>]

    Connect-SFCluster [-Target] <String> [-UserName] <String> [-Password]
    <String> [[-Node] [<SwitchParameter>]] [[-VersionApi] [<Single>]]
    [<CommonParameters>]

DESCRIPTION
    Establishes a connection to a SolidFire Cluster or node. Validates the
    provided credentials and checks the Element OS version of the target
```

Parameter Sets

Many of the functions for SolidFire PowerShell Tools have parameter sets to allow multiple use cases. For example, parameter sets are used with the creation and modification of SolidFire objects, such as Accounts, Volumes, and Volume Access Groups.

You can identify parameter sets by using `Get-Help` for the function and reviewing the content under the Syntax section.

Managing Connections to a SolidFire Cluster

All of the functions in the SolidFire PowerShell Tools make direct calls to the SolidFire API. In order to manage authentication efficiently, a connection function has been developed for collecting target and authentication information.

Connecting to a SolidFire Cluster

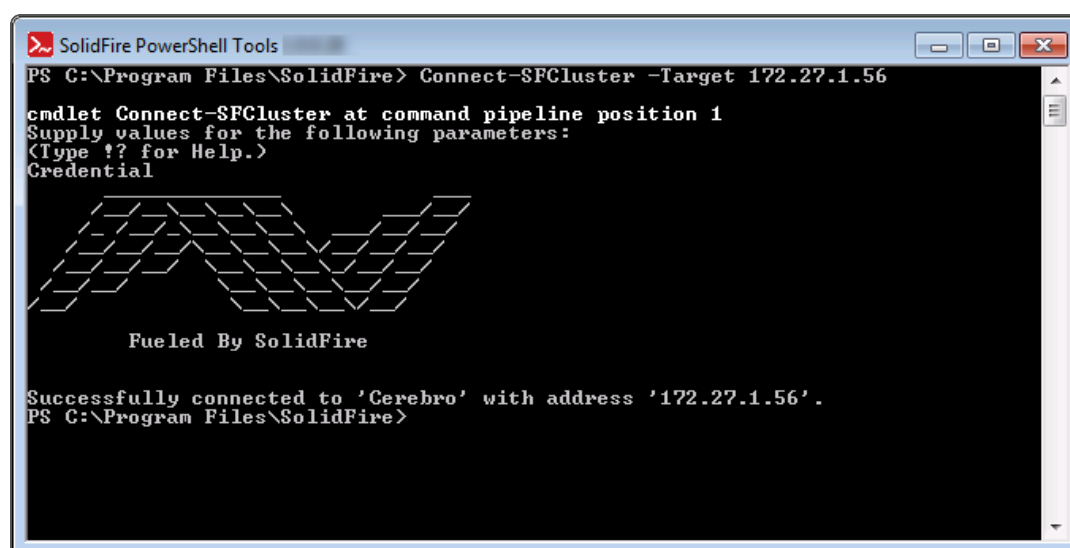
Use `Connect-SFCluster` to connect to a SolidFire cluster. The function collects SolidFire connection information, including target and authentication information from the user. `Connect-SFCluster` also supports connections to multiple SolidFire clusters.

By default, the `Connect-SFCluster` cmdlet queries the target cluster and sets the connection information to the latest API version on the cluster. This could also change the URI property to the version of Element OS you are using. See [Changing API Versions](#) to specify an API.

Procedure

1. In the command line interface, type `Connect-SFCluster -Target <address>`.

The following example shows a successful connection.



NOTE: If your connection to a SolidFire cluster is successful, the function `Connect-SFCluster` stores credentials and target information into a global variable `$SFConnection`. Multiple connections are also supported, and each successful connection is stored in the global array variable `$SFconnections`. See [Global Variables for All Functions](#).

Connecting to a SolidFire Node

Use the `Connect-SFCluster` function with a `-Node` switch parameter to connect to a specific SolidFire node.

Procedure

1. In the command line interface, type `Get-SFNode | Select-Name, ManagementIP, NodeID` to get the IP address for the node.
2. Include `-Node` and provide the node IP address in order to connect.

```
Connect-SFCluster -Target <NodeIP> -UserName <AdminAccount> -Node:$true
```

Disconnecting from a SolidFire Cluster or Node

Use the `Disconnect-SFCluster` function to disconnect from a SolidFire cluster. The function also clears the `$SFConnection` and `$SFconnections` global variables from the session. This makes it easier to secure the shell if you wish to keep it active or work

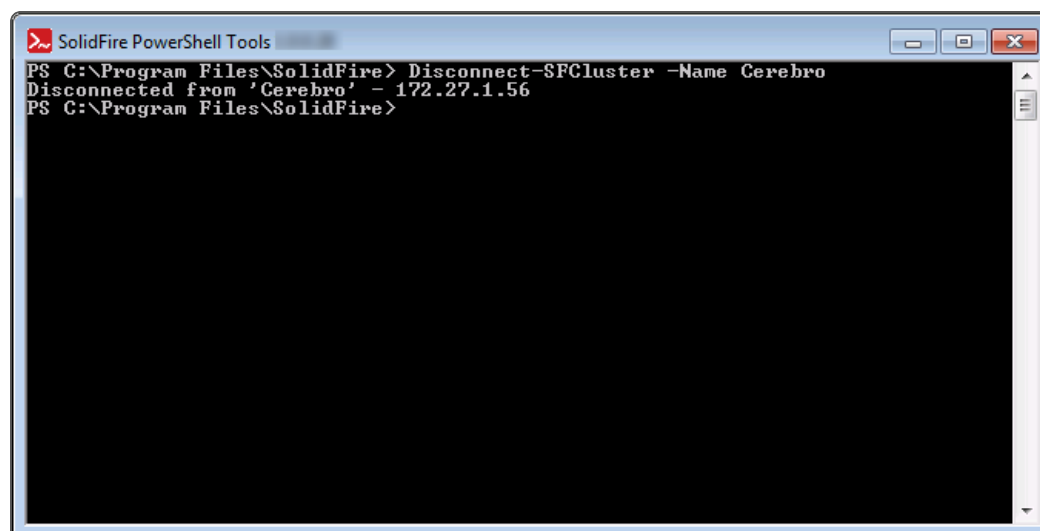
with a different SolidFire cluster.

You can disconnect a specific connection using the name of the connection from the `$SFConnection` or `$SFConnections` global variables. This name is either the cluster or the node name. See [Global Variables for All Functions](#) for an example.

Procedure

1. In the command line interface, type `Disconnect-SFCluster` to disconnect from the cluster. You can add an optional extension `-Name <node or cluster name>` or optional `-Target` parameter to specify the IP address instead of the name.

The following example shows a successful disconnection.



Changing API Versions

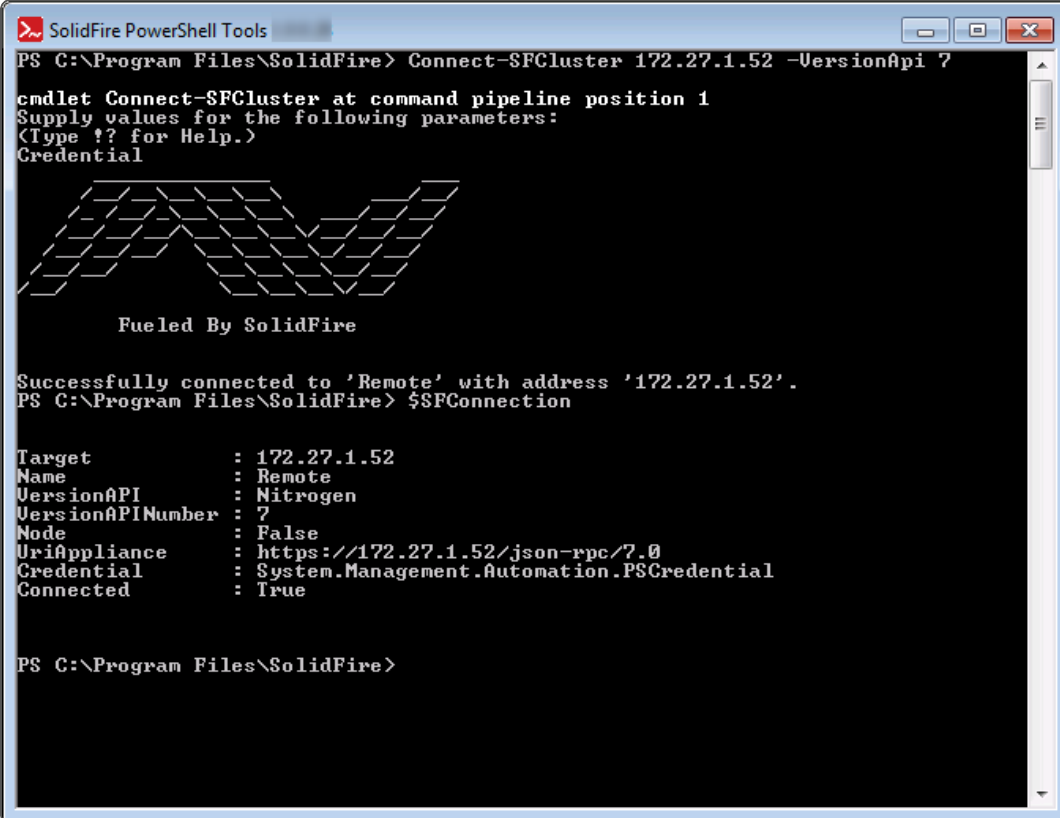
Use `-VersionApi` to specify an SolidFire API version. By default, the SolidFire PowerShell Tools module will use the most recent version of the SolidFire API available.

NOTE: Changing versions might produce unexpected results based on availability of features and possible API method changes between releases. Even if a connection to an API version works, not all new features might be available in the SolidFire PowerShell Tools module version that you have installed.

Procedure

1. In the command line interface, type `$cred = Get-Credential`.
2. Type `Connect-SFCluster -Target <address> -Credential $cred -VersionApi <version number>`.

The following example demonstrates connecting to a SolidFire cluster with API version 7.0 (Nitrogen).



```

PS C:\Program Files\SolidFire> Connect-SFCluster 172.27.1.52 -VersionAPI 7

cmdlet Connect-SFCluster at command pipeline position 1
Supply values for the following parameters:
(Type ?? for Help.)
Credential

Fueled By SolidFire

Successfully connected to 'Remote' with address '172.27.1.52'.
PS C:\Program Files\SolidFire> $SFConnection

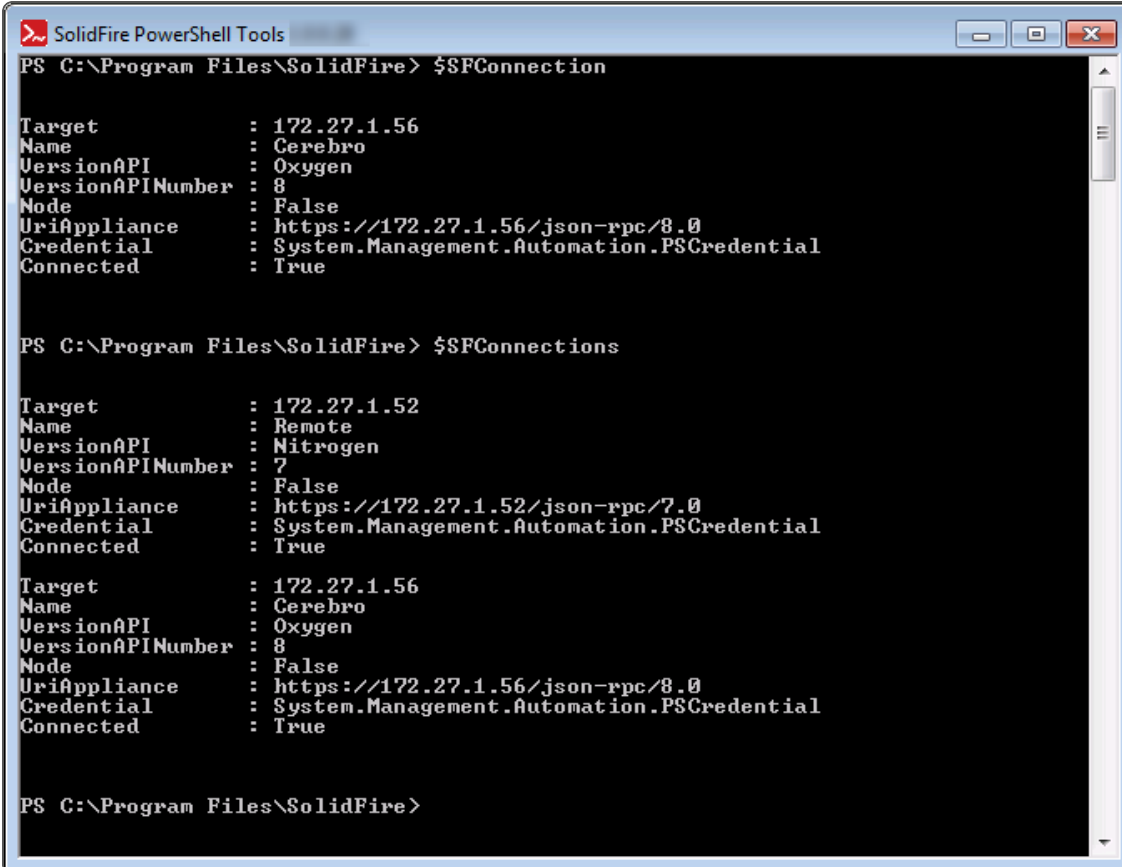
Target           : 172.27.1.52
Name              : Remote
VersionAPI        : Nitrogen
VersionAPINumber  : 7
Node              : False
UriAppliance      : https://172.27.1.52/json-rpc/7.0
Credential        : System.Management.Automation.PSCredential
Connected        : True

PS C:\Program Files\SolidFire>

```

Global Variables for All Functions

If your connection to a SolidFire cluster is successful, the function `Connect-SFCluster` stores credentials and target information in a global variable `$SFConnection`. The information in this variable is used in API calls of other SolidFire PowerShell Tools functions. Multiple connections are also supported, and each successful connection is stored in the global array variable `$SFConnections`.

A screenshot of a PowerShell console window titled "SolidFire PowerShell Tools". The window has a blue title bar and standard Windows window controls. The background is black with white text. The prompt is "PS C:\Program Files\SolidFire>". The first command is "\$SFConnection", which returns a list of properties for a single connection: Target (172.27.1.56), Name (Cerebro), VersionAPI (Oxygen), VersionAPINumber (8), Node (False), UriAppliance (https://172.27.1.56/json-rpc/8.0), Credential (System.Management.Automation.PSCredential), and Connected (True). The second command is "\$SFConnections", which returns a list of properties for two connections. The first connection has Target (172.27.1.52), Name (Remote), VersionAPI (Nitrogen), VersionAPINumber (7), Node (False), UriAppliance (https://172.27.1.52/json-rpc/7.0), Credential (System.Management.Automation.PSCredential), and Connected (True). The second connection has Target (172.27.1.56), Name (Cerebro), VersionAPI (Oxygen), VersionAPINumber (8), Node (False), UriAppliance (https://172.27.1.56/json-rpc/8.0), Credential (System.Management.Automation.PSCredential), and Connected (True). The prompt is "PS C:\Program Files\SolidFire>".

```
PS C:\Program Files\SolidFire> $SFConnection

Target      : 172.27.1.56
Name        : Cerebro
VersionAPI  : Oxygen
VersionAPINumber : 8
Node        : False
UriAppliance : https://172.27.1.56/json-rpc/8.0
Credential   : System.Management.Automation.PSCredential
Connected   : True

PS C:\Program Files\SolidFire> $SFConnections

Target      : 172.27.1.52
Name        : Remote
VersionAPI  : Nitrogen
VersionAPINumber : 7
Node        : False
UriAppliance : https://172.27.1.52/json-rpc/7.0
Credential   : System.Management.Automation.PSCredential
Connected   : True

Target      : 172.27.1.56
Name        : Cerebro
VersionAPI  : Oxygen
VersionAPINumber : 8
Node        : False
UriAppliance : https://172.27.1.56/json-rpc/8.0
Credential   : System.Management.Automation.PSCredential
Connected   : True

PS C:\Program Files\SolidFire>
```

Using SolidFire PowerShell Tools with Other Modules and Snap-ins

Several vendors have produced resources for PowerShell to help manage their solution. These resources include PowerCLI by VMware and the UCSPowerTool by Cisco. It is possible, when used in conjunction with the SolidFire PowerShell Tools, to report or automate multiple layers of the infrastructure within a single script. This requires having each module or snap-in loaded in the PowerShell session.

Appendix A — Available Cmdlets

This section describes all cmdlets that are available in SolidFire PowerShell Tools version 1.0. Cmdlet descriptions, syntax, and return values are included.

NOTE: For return object descriptions, see [Return Object Descriptions](#).

Common Parameters

All cmdlets, with the exception of `Connect-SFCluster` and `Disconnect-SFCluster`, have the common parameter `Target`. This common parameter is not in the following examples for each cmdlet but can be assumed to be present. The embedded Help within PowerShell Tools has the common parameter in its examples.

If the `-Target <String[]>` parameter is included in the cmdlet, the cmdlet will run against all connections in `$SFConnections` whose name or target (IP address) matches using a wildcard pattern match. Results are written to output as normal without indicating the target against which the cmdlet was run.

Each cmdlet is configured to be run on either a cluster or node. Before processing the cmdlet against any target, the cmdlet will check the connection to make sure it matches the intended cluster or node. If there is no match, a non-terminating error message (as in the following example) appears that states it is skipping the command:

`Get-SFNetworkConfig : Skipping command on connection 'Connection Name' . CmdLet requires Node connection .`

All cmdlets will execute against all matching connections.

Add-SFDrive

Adds one or more available drives to the cluster, enabling the drives to host a portion of the cluster's data. When a new drive is made available to the cluster, either through a newly added node or by being inserted into an existing node, it is marked as "available" and must be added using `Add-SFDrives` before it is utilized. Use the `Get-SFDrives` method to display drives that are "available" to be added. When multiple drives are being added, it is more efficient to add them in a single `Add-SFDrives` method rather than using multiple individual methods with a single drive. This reduces the amount of data balancing that must occur to even out the storage load on the cluster.

When adding a drive, the system automatically determines the "type" of drive it should be.

The method returns immediately, however it may take some time for the data in the cluster to be fully rebalanced using the newly added drives. As the new drive(s) are syncing on the system, you can use the `Get-SFSyncJob` method to monitor drive(s) rebalancing and the progress of adding the new drive.

Syntax

```
Add-SFDrive [-DriveID] <Int64[]>
```

Return Value

Nothing

Add-SFInitiatorToVolumeAccessGroup

Adds initiator(s) to the existing initiators in a volume access group. Initiators can only exist in a single volume access group. If an initiator exists in another volume access group, an error is thrown to notify the user.

Syntax

```
Add-SFInitiatorToVolumeAccessGroup [-VolumeAccessGroupID] <Int64> [-Initiators] <String[]>
```

Return Value

SFVolumeAccessGroup

Add-SFNode

Adds a pending node to a SolidFire cluster. Identify pending nodes using the `Get-SFPendingNode` cmdlet. It accepts piped input from `Get-SFPendingNode`.

Syntax

```
Add-SFNode [-PendingNodeID] <Int64[]>
```

Return Value

SFNodeWaitingToJoin[]

Add-SFSnmpNetwork

SNMP Network object is used to configure SNMP on the cluster nodes. SNMP v3 is supported on SolidFire clusters. This object can be retrieved with `Get-SFSnmpInfo` or `Get-SFSnmpAcl`.

To disconnect all sessions use the wildcard *

Syntax

```
Add-SFSnmpNetwork [-Access] <String> [-CIDR] <Int32> [-Community] <String> [-Network] <String>
```

Return Value

SFSnmpAcl

Add-SFSnmpTrapRecipient

SNMP Trap Recipient is a host that receives the traps generated by the Cluster Master.

Syntax

```
Add-SFSnmpTrapRecipient [-Host] <String> [-Community] <String> [-Port] <int32>
```

Return Value

SFSnmpTrapInfo

Add-SFSnmpUsmUser

SNMP v3 USM Users Object is used to configure SNMP on the cluster. SNMP v3 is supported on SolidFire clusters. This object can be retrieved with `Get-SFSnmpInfo` or `Get-SFSnmpAcl`.

Syntax

```
Add-SFSnmpUsmUser [-Access] <String> [-Name] <String> [-Password] <String> [-Passphrase] <String> [-SecLevel] <String>
```

Return Value

SFSnmpAcl

Add-SFVolumeToVolumeAccessGroup

Adds volume(s) to the existing volumes in a volume access group. Volumes can exist in multiple volume access groups.

Syntax

```
Add-SFVolumeToVolumeAccessGroup [-VolumeAccessGroupID] <Int64> [-VolumeID] <Int64 []>
```

Return Value

SFVolumeAccessGroup

Connect-SFCluster

Establishes a connection to a SolidFire cluster or node. Validates the credentials provided and checks the Element OS version of the target cluster. It then sets the `-VersionAPI` parameter to match the version of the cluster.

Connection information is stored in the `global:$SFConnection` variable.

To change the target API version, use the `-versionAPI` parameter.

Multiple connections are supported. Each successful connection is stored in the `global:$SFconnections` variable. The default connection used is present in the `global:$SFconnection` variable.

Syntax

```
Connect-SFCluster [-Target] <String> [-Credential] <PSCredential> [-Node] [[-VersionApi] <Single>]
```

```
Connect-SFCluster [-Target] <String> [-UserName] <String> [-Password] <String> [-Node] [[-VersionApi] <Single>]
```

Return Value

Fueled By SolidFire

Successfully connected to '[Name]' with address '[Target]'.

Disconnect-SFCluster

Used to disconnect a cluster or node in the session. Removes the connection from `global:$SFConnection` and `global:$SFConnections` respectively.

To disconnect all sessions use the wildcard `*`

Syntax

```
Disconnect-SFCluster [[-Target] <String[]>]
```

```
Disconnect-SFCluster [-Name] <String[]>
```

Return Value

Fueled By SolidFire

Successfully connected to '[Name]' with address '[Target]'.

Enable-SFSnmp

Enables SNMP on the cluster nodes. The values set with this interface apply to all nodes in the cluster, and the values that are passed replace, in whole, all values set in any previous call to `Enable-SFSnmp`.

Syntax

```
Enable-SFSnmp [-SnmpV3Enabled] <bool>
```

Return Value

`SFSnmpState`

Get-SFAccount

Gets information about volume accounts on a SolidFire cluster. Lists all active volume accounts on the cluster by default. To get an account by name, use the `-UserName` parameter or supply the name after the cmdlet. Wildcards can be used for the username, such as `J*` or `*oe*`. Multiple selections can be submitted. Separate selections with a comma.

Syntax

```
Get-SFAccount [[-AccountID] <Int64[]>]
```

```
Get-SFAccount [[-UserName] <String[]>]
```

Return Value

`SFAccount[]`

Get-SFAccountEfficiency

Retrieves storage efficiency information about a volume account. Only the account given as a parameter in this cmdlet is used to compute capacity.

Efficiency metrics include the following:

- Compression
- Deduplication
- Thin provisioning

To get the most recent metrics prior to the completion of a garbage collection event, use the `-ForceGet` parameter.

Syntax

```
Get-SFAccountEfficiency [-AccountID] <Int64[]> [-ForceGet]
```

```
Get-SFAccountEfficiency [-Account] <SFAccountInfoDetails[]> [-ForceGet]
```

Return Value

`SFEfficiency`

Get-SFActiveNode

Retrieves information about the active SolidFire nodes in the cluster. To collect information about a specific node, use either the node ID or the node's name.

Syntax

```
Get-SFActiveNode [[-NodeID] [<Int64[]>]]
```

```
Get-SFActiveNode [[-Name] [<String[]>]]
```

Return Value

```
SFNode[]
```

Get-SFAsyncResult

Retrieves the result of asynchronous method calls. Some method calls are long running and do not complete when the initial response is sent. To obtain the result of the method call, polling with `Get-SFAsyncResult` is required. `Get-SFAsyncResult` returns the overall status of the operation (in progress, completed, or error) in a standard fashion, but the actual data returned for the operation depends on the original method call, and the return data is documented with each method. The result for a completed asynchronous method call can only be retrieved once. Once the final result has been returned, later attempts return an error.

Syntax

```
Get-SFAsyncResult [-AsyncResultID] <String>
```

Return Value

```
SFAsyncResult
```

Get-SFBootstrapConfig

Returns the cluster name and node name from the bootstrap configuration file. This API method should be performed on an individual node before it has been configured into a cluster. The resulting information from this method is used in the Cluster Configuration UI when the cluster is created.

Syntax

```
Get-SFBootstrapConfig
```

Return Value

```
SFBootstrapConfigInfo
```

Get-SFBulkVolumeJob

Returns information about each bulk volume read or write operation that is occurring in the system.

Syntax

```
Get-SFBulkVolumeJob [[-VolumeID] <Int64[]>]]
```

```
Get-SFBulkVolumeJob [[-Volume] <SFVolumeInfo[]>]]
```

Return Value

```
SFBulkVolumeJobInfo[]
```

Get-SFClusterAdmin

Returns the list of all cluster administrators for the cluster. There can be several cluster administrators that have different levels of permissions. There can be only one primary cluster administrator in the system. The primary Cluster Admin is the administrator that was created when the cluster was created. LDAP administrators can also be created when setting up an LDAP system on the cluster.

Syntax

```
Get-SFClusterAdmin
```

Return Value

```
SFClusterAdmin[]
```

Get-SFClusterCapacity

Returns high-level capacity measurements for an entire cluster. The fields returned from this method can be used to calculate efficiency rates that are displayed in the Element user interface. The efficiency calculations can be used in scripts to return the efficiency rates for thin provisioning, deduplication, compression, and overall efficiency. Refer to the SolidFire API for efficiency calculation formulas.

Syntax

```
Get-SFClusterCapacity
```

Return Value

```
SFClusterCapacity
```

Get-SFClusterConfig

Returns information about the cluster configuration the node uses to communicate with its cluster.

Syntax

```
Get-SFClusterConfig
```

Return Value

```
SFClusterConfig
```

Get-SFClusterFault

Gets information about any faults detected on the cluster. Both current faults as well as faults that were previously detected and resolved can be returned. Faults in the system are cached every 30 seconds.

Syntax

```
Get-SFClusterFault [[-FaultType] <String[]>]
```

Return Value

```
SFClusterFault[]
```

Get-SFClusterFullThreshold

Retrieves the stages that are set for cluster fullness levels. All levels are returned when this cmdlet is entered.

There are five stages of cluster fullness:

- **stage1Happy** – No alerts or error conditions.
- **stage2AwareThreshold** – Three nodes of capacity available.
- **stage3LowThreshold** – Two nodes of capacity available.
- **stage4CriticalThreshold** – One node of capacity available. No new volumes or clones can be created.
- **stage5CompletelyConsumedThreshold** – Completely consumed. Cluster is read-only and iSCSI connection is maintained but all writes are suspended.

Syntax

```
Get-SFClusterFullThreshold
```

Return Value

```
SFClusterFullThresholdInfo
```

Get-SFClusterInfo

Retrieves configuration information about the cluster.

Syntax

```
Get-SFClusterInfo
```

Return Value

```
SFClusterInfo
```

Get-SFClusterMasterNodeID

Returns the ID of the node that can perform cluster-wide administrative tasks and holds the storage virtual IP (SVIP) and management virtual IP (MVIP).

Syntax

```
Get-SFClusterMasterNodeID
```

Return Value

```
SFClusterMasterNode
```

Get-SFClusterStats

Returns high-level activity measurements for the cluster. Values returned are cumulative from the creation of the cluster.

Syntax

```
Get-SFClusterStats
```

Return Value

```
SFClusterStats
```

Get-SFClusterVersionInfo

Returns information about the Element software version running on each node in the cluster. Information about nodes that are currently in the process of upgrading software is also returned.

Syntax

```
Get-SFClusterVersionInfo
```

Return Value

```
SFClusterVersionInfo
```

Get-SFConfig

Retrieves all configuration information for the node. This cmdlet includes the same information available in both `Get-SFClusterConfig` and `Get-SFNetworkConfig` API methods.

Syntax

```
Get-SFConfig
```

Return Value

```
SFConfigNode
```

Get-SFDefaultQoS

Retrieves the default QoS values that are set for a volume if QoS is not supplied.

Syntax

```
Get-SFDefaultQoS
```

Return Value

```
SFDefaultQoS
```

Get-SFDeletedVolume

Returns the entire list of volumes that have been marked for deletion or purged from the system. The results can be filtered by `VolumeID` or `Name`.

Syntax

```
Get-SFDeletedVolume [-VolumeID] [<Int64[]>]
```

```
Get-SFDeletedVolume [-Name] [<String[]>]
```

Return Value

```
SFVolumeInfo[]
```

Get-SFDrive

Retrieves the list of the drives that exist in the cluster's active nodes. This list includes drives that have been added as volume metadata or block drives as well as drives that have not been added and are available are returned.

Syntax

```
Get-SFDrive [[-DriveID] [<Int64[]>]]
```

Return Value

```
SFDrive[]
```

Get-SFDriveConfig

Displays drive information for expected slice and block drive counts as well as the number of slices and block drives that are currently connected to the node.

Syntax

```
Get-SFDriveConfig
```

Return Value

```
SFDriveConfig
```

Get-SFDriveHardwareInfo

Returns all the hardware info for the given drive. This generally includes manufacturers, vendors, versions, and other associated hardware identification information.

Syntax

```
Get-SFDriveHardwareInfo [-DriveID] <Int64[]>
```

Return Value

```
SFDriveHardwareInfo
```

Get-SFDriveStats

Retrieves high-level activity measurements for a SolidFire drive. Values are cumulative from the addition of the drive to the cluster. Some values are specific to block drives. Statistical data might not be returned for both block and metadata drives when running this cmdlet.

Syntax

```
Get-SFDriveStats [-DriveID] <Int64>
```

Return Value

```
SFDriveStats
```

Get-SFEvent

Retrieves events detected on the cluster, sorted from oldest to newest.

Syntax

```
Get-SFEvent [[-StartEventId] [<Int64>]] [[-EndEventId] [<Int64>]]
```

Return Value

SFEvent

Get-SFGroupSnapshot

Returns information about all group snapshots that have been created.

Syntax

```
Get-SFGroupSnapshot [[-GroupSnapshotID] <Int64[]>]]
```

```
Get-SFGroupSnapshot [[-GroupSnapshotName] <String[]>]]
```

```
Get-SFGroupSnapshot [[-VolumeID] <Int64[]>]]
```

```
Get-SFGroupSnapshot [[-Volume] <SFVolumeInfo[]>]]
```

Return Value

SFGroupSnapshot[]

Get-SFIscsiSession

Returns information about the iSCSI session connection for each volume.

Syntax

```
Get-SFIscsiSession
```

Return Value

SFIscsiSession

Get-SFNetworkConfig

Displays the network configuration information for a node.

Syntax

```
Get-SFNetworkConfig
```

Return Value

SFNetworkConfig

Get-SFNtpInfo

Returns the current network time protocol (NTP) configuration information.

Syntax

```
Get-SFNtpInfo
```

Return Value

SFNtpInfo

Get-SFPendingNode

Retrieves any pending SolidFire nodes that could be added to the cluster. Pending nodes are running and configured to join the cluster, but have not yet been added using the `Add-SFNode` cmdlet. To get information on multiple specific pending nodes, you can use an array of node IDs or names.

Syntax

```
Get-SFPendingNode [[-PendingNodeID] [<Int64[]>]]
```

```
Get-SFPendingNode [[-Name] [<String[]>]]
```

Return Value

SFPendingNode

Get-SFSnapshot

Returns the attributes of each snapshot taken on the volume. Information about snapshots that reside on the target cluster will be displayed on the source cluster when this method is called from the source cluster.

Syntax

```
Get-SFSnapshot [[-SnapshotID] [<Int64[]>]]
```

```
Get-SFSnapshot [[-SnapshotName] [<String[]>]]
```

```
Get-SFSnapshot [[-VolumeID] [<Int64[]>]]
```

```
Get-SFSnapshot [[-Volume] [<SFVolumeInfo[]>]]
```

Return Value

SFSnapshot[]

Get-SFSnmpAcl

Returns the current SNMP access permissions on the cluster nodes. It returns an array of networks and `usrUsers`.

Syntax

```
Get-SFSnmpAcl
```

Return Value

SFSnmpAcl

Get-SFSnmpInfo

Returns the current simple network management protocol (SNMP) configuration information.

Syntax

```
Get-SFSnmpInfo
```

Return Value

SFSnmpInfo

Get-SFSnmpState

Returns the current state of the SNMP feature.

Syntax

```
Get-SFSnmpState
```

Return Value

SFSnmpState

Get-SFSnmpTrapInfo

Returns current SNMP trap configuration information.

Syntax

```
Get-SFSnmpTrapInfo
```

Return Value

SFSnmpTrapInfo

Get-SFSyncJob

Returns information about synchronization jobs that are running on a SolidFire cluster. Synchronization jobs that are returned with this method are "slice," "clone," and "remote."

Syntax

```
Get-SFSyncJob
```

Return Value

SFSyncJob[]

Get-SFVolume

Returns a list of volumes that are in a cluster. You can specify the volumes you want to return in the list by using the available parameters. If you want deleted volumes included in the return list, use the `IncludeDeleted` switch.

Syntax

```
Get-SFVolume [-VolumeID] <Int64[]> [-IncludeDeleted]
Get-SFVolume [-VolumeName] <String[]> [-IncludeDeleted]
Get-SFVolume [-AccountID] <Int64[]> [-IncludeDeleted]
Get-SFVolume [-Account] <SFAccount[]> [-IncludeDeleted]
```

Return Value

SFVolume[]

Get-SFVolumeAccessGroup

Returns information about a SolidFire volume access group. If no `VolumeAccessGroupID` or `VolumeAccessGroupName` is specified, information for all volume access groups is returned.

Syntax

```
Get-SFVolumeAccessGroup [-VolumeAccessGroupID] [<Int64[]>]
```

```
Get-SFVolumeAccessGroup [-VolumeAccessGroupName] [<String[]>]
```

Return Value

SFVolumeAccessGroup[]

Get-SFVolumeAccessGroupEfficiency

Returns efficiency information about all volumes in a volume access group. Efficiency information is related to all volumes in the volume access group.

Efficiency metrics include the following:

- Compression
- Deduplication
- Thin provisioning

Syntax

```
Get-SFVolumeAccessGroupEfficiency [-VolumeAccessGroupID] <Int64[]>
```

```
Get-SFVolumeAccessGroupEfficiency [-VolumeAccessGroup] <SFVolumeAccessGroupInfo[]>
```

Return Value

SFEfficiency

Get-SFVolumeEfficiency

Retrieves storage efficiency information about a volume. Only the volume given as a parameter in this cmdlet is used to compute the capacity.

Efficiency metrics include the following:

- Compression
- Deduplication
- Thin provisioning

To get the most recent metrics prior to the completion of a garbage collection event, use the `-ForceGet` parameter.

Syntax

```
Get-SFVolumeEfficiency [-VolumeID] <Int64> [[-ForceGet] [<SwitchParameter>]]
```

Return Value

SFEfficiency

Get-SFVolumeStats

Returns high-level activity measurements for every volume by volume ID, account ID, access group ID, or access group. Values are cumulative from the creation of the volume.

Syntax

```
Get-SFVolumeStats [-VolumeID] <Int64[]>]
```

```
Get-SFVolumeStats [-AccountID] <Int64[]>]
```

```
Get-SFVolumeStats [-VolumeAccessGroupID] <Int64[]>]
```

```
Get-SFVolumeStats [-VolumeAccessGroup] [<SFVolumeAccessGroupInfo[]>]
```

Return Value

SFVolumeStats[]

Invoke-SFSecureEraseDrive

Removes any residual data from drives that have a status of "available." This can be used when, for example, replacing a drive that contained sensitive data at its end-of-life. `Invoke-SFSecureEraseDrive` uses a Security Erase Unit command to write a predetermined pattern to the drive and resets the encryption key on the drive. The method may take up to two minutes to complete, so it is an asynchronous method. `Get-SFAsyncResult` can be used to check on the status of the secure erase operation. Use the `Get-SFDrive` method to obtain the drive IDs for the drives you want to secure erase.

Syntax

```
Invoke-SFSecureEraseDrive [-DriveID] <Int64[]>
```

```
Invoke-SFSecureEraseDrive [-Drive] <SFDrive[]>
```

Return Value

SFAsyncHandle

New-SFAccount

Adds a new account to the system. New volumes can be created under the new account. The CHAP settings specified for the account apply to all volumes owned by the account.

Syntax

```
New-SFAccount [-UserName] <String> [[-InitiatorSecret] [<String>]] [[-TargetSecret] [<String>]] [[-Attributes] [<Hashtable>]]
```

Return Value

SFAccount[]

New-SFClone

Creates a copy of the volume. This method is asynchronous and takes a variable amount of time to complete. The cloning process begins when the `New-SFClone` cmdlet is called that is representative of the state of the volume at the time when the API method is issued. `Get-SFAsyncResult` can be used to determine when the cloning process is complete and the new volume is available for connections. `Get-SFSyncJob` can be used to see the progress of the clone creation process.

NOTE: The initial attributes and quality of service settings for the volume are inherited from the volume being cloned. If different settings are required, change settings using `Set-SFVolume`.

NOTE: Cloned volumes do not inherit volume access group membership(s) from the source volume.

Syntax

```
New-SFClone [-VolumeID] <Int64> [-Name] <String> [[-NewAccountID] [<Int64>]] [[-NewSize] [<Int64>]]
[[-Access] [<String>]] [[-SnapshotID] [<Int64>]] [[-Attributes] [<Hashtable>]]
```

Return Value

SFCloneResponse

New-SFCloneMultiple

Creates a clone of a group of specified volumes. A consistent set of characteristics can be assigned to a group of multiple volumes when those volumes are cloned together.

If `groupSnapshotID` will be used to clone the volumes in a group snapshot, the group snapshot must be first created using `New-SFGroupSnapshot` or the SolidFire Element Web UI. Using `groupSnapshotID` is optional when cloning multiple volumes.

NOTE: Cloning multiple volumes is allowed if cluster fullness is at stage 2 or 3. Clones cannot be created when cluster fullness is at stage 4 or 5.

Syntax

```
New-SFCloneMultiple [-Volume] <SFCloneVolume[]> [[-Access] [<String>]] [[-GroupSnapshotID]
[<Int64>]] [[-NewAccountID] [<Int64>]]
```

```
New-SFCloneMultiple [-VolumeID] <Int64[]> [[-Access] [<String>]] [[-GroupSnapshotID] [<Int64>]] [[-
NewAccountID] [<Int64>]]
```

Return Value

SFCloneMultipleResponse

New-SFCluster

The `CreateCluster` method is used to initialize the node in a cluster that has ownership of the MVIP and SVIP addresses. Each new cluster is initialized using the MIP of the first node in the cluster. This method also automatically adds all nodes being configured into the cluster. The method is used only once each time a new cluster is initialized.

NOTE: You need to log into the node that is used as the master node for the cluster. Once logged in, run the `Get-SFBootstrapConfig` cmdlet on the node to get the IP addresses for the remaining nodes that you want to include in the cluster. After you have the IP addresses, run the `New-SFCluster` cmdlet.

Syntax

```
New-SFCluster [-Mvip] <String> [-Svip] <String> [-Username] <String> [-Password] <String> [-Nodes]
<String[]> [[-ReplicaCount] <Int32>]] [[-Attributes] [<Hashtable>]]
```

Return Value

Nothing

New-SFClusterAdmin

Adds a new Cluster Admin. A Cluster Admin can be used to manage the cluster using the API and management tools. Cluster Admins are completely separate and unrelated to standard tenant accounts.

Each Cluster Admin can be restricted to a subset of the API. SolidFire recommends using multiple Cluster Admins for different users and applications. Each Cluster Admin should be given the minimal permissions necessary to reduce the potential impact of credential compromise.

`Get-SFClusterAdmin` returns a list of all cluster administrators after the addition.

Syntax

```
New-SFClusterAdmin [-UserName] <String> [-Password] <String> [-Access] <String[]> [[-Attributes]
[<Hashtable>]]
```

Return Value

`SFClusterAdmin[]`

New-SFGroupSnapshot

Creates a point-in-time copy of a group of volumes. The snapshot can then be used later as a backup or rollback to ensure the data on the group of volumes is consistent for the point in time in which the snapshot was created.

NOTE: Snapshots cannot be created when cluster fullness is at stage 4 or 5. You can check cluster fullness with the `Get-SFClusterFullThreshold`.

Syntax

```
New-SFGroupSnapshot [-VolumeID] <Int64[]> [[-Name] <String>]] [[-Attributes] [<Hashtable>]]
```

Return Value

`SFGroupSnapshot`

New-SFSnapshot

Creates a point-in-time copy of a volume. A snapshot can be created from any volume or from an existing snapshot. If a `SnapshotID` is not provided with this API method, a snapshot is created from the volume's active branch. If the volume from which

the snapshot is created is being replicated to a remote SolidFire cluster, the snapshot can also be replicated to the same target.

NOTE: Creating a snapshot is allowed if cluster fullness is at stage 2 or 3. Snapshots cannot be created when cluster fullness is at stage 4 or 5.

Syntax

```
New-SFSnapshot [-VolumeID] <Int64> [[-SnapshotID] <Int64>]] [[-Name] [<String>]] [[-Attributes]
[<Hashtable>]]
```

```
New-SFSnapshot [-Volume] <SFVolumeInfo> [[-SnapshotID] <Int64>]] [[-Name] [<String>]] [[-
Attributes] [<Hashtable>]]
```

Return Value

SFSnapshot

New-SFVolume

Creates a new SolidFire volume. Requires that an account and size be specified. The QoS profile does not have to be specified. If no values are provided, the system's DefaultQoS profile is used. Use the Get-SFDefaultQoS cmdlet to identify the default QoS profile. Burst time is static to 60 and cannot be changed.

Syntax

```
New-SFVolume [-Name] <String> [-AccountID] <Int64> [-Enable512e] [<Boolean>] [-TotalSize] <Int64> [[-
GB] [<SwitchParameter>]] [[-GiB] [<SwitchParameter>]] [[-Attributes] [<Hashtable>]] [[-MinIOPS]
[<Int32>]] [[-MaxIOPS] [<Int32>]] [[-BurstIOPS] [<Int32>]]
```

```
New-SFVolume [-Name] <String> [-Account] <SFAccount> [-Enable512e] [<Boolean>] [-TotalSize] <Int64>
[[[-GB] [<SwitchParameter>]] [[-GiB] [<SwitchParameter>]] [[-Attributes] [<Hashtable>]] [[-MinIOPS]
[<Int32>]] [[-MaxIOPS] [<Int32>]] [[-BurstIOPS] [<Int32>]]
```

Return Value

SFVolume

New-SFVolumeAccessGroup

Creates a new volume access group. The new volume access group must be given a name when it is created. Entering initiators and volumes is optional when creating a volume access group. Once the group is created, volumes and initiator IQNs can be added. Any initiator IQN that is successfully added to the volume access group is able to access any volume in the group without CHAP authentication.

Volume access group limits:

- A volume access group can contain up to sixty-four initiator IQNs.
- An initiator can only belong to only one volume access group.
- A volume access group can contain up to two-thousand volumes.
- Each volume access group can belong to a maximum of four other volume access groups.

Syntax

```
New-SFVolumeAccessGroup [-Name] <String> [[-Initiators] [<String[]>]] [[-VolumeIDs] [<Int64[]>]] [[-
Attributes] [<Hashtable>]]
```

Return Value

SFVolumeAccessGroup

Remove-SFAccount

Removes an existing account. All volumes must be deleted and purged on the account before the account can be removed. If deletion of any volume on the account is still pending, `Remove-SFAccount` cannot be used. Use `Remove-SFVolume` to delete and purge the volumes.

Syntax

```
Remove-SFAccount [-AccountId] <Int64[]>
```

```
Remove-SFAccount [-Account] <SFAccount[]>
```

Return Value

SFAccount[]

Remove-SFClusterAdmin

Removes a Cluster Admin. The primary Cluster Admin cannot be removed.

Syntax

```
Remove-SFClusterAdmin [-ClusterAdminID] <Int64[]>
```

```
Remove-SFClusterAdmin [-ClusterAdmin] <SFClusterAdmin[]>
```

Return Value

SFClusterAdmin[]

Remove-SFDeletedVolume

Immediately and permanently purges a volume which has been deleted. A volume must first be deleted using `Remove-SFVolume` before it can be purged. Volumes are purged automatically after a period of time, so usage of this method is not typically required.

Syntax

```
Remove-SFDeletedVolume [-VolumeID] <Int64[]>
```

Return Value

Nothing

Remove-SFDrive

Remove drives pro-actively that are part of the cluster. For example, when reducing cluster capacity or preparing to replace drives at their end-of-life, using the `Remove-SFDrive` method ensures that any data on the drives to be removed is migrated to other drives in the cluster prior to removal. This is an asynchronous method. Depending on the total capacity of the drives being removed, it may take several minutes to migrate all of the data. The `Get-SFAsyncResult` method can be used to check on the status of the remove operation. When multiple drives are being removed, it is more efficient to remove them in a single method rather than through multiple individual methods with a single drive. This reduces the amount of data balancing that must occur to even out the storage load on the cluster.

Drives with a "failed" status can also be removed using `Remove-SFDrive`. When a drive with a "failed" status is removed, it is not returned to an "available" or "active" status. The drive is unavailable for use in the cluster.

Use the `Get-SFDrive` method to obtain the drive IDs for the drives you want to remove.

Syntax

```
Remove-SFDrive [-DriveID] [<Int64[]>]
```

```
Remove-SFDrive [-Drive] [<SFDrive[]>]
```

Return Value

SFAsyncHandle

Remove-SFGroupSnapshot

Removes a group snapshot. The `SaveMembers` parameter can be used to preserve all the snapshots that were made for the volumes in the group, but the group association will be removed.

Syntax

```
Remove-SFGroupSnapshot [-GroupSnapshotID] <Int64[]> [[-SaveMembers] [<SwitchParameter>]]
```

Return Value

Nothing

Remove-SFInitiatorFromVolumeAccessGroup

Removes initiator(s) from the existing initiators in a volume access group. Initiators can exist in only one volume access group. This cmdlet supports an array or list of initiators.

Syntax

```
Remove-SFInitiatorFromVolumeAccessGroup [-VolumeAccessGroupID] <Int64> [-Initiators] <String[]>
```

Return Value

SFVolumeAccessGroup

Remove-SFNode

Removes one or more nodes that should no longer participate in the cluster. Before removing a node, all drives it contains must first be removed with `Remove-SFDrive`. A node cannot be removed until the `Remove-SFDrive` process has completed and all data has been migrated from the node.

Once all data is removed, a node registers itself as a pending node and can be added again or shut down, which removes it from the Pending Node list.

Syntax

```
Remove-SFNode [-NodeID] <Int64[]>
```

Return Value

Nothing

Remove-SFSnapshot

Deletes a snapshot. A snapshot that is the "active" snapshot cannot be deleted. You must rollback and make another snapshot "active" before the current snapshot can be deleted. After a snapshot is deleted, this cmdlet returns an array of all remaining SFSnapshots.

Syntax

```
Remove-SFSnapshot [-SnapshotID] <Int64[]>
```

Return Value

```
SFSnapshot[]
```

Remove-SFVolume

Marks an active volume for deletion. It is purged (permanently deleted) after the cleanup interval elapses.

After making a request to delete a volume, any active iSCSI connections to the volume are immediately terminated, and no further connections are allowed while the volume is in this state. The volume is not returned in target discovery requests.

Any snapshots of a volume that has been marked to delete are not removed until the volume is purged from the system. If a volume is marked for deletion, and it has a bulk volume read or bulk volume write operation in progress, the bulk volume read or write process is stopped.

If the volume you delete is paired with a volume, replication between the paired volumes is suspended and no data is transferred to it or from it while in a deleted state. The remote volume the deleted volume was paired with enters into a `PausedMisconfigured` state, and data is no longer sent to it or from the deleted volume. Until the deleted volume is purged, it can be restored and data transfers resumes. If the deleted volume is purged from the system, the volume it was paired with enters into a `StoppedMisconfigured` state and the volume pairing status is removed. The purged volume becomes permanently unavailable.

Syntax

```
Remove-SFVolume [-VolumeID] <Int64[]>
```

Return Value

```
Nothing
```

Remove-SFVolumeAccessGroup

Removes a volume access group from the cluster. Volume access groups can be removed even if there are active volumes associated with them.

Syntax

```
Remove-SFVolumeAccessGroup [-VolumeAccessGroupID] <Int64[]>
```

Return Value

```
Nothing
```

Remove-SFVolumeFromVolumeAccessGroup

Removes volume(s) from the existing volumes in a volume access group. Volumes can exist in multiple volume access groups.

Syntax

```
Remove-SFVolumefromVolumeAccessGroup [-VolumeAccessGroupID] <Int64> [-VolumeID] <Int64[]>
```

Return Value

```
SFVolumeAccessGroup
```

Set-SFAccount

Modifies an existing account. When locking an account, any existing connection from that account is immediately terminated. When changing CHAP settings, any existing connection continues to be active, and the new CHAP values are only used on subsequent connection or reconnection.

Syntax

```
Set-SFAccount [-AccountID] <Int64> [[-Status] [<String>]] [[-InitiatorSecret] [<String>]] [[-TargetSecret] [<String>]] [[-Attributes] [<Hashtable>]]
```

Return Value

```
SFAccount
```

Set-SFClusterConfig

Sets the configuration the node uses to communicate with the cluster it is associated with. You can only change the host name and Management Interface (mipi). To display the current cluster interface settings for a node, run the `Get-SFClusterConfig` cmdlet.

Syntax

```
Set-SFClusterConfig [[-Name] [<String>]] [[-Mipi] [<String>]]
```

Return Value

```
SFClusterConfig
```

Set-SFNetworkConfig

Sets the network configuration for a node. To see the states in which these objects can be modified, refer to the SolidFire API. To display the current network settings for a node, run `Get-SFNetworkConfig`.

Caution: Changing the "bond-mode" on a node can cause a temporary loss of network connectivity.

Caution: Changing the IP address of a node can cause a loss of network connectivity. Be prepared to reconnect if this command times out.

Syntax

```
Set-SFNetworkConfig [-Network] <String> [[-BondMode] [<String>]] [[-Address] [<String>]] [[-Netmask] [<String>]] [[-Gateway] [<String>]] [[-Mtu] [<Int64>]] [[-DnsNameservers] [<String[]>]] [[-DnsSearch] [<String[]>]] [[-Status] [<String>]]
```

Return Value

SFNetworkConfig

Set-SFNtpInfo

Configures the NTP on cluster nodes. The values set with this interface apply to all nodes in the cluster. The nodes can only be configured as a server where a host is selected to administrate the networking and/or a broadcast client where each host sends each message to each peer.

Syntax

```
Set-SFNtpInfo [-Servers] <string[]> [[-BroadcastClient] [<Boolean>]]
```

Return Value

SFNtpInfo

Set-SFSnmpAcl

Configures SNMP access permissions on the cluster nodes. The values set with this interface apply to all nodes in the cluster, and the values that are passed replace, in whole, all values set in any previous call to `Set-SFSnmpACL`. Values set with this interface replace all `Network` or `UsmUsers` values set with the older `Set-SFSnmpInfo`.

Syntax

```
Set-SFSnmpAcl [[-Networks] [<SFSnmpNetwork[]>]] [[-UsmUsers] [<SFSnmpV3User[]>]]
```

Return Value

SFSnmpAcl

Set-SFSnmpInfo

Configures SNMP v2 and v3 on the cluster nodes. The values set with this interface apply to all nodes in the cluster, and the values that are passed replace, in whole, all values set in any previous call to `Set-SFSnmpInfo`.

Syntax

```
Set-SFSnmpInfo [[-Networks] [<SFSnmpNetwork[]>]] [[-Enabled] [<Boolean>]] [[-SnmpV3Enabled] [<Boolean>]] [[-UsmUsers] [<SFSnmpV3User[]>]]
```

Return Value

SFSnmpInfo

Set-SFSnmpTrapInfo

Enables and disables the generation of SolidFire SNMP notifications (traps) and specifies the set of network host computers that are to receive the notifications. The values passed with each `Set-SFSnmpTrapInfo` method replaces all values set in any previous method to `Set-SFSnmpTrapInfo`.

Syntax

```
Set-SFSnmpTrapInfo [-TrapRecipients] <SFSnmpTrapRecipient[]> [[-ClusterFaultTrapsEnabled] [<Boolean>]] [[-ClusterFaultResolvedTrapsEnabled] [<Boolean>]] [[-ClusterEventTrapsEnabled]
```



```
[<Boolean>]]
```

Return Value

```
SFSnmpTrapInfo
```

Set-SFVolume

Modifies settings on an existing volume. Modifications can be made to one volume at a time, and changes take place immediately. When extending the size of a volume that is being replicated, the target "Replication Target" volume should first be increased in size and then the source "Read / Write" volume can be resized. It is recommended that both the target and the source volumes be the same size.

Syntax

```
Set-SFVolume [-VolumeID] <Int64[]> [[-AccountId] [<Int64>]] [[-Access] [<String>]] [[-Attributes]
[<Hashtable>]] [[-SetCreateTime] [<String>]] [[-MinIOPS] [<Int64>]] [[-MaxIOPS] [<Int64>]] [[-
BurstIOPS] [<Int64>]] [[-TotalSize] [<Int64>]] [[-GB] [<SwitchParameter>]] [[-GiB]
[<SwitchParameter>]]
```

Return Value

```
SFVolume
```

Set-SFVolumeAccessGroup

Updates the name or attributes of a volume access group. To add or remove initiators to a volume access group, use the `Add-SFInitiatorToVolumeAccessGroup` or `Remove-SFInitiatorToVolumeAccessGroup` cmdlets. To add or remove volumes in a volume access group, use the `Add-SFVolumeToVolumeAccessGroup` and `Remove-SFVolumeToVolumeAccessGroup`.

Syntax

```
Set-SFVolumeAccessGroup [-VolumeAccessGroupID] <Int64> [[-Name] [<String>]] [[-Attributes]
[<Hashtable>]]
```

Return Value

```
SFVolumeAccessGroup[]
```

Appendix B — Return Object Descriptions

SFSnmpAcl	
Networks	SFSnmpNetwork[]
UsmUsers	SFSnmpV3User[]
SFSnmpNetwork	
Access	String
CIDR	Int
Community	String
Network	String
SFSnmpV3User	
Access	String
Password	String
PassPhrase	String
SecLevel	String
SFSnmpTrapInfo	
TrapRecipients	SFSnmpTrapRecipient[]
ClusterFaultTrapsEnabled	Boolean
ClusterFaultResolvedTrapsEnabled	Boolean
ClusterEventTrapsEnabled	Boolean
SFSnmpTrapRecipient	
Host	String
Community	String
Port	Int
SFSnmpState	
Enabled	Boolean
SnmpV3Enabled	Boolean
SFSnmpInfo	
Enabled	Boolean
SnmpV3Enabled	Boolean
Networks	SFSnmpNetwork[]
UsmUsers	SFSnmpV3User[]
SFNtplInfo	
BroadcastClient	Boolean
Servers	String[]

SFAccount	
AccountID	Int64
UserName	String
AccountStatus	String
InitiatorSecret	String
TargetSecret	String
Attributes	Dictionary<String, String>
VolumeID	Int64[]
SFIscsiSession	
AccountID	Int64
AccountName	String
DriveID	Int64
InitiatorIP	String
InitiatorName	String
NodeID	Int64
ServiceID	Int64
SessionID	Int64
TargetName	String
TargetIP	String
VirtualNetworkID	Int64
VolumeID	Int64
SFCloneMultipleResponse	
AsyncHandle	Int64
GroupCloneID	Int64
Members	SFCloneMultipleMember[]
SFCloneMultipleMember	
SrcVolumeID	Int64
VolumeID	Int64
SFClusterAdmin	
Access	String[]
Attributes	Dictionary<String, String>
ClusterAdminID	Int64
UserName	String
SFSnapShot	
Attributes	Dictionary<String, String>

GroupID	Int64
Name	String
SnapshotID	Int64
Status	String
TotalSize	Int64
VolumeID	Int64
SFVolume	
Access	String
CreateTime	DateTime
DeleteTime	DateTime
PurgeTime	DateTime
AccountID	Int64
Attributes	Hashtable
Enable512e	Boolean
lqn	String
VolumeName	String
QoS	SFQoS
ScsiEUIDeviceID	String
ScsiNAADeviceID	String
Status	String
TotalSize	Int64
VolumeID	Int64
SliceCount	Int64
VolumeAccessGroups	Int64[]
VolumePairs	SFVolumePairInfo[]
SFQoS	
Curve	SFCurveForQoS
BurstTime	Int64
BurstIOPS	Int64
MaxIOPS	Int64
MinIOPS	Int64
SFVolumePairInfo	
ClusterPairID	Int64
RemoteVolumeID	Int64
RemoveVolumeName	String

VolumePairUUID	String
SFCurveForQoS	
IOPS_at_4K	Int64
IOPS_at_8K	Int64
IOPS_at_16K	Int64
IOPS_at_32K	Int64
IOPS_at_64K	Int64
IOPS_at_128K	Int64
IOPS_at_256K	Int64
IOPS_at_512K	Int64
IOPS_at_1024K	Int64
SFVolumeAccessGroupInfo	
VolumeAccessGroupName	String
VolumeAccessGroupID	Int64
Initiators	String[]
VolumeID	Int64[]
DeletedVolumeID	Int64[]
Attributes	Dictionary<String, String>
SFGroupSnapshot	
Attributes	Dictionary<String, String>
CreateTime	DateTime
GroupSnapshotID	Int64
Members	SFGroupSnapshotMember[]
Name	String
Status	String
SFGroupSnapshotMember	
Checksum	String
SnapshotID	Int64
VolumeID	Int64
SFAccountInfoDetail	
AccountID	Int64
UserName	String
AccountStatus	String
InitiatorSecret	String
TargetSecret	String

Attributes	Dictionary<String, String>
VolumeID	Int64[]
SFAsyncResult	
ID	Int64
Status	String
Error	SFError
Result	SFAsyncResultMessage
SFAsyncResultMessage	
Message	String
SFAsyncHandle	
AsyncHandle	Int64
SFError	
Code	Int64
Name	String
Message	String
SFBootstrapConfigInfo	
ClusterName	String
NodeName	String
Nodes	SFNodeWaitingToJoin[]
Version	String
SFNodeWaitingToJoin	
Name	String
Version	String
NodeID	Int64
PendingNodeID	Int64
Mip	IPAddress
Cip	IPAddress
Sip	IPAddress
SFBulkVolumeJobInfo	
Attributes	Dictionary<String, String>
BulkVolumeID	Int64
Format	String
CreateTime	DateTime
Key	String
PercentComplete	Int64

SourceVolumeID	Int64
Status	String
Type	String
SFClusterCapacity	
ActiveBlockSpace	Int64
ActiveSessions	Int64
AverageSessions	Int64
AverageIOPS	Int64
ClusterRecentIOSize	Int64
CurrentIOPS	Int64
MaxIOPS	Int64
MaxOverProvisionableSpace	Int64
MaxProvisionedSpace	Int64
MaxUsedMetadataSpace	Int64
MaxUsedSpace	Int64
NonZeroBlocks	Int64
PeakActiveSessions	Int64
PeakIOPS	Int64
ProvisionedSpace	Int64
TotalOps	Int64
TimeStamp	DateTime
UniqueBlocks	Int64
UniqueBlocksUsedSpace	Int64
UsedMetadataSpace	Int64
UsedMetadataSpaceInSnapshots	Int64
UsedSpace	Int64
ZeroBlocks	Int64
SFEfficiency	
Compression	Int64
DeDuplication	Double
ThinProvisioning	Double
TimeStamp	DateTime
SFClusterConfig	
Cipi	String
Cluster	String

Creator	String
Ensemble	String[]
Mipi	String
Name	String
NodeID	Int64
PendingNodeID	Int64
Role	String
Sipi	String
State	String
SFCluster	
CIP	String
ClusterName	String
Ensemble	String[]
Mip	String
HostName	String
NodeID	String
PendingNodeID	String
SIP	String
ClusterMembership	String
SFClusterFault	
ClusterFaultID	Int64
Code	String
Data	Object
Date	DateTime
Details	String
DriveID	Int64
NodeHardwareFaultID	Int64
NodeID	Int64
Resolved	Boolean
ResolvedDate	DateTime
ServiceID	String
Severity	String
Type	String
SFClusterFullThreshold	
BlockFullness	String

Fullness	String
MaxMetadataOverProvisionFactor	Int64
MetadataFullness	String
SliceReserveUsedThresholdPct	Int64
Stage2AwareThreshold	Int64
Stage3LowThreshold	Int64
Stage4CriticalThreshold	Int64
SumTotalClusterBytes	Int64
SumTotalMetadatClusterBytes	Int64
SumUsedClusterBytes	Int64
SumUsedMetadataClusterBytes	Int64
SFClusterInfo	
Attributes	Dictionary<String, String>
EncryptionAtRestState	String
Ensemble	String[]
Mvip	String
MvipNodeID	String
Name	String
RepCount	Int64
Svip	String
SvipNodeID	String
UniquelD	String
Uuid	String
SFClusterMasterNode	
NodeID	Int64
SFClusterStats	
ClusterUtilization	Single
ReadBytes	Int64
ReadOps	Int64
TimeStamp	DateTime
WriteBytes	Int64
WriteOps	Int64
SFClusterVersion	
ClusterVersionInfo	SFNodeVersionInfo[]
ClusterVersion	String

ClusterAPIVersion	String
SoftwareVersionInfo	SFSoftwareVersionInfo
SFNodeVersion	
NodeID	Int64
NodeVersion	String
NodeInternalRevision	String
SFSoftwareVersion	
CurrentVersion	String
NodeID	Int64
PackageName	String
PendingVersion	String
StartTime	DateTime
SFConfigNode	
Cluster	SFCluster
Network	SFNetwork
SFNetwork	
Bond10G	SFNetworkConfig
Bond1G	SFNetworkConfig
SFNetworkConfig	
IsDefault	Boolean
Address	String
Auto	Boolean
BondDownDelay	Int64
BondFailOverMac	String
BondPrimaryReselect	String
BondLacpRate	String
BondMiiMon	Int64
BondMode	String
BondSlaves	String
BondUpDelay	Int64
Broadcast	String
DnsNameservers	String
DnsSearch	String
Family	String
Gateway	String

MacAddress	String
MacAddressPermanent	String
Method	String
Mtu	String
Netmask	String
Network	String
Physical	SFPhysicalAdaptor
Routes	String[]
Status	String
SymmetricRouteRules	String[]
UpAndRunning	Boolean
SFPhysicalAdapter	
Address	String
MacAddress	String
MacAddressPermanent	String
Mtu	String
Netmask	String
Network	String
UpAndRunning	Boolean
SFDefaultQoS	
Curve	SFCurveForQoS
BurstTime	Int64
BurstIOPS	Int64
MaxIOPS	Int64
MinIOPS	Int64
SFDrive	
Attributes	Dictionary<String, String>
Capacity	Int64
DriveID	Int64
NodeID	Int64
Serial	String
Slot	Int64
Status	String
Type	String
SFDriveConfig	

NumBlockActual	Int64
NumBlockExpected	Int64
NumSliceActual	Int64
NumSliceExpected	Int64
NumTotalActual	Int64
NumTotalExpected	Int64
Drives	SFDriveConfigDetail[]
SFDriveConfigDetail	
CanonicalName	String
Connected	Boolean
Dev	Int64
DevPath	String
DriveType	String
FsType	String
IsMounted	Boolean
MountPoint	String
Name	String
Path	String
PathLink	String
Product	String
ScsiCompatID	String
SecurityEnabled	Boolean
SecurityFrozen	Boolean
SecurityLocked	Boolean
SecuritySupported	Boolean
Size	Int64
Slot	Int64
SmartSsdWriteCapable	Boolean
Uuid	String
Vendor	String
Version	String
SFDriveHardwareInfo	
Description	String
Dev	String
DevPath	String

DriveSecurityAtMaximum	Boolean
DriveSecurityFrozen	Boolean
DriveSecurityLocked	Boolean
LogicalName	String
Product	String
SecurityFeatureEnabled	String
SecurityFeatureSupported	String
Serial	String
Size	Int64
Uuid	String
Version	String
SFDriveStats	
ActiveSessions	Int64
FailedDieCount	Int64
LifeRemainingPercent	Int64
LifetimeReadBytes	Int64
LifetimeWriteBytes	Int64
PowerOnHours	Int64
ReadBytes	Int64
ReadOps	Int64
ReallocatedSectors	Int64
ReserveCapacityPercent	Int64
TimeStamp	DateTime
TotalCapacity	Int64
UsedCapacity	Int64
UsedMemory	Int64
WriteBytes	Int64
WriteOps	Int64
SFEvent	
Details	object
DriveID	Int64
EventID	Int64
EventInfoType	String
Message	String
NodeID	Int64

ServiceID	Int64
Severity	Int64
TimeOfReport	DateTime
TimeOfPublish	DateTime
SFNode	
AssociatedFServiceID	Int64
AssociatedMasterServiceID	Int64
Attributes	Dictionary<String, String>
Cip	String
Cipi	String
FibreChannelTargetPortGroup	String
MacAddressID	String
Mip	String
Mipi	String
Name	String
NodeID	Int64
PlatformInfo	SFPlatformInfo
Sip	String
Sipi	String
SoftwareVersion	String
Uuid	String
VirtualNetworks	SFVirtualNetwork[]
SoftwareInfo	SFSoftwareInfo
SFPendingNode	
AssignedNodeID	Int64
AssociatedFServiceID	Int64
AssociatedMasterServiceID	Int64
Attributes	Dictionary<String, String>
Compatible	Boolean
Cip	String
Cipi	String
FibreChannelTargetPortGroup	String
MacAddressID	String
Mip	String
Mipi	String

Name	String
NodeID	Int64
PendingNodeID	Int64
PlatformInfo	SFPlatformInfo
Sip	String
Sipi	String
SoftwareVersion	String
Uuid	String
VirtualNetworks	SFVirtualNetwork[]
SFVirtualNetwork	
Address	String
VirtualNetworkID	Int64
SFSoftwareInfo	
CurrentSoftware	SFCurrentSoftware[]
SFCurrentSoftware	
Description	String
PackageName	String
VersionNumber	String
SFPlatformInfo	
ChassisType	String
CpuModel	String
NodeMemoryGB	Int64
NodeType	String
SFSyncJob	
BytesPerSecond	Single
CurrentBytes	Int64
DstServiceID	Int64
ElapsedTime	Single
PercentComplete	Single
RemainingTime	Single
SliceID	Int64
SrcServiceID	Int64
TotalBytes	Int64
Type	String
CloneID	Int64

DstVolumeID	Int64
NodeID	Int64
SnapshotID	Int64
SrcVolumeID	Int64
BlocksPerSecond	Single
Stage	Int64
SFVolumeStats	
AccountID	Int64
ActualIOPS	Int64
AsyncDelay	String
AverageIOPSize	Int64
BurstIOPSCredit	Int64
ClientQueueDepth	Int64
NonZeroBlocks	Int64
ReadBytes	Int64
ReadLatencyUSec	Int64
ReadOps	Int64
Throttle	Single
TimeStamp	DateTime
TotalLatencyUSec	Int64
UnalignedReads	Int64
UnalignedWrites	Int64
VolumeAccessGroups	Int64[]
VolumeID	Int64
VolumeSize	Int64
VolumeUtilization	Single
WriteBytes	Int64
WriteLatencyUSec	Int64
WriteOps	Int64
ZeroBlocks	Int64
SFCloneResponse	
AsyncHandle	Int64
CloneID	Int64
Curve	SFCurveForQoS
VolumeID	Int64

Contacting SolidFire PowerShell Tools Support

If you have any questions or comments about this product, contact powershell@solidfire.com. Your feedback helps us focus our efforts on new features and capabilities.



1600 Pearl Street, Suite 200
Boulder, Colorado 80302

Phone: 720.523.3278

Email: info@solidfire.com

Web: www.solidfire.com

SolidFire Support: www.solidfire.com/support/

9/30/15