SOLIDFIRE

SolidFire PowerShell Tools User Guide

Version: 1.1

12/17/15

SolidFire Legal Notices

TABLE OF CONTENTS

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 BrickFTP 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.
	Microsoft® Windows® 10	PowerShell is installed by default.
	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

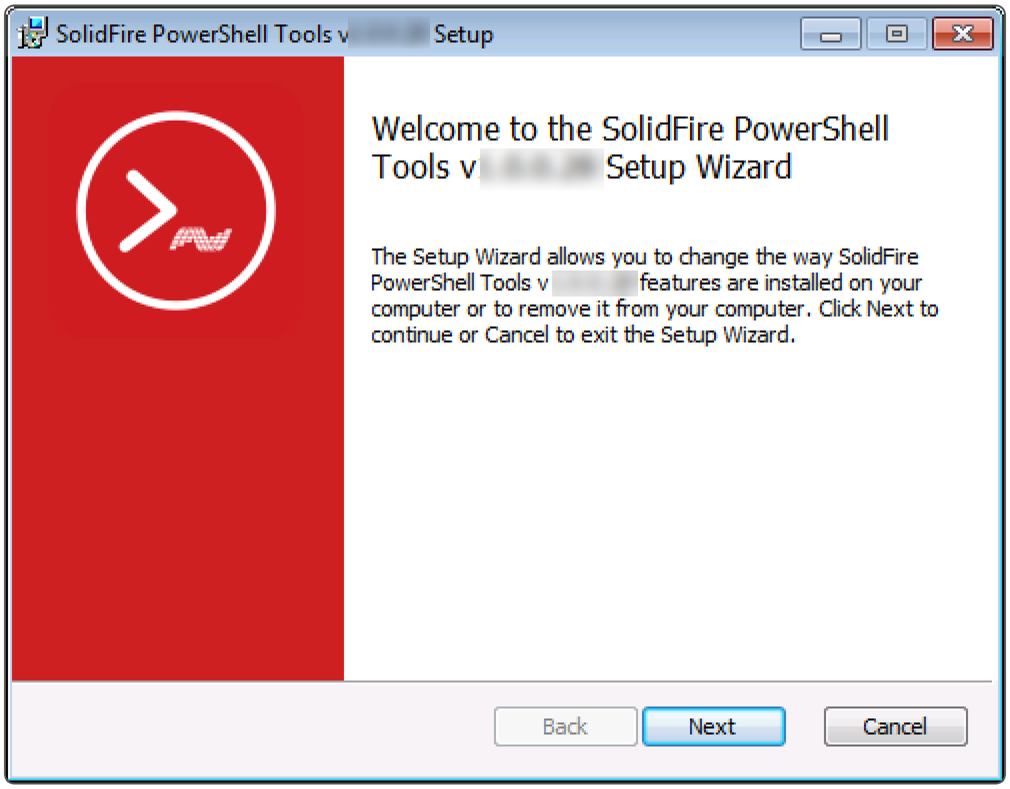
- Administrative privileges to be able to complete the installation.
- 64-bit operating system needed to run the installer.

- 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.



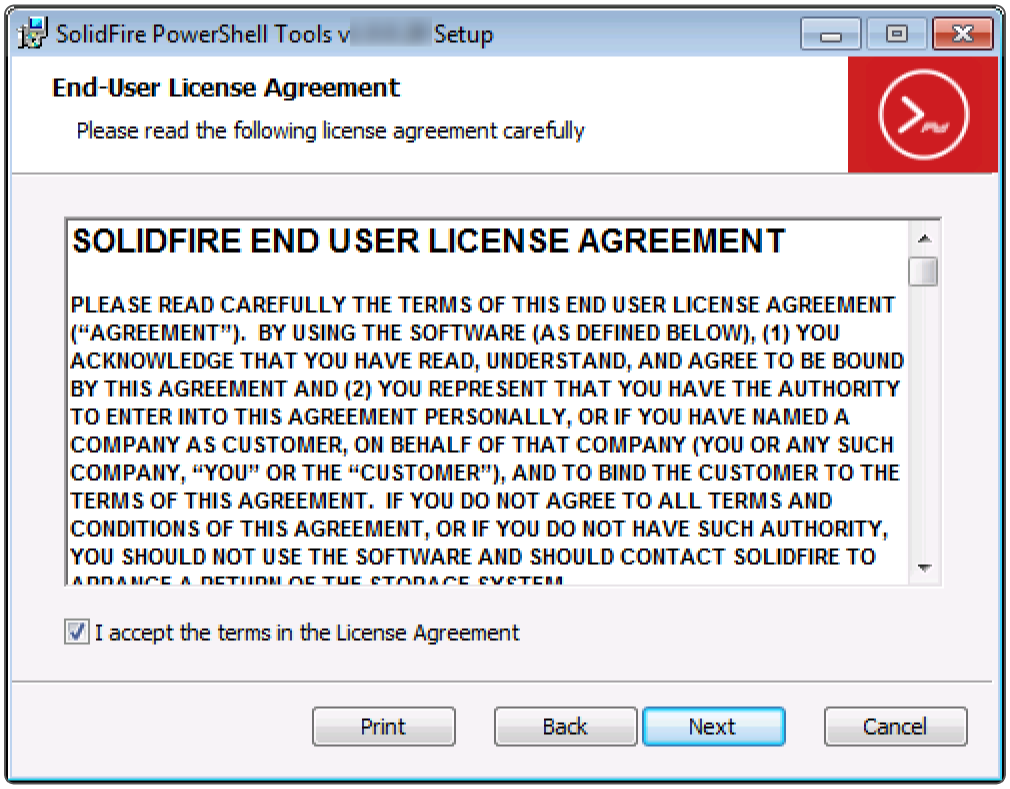
SolidFire PowerShell Tools v[redacted] Setup

Welcome to the SolidFire PowerShell Tools v[redacted] Setup Wizard

The Setup Wizard allows you to change the way SolidFire PowerShell Tools v[redacted] features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.

3. Click **Next**.

   The *SolidFire End User License Agreement* appears.


SolidFire PowerShell Tools v Setup
End-User License Agreement
Please read the following license agreement carefully

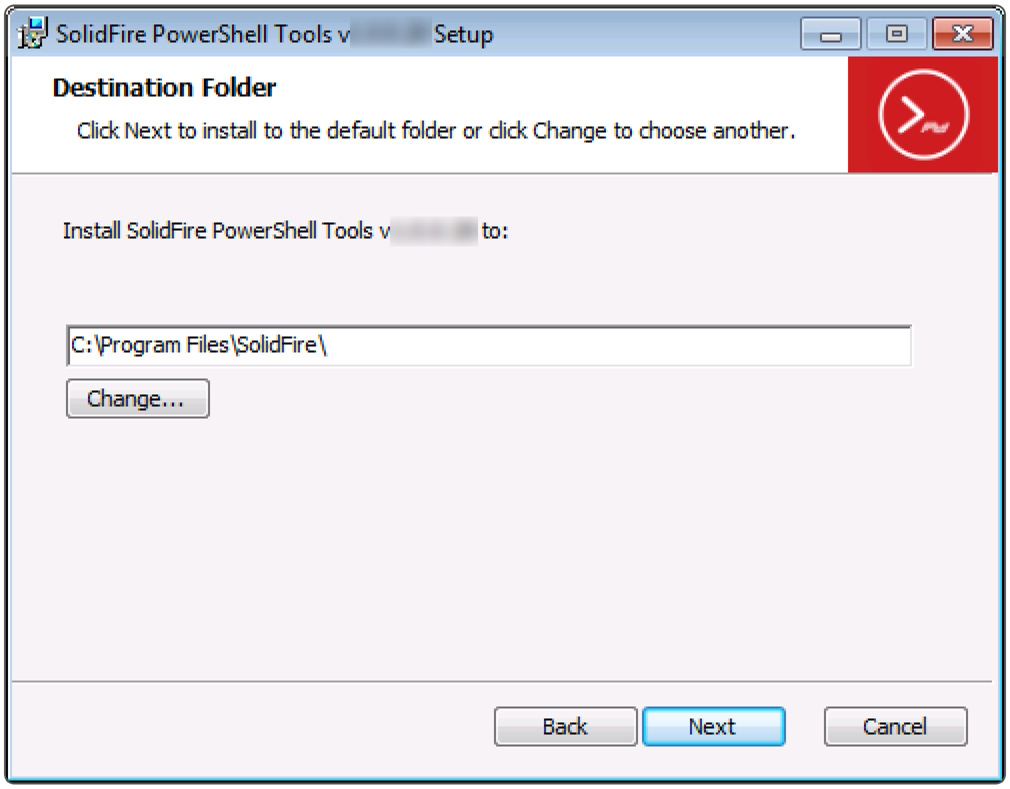
SOLIDFIRE END USER LICENSE AGREEMENT

PLEASE READ CAREFULLY THE TERMS OF THIS END USER LICENSE AGREEMENT ("AGREEMENT"). BY USING THE SOFTWARE (AS DEFINED BELOW), (1) YOU ACKNOWLEDGE THAT YOU HAVE READ, UNDERSTAND, AND AGREE TO BE BOUND BY THIS AGREEMENT AND (2) YOU REPRESENT THAT YOU HAVE THE AUTHORITY TO ENTER INTO THIS AGREEMENT PERSONALLY, OR IF YOU HAVE NAMED A COMPANY AS CUSTOMER, ON BEHALF OF THAT COMPANY (YOU OR ANY SUCH COMPANY, "YOU" OR THE "CUSTOMER"), AND TO BIND THE CUSTOMER TO THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL TERMS AND CONDITIONS OF THIS AGREEMENT, OR IF YOU DO NOT HAVE SUCH AUTHORITY, YOU SHOULD NOT USE THE SOFTWARE AND SHOULD CONTACT SOLIDFIRE TO

[✓] I accept the terms in the License Agreement

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.



Destination Folder

Click Next to install to the default folder or click Change to choose another.

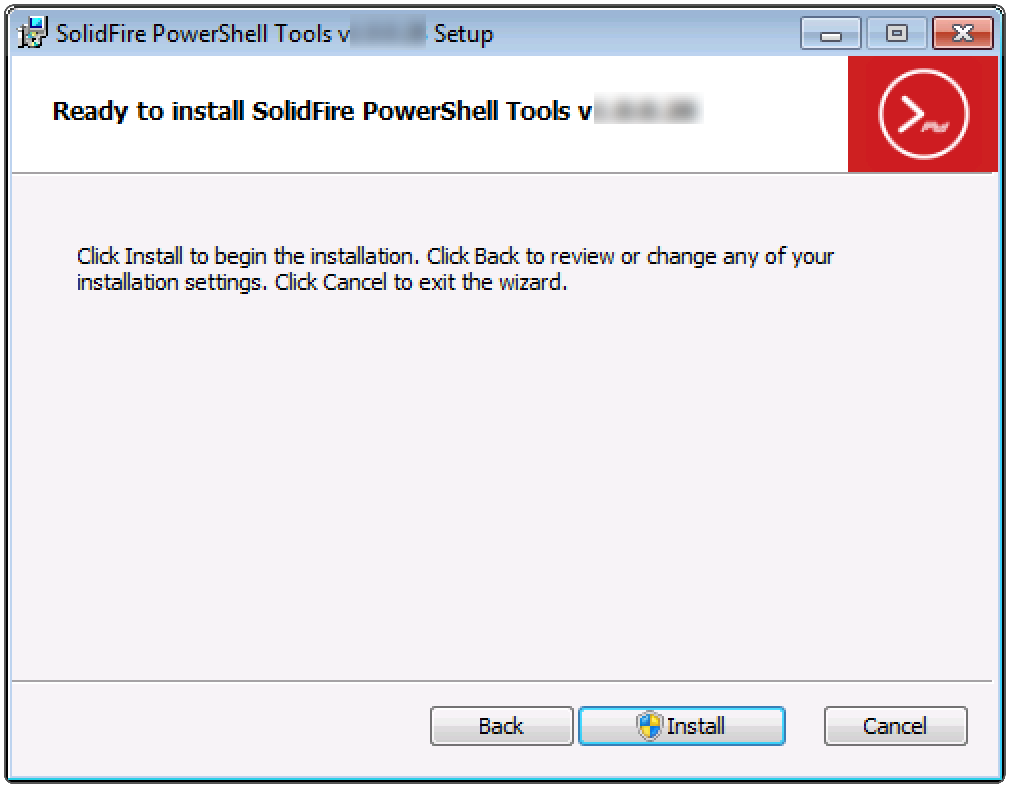
Install SolidFire PowerShell Tools v[redacted] to:

C:\Program Files\SolidFire\

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.


SolidFire PowerShell Tools v[redacted] Setup

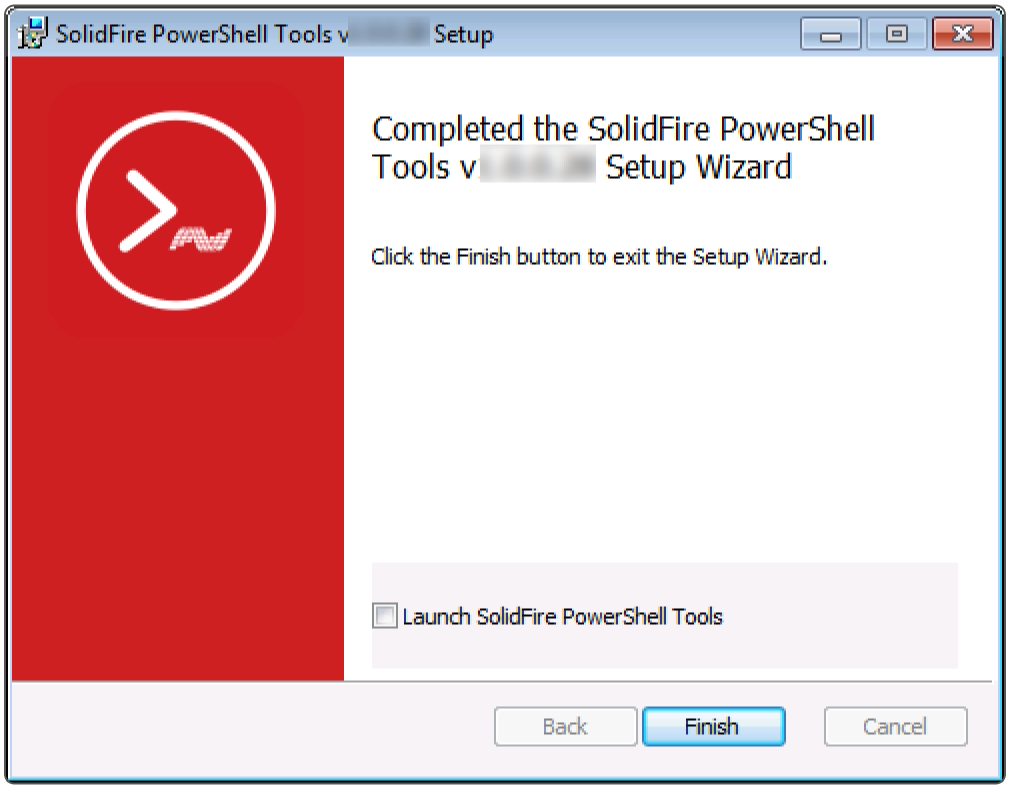
Ready to install SolidFire PowerShell Tools v[redacted]

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

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.

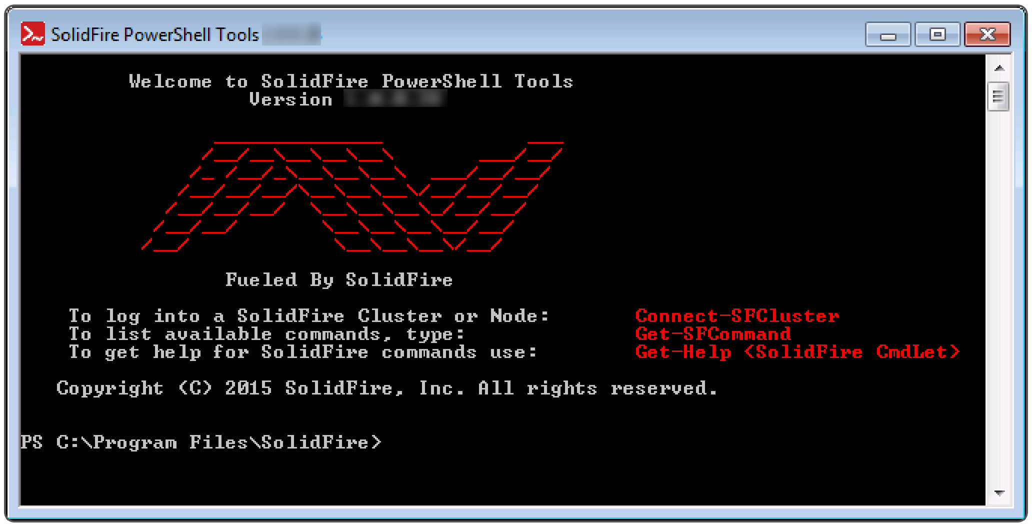

SolidFire PowerShell Tools v[redacted] Setup

Completed the SolidFire PowerShell Tools v[redacted] Setup Wizard

Click the Finish button to exit the Setup Wizard.

☐ Launch SolidFire PowerShell Tools

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


SolidFire PowerShell Tools [redacted]

```
         Welcome to SolidFire PowerShell Tools
                   Version [redacted]

                 Fueled By SolidFire

    To log into a SolidFire Cluster or Node:       Connect-SFCluster
    To list available commands, type:              Get-SFCommand
    To get help for SolidFire commands use:        Get-Help <SolidFire CmdLet>

   Copyright (C) 2015 SolidFire, Inc. All rights reserved.

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

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 ways to access available functions for SolidFire PowerShell Tools, manage connections to a SolidFire node, and find additional cmdlet parameter and return object information.

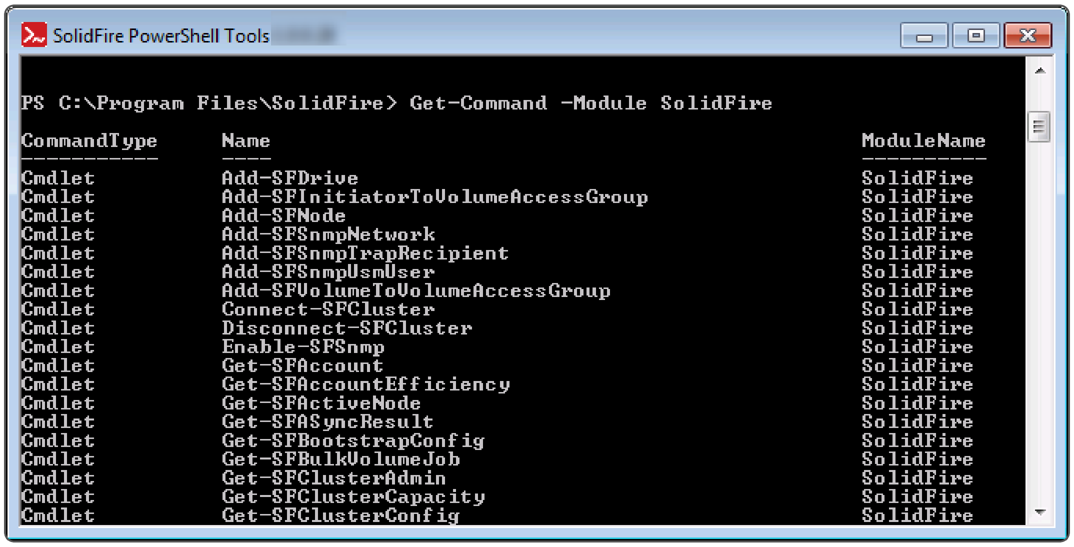
Listing Available Functions

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`.

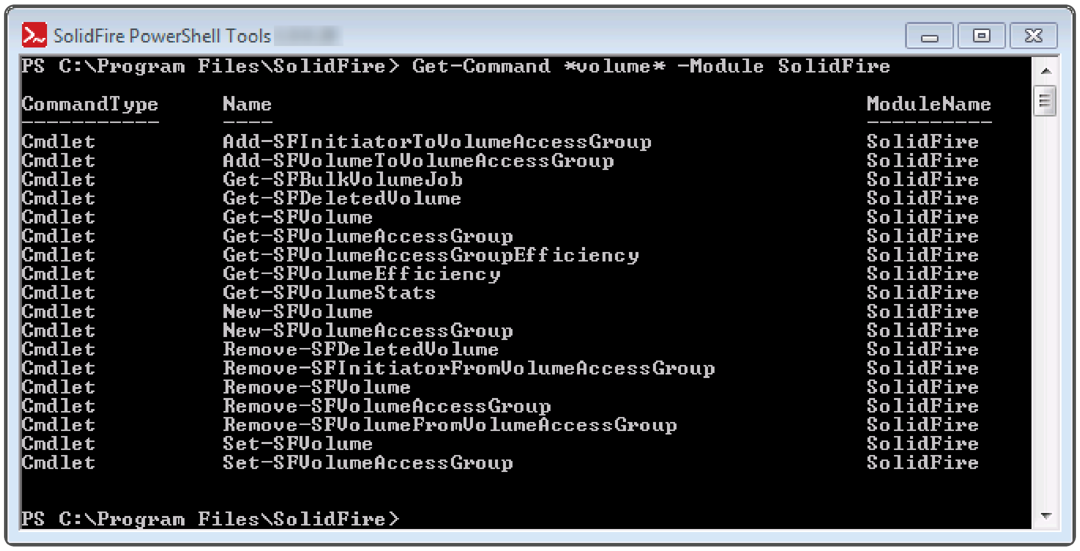
   The list of available commands appears.



```
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>
```

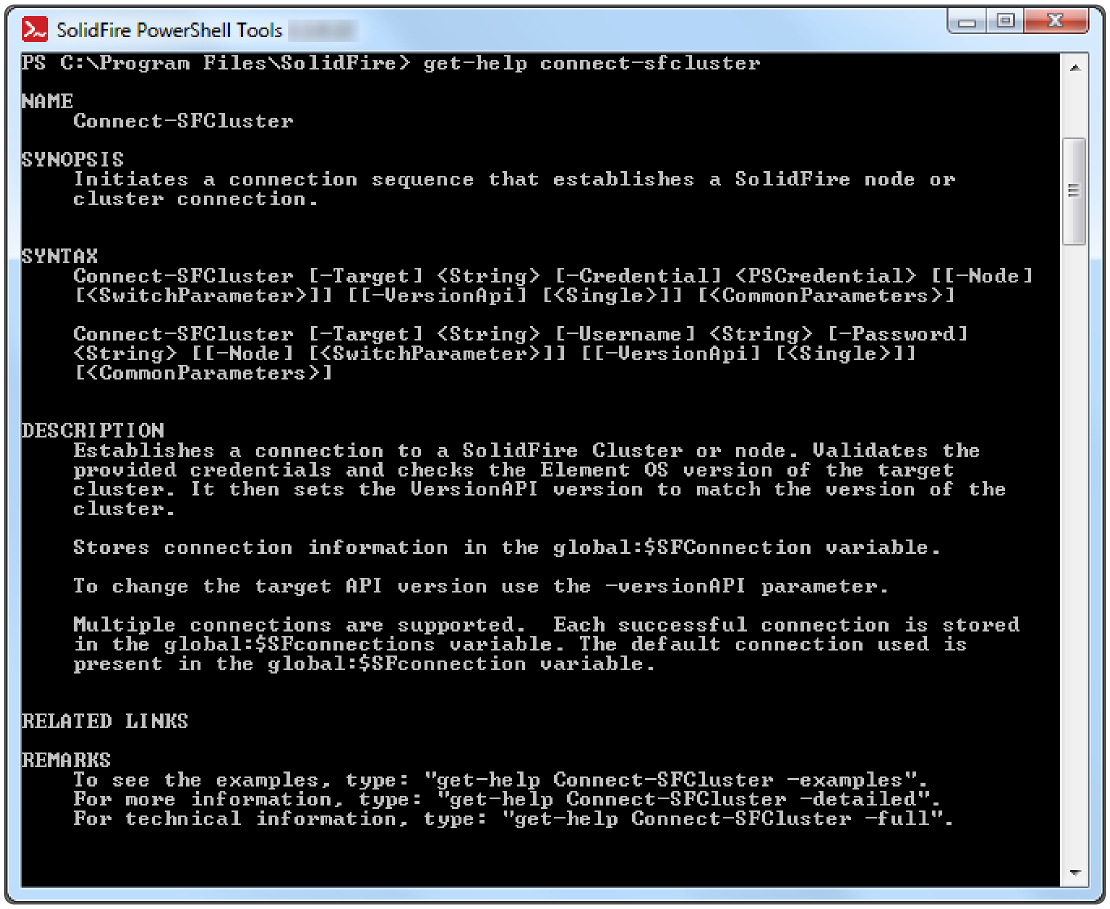
Accessing Embedded Help

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.



SolidFire PowerShell Tools

```
PS C:\Program Files\SolidFire> get-help connect-sfcluster

NAME
    Connect-SFCluster

SYNOPSIS
    Initiates a connection sequence that establishes a SolidFire node or
    cluster connection.


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
    cluster. It then sets the VersionAPI version to match the version of the
    cluster.

    Stores connection information 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.


RELATED LINKS

REMARKS
    To see the examples, type: "get-help Connect-SFCluster -examples".
    For more information, type: "get-help Connect-SFCluster -detailed".
    For technical information, type: "get-help Connect-SFCluster -full".
```

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

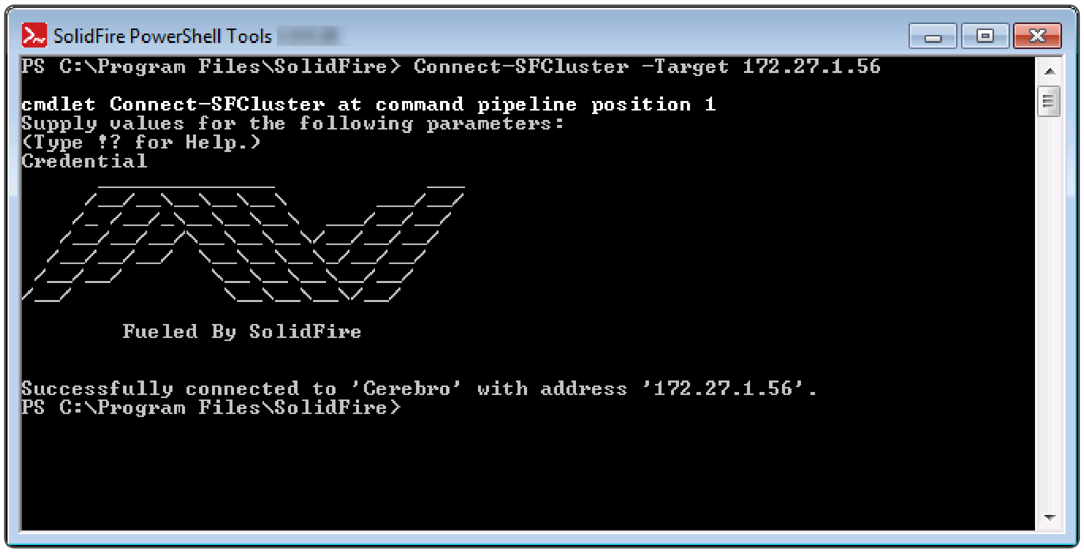
Use `Connect-SFCluster` to connect to a SolidFire cluster. The function collects SolidFire connection information, including target and authentication information from the user. `Connect-SFCluster` also supports connections to multiple SolidFire clusters.

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.



```
PS C:\Program Files\SolidFire> Connect-SFCluster -Target 172.27.1.56

cmdlet Connect-SFCluster at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
Credential
      ______________            ___
     /__/__\__\__\__\       ___/__/
    /_ /__/_\__\__\__\  ___/__/__/
   /__/__/__/\__\__\__\/__/__/__/
  /__/__/__/  \__\__\__\_/__/__/
 /__/__/       \__\__\__\__/__/
/__/            \__\__\__\/__/

        Fueled By SolidFire


Successfully connected to 'Cerebro' with address '172.27.1.56'.
PS C:\Program Files\SolidFire>
```

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:
   ```

Disconnecting from a SolidFire Cluster or Node

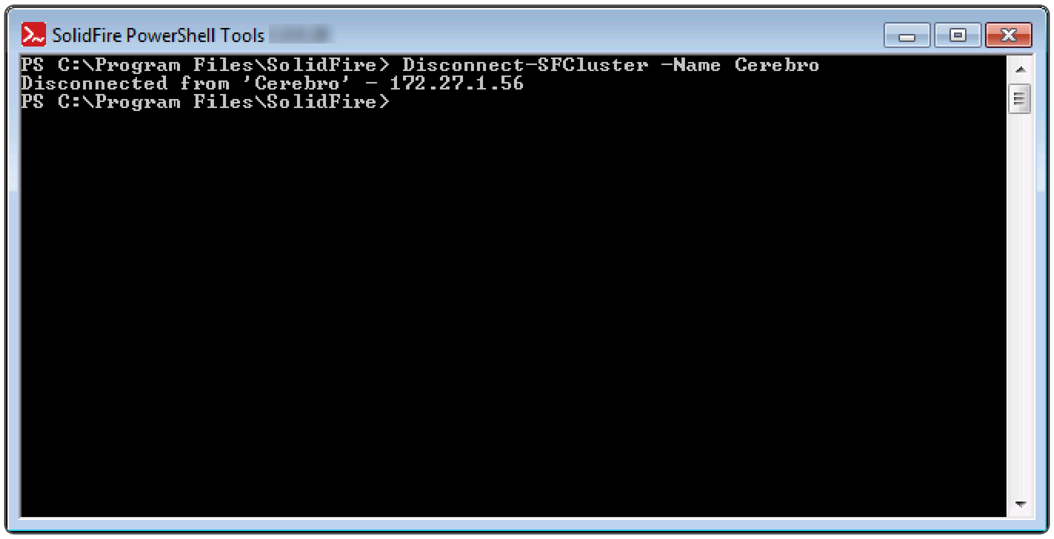
Use the `Disconnect-SFCluster` function to disconnect from a SolidFire cluster. The function also clears the `$SFConnection` and `$SFConnections` global variables from the session. This makes it easier to secure the shell if you wish to keep it active or work with a different SolidFire cluster.

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.



```
PS C:\Program Files\SolidFire> Disconnect-SFCluster -Name Cerebro
Disconnected from 'Cerebro' - 172.27.1.56
PS C:\Program Files\SolidFire>
```

Changing API Versions

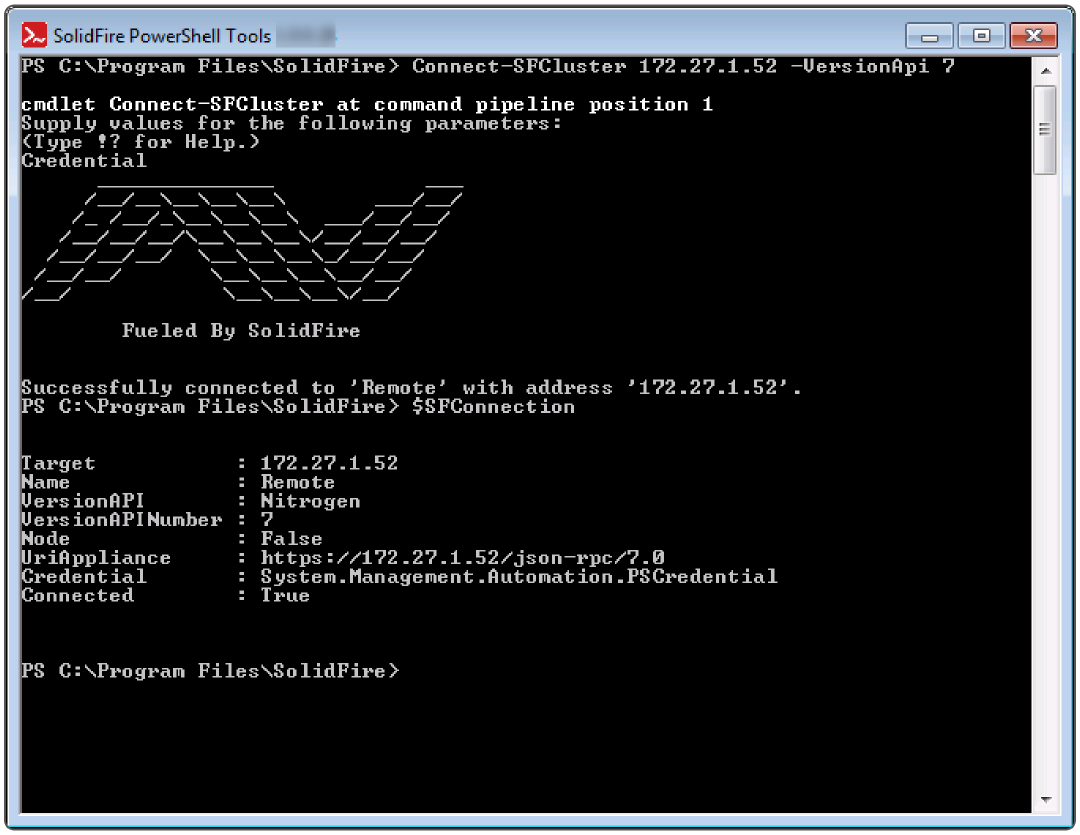
Use `-VersionApi` to specify a SolidFire API version. By default, the SolidFire PowerShell Tools module will use the most recent version of the SolidFire API available.

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>`.

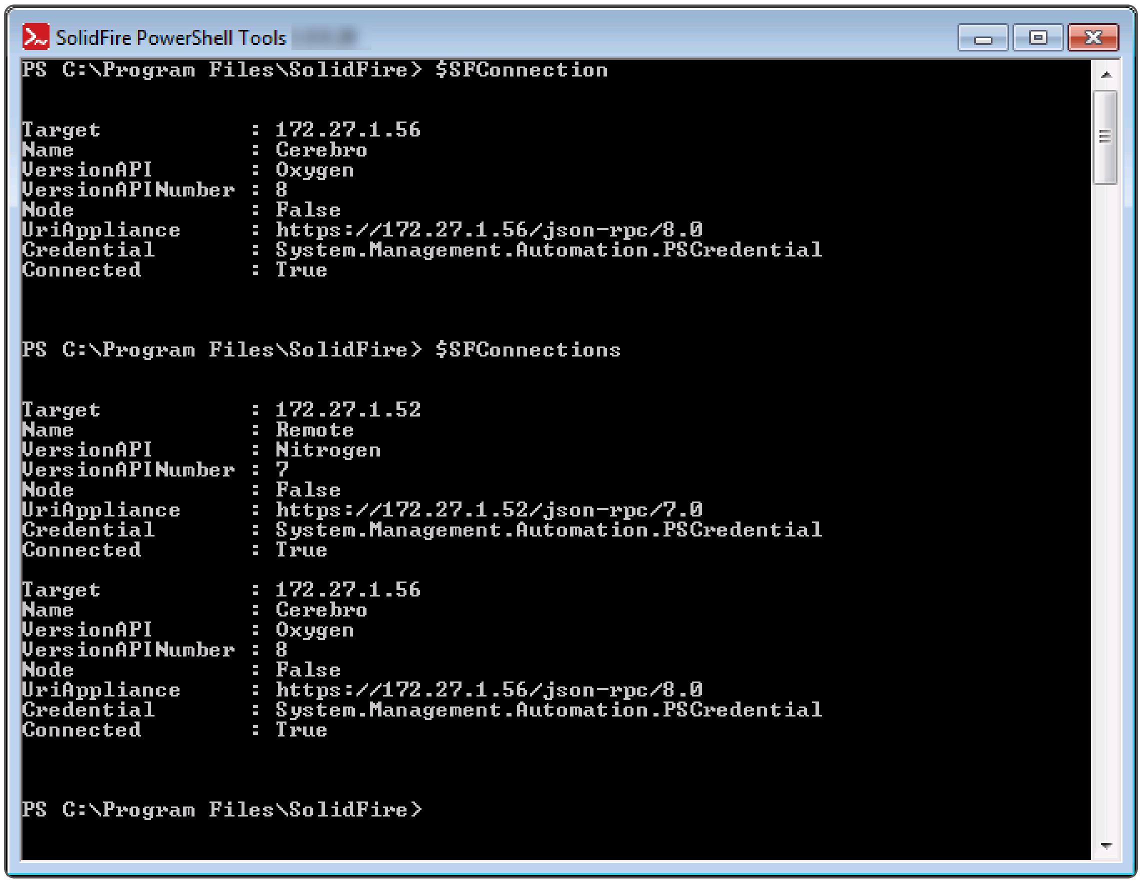
   The following example demonstrates connecting to a SolidFire cluster with API version 7.0 (Nitrogen).



```
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`.



```
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>
```

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 Get-Help 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.

Return Object Descriptions

Return values are fully documented as part of the C# SDK documentation that is available online. There are three methods for inspecting cmdlet return values:

- *Accessing Return Value Reference Documentation*
- *Accessing Return Values Using Get-Help*
- *Leveraging Get-Member to Inspect Return Objects*

Accessing Return Value Reference Documentation

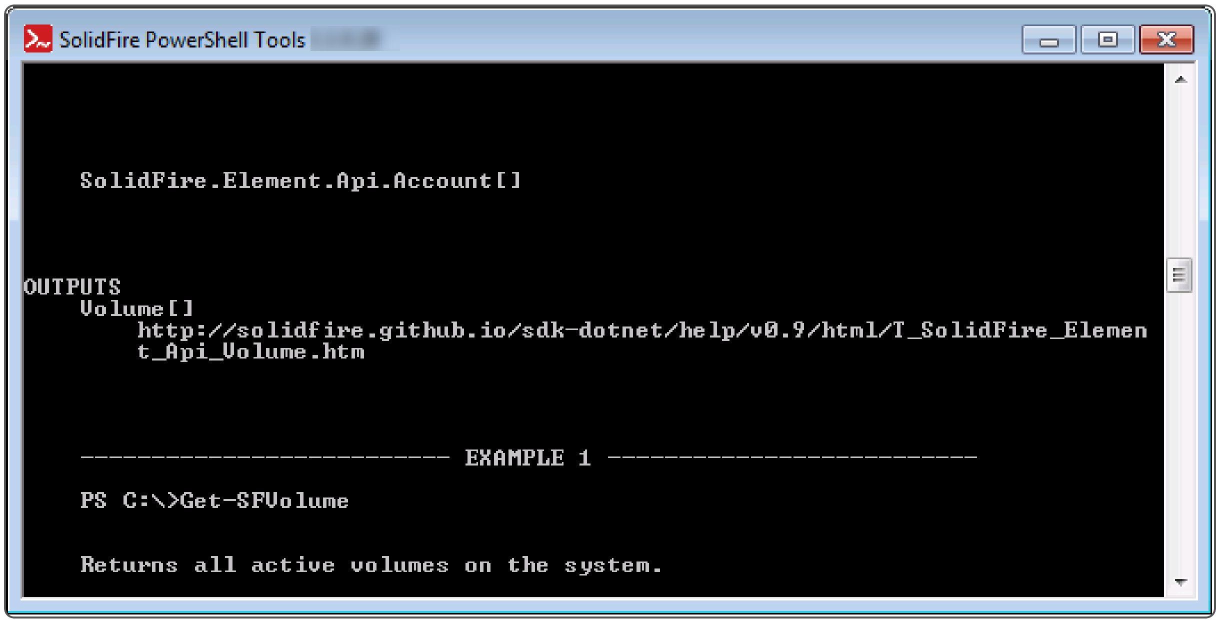
Each return value is documented as part of the online documentation for the SolidFire C# SDK. This documentation can be found on GitHub.

Accessing Return Values Using Get-Help

For any cmdlet included in the SolidFire PowerShell Tools, type `Get-Help <cmdlet name> -Full` to return the following:

- A specific return type for the cmdlet that is described in the Outputs section.
- A URL to the related SolidFire C# SDK reference page on GitHub.
- Examples of cmdlet use.

The following is an example of `Get-Help Get-SFVolume -Full`:



```
    SolidFire.Element.Api.Account[]



OUTPUTS
    Volume[]
        http://solidfire.github.io/sdk-dotnet/help/v0.9/html/T_SolidFire_Elemen
        t_Api_Volume.htm



    -------------------------- EXAMPLE 1 --------------------------

    PS C:\>Get-SFVolume

    Returns all active volumes on the system.
```

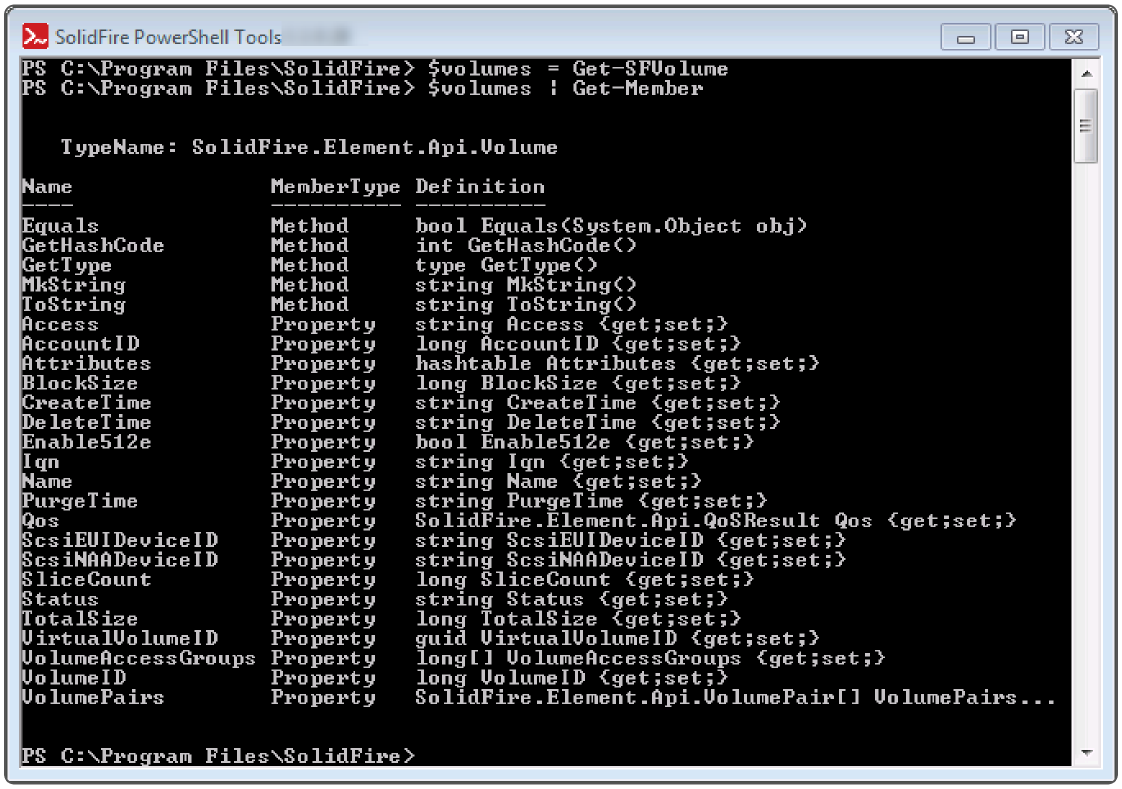
Leveraging Get-Member to Inspect Return Objects

Use the built-in SolidFire PowerShell Tools `Get-Member` cmdlet to inspect return values.

Procedure

1. In the command line interface, type `$<variable> = <Cmdlet with appropriate parameters>`
2. Type `$<variable> | Get-Member`.

   The following example shows the result for `Get-SFVolume` with each return property listed.



```
PS C:\Program Files\SolidFire> $volumes = Get-SFVolume
PS C:\Program Files\SolidFire> $volumes | Get-Member


   TypeName: SolidFire.Element.Api.Volume

Name                MemberType Definition
----                ---------- ----------
Equals              Method     bool Equals(System.Object obj)
GetHashCode         Method     int GetHashCode()
GetType             Method     type GetType()
MkString            Method     string MkString()
ToString            Method     string ToString()
Access              Property   string Access {get;set;}
AccountID           Property   long AccountID {get;set;}
Attributes          Property   hashtable Attributes {get;set;}
BlockSize           Property   long BlockSize {get;set;}
CreateTime          Property   string CreateTime {get;set;}
DeleteTime          Property   string DeleteTime {get;set;}
Enable512e          Property   bool Enable512e {get;set;}
Iqn                 Property   string Iqn {get;set;}
Name                Property   string Name {get;set;}
PurgeTime           Property   string PurgeTime {get;set;}
Qos                 Property   SolidFire.Element.Api.QoSResult Qos {get;set;}
ScsiEUIDeviceID     Property   string ScsiEUIDeviceID {get;set;}
ScsiNAADeviceID     Property   string ScsiNAADeviceID {get;set;}
SliceCount          Property   long SliceCount {get;set;}
Status              Property   string Status {get;set;}
TotalSize           Property   long TotalSize {get;set;}
VirtualVolumeID     Property   guid VirtualVolumeID {get;set;}
VolumeAccessGroups  Property   long[] VolumeAccessGroups {get;set;}
VolumeID            Property   long VolumeID {get;set;}
VolumePairs         Property   SolidFire.Element.Api.VolumePair[] VolumePairs...


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

3. If objects are more than one layer deep (see the QoS property in the example from the previous step), examine additional layers using the dot operator `$<variable.property> | Get-Member`.

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.

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.