



NetApp SolidFire PowerShell Tools User Guide

Version: 1.3

1/24/17

Copyright Information

Copyright © 1994-2017 NetApp, Inc. All Rights Reserved.

No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this document may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark Information

NetApp, the NetApp logo, AltaVault, ASUP, AutoSupport, Campaign Express, Cloud ONTAP, Clustered Data ONTAP, Customer Fitness, Data ONTAP, DataMotion, Fitness, Flash Accel, Flash Cache, Flash Pool, FlexArray, FlexCache, FlexClone, FlexPod, FlexScale, FlexShare, FlexVol, FPolicy, GetSuccessful, LockVault, Manage ONTAP, Mars, MetroCluster, MultiStore, NetApp Insight, OnCommand, ONTAP, ONTAPI, RAID DP, RAID-TEC, SANtricity, SecureShare, Simplicity, Simulate ONTAP, SnapCenter, Snap Creator, SnapCopy, SnapDrive, SnapIntegrator, SnapLock, SnapManager, SnapMirror, SnapMover, SnapProtect, SnapRestore, Snapshot, SnapValidator, SnapVault, StorageGRID, Tech OnTap, Unbound Cloud, WAFL, SolidFire, Element, Active IQ, SolidFire Helix and the helix design and other names are trademarks or registered trademarks of NetApp, Inc., in the United States, and/or other countries. All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such. A current list of NetApp trademarks is available on the web at <http://www.netapp.com/us/legal/netapptmlist.aspx>.

TABLE OF CONTENTS

Introduction	1
Software Prerequisites	1
Supported OS and OS-level Virtualization	1
Installing SolidFire PowerShell Tools on Windows	2
Installing SolidFire PowerShell Tools on MacOS	6
Installing SolidFire PowerShell Tools on Linux	7
Installing SolidFire PowerShell Tools as Docker Container	8
Successful Installation	9
Upgrading SolidFire PowerShell Tools on Windows	10
How to Use SolidFire PowerShell Tools	11
Listing Available Functions	11
Accessing Embedded Help	12
Parameter Sets	12
Managing Connections to a SolidFire Cluster	12
Connecting to a SolidFire Cluster	12
Connecting to a SolidFire Node	13
Disconnecting from a SolidFire Cluster or Node	14
Changing API Versions	14
Global Variables for All Functions	15
Common Parameters	16
Return Object Descriptions	16
Accessing Return Value Reference Documentation	17
Accessing Return Values Using Get-Help	17
Leveraging Get-Member to Inspect Return Objects	17
Using SolidFire PowerShell Tools with Other Modules and Snap-ins	18
Contacting SolidFire PowerShell Tools Support	18

Introduction

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. You can use 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 SolidFire PowerShell Tools. Before you use SolidFire PowerShell Tools, you should have an understanding of Windows PowerShell functions. The SolidFire PowerShell Tools module can be obtained through the SolidFire Support [BrickFTP](#) site or [GitHub](#).

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.
SolidFire Element OS		Element versions 7 through 9
.NET framework		4.5.1 or later

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

Supported OS and OS-level Virtualization

The following operating systems and container software are supported:

OS and Containers	Description
Microsoft® Windows® 8.1	Windows PowerShell is installed by default. Install the KB2883200 update.
Microsoft® Windows® 7 SP1	Windows PowerShell is supported but not installed.
Microsoft® Windows® 10	Windows PowerShell is installed by default.
Windows® Server 2012 R2 64-bit	Windows PowerShell is installed by default.
Windows® Server 2016	Windows PowerShell is installed by default.
Mac OS 10.11	Windows PowerShell for Mac is not installed by default.
Linux	Windows PowerShell for Linux is not installed by default.
Docker	Runs a container with Ubuntu and the SolidFire PowerShell tools pre-installed.

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

Installing SolidFire PowerShell Tools on Windows

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.

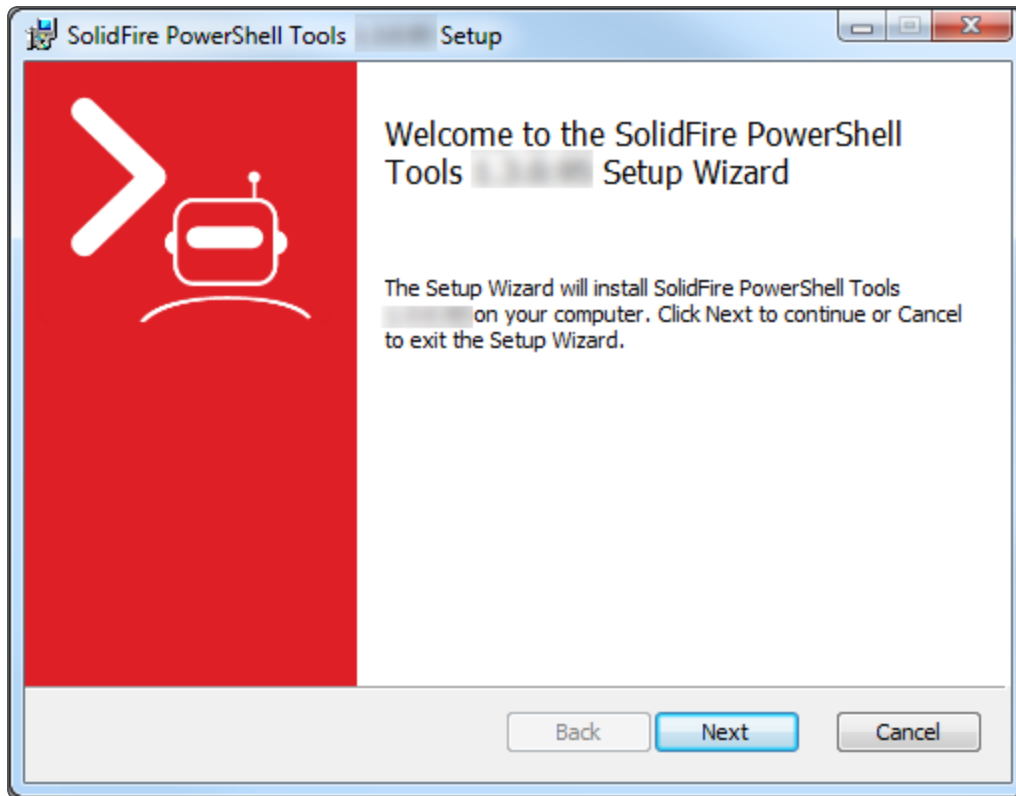
Prerequisites

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

Procedure

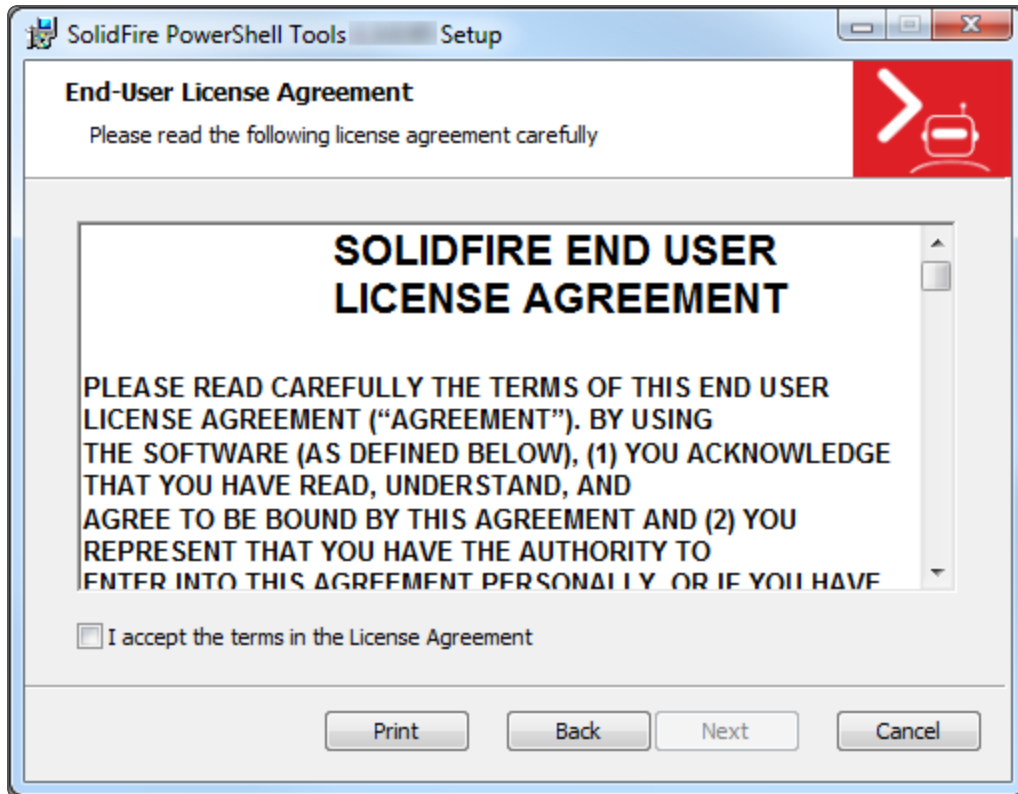
1. Download the installer from SolidFire [BrickFTP](#) or [GitHub](#) to a local or network directory that is accessible to the Windows system from which you will be running PowerShell commands.
2. Double-click the `SolidFire_PowerShell_<version number>-install.msi` installer.

The *Welcome* window appears.



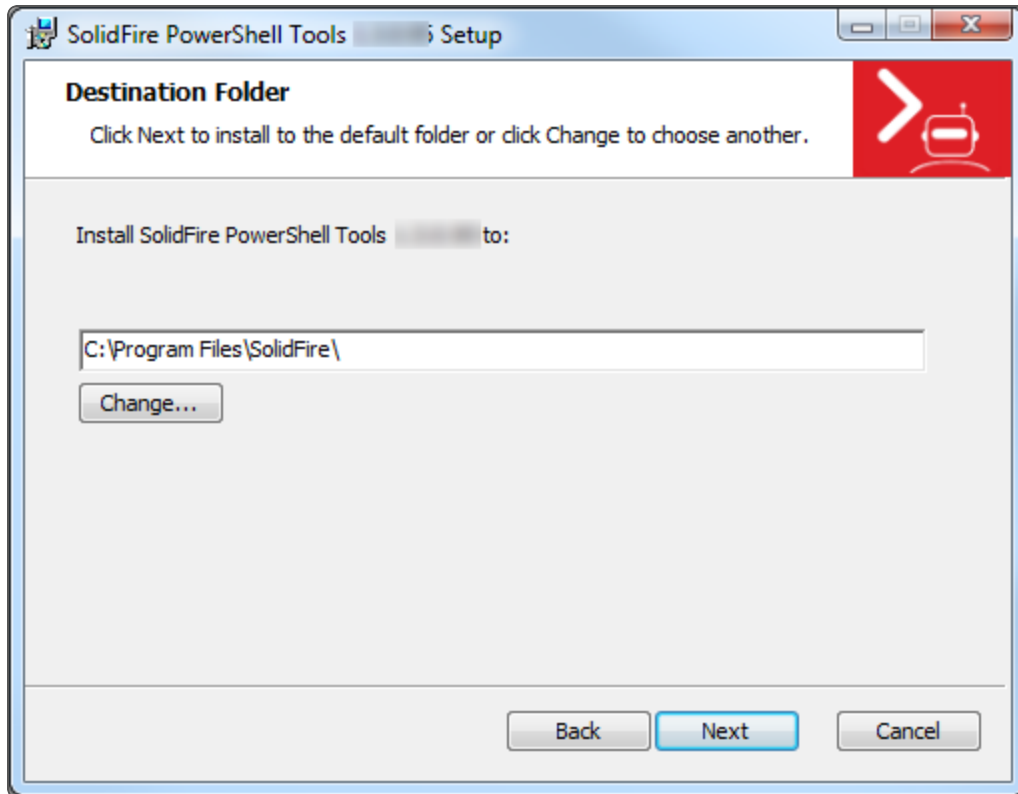
3. Click **Next**.

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**.

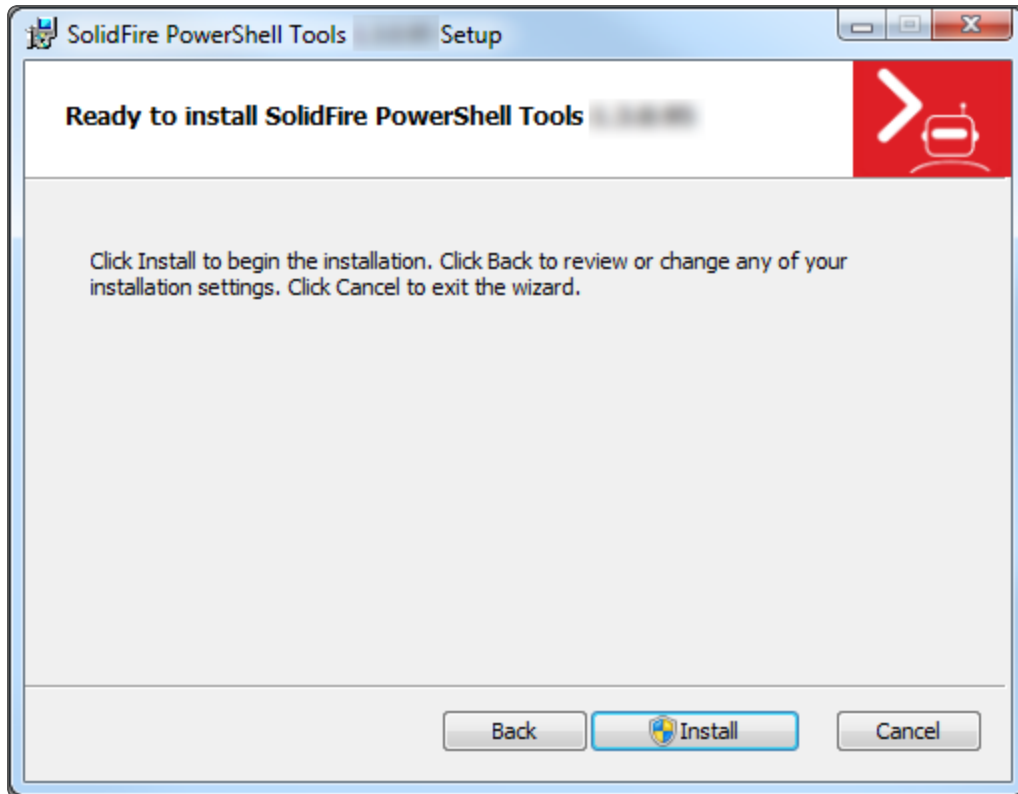
The *Destination Folder* window appears.



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**.

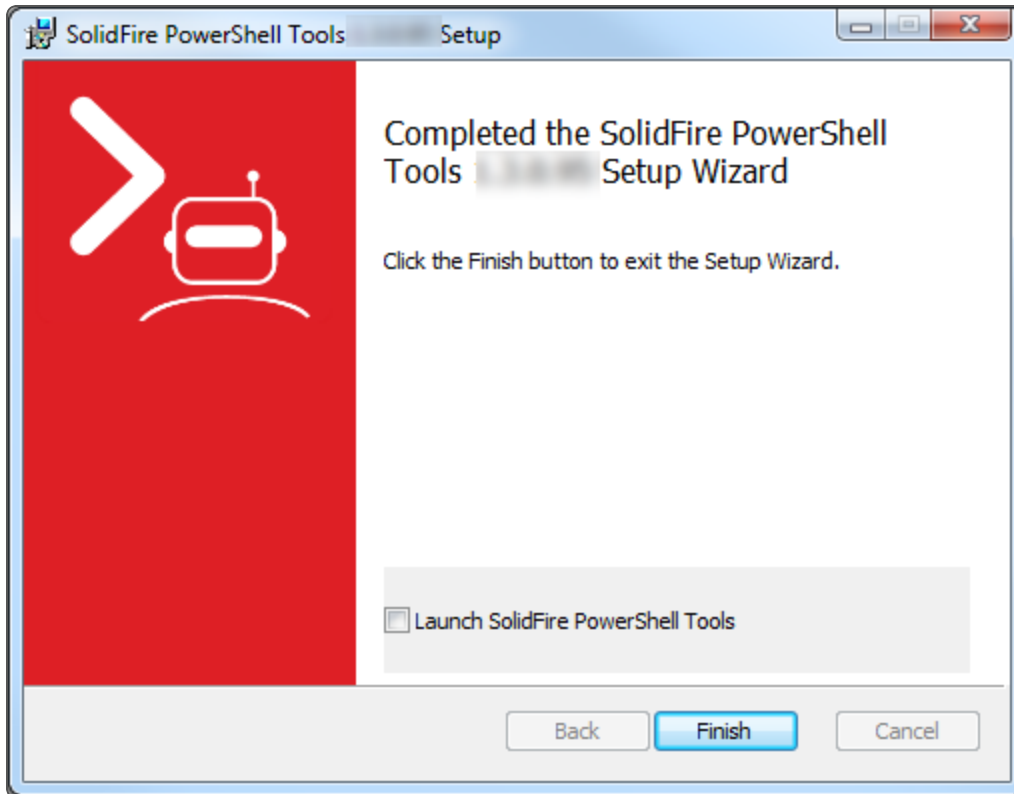
The *Ready to install* window appears.



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**.

Installing SolidFire PowerShell Tools on MacOS

You can install SolidFire PowerShell Tools as a module on MacOS once you have installed PowerShell for Mac. 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.

Prerequisites

- Run installation commands in a terminal session from the machine that will contain the PowerShell Tools installation. Commands can be run from any directory.
- Sudo access is needed to complete the installation.
- 64-bit operating system.
- Review [Software Prerequisites](#).

Procedure

1. Install PowerShell for Mac:

NOTE: For additional instructions, see the PowerShell for Mac [installation page](#).

- a. Install the dependencies with [Homebrew](#).

```
brew install openssl  
brew install curl --with-openssl
```

- b. Download the PowerShell for Mac package.

```
wget https://github.com/PowerShell/PowerShell/releases/download/v6.0.0-alpha.13/powershell-6.0.0-alpha.13.pkg
```

- c. Install the PowerShell package on your machine.

```
sudo installer -pkg powershell-6.0.0-alpha.13.pkg -target /
```

2. Install SolidFire PowerShell Tools.

- a. Create a directory for the SolidFire PowerShell Tools module.

```
mkdir -p /usr/local/share/powershell/Modules/SolidFire
```

- b. Download the SolidFire PowerShell Tools Module DLLs from GitHub.

```
cd /usr/local/share/powershell/Modules/SolidFire && curl --remote-name-all
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/DotNetSdk.dll
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/Newtonsoft.Json.dll
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/SolidFire.dll
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/SolidFire.psd1
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/Initialize-
SFEnvironment.ps1
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/SolidFire.dll-
help.xml >/dev/null && cd -
```

- c. [OPTIONAL] Set up the SolidFire PowerShell Tools initializer script:

```
echo ". /usr/local/share/powershell/Modules/SolidFire/Initialize-SFEnvironment.ps1" >
/usr/local/microsoft/powershell/6.0.0-alpha.13/profile.ps1
```

- d. Launch PowerShell session:

```
powershell
```

Installing SolidFire PowerShell Tools on Linux

You can install SolidFire PowerShell Tools as a module on Linux once you have installed PowerShell. 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.

Prerequisites

- Run installation commands in a terminal session from the machine that will contain the PowerShell Tools installation. Commands can be run from any directory.
- Sudo access is needed to complete the installation.
- 64-bit operating system.
- Review [Software Prerequisites](#).

Procedure

1. Install PowerShell for Linux:

- [Ubuntu 14.04](#)

- a. Download the Debian package **powershell_6.0.0-alpha.13-1ubuntu1.14.04.1_amd64.deb** from the PowerShell [releases page](#) onto your Ubuntu machine.
- b. Type the following commands in the terminal:

```
sudo dpkg -i powershell_6.0.0-alpha.13-1ubuntu1.14.04.1_amd64.deb
sudo apt-get install -f
```

NOTE: The Debian install command `dpkg -i` fails if there are unmet dependencies. The command `apt-get install -f` resolves these dependencies and completes the configuration of the PowerShell package.

- **Ubuntu 16.04**

- a. Download the Debian package **powershell_6.0.0-alpha.13-1ubuntu1.16.04.1_amd64.deb** from the PowerShell [releases page](#) onto your Ubuntu machine.
- b. Type the following commands in the terminal:

```
sudo dpkg -i powershell_6.0.0-alpha.13-1ubuntu1.16.04.1_amd64.deb
sudo apt-get install -f
```

NOTE: The Debian install command `dpkg -i` fails if there are unmet dependencies. The command `apt-get install -f` resolves these dependencies and completes the configuration of the PowerShell package.

- **CentOS 7**

- a. Download the RPM package **powershell-6.0.0_alpha.13-1.el7.centos.x86_64.rpm** from the PowerShell [releases page](#) onto your CentOS machine.

NOTE: This package works on Oracle Linux 7 and Red Hat Enterprise Linux 7.

- b. Type the following commands in the terminal:

```
sudo yum install ./powershell-6.0.0_alpha.13-1.el7.centos.x86_64.rpm
```

2. Install SolidFire PowerShell Tools.

- a. Create a directory for the SolidFire PowerShell Tools module.

```
mkdir -p /usr/local/share/powershell/Modules/SolidFire
```

- b. Download the SolidFire PowerShell Tools Module DLLs from GitHub.

```
cd /usr/local/share/powershell/Modules/SolidFire && curl --remote-name-all
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/DotNetSdk.dll
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/Newtonsoft.Json.dll
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/SolidFire.dll
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/SolidFire.psd1
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/Initialize-
SFEnvironment.ps1
https://raw.githubusercontent.com/solidfire/PowerShell/release1.3/packages/SolidFire.dll-
help.xml >/dev/null && cd -
```

- c. [OPTIONAL] Set up the SolidFire PowerShell Tools initializer script:

```
echo ". /usr/local/share/powershell/Modules/SolidFire/Initialize-SFEnvironment.ps1" >
/usr/local/microsoft/powershell/6.0.0-alpha.13/profile.ps1
```

- d. Launch PowerShell session:

```
powershell
```

Installing SolidFire PowerShell Tools as Docker Container

You can install SolidFire PowerShell Tools as a Docker container. 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.

Prerequisites

- Run installation commands in a terminal session from the machine that will contain the PowerShell Tools installation. Commands can be run from any directory.
- A [Docker engine](#) is installed and running on your host machine.

- 64-bit operating system.
- Review [Software Prerequisites](#).

Procedure

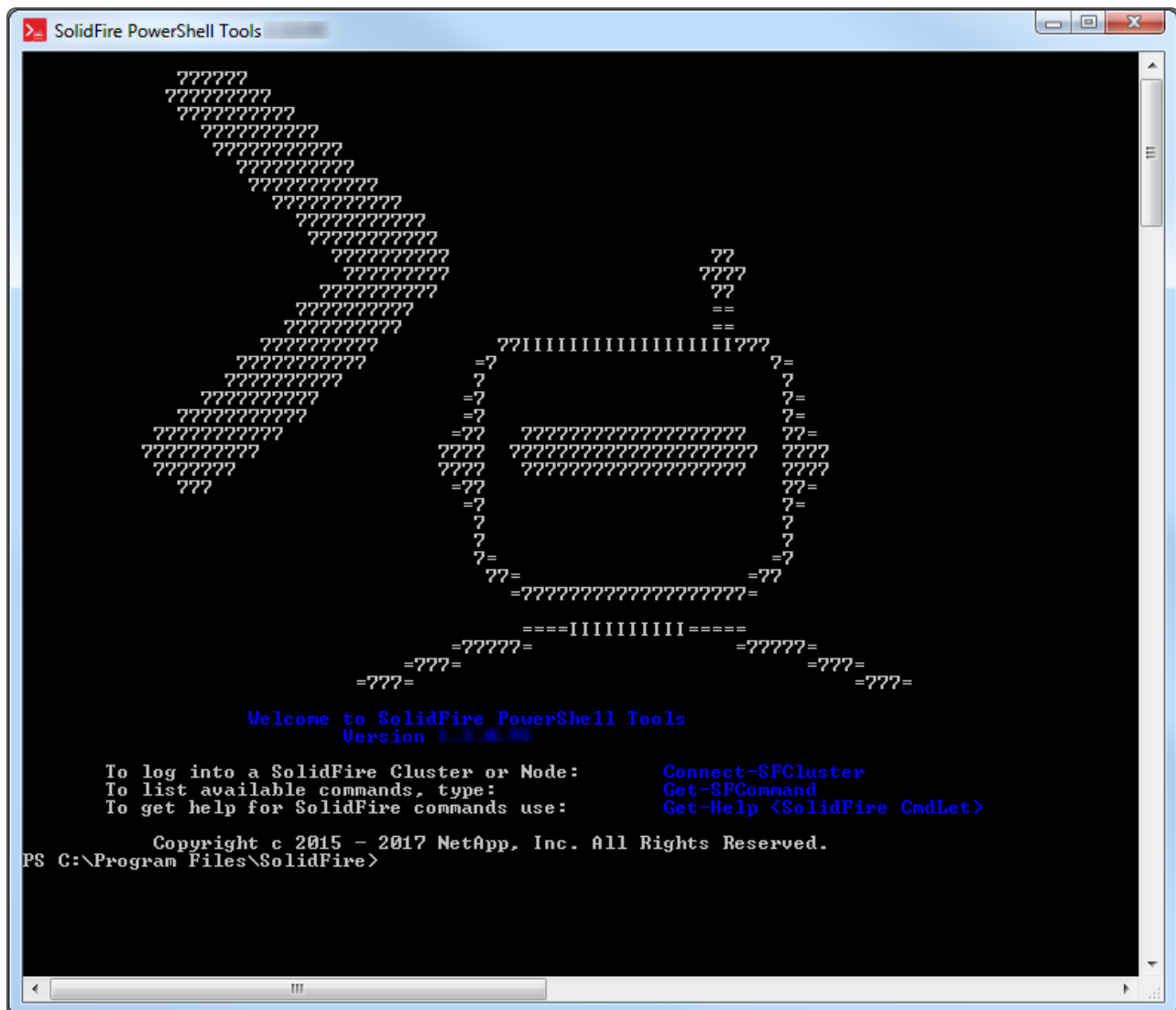
1. From a terminal, type this command to pull and run the latest SolidFire PowerShell Tools image from the NetApp SolidFire public Docker repo:

```
docker run -it -v $(pwd):/scripts solidfire/powershell
```

NOTE: This command downloads the SolidFire PowerShell Tools Docker image, starts a new container, and opens a shell from your terminal with the SolidFire module loaded and ready to use. Any scripts in your host's present working directory (PWD) are available at **/scripts** once you are in the shell.

Successful Installation

After successful installation, SolidFire PowerShell Tools opens in a customized PowerShell window.



Upgrading SolidFire PowerShell Tools on Windows

To upgrade SolidFire PowerShell Tools on Windows, 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 SolidFire PowerShell Tools on Windows*](#).

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-SFSnmpUser	SolidFire
Cmdlet	Add-SFVolumeToVolumeAccessGroup	SolidFire
Cmdlet	Complete-SFClusterPairing	SolidFire
Cmdlet	Complete-SFVolumePairing	SolidFire
Cmdlet	Connect-SFCluster	SolidFire
Cmdlet	Copy-SFVolume	SolidFire
Cmdlet	Disable-SFEncryptionAtRest	SolidFire
Cmdlet	Disable-SFSnmp	SolidFire
Cmdlet	Disconnect-SFCluster	SolidFire
Cmdlet	Enable-SFEncryptionAtRest	SolidFire
Cmdlet	Enable-SFFeature	SolidFire
Cmdlet	Enable-SFSnmp	SolidFire
Cmdlet	Get-SFAccount	SolidFire
Cmdlet	Get-SFAccountEfficiency	SolidFire
Cmdlet	Get-SFActiveNode	SolidFire
Cmdlet	Get-SFASyncResult	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	Copy-SFVolume	SolidFire
Cmdlet	Get-SFDeletedVolume	SolidFire
Cmdlet	Get-SFVirtualVolume	SolidFire
Cmdlet	Get-SFVolume	SolidFire
Cmdlet	Get-SFVolumeStatsByVirtualVolume	SolidFire
Cmdlet	Invoke-SFRestoreDeletedVolume	SolidFire
Cmdlet	New-SFVolume	SolidFire
Cmdlet	Remove-SFDeletedVolume	SolidFire
Cmdlet	Remove-SFVolume	SolidFire
Cmdlet	Set-SFVolume	SolidFire

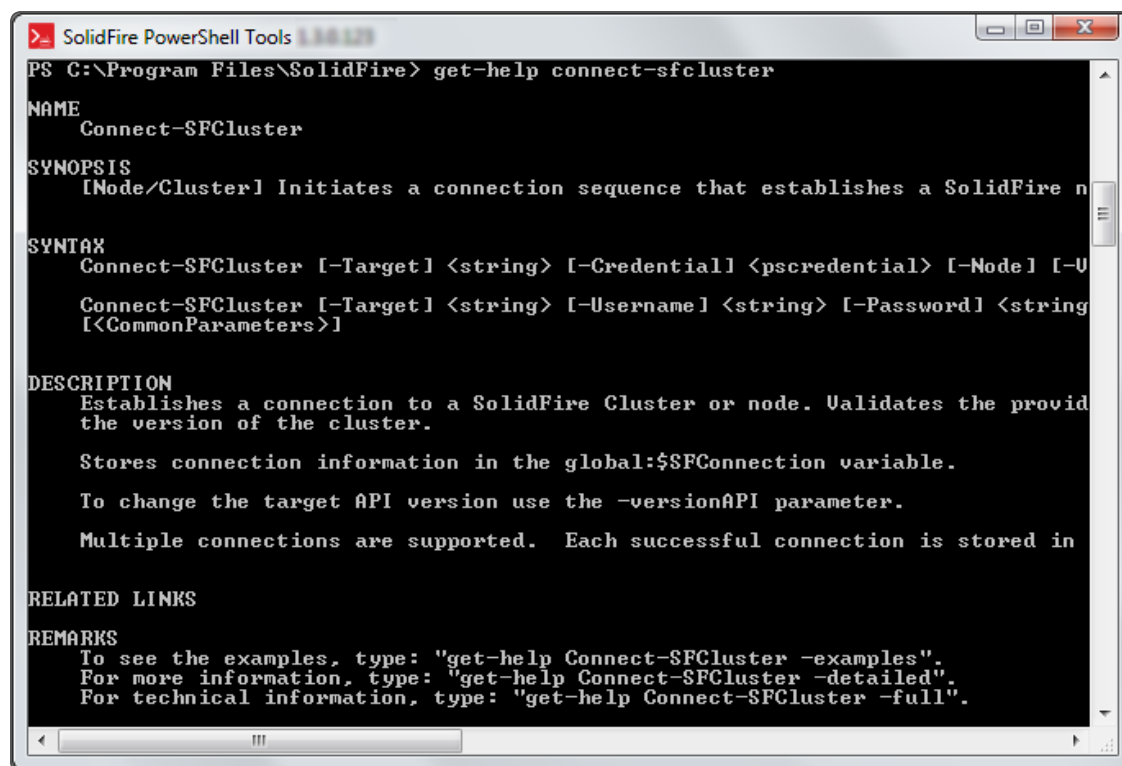
Accessing Embedded Help

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
    [Node/Cluster] Initiates a connection sequence that establishes a SolidFire n

SYNTAX
    Connect-SFCluster [-Target] <string> [-Credential] <pscredential> [-Node] [-U
    Connect-SFCluster [-Target] <string> [-Username] <string> [-Password] <string
    [<CommonParameters>]

DESCRIPTION
    Establishes a connection to a SolidFire Cluster or node. Validates the provid
    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

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".
```

NOTE: To view full cmdlet help, see [Accessing Return Values Using Get-Help](#).

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 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 "10.117.60.15"

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

Target           : 10.117.60.15
Name             : HI-FC
Port             :
VersionApiNumber : 9
VersionApiName   : Fluorine
Node             : False
Uri              : https://10.117.60.15/json-rpc/9.0
RequestCount     : 3
Credential       : System.Management.Automation.PSCredential
Limits           : {"AccountCountMax" = 5000, "AccountNameLengthMax" = 64, "Acco
                  "ClusterPairsCountMax" = 4, "InitiatorNameLengthMax" = 224, "
                  4096, "SecretLengthMax" = 16, "SecretLengthMin" = 12, "Snapsh
                  16383, "VolumeAccessGroupNameLengthMax" = 64, "VolumeAccessGr
                  "InitiatorAliasLengthMax" = 224, "VolumeBurstIOPSMax" = 20000
                  "VolumeMinIOPSMax" = 15000, "VolumeMinIOPSMin" = 50, "VolumeN
                  "VolumesPerAccountCountMax" = 2000, "VolumesPerGroupSnapshotM

Timeout         :
Element         : SolidFire.Element.Api.SolidFireElement

              ??
            ???
            ??
            ==
          ??IIIIIIIIIIIIIIII??
        =?              ?=
        ?              ?
        =?              ?=
        =?              ?=
        =??  ??????????????????  ??=
        ?????  ??????????????????  ?????
        ?????  ??????????????????  ?????
        =??              ??=
        =?              ?=
        ?              ?
        ?              =?
        ??=            =??
        =????????????????????=
        =?????=?      =?????=
        =???=        =???=
        =???=        =???=

              Fueled By NetApp SolidFire

Successfully connected to 'HI-FC' with address '10.117.60.15'.

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

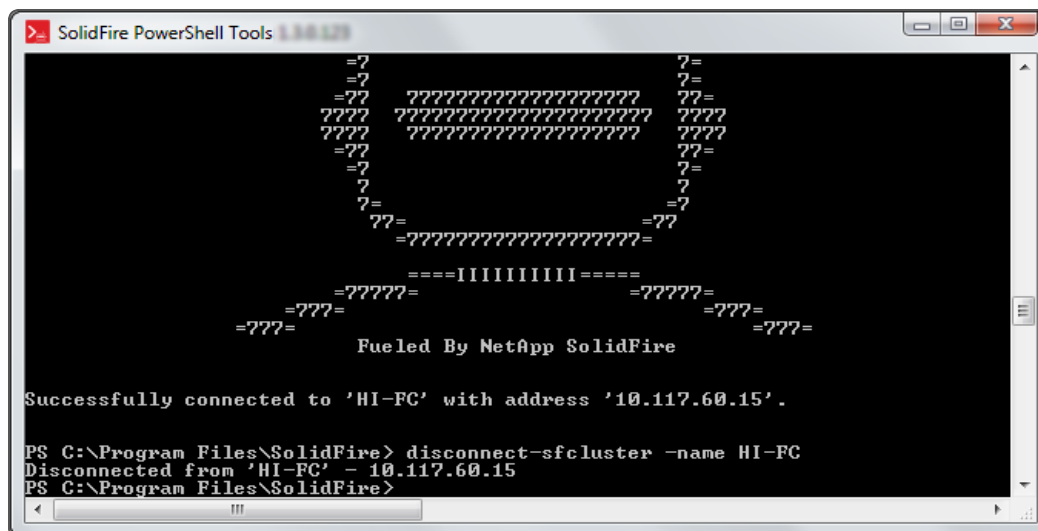
Use the `Disconnect-SFCluster` function to disconnect from a SolidFire cluster. The function also clears the `$SFConnection` and `$SFConnections` global variables from the session. This makes it easier to secure the shell if you wish to keep it active or work with a different SolidFire cluster.

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.

**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 10.117.60.15 -versionAPI 7

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

Target          : 10.117.60.15
Name            : HI-FC
Port           : 
VersionApiNumber : 7
VersionApiName  : Nitrogen
Node           : False
Uri            : https://10.117.60.15/json-rpc/7.0
RequestCount    : 5
Credential      : System.Management.Automation.PSCredential
Limits         : {"AccountCountMax" = 5000, "AccountNameLengthMax" = 64, "Acco
                  "ClusterPairsCountMax" = 4, "InitiatorNameLengthMax" = 224, "
                  4096, "SecretLengthMax" = 16, "SecretLengthMin" = 12, "Snapsh
                  16383, "VolumeAccessGroupNameLengthMax" = 64, "VolumeAccessGr
                  "InitiatorAliasLengthMax" = 224, "VolumeBurstIOPSMax" = 20000
                  "VolumeMinIOPSMax" = 15000, "VolumeMinIOPSMin" = 50, "VolumeN
                  "VolumesPerAccountCountMax" = 2000, "VolumesPerGroupSnapshotM

Timeout        : 
Element        : SolidFire.Element.Api.SolidFireElement

              ??
            ???
            ??
              ==
      77IIIIIIIIIIIIII777
    =?              ?=
    ?              ?
    =?              ?=
    ?              ?=
    =??  ??????????????????  ??=
    ???? ??????????????????  ???
    ???? ??????????????????  ???
    =??              ??=
    ?              ?
    ?              ?
    ??=             =??
    =????????????????=
      ==IIIIIIII==
    =?????=?       =?????=?
    =???=?         =???=?
    =???=?         =???=?

          Fueled By NetApp SolidFire

Successfully connected to 'HI-FC' with address '10.117.60.15'.

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` and `SFConnection`. These common parameters are 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` 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 without indicating the target against which the cmdlet was run.

If the `-SFConnection` parameter is included in the cmdlet, the cmdlet will run against the specific `SFConnection` that was handed in through that parameter. You can inspect the `$SFConnections` session variable to find a specific `SFConnection` or you can pass the result of `Connect-SFCluster` into it.

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 .NET 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

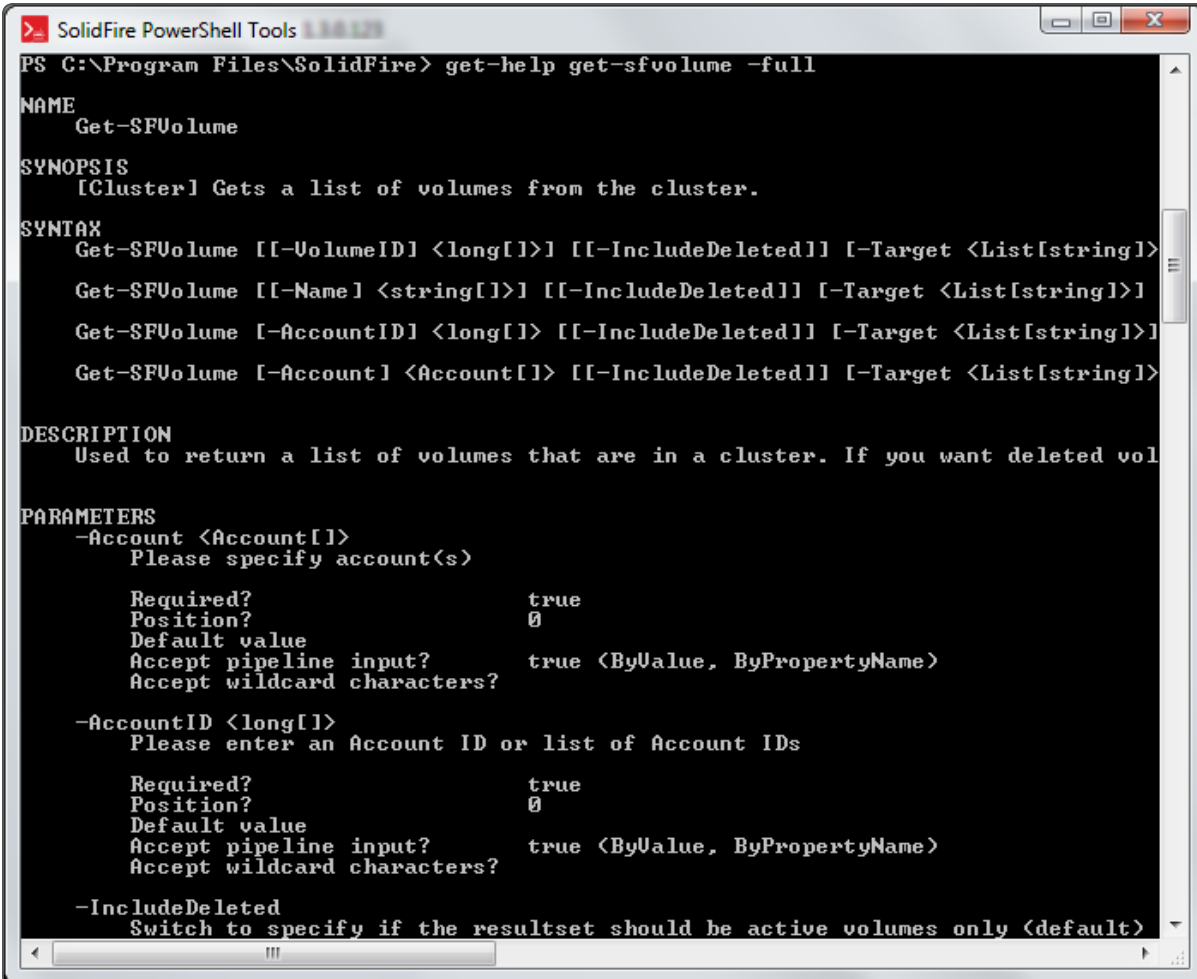
Each return value is documented as part of the online documentation for the SolidFire .NET SDK. This documentation can be found on [GitHub](#).

Accessing Return Values Using Get-Help

For any cmdlet included in 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 .NET SDK reference page on GitHub.
- Examples of cmdlet use.

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



```
PS C:\Program Files\SolidFire> get-help get-sfvolume -full

NAME
    Get-SFVolume

SYNOPSIS
    [Cluster] Gets a list of volumes from the cluster.

SYNTAX
    Get-SFVolume [[-VolumeID] <long[]>] [[-IncludeDeleted]] [-Target <List[string]>]
    Get-SFVolume [[-Name] <string[]>] [[-IncludeDeleted]] [-Target <List[string]>]
    Get-SFVolume [-AccountID] <long[]> [[-IncludeDeleted]] [-Target <List[string]>]
    Get-SFVolume [-Account] <Account[]> [[-IncludeDeleted]] [-Target <List[string]>]

DESCRIPTION
    Used to return a list of volumes that are in a cluster. If you want deleted vol

PARAMETERS
    -Account <Account[]>
        Please specify account(s)

        Required?                true
        Position?                 0
        Default value
        Accept pipeline input?    true (ByValue, ByPropertyName)
        Accept wildcard characters?

    -AccountID <long[]>
        Please enter an Account ID or list of Account IDs

        Required?                true
        Position?                 0
        Default value
        Accept pipeline input?    true (ByValue, ByPropertyName)
        Accept wildcard characters?

    -IncludeDeleted
        Switch to specify if the resultset should be active volumes only (default)
```

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;}
Ign             Property    string Ign {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 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, reach out to the development community at [ThePub](#). We also monitor the [GitHub PowerShell](#) repository for open issues or pull requests. Your feedback helps us focus our efforts on new features and capabilities.



1048 Pearl Street, Suite 250
Boulder, Colorado 80302

Phone: 720.523.3278
Email: info@solidfire.com

Web: solidfire.com
Support: www.solidfire.com/support/

1/24/17