NetApp

NetApp SolidFire PowerShell Tools User Guide

Version: 1.4

6/12/17

Copyright Information

Trademark Information

Table of Contents

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 9.1
.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 Non-Windows OS

You can install SolidFirePowerShell Tools as a Docker container or as a module on MacOS or Linux.

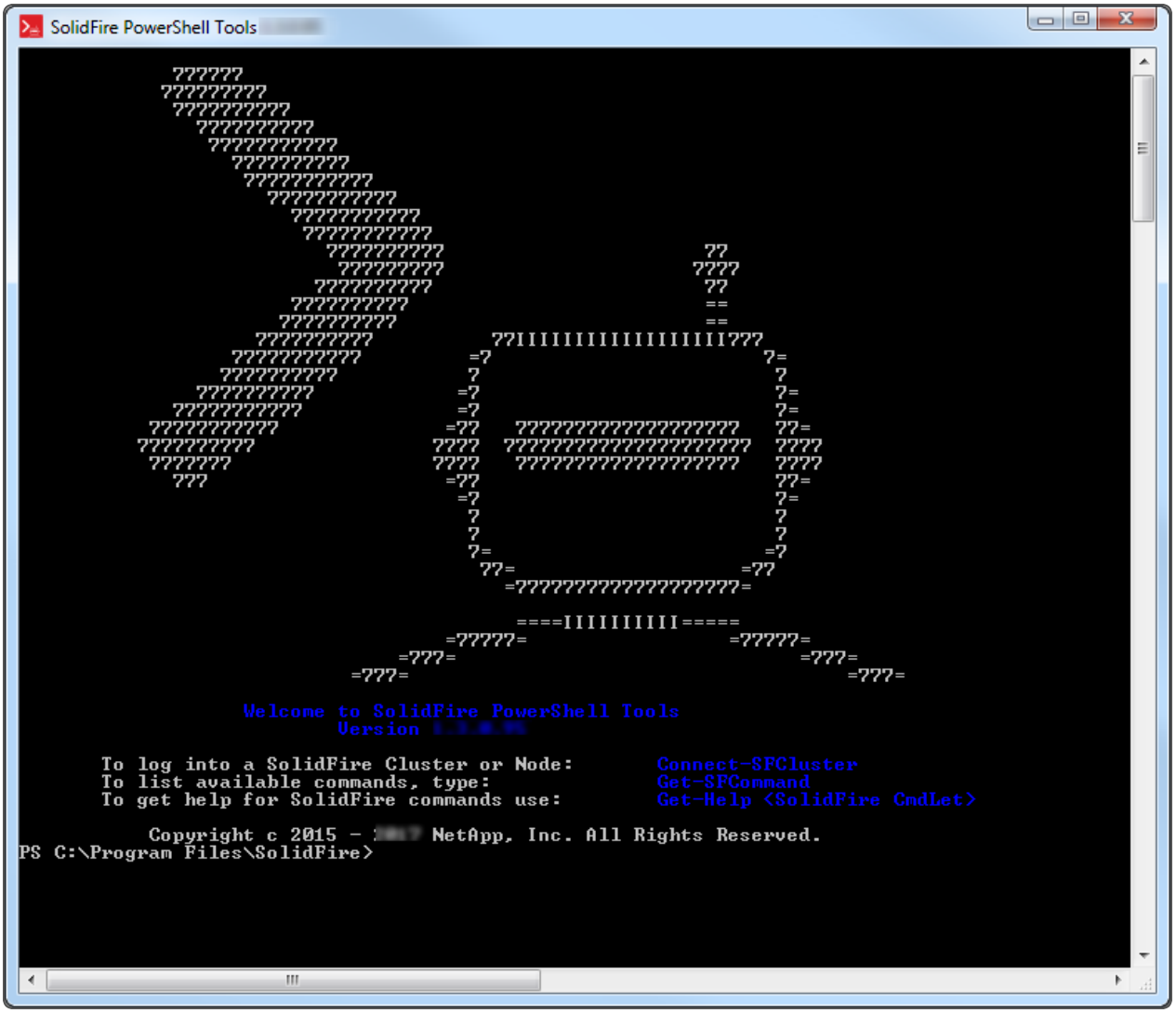
Procedure

1. See the README.md instructions on GitHub for your installation:
   - MacOS
   - Linux

- Docker

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.

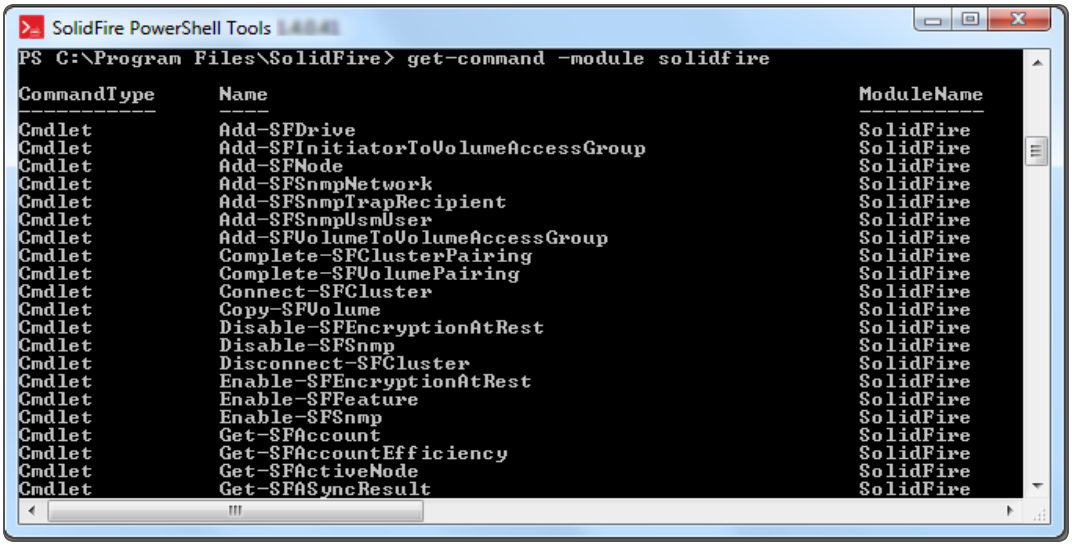
Listing Available Functions

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

Procedure

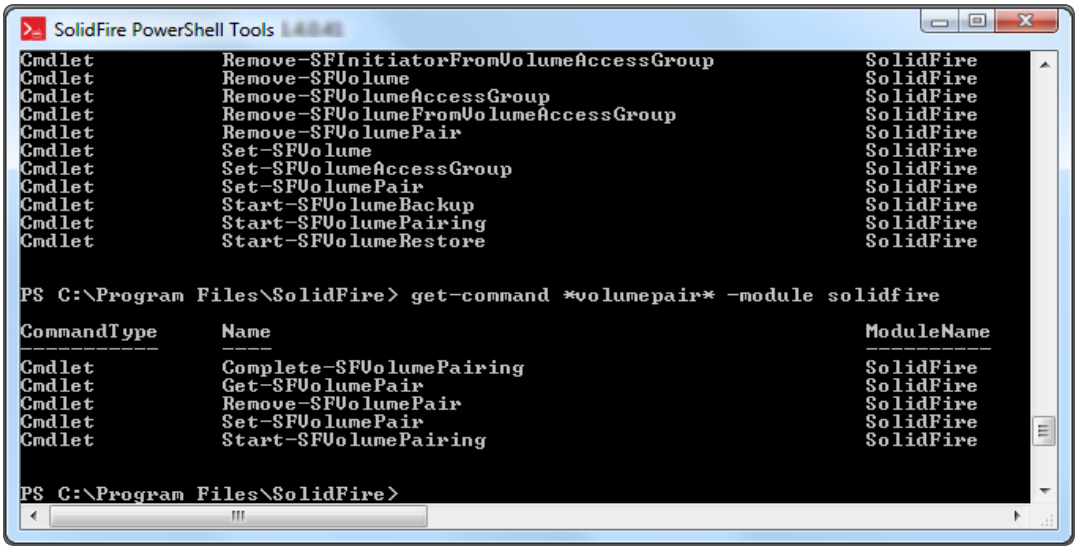
1. In the command line interface, type `Get-Command -Module SolidFire`.

   The list of available commands appears.



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

   The filtered list of available commands appears.



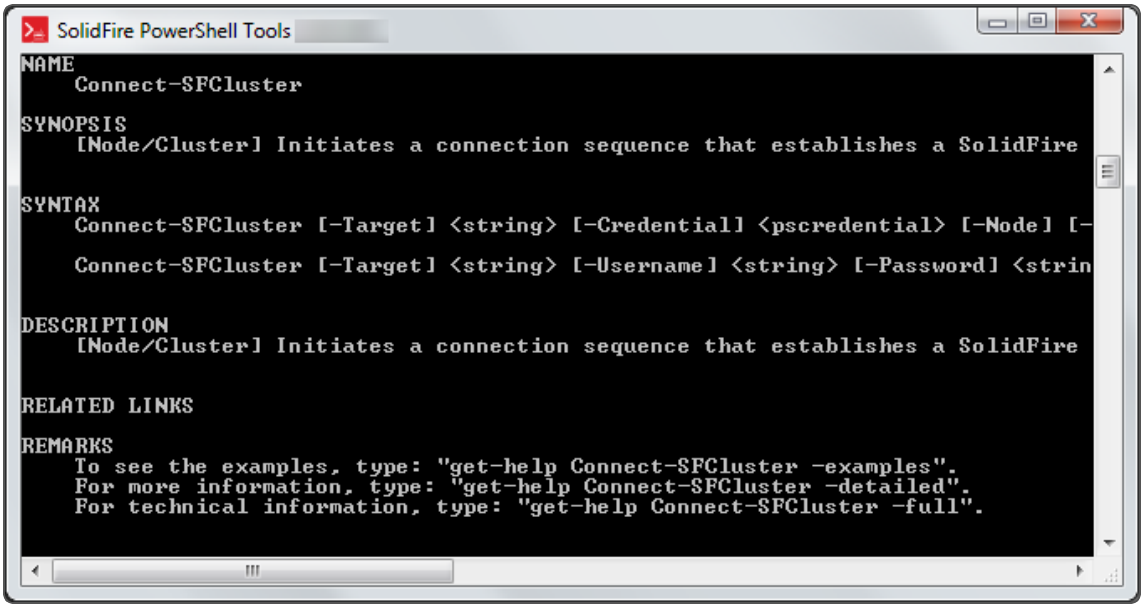
Accessing Embedded Help

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.



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

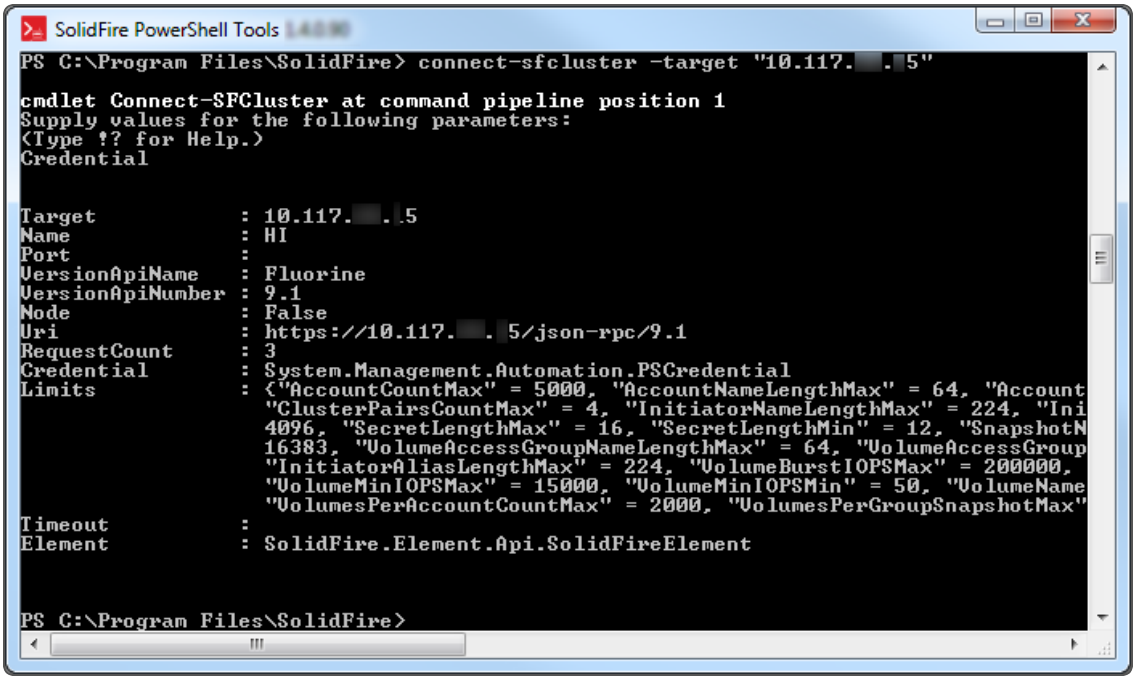
Use `Connect-SFCluster` to connect to a SolidFire cluster. The function collects SolidFire connection information, including target and authentication information from the user. `Connect-SFCluster` also supports connections to multiple SolidFire clusters.

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.



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.

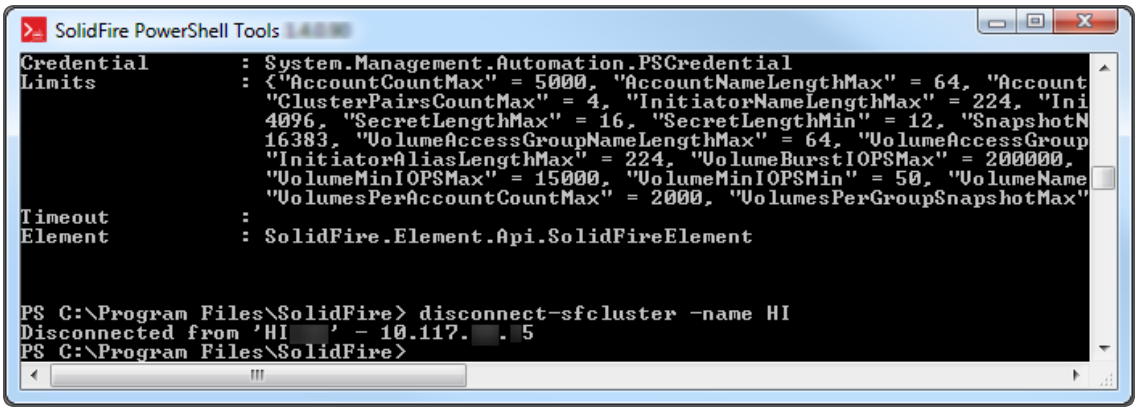
You can disconnect a specific connection using the name of the connection from the `$SFConnection` or `$SFConnections` global variables. This name is either the cluster or the node name. See *Global Variables for All Functions* for an example.

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

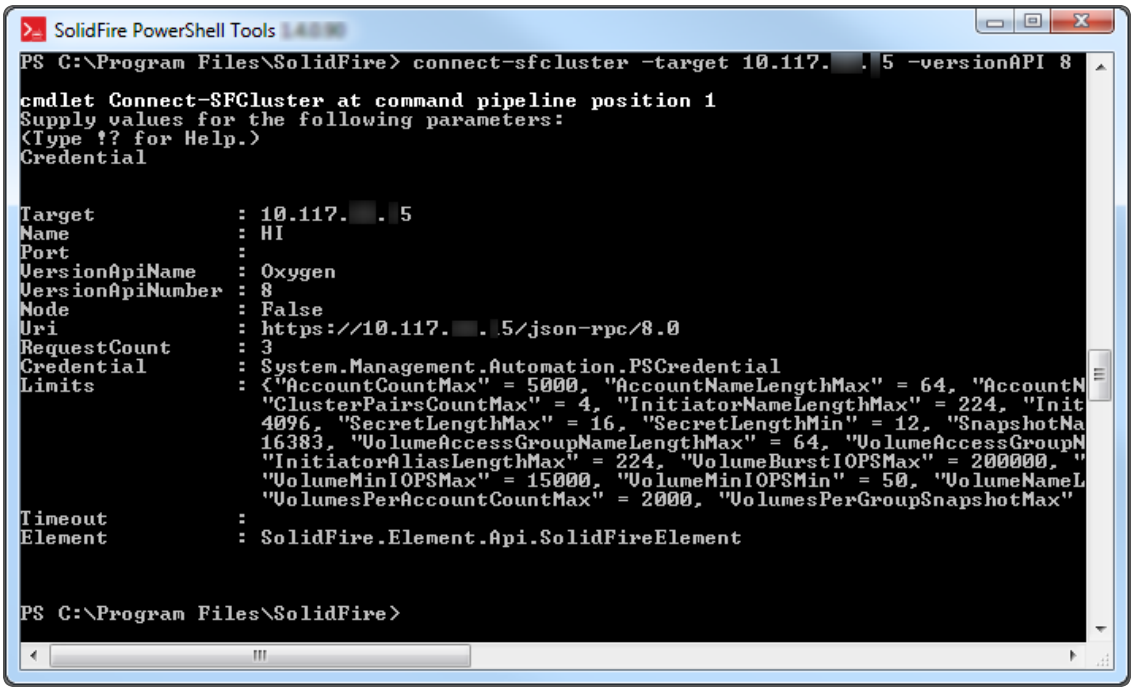
Use `-VersionApi` to specify a SolidFire API version. By default, the SolidFire PowerShell Tools module will use the most recent version of the SolidFire API available.

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

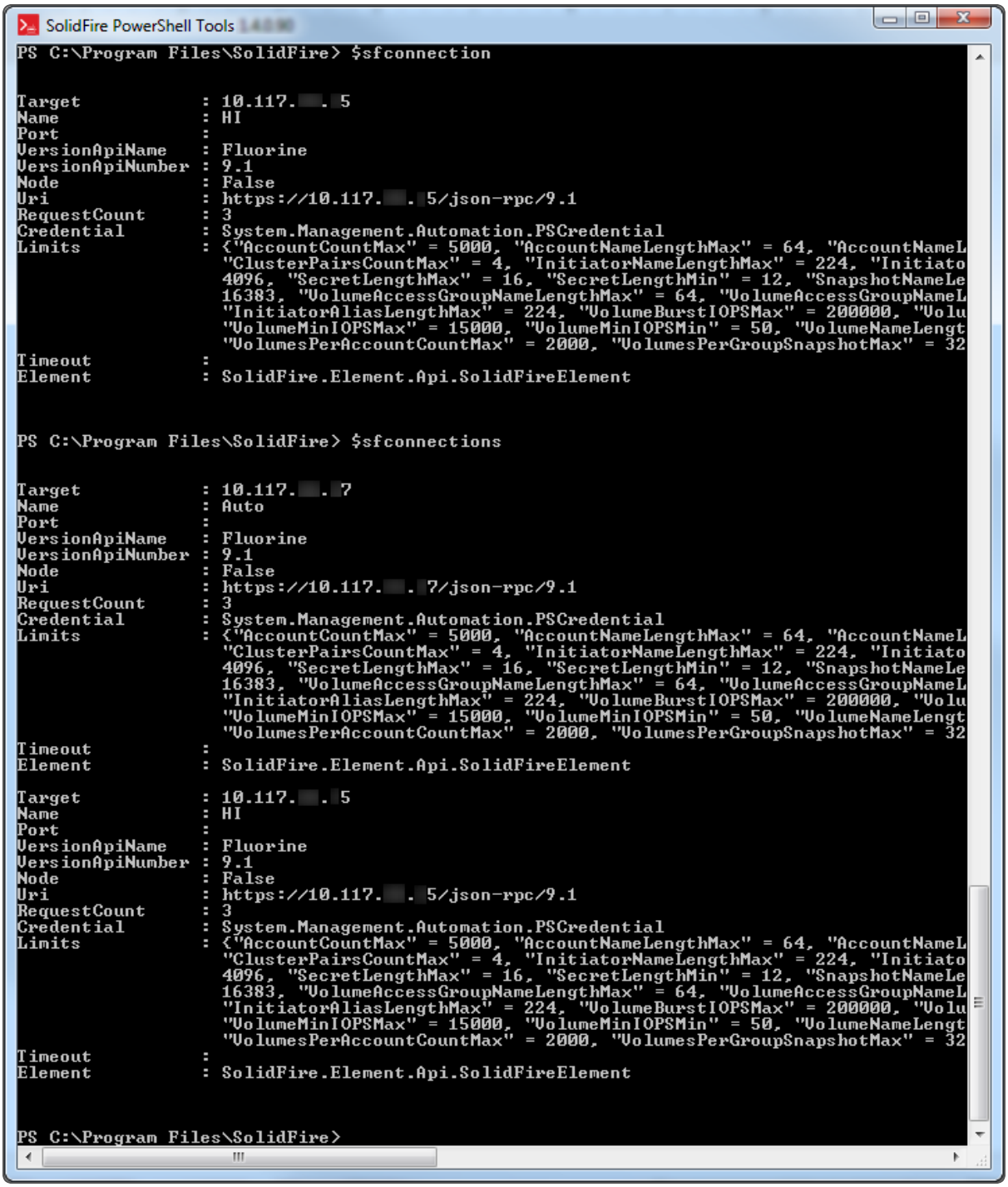
1. Type `Connect-SFCluster -Target <address> -VersionApi <version number>`.

   The following example demonstrates connecting to a SolidFire cluster with API version 8.0 (Oxygen).



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`.



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

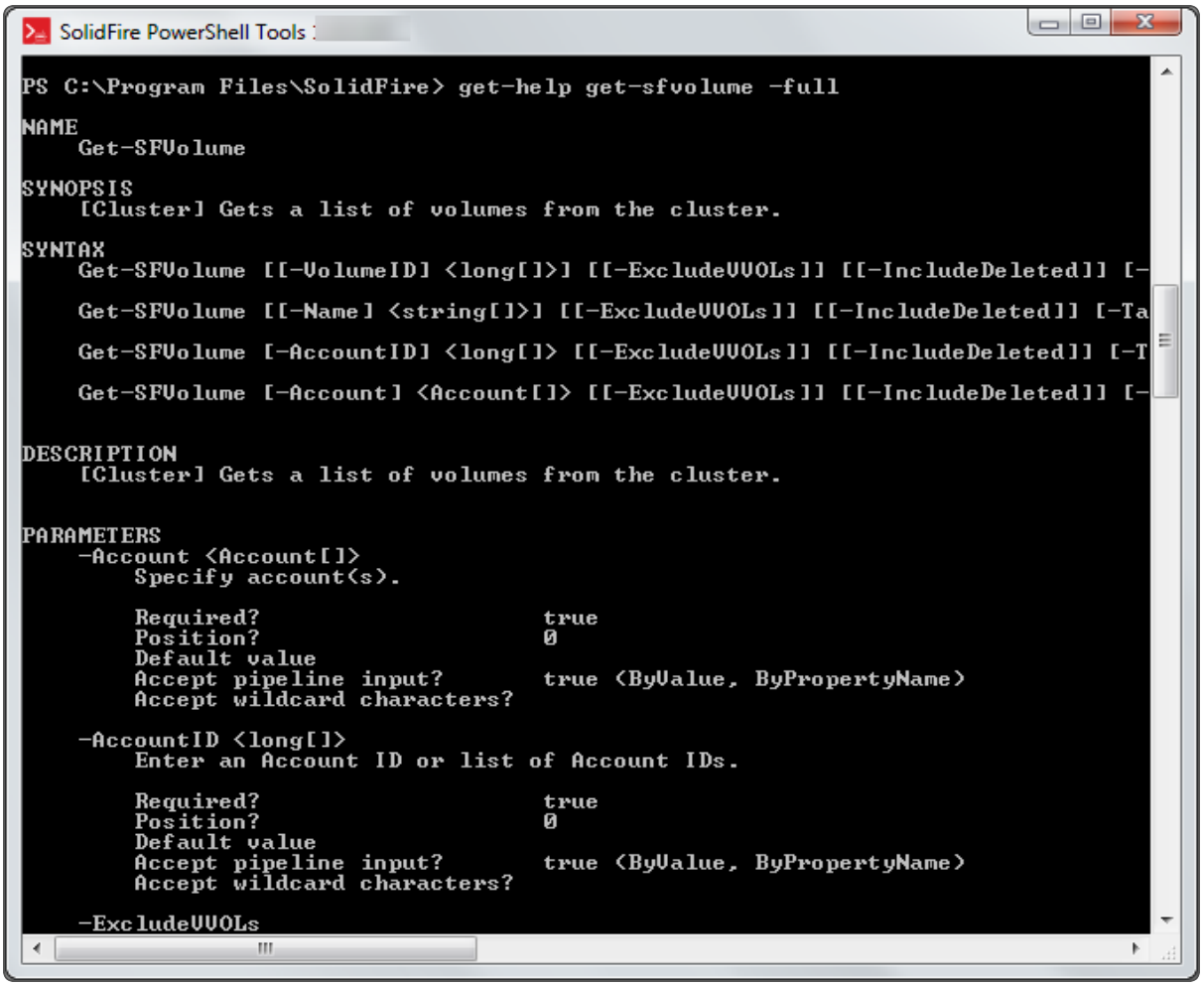
Each return value is documented as part of the online documentation for the SolidFire .NET SDK. This documentation can be found on GitHub.

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`:



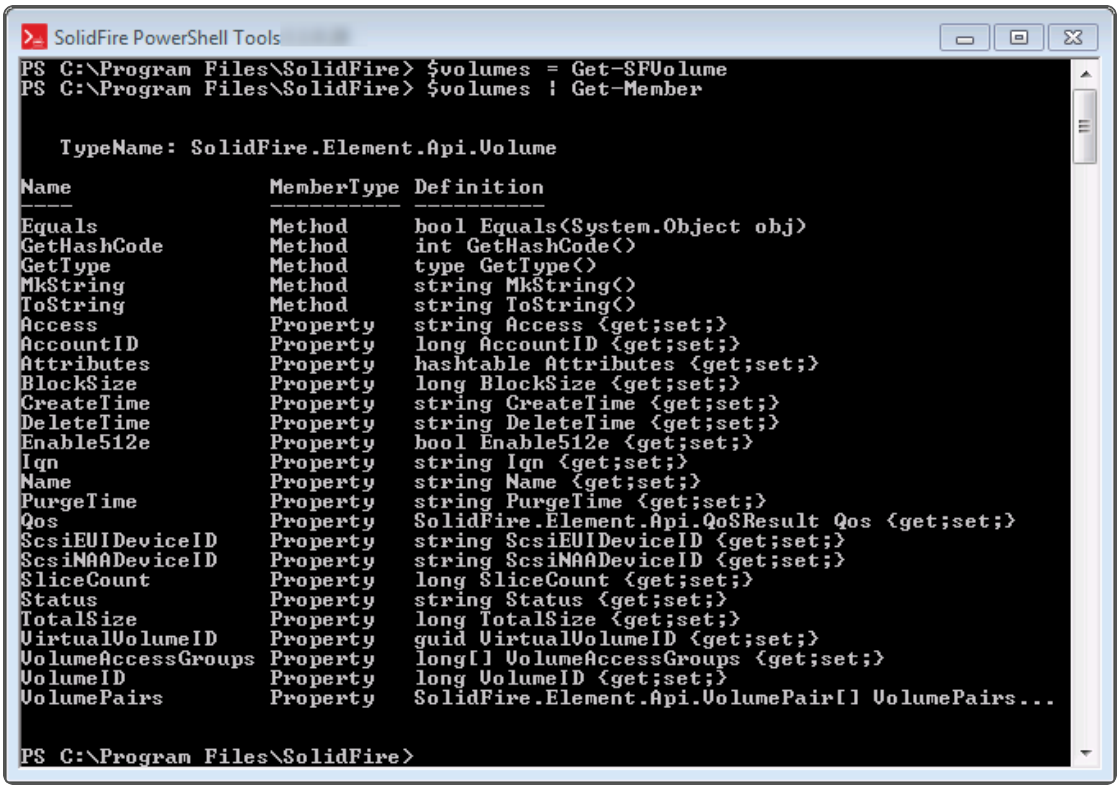
Leveraging Get-Member to Inspect Return Objects

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.



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`.

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/

6/12/17