Technical Report

Data ONTAP PowerShell Toolkit Primer

(First-Steps Guide for Data ONTAP Automated Management Using PowerShell)

Toyohiko Kambara,
Supported by Field Center of Innovation,
NetApp
August 2012 | TR-4161

First-Steps Guide for Data ONTAP Automated Management Using PowerShell

This report describes the basic operation procedures for automated management using PowerShell Toolkit, a management utility provided by NetApp. It provides NetApp storage administrators who are new to PowerShell with the basics of PowerShell and typical cmdlets of PowerShell Toolkit to quickly allow them to perform their day-to-day management tasks using PowerShell-based scripts. We hope this report will provide initial support to you in Data ONTAP automated management using Powershell.

Table of Contents

List of Figures and Tables

1 Introduction

We are sure that many of you who are engaged in day-to-day system management using Data ONTAP are accustomed to operations using native commands available in Data ONTAP on a terminal emulator installed on your desktop computer via Telnet, SSH, or rsh, rather than GUI-based operations from FilerView or System Manager.

At the same time, progress in virtualization technology and increased need for cloud-based infrastructure have resulted in an explosive increase in the number of systems to be managed by one administrator and a proportional increase in the complexity of storage management.

One important point in management operations is to maintain a familiar operation environment while enhancing the quality and efficiency of work. Nowadays, administrators are required to attain high efficiency in their management operations, advancing from a level of changing routine administrative procedures into batch processing to improve work efficiency to a level of automating the procedures in collaboration with orchestration tools and allowing users themselves to perform a series of tasks by a simple operation (self-service).

In many of the front-line tasks of Data ONTAP management, however, administrators come across problems, for example when using the familiar Data ONTAP native commands as the basis and copy-and-pasting text files that contain work procedures onto terminal emulators or extensively using rsh or expect and creating shell scripts. Due to the conditions described above, we have begun to hear from many administrators that they are suffering from increasingly complex management tasks in proportion with an increase in the number of management targets.

Data ONTAP PowerShell Toolkit, described in this report, is a powerful utility that provides support to administrators engaged in front-line management tasks.

Since the release of Data ONTAP PowerShell Toolkit, many have enjoyed the convenience of this utility. However, it is also true that others are hesitant to learn and use PowerShell as a new management interface, and are unable to shift away from using the familiar native commands.

These people must be wondering why they need to learn about a new interface again when it is the same command-line-based interface as the one they already know. The more experienced the administrator, the more likely they are to feel this way.

However, if you take the first step into Data ONTAP management using PowerShell, you will be on your way to experiencing the surprises and delights of the tool's power, innovation, convenience, and expandability, i.e., a whole new world that stimulates the imagination of engineers.

For engineers, a guide is a must when they start to handle new technologies, especially technologies that serve as operation interfaces.

This report has been written to guide the Data ONTAP administrators from the day-to-day management tasks to the first door to a world of automation and self-service.

I would be honored if you could take up and start reading this report casually without taking it too seriously.

I would like to express my sincere gratitude to many partners and users who informed me of the kind of needs, motivations, and backgrounds there are to automation in the front-line Data ONTAP management jobs.

I hope this report will provide support to you as the first-steps guide to Data ONTAP automated management using PowerShell.

2 About Data ONTAP PowerShell Toolkit

Data ONTAP PowerShell Toolkit (PSTK) is a PowerShell extension module that allows you to control Data ONTAP, a storage management OS installed on the NetApp FAS series, from a PowerShell environment.

Usually, you manage Data ONTAP from FilerView or System Manager GUI, or by directly logging in to Data ONTAP and using CLI. PSTK allows you to perform most of the common management tasks using Powershell cmdlets.

PSTK is developed and maintained by the Management Framework Integration Team of NetApp and is distributed free of charge through the user community site.

"NetApp Community," a community site for NetApp users

NetApp Community (http://community.netapp.com) is a technical community site for NetApp product users, partners, and employees. This site, where storage administrators exchange information and various technical documents are disclosed, provides you with a wide range of extremely useful information.

PSTK is one of the many spaces in the community site, and you are welcome to participate and upload your own scripts that you have written according to this report.

Figure 1) NetApp Community home page

Space: PowerShell Toolkit
"Product & Solutions" -> "Microsoft Applications and Environments" -> "PowerShell Toolkit"
https://communities.netapp.com/community/products_and_solutions/microsoft/powershell

PSTK is a free utility developed to provide support for Data ONTAP administrators to improve the efficiency of daily tasks. It is continuously enhanced with additional functionalities and bug fixes based on feedback from the community site. Additionally, the user community site is where users share scripts created with PSTK, and you can download any of the scripts actually created and used by administrators worldwide and use it as your own sample.

PSTK is not supported by maintenance support services as it is a free utility. However, it is used as a core engine for various NetApp storage management software products, and is a fully reliable and useful tool for day-to-day management tasks.

2.1 Features of Data ONTAP PowerShell Toolkit in Data ONTAP Management

PSTK provides more than 1,000 commands for Data ONTAP management, called cmdlets. Cmdlets are a major feature of PowerShell that are highly convenient and can be used intuitively and administrators with PowerShell 2.0 environment can start using PSTK immediately.

Furthermore, PSTK is capable of handling return values of commands as objects, which is also a feature of PowerShell itself. Thus, it can significantly decrease text formatting and filter processing for connecting commands, which is a troublesome task in the UNIX/Linux shell environment.

If you have created scripts using the standard functions of Data ONTAP such as rsh, you probably found it difficult to implement return value control, conditional branching, error handling, and exception processing. PSTK significantly reduces such complexity in script authoring because control is available in an execution environment.

Nowadays, the Microsoft System Center product and the VMware vSphere vCenter product also provide cmdlets and APIs using PowerShell. Likewise, the NetApp software related to the Microsoft and VMWare products provide the PowerShell cmdlets as their CLI. Using these PowerShell cmdlets with PSTK allows you to seamlessly automate day-to-day management tasks in virtualization environments as well as to easily create useful batch scripts - one of the major advantages of using PSTK.

2.2 How Data ONTAP PowerShell Toolkit Works

The NetApp storage management software products including Data ONTAP provide APIs for developers, based on the "Open Management Framework". These APIs are shared not only by other NetApp software products such as OnCommand System Manager, Unified Manager, and SnapManager series but also by the third-party collaboration modules and orchestration software for NetApp.

About Open Management Framework

NetApp releases the storage management APIs for developers together with a development support kit called NetApp Manageability SDK (NMSDK). NMSDK is available via download from the NOW site (http://support.netapp.com/NOW) to users of NetApp products and engineers of NetApp partner companies by filling out a simple questionnaire form including purposes of use.

NMSDK consists of API libraries for external control of Data ONTAP and OnCommand Unified Manager (Core API, ONTAP-API for controlling Data ONTAP, and DFM-API for controlling OnCommand Unified Manager) and a development support kit, and supports major object-oriented languages such as Java, Perl, .Net, and C#.

Compared to PowerShell Toolkit, NMSDK provides more powerful and varied control, but less ease of use from the administrative point of view as it poses higher technical difficulties to developers.

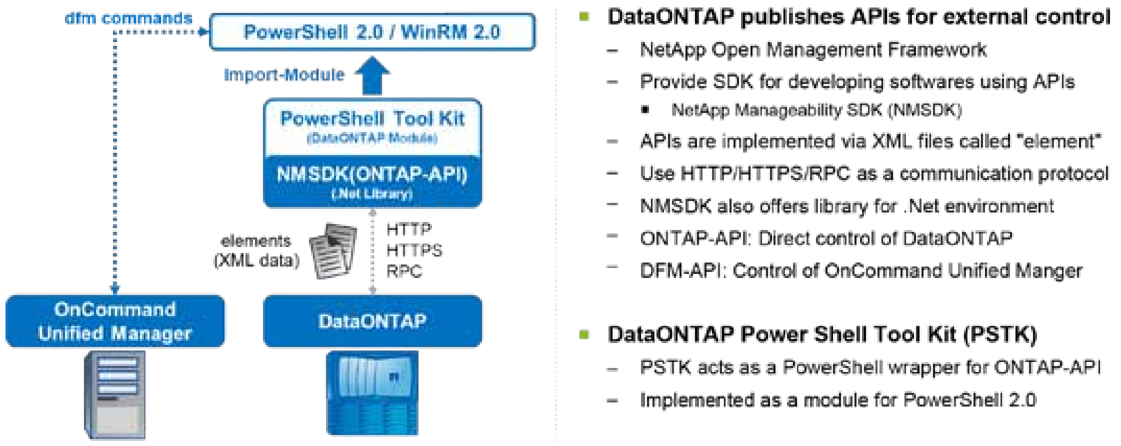
NMSDK is a kit for developers to develop full-fledged orchestration tools or NetApp collaboration software for ISVs. NetApp recommends NMSDK if your reason for automation is one of the above.

The external control APIs provided by Data ONTAP and OnCommand Unified Manager are implemented by exchanging XML files - called elements, that store control instructions and information - via HTTP, HTTPS, or RPC. NMSDK is also a set of libraries for handling these XML data as objects in each language.

PSTK uses the .NET library in NMSDK and acts as a PowerShell wrapper for the Data ONTAP control API called "ONTAP-API". This makes it possible to execute almost all the functions included in ONTAP-API through PowerShell.

Although the NMSDK functions called DFM-API provided by OnCommand Unified Manager cannot be used directly from PSTK, using native commands provided by OnCommand Unified Manager for Windows (dfm commands) or cmdlets for PowerShell enables seamless creation of scripts.

Figure 2) How Data ONTAP PowerShell Toolkit works



- **DataONTAP publishes APIs for external control**
  - NetApp Open Management Framework
  - Provide SDK for developing softwares using APIs
    - NetApp Manageability SDK (NMSDK)
  - APIs are implemented via XML files called "element"
  - Use HTTP/HTTPS/RPC as a communication protocol
  - NMSDK also offers library for .Net environment
  - ONTAP-API: Direct control of DataONTAP
  - DFM-API: Control of OnCommand Unified Manger
- **DataONTAP Power Shell Tool Kit (PSTK)**
  - PSTK acts as a PowerShell wrapper for ONTAP-API
  - Implemented as a module for PowerShell 2.0

2.3 NetApp Automation Solutions

This section describes the NetApp products for automation.

NetApp provides products and utilities that are designed for certain automation purposes, and it is important to select one that suits your own needs.

PSTK, described in this report, is mainly intended for administrators who are engaged in day-to-day storage management tasks using various functions of Data ONTAP. A principal objective of this utility is to help them improve work efficiency by creating batch scripts for complex operations. Furthermore, PSTK is also adopted as an execution engine for more powerful automation products such as Workflow Automation, listed below, and languages for creating custom functions. If a more advanced usage is required, you will be able to tap into the knowledge acquired from PSTK and the scripts created using it.

Table 1) Automation solutions available from NetApp

Product/Utility	Features	Target
OnCommand Unified Manager (Provisioning Manager function)	• Automation software provided by NetApp • Can be used via wizard-type GUI operations according to a predefined framework • A GUI-defined storage configuration can be used as templates called "storage services" and automatically deployed through CLI or NMSDK	• Administrators who want to easily automate tasks in a simple storage configuration • Developers who want to quickly implement predefined storage services

Product/Utility	Features	Target
PowerShell Toolkit (PSTK)	• Cmdlets in PowerShell environment • An extensive range of commands helps you create scripts easily	• Administrators who want to batch-process their management tasks on complex volume/network configurations where storage is used for business systems or virtualization environments • Intended audience of this report
Workflow Automation	• Allows you to create a flow of automation jobs using predefined functions • In addition to predefined functions, you can also create custom functions using PSTK or NMSDK	• Developers of automated control processing for large-scale tenant deployment or configuration changes based on usage in standardized system configurations
NetApp Manageability SDK	• Direct control of NetApp products • Advanced development support kit using object-oriented languages such as Java, C#, and .Net	• Developers who want to develop full-fledged orchestration tools or NetApp collaboration software modules

3 Preparing the Environment for Using PowerShell Toolkit

This section describes how to prepare the environment for using PSTK.

3.1 Preparing Data ONTAP

First, you need to prepare Data ONTAP.

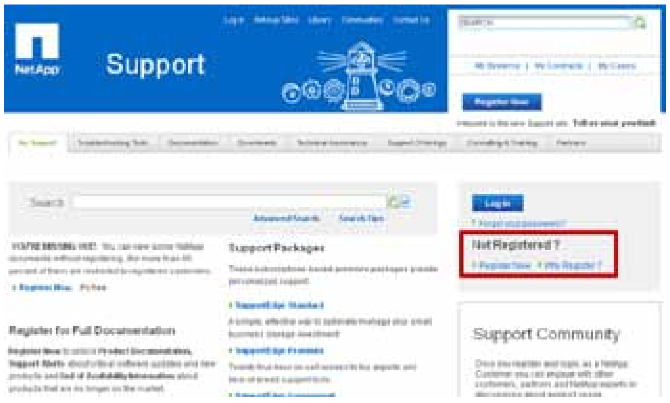
If you find there is no free FAS storage unit available for personal use, don't worry.

NetApp provides a free Data ONTAP simulator on the support site (former NOW site) available via download to NetApp product users and NetApp partners. In other words, any reader of this report who wishes to use PSTK can obtain the Data ONTAP simulator.

Registering Support Site Account

The NetApp support site provides product/technical information. The account that you register and use on this site acts a single sign-on account for other Web sites provided by NetApp. Enter the URL (http://support.netapp.com/) to access the home page and click "Register Now" located on the right.

Figure 3) Support site home page



Any NetApp product user or NetApp partner can register a support site account free of charge. We recommend that you take this opportunity to register an account if you do not have one yet. You can use the account with guest or user permissions as soon as the registration is complete, but it usually takes two or three business days for verification to upgrade it to NetApp partner permission. After the verification process is completed, the account is automatically upgraded. Please check your account status several days after registration.

Data ONTAP Simulator
The Data ONTAP simulator software is available for download from the support site (http://support.netapp.com/). It runs on a virtual desktop environment such as VMware Player. The Installation Guide and a simulator license are included with the software. To download the software, access the support site and select Download -> Utility ToolChest -> Simulate ONTAP. (http://support.netapp.com/NOW/cgi-bin/simulator) • For Data ONTAP 7.x, it is provided as an application program that runs on Linux. Therefore, set up Linux as the guest OS and then install and configure the simulator. It is supported in both 32-bit and 64-bit operation environments because Data ONTAP 7.x is a 32-bit OS. • For Data ONTAP 8.x, it is provided as a virtual machine that runs on VMware Player, VMware Workstation, and vSphere ESX Server. It requires a 64-bit operation environment because Data ONTAP 8.x is a 64-bit OS.

As a simulator, Data ONTAP simulator has some limitations. Unavailable functions and features include the Fibre Channel Protocol support, some Data ONTAP functions such as the MetroCluster function that depend on physical mechanisms, and the capability to create TB-level high-capacity volumes. However, it is fully usable as a development environment because it provides major functions such as the IP-based communication protocols (iSCSI, NFS, and CIFS), MultiStore, FlexVol, Snapshot, and FlexClone.

Data ONTAP Simulator runs as a guest OS on virtualization environments such as VMware Player and VMware Workstation, which are installed on commonly used platforms such as Windows, Linux, and MacOS.

In other words, one of the biggest advantages of PSTK is that a development environment can be prepared on administrators' own computers.

We will skip the detailed steps for installing and configuring Data ONTAP Simulator here. The following sections assume the simulator is installed and configured as specified in the tables below.

Table 2) Data ONTAP Simulator environment used in this report

Data ONTAP Simulator	Guest OS	Virtual Environment
Data ONTAP Simulator 7.3.5	CentOS 5.6 (32-bit)	VMware Player 4.0.2

Table 3) Data ONTAP Simulator default settings used in this report

Simulator Setting	Value
Changed value at initial deployment of Simulator (setup.sh)	Addition of twenty-two 129MB HDDs (24 HDDs in total)
hostname	dots1
IPv6	no
virtual network interfaces	no
IP address for ns0	192.168.151.101

Simulator Setting	Value
netmask for ns0	255.255.255.0
media type for ns0	auto
IP address for ns1	192.168.151.102
netmask for ns1	255.255.255.0
media type for ns1	auto
setup through the web	no
default gateway	192.168.151.254
administration host	192.168.151.1 (Address of the desktop computer)
timezone	GMT
filer located	PSTK
language for multi-protocol files	C
DNS	no
NIS	no
ACP for SAS	no
Password	password
WINS	Ctrl-c (Suspend)

NMSDK ONTAP-API, on which PSTK is based, maintains backward compatibility in principle. The functions, settings, and attributes that are added to new versions of Data ONTAP are set as additional elements and parameters so that previous elements and parameters are maintained.

To perform actual control, you need to first create a session for Data ONTAP and specify the API version to be used for ONTAP-API, thus ensuring the backward compatibility. (In PSTK, you need not trouble yourself with the version specification).

As described earlier, PSTK communicates with Data ONTAP via HTTP, HTTPS, or RPC. Although the Data ONTAP settings listed below are required to use PSTK, these settings are enabled by default and you don't have to change any settings. Another big advantage of this tool is that you can perform PSTK-based management right after the initial setup if the environment is connected to network.

Table 4) Data ONTAP settings required for using PSTK

Data ONTAP Setting	Value	Description
Password for administrative user ("root" in this report)	(To be supplied by user)	
options httpd.admin.enable	on	Enables administrative access via HTTP.
options httpd.admin.ssl.enable	on	Enables administrative access via HTTPS.

3.2 Deploying PowerShell

PSTK requires following PowerShell environment.

Table 5) Required operating environment of Data ONTAP PowerShell Toolkit

Required operating environment of Data ONTAP PowerShell Toolkit
• PowerShell 2.0 • WinRM 2.0 (Windows Remote Management) * These are provided by Microsoft as a component of Windows Management Framework.

These are pre-installed on Windows 7 and Windows 2008 Server R2. On other Windows versions, you need to download and install them yourself.

They are available via free download from the Microsoft Web site. Please obtain and install the ones compatible with your Windows version.

Table 6) Windows OS versions compatible with PowerShell

Windows OS versions compatible with PowerShell 2.0 (Versions that you need to install yourself)
• Windows Server 2008 (SP1) and Windows Server 2003 (SP2) • Windows Vista (SP1 and SP2) and Windows XP (SP3) * Check with Microsoft for the latest information.

(Tea Break) About PowerShell Console

PowerShell is a very advanced and powerful tool. However, the terminal console that comes with it is not very easy to use.

When creating a script, you require a user-friendly and productive terminal console as well as a text editor that understands the language syntax to support you. For your information, I would like to mention some terminal consoles that I use.

1. PowerShell ISE

When PowerShell is installed, a development utility called PowerShell ISE is installed as well.

It is a very useful editor for writing a script with three panes for input, output, and text editor displayed in one window.

In the text editor, you can press the Tab key to autocomplete the text, which significantly decreases the keyboard input when writing a script.

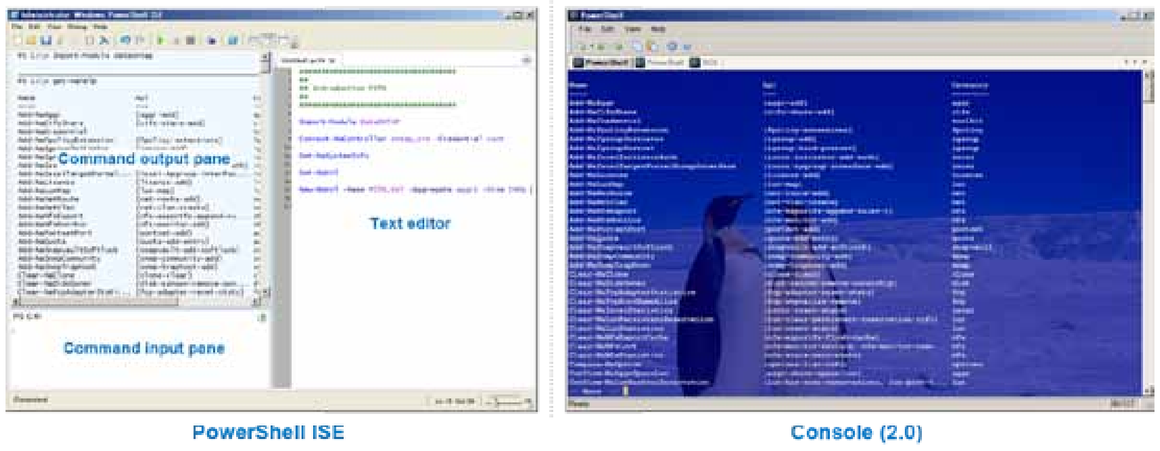
2. Console

This is an open-source terminal emulator for the Windows shell available from SourceForge.net.

With the tab-switching-type screen, this highly user-friendly terminal emulator allows you to launch more than one PowerShell session and customize the keyboard shortcuts and mouse operations (e.g. clicking the left mouse button to copy and the right button to paste) according to your preferences.

http://sourceforge.net/projects/console/

Figure 4) Screenshots of PowerShell ISE and Console (Version 2.0)



3.3 Preparation of PowerShell Environment

After installing PowerShell, make sure that its version supports PSTK.

Check the PowerShell version using the following commands: Major version of 2 with Minor version of 0 or higher is OK.

PS command 1) Check the PowerShell version

```
PS C:\> $host.version

Major  Minor  Build  Revision
-----  -----  -----  --------
2      0      -1     -1                <- Check that Major version is 2 and Minor version is 0 or higher.
```

We now change the Execution Policy for the created script so that it can be executed without trouble. By default, PowerShell does not allow execution of a script file for security purposes.

In the example below, the policy for a script file acquired via network is set to RemoteSigned, which requires a digital signature. The user profile file described later is also a kind of script and requires at least the RemoteSigned execution policy. For more information about execution policies, refer to the information provided by Microsoft.

PS command 2) Check the script execution policy (default status)

```
PS C:\> Get-ExecutionPolicy

Restricted
```

PS command 3) Changing the script execution policy

```
PS C:\> Set-ExecutionPolicy RemoteSigned

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution
policy might expose you to the security risks described in the about_Execution_Policies help
topic. Do you want to change the execution policy?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y
```

Now create a profile file that configures the user environment in PowerShell.

In the PowerShell environment, the profile file location is stored in the $profile variable. You need to create this variable as it is not created by default. The following shows an example for a Windows XP SP3 environment.

PS command 4) Check the profile file path

```
PS C:\> $profile

C:\Documents and Settings\<user name>\My Documents\WindowsPowerShell
                                                            \Microsoft.PowerShell_profile.ps1
```

PS command 5) Creation of profile file and editing using Notepad

```
PS C:\> New-Item -Path $profile -ItemType file -Force

PS C:\> Test-Path $profile

True                          <- True is returned if the file already exists.

PS C:\> notepad $profile      <- Edit it using a text editor such as Notepad.
```

After installing PSTK, add the following line in the profile file created. This makes PSTK available automatically next time PowerShell is started.

PS command 6) Add a line to PowerShell profile file (importing PSTK module)

```
Import-Module DataONTAP;
```

The basic environment for using PSTK is ready now.

Continue to the next section to start using PSTK.

4 Deploying and Configuring PowerShell Toolkit

So far you have prepared the environment for using PSTK. Now you can download and integrate PSTK into your PowerShell environment.

4.1 Downloading and Installing PowerShell Toolkit

PSTK is available to download from NetApp Community, a NetApp community site.

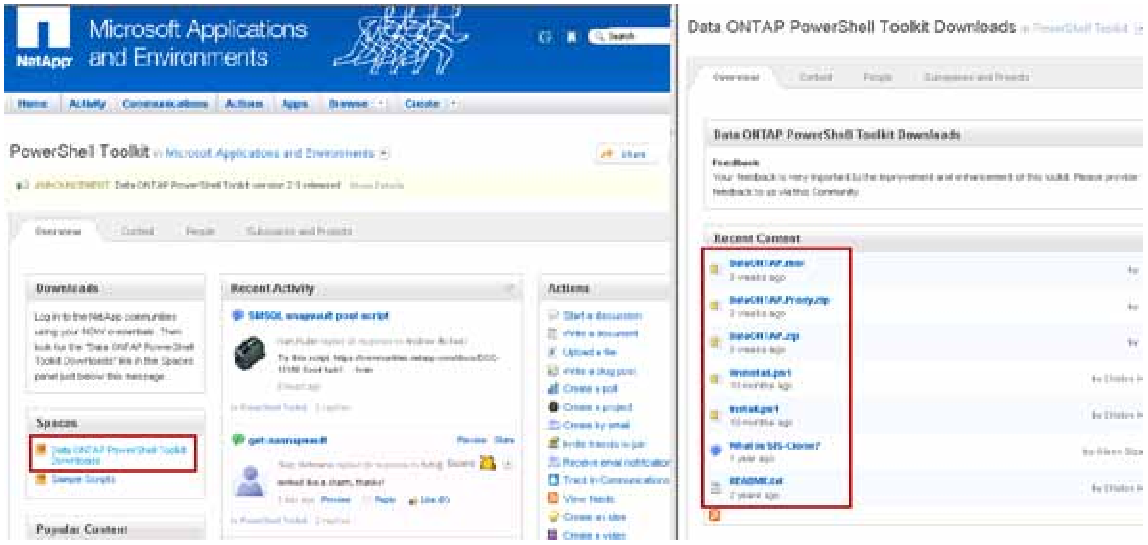
Log in to the site and navigate to "PowerShell Toolkit" space. The PSTK download space is visible only if you have logged in.

Download PowerShell Toolkit

The download space is only displayed if you logged in to NetApp Community with the support site account.

The "DataONTAP.zip" file is the PSTK module.

Figure 5) Download PowerShell Toolkit



URL: https://communities.netapp.com/community/products_and_solutions/microsoft/powershell/data_ontap_powershell_toolkit_downloads

The download space contains the files listed below. You need at least the main PSTK module "Data ONTAP.zip" file. The following outlines all of the available files for your reference.

Table 7) List of files available from download site

File Name	Description
DataONTAP.zip	PSTK module: Extract the zip file into the Modules folder of PowerShell
What is SIS-Clone?	Explains the SIS-Clone function
README.txt	Quick guide for tasks such as installation
Uninstall.ps1	PowerShell script for uninstalling PSTK: Not particularly required as it can also be manually uninstalled
Install.ps1	PowerShell script for installing PSTK: Not particularly required as it can also be manually installed

After downloading the "DataONTAP.zip" file, extract it into the folder where you want to store the PowerShell module. With PowerShell, you can group cmdlets used for each roles and features deployed in Windows into a "module" if required. PSTK is implemented as a module.

There are two folders for storing PowerShell modules as shown below. Specify the one that suits your usage. Files for the module must exist directly under the DataONTAP folder, such as "\Modules\DataONTAP\<filename>.dll files".

PS command 7) Check the folder path for storing modules

```
PS C:\> $env:PSModulePath

C:\Documents and Settings\<user name>\My Documents\WindowsPowerShell\Modules;
                                          C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules\
```

Table 8) Two folder paths for storing PowerShell Toolkit

Path	When
%windir%\System32\WindowsPowerShell\v1.0\Modules	Used by all users
%UserProfile%\Documents\WindowsPowerShell\Modules	Used only by a specific user

After extracting the module successfully, use the following command to check if it is recognized correctly.

PS command 8) Check if the PowerShell Toolkit module is recognized

```
PS C:\> Get-Module -ListAvailable

ModuleType Name                                 ExportedCommands
---------- ----                                 ----------------
Manifest   BitsTransfer                         {}
Manifest   DataONTAP                            {}                      <- OK if "DataONTAP" is displayed
```

As of the writing of this report, the latest PSTK version is 2.1. This report assumes the use of version 2.0.

Table 9) Target version of PowerShell Toolkit in this report

Target version of PowerShell Toolkit in this report
Data ONTAP PowerShell Toolkit 2.0

4.2 Configuring PowerShell Toolkit

After extracting the module, start PowerShell and import the module.

PS command 9) Import the PowerShell Toolkit module

```
PS C:\> Import-Module Data ONTAP
```

Use the "Import-Module" command to import the PSTK module. If this line is added to the profile file, the PSTK module is automatically imported every time PowerShell is started.

When you create a script using PSTK, also make sure to add the same command on the first line of the script to import the module.

To check the module that has been imported in the current PowerShell session, use the "Get-Module" cmdlet.

PS command 10) Check the module used in PowerShell session

```
PS C:\> Get-Module

ModuleType Name                   ExportedCommands
---------- ----                   ----------------
Manifest   DataONTAP              {Invoke-NaSnapmirrorThrottle, Resume-NcJob...
```

If the module has been imported successfully, a list of cmdlets in PSTK can be displayed.

PS command 11) Display cmdlets included in PowerShell Toolkit

```
PS C:\> Get-NaHelp | more

Name                                       Api                                       Category
----                                       ---                                       --------
Add-NaAggr                                 {aggr-add}                                aggr
Add-NaCifsShare                            {cifs-share-add}                          cifs
Add-NaCredential                                                                     toolkit
<snip>
-- More  --
```

4.3 Displaying Details of Commands in PowerShell Toolkit

This section describes the procedure to display the help for cmdlets, which is useful for future use.

In addition to the help for PowerShell itself, you can also use the cmdlets "`Get-NaHelp`" and "`Show-NaHelp`."

PS command 12) Display the list of PowerShell Toolkit cmdlets using `Get-NaHelp`

```
PS C:\> Get-NaHelp

Name                                       Api                                       Category
----                                       ---                                       --------
Add-NaAggr                                 {aggr-add}                                aggr
Add-NaCifsShare                            {cifs-share-add}                          cifs
Add-NaCredential                                                                     toolkit
<snip>
```

The cmdlets included in PSTK are classified into "categories".

PS command 13) Display the list of categories using `Get-NaHelp` cmdlet with `-CategoryList` option

```
PS C:\> Get-NaHelp -CategoryList
aggr
cf
cifs
clock
clone
<snip>
```

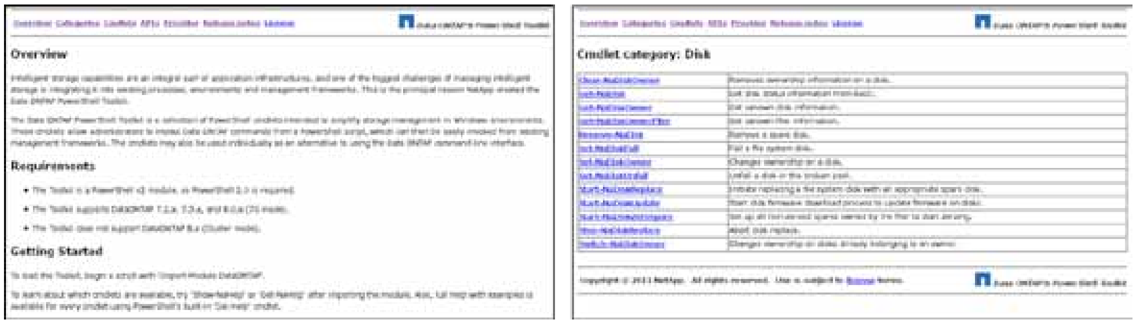
While "`Get-NaHelp`" lists cmdlets or categories, the "`Show-NaHelp`" command allows you to display HTML-based Help that is included with PSTK on your browser.

You can simply click a category to display cmdlets for various operations in the category.

PS command 14) Display HTML Help using `Show-NaHelp` cmdlet

```
PS C:\> Show-NaHelp
```

Figure 6) HTML Help screens displayed by the `Show-NaHelp` cmdlet



You can also check options and syntax for a cmdlet by entering the following cmdlet.

PS command 15) Check options and syntax for a cmdlet using `Get-Help` cmdlet

```
PS C:\> Get-Help Get-NaVol
             or
PS C:\> Get-NaVol -?

NAME
    Get-NaVol

SYNTAX
    Get-NaVol [[-Names] <String[]>] [-Aggregate <String>] [-Controller <NaController>]
[-Verbose] [-Debug] [-ErrorAction <ActionPreference>] [-WarningAction <ActionPreference>]
[-ErrorVariable <String>] [-WarningVariable <String>] [-OutVariable <String>]
[-OutBuffer <Int32>]
```

If you enter `Get-NaHelp` followed by a PSTK cmdlet, the API name of .Net and the cmdlet category name are displayed.

PS command 16) Use `Get-NaHelp` cmdlet followed by a PSTK cmdlet

```
PS C:\> Get-NaHelp Get-NaVol

Name                                      Api                                       Category
----                                      ---                                       --------
Get-NaVol                                 {volume-list-info}                        volume
```

If you enter "`Get-NaHelp`" followed by a category name, a list of cmdlets for that category is displayed.

PS command 17) Use `Get-NaHelp` followed by a category name

```
PS C:\> Get-NaHelp -Category volume

Name                                        Api                                       Category
----                                        ---                                       --------
Get-NaVol                                   {volume-list-info}                        volume
Get-NaVolAutosize                           {volume-autosize-get}                     volume
Get-NaVolCharmap                            {volume-charmap-get}                      volume
Get-NaVolCloneSplit                         {volume-clone-split-status}               volume
Get-NaVolCloneSplitEstimate                 {volume-clone-split-estimate}             volume
Get-NaVolContainer                          {volume-container}                        volume
Get-NaVolLanguage                           {volume-get-language}                     volume
<snip>
```

You may have already noticed that cmdlets included in PSTK follow a certain naming convention. PSTK cmdlets, like native PowerShell cmdlets, are named according to the naming convention described below. It is defined so that you can easily understand which cmdlets control Data ONTAP as you write a script. If you apply this naming convention when writing your own script, the resulting script code would become very readable.

Table 10) Naming convention for PowerShell Toolkit cmdlets

Naming Convention for PowerShell Toolkit cmdlets	
<verb> - Na<abbreviated_category_name><operation_description>	Example: `Set-NaSnapshotAutodelete`
With Na: Control instruction to Data ONTAP Without Na: Format conversion of numbers, etc.	Example: `ConvertTo-DateTime`

Below is a list of <verbs> commonly used in PSTK and their usage.

Table 11) Common verbs used in PowerShell Toolkit cmdlets

Verb	Usage
`Get-`	Gets an object and its data. Gets a target object and passes it to a subsequent cmdlet, etc.
`Set-`	Changes attribute or setting of a target object.
`New-`	Creates a new object.
`Add-`	Adds an object or a setting value to an existing object.
`Remove-`	Removes existing objects and settings.
`Invoke-`	Starts Data ONTAP background processes (such as A-SIS and SnapMirror).
`Enable-`	Enables Data ONTAP functions and protocols.
`Disable-`	Disables Data ONTAP functions and protocols.

5 Connecting to Data ONTAP

Now you are ready to use PSTK, let's start Simulator and control Data ONTAP.

When you use PSTK either interactively with the CLI or through batch processing using a script, the following workflow is basically used.

Table 12) Control flow using PowerShell Toolkit

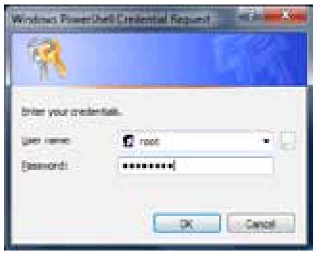
Control Flow using Data ONTAP PowerShell Toolkit	
1	Connect to Data ONTAP using `Connect-NaController` cmdlet
2	Perform tasks using cmdlets
3	Exit PowerShell session (disconnect from Data ONTAP)

5.1 Connecting with `Connect-NaController` Cmdlet

Connect to Data ONTAP using "`Connect-NaController`" cmdlet.

PS command 18) Use of `Connect-NaController` cmdlet (1)

```
PS C:\> Connect-NaController dots1 -Credential root -HTTP

* The password input screen appears. Enter "password," the specified root user password in dots1.
```



```
Name          Address           Ontapi   Version
----          -------           ------   -------
dots1         192.168.151.101   1.14     NetApp Release 7.3.5: Mon Nov 22 08:37 PM:51 PST 2010

PS C:\>
```

In the examples of this report, HTTP is explicitly specified as the communication protocol. As described before, HTTP, HTTPS, or RPC is used for communications with Data ONTAP. RPC is used by default when none is specified. Specify a protocol suitable for your environment.

Once the password is entered and you are connected to Data ONTAP, information about Data ONTAP is shown and the prompt is displayed again. Now you have completed the connection to Data ONTAP.

5.2 Using PSCredential Object during Connection

The screen that appears to prompt you for password may seem odd.

You would think that it should be possible to specify the password in an argument for interactive CLI.

For security reasons, PowerShell handles credentials such as user permissions using the PSCredential object. "`Connect-NaController`" cmdlet also uses the PSCredential object for the "`-Credential`" argument.

PS command 19) Use of `Connect-NaController` cmdlet (2)

```
PS C:\> Get-Help Connect-NaController

NAME
    Connect-NaController

SYNTAX
    Connect-NaController [-Name] <String> [-Port <Nullable`1>] [-Credential <PSCredential>]
 [-HTTPS] [-HTTP] [-RPC] [-Transient] [-Vfiler <String>] [-Timeout <Nullable`1>]
 [-Verbose] [-Debug] [-ErrorAction <ActionPreference>] [-WarningAction <ActionPreference>]
 [-ErrorVariable <String>] [-WarningVariable <String>] [-OutVariable <String>]
 [-OutBuffer <Int32>]
```

Depending on the value for the `-Credential` argument, PowerShell performs one of the following.

Table 13) Value for `-Credential` argument and corresponding PowerShell behavior

Value	PowerShell Behavior
No value	Popup window appears, prompting for user name and password.
Text String	The text is treated as user name and popup window appears, prompting for password.
PSCredential object	Authentication is performed using the object as credentials.

Now, create the PSCredential object assuming you would write a script for connection. Then connect to Data ONTAP again.

To create a PSCredential object, you need the user name (String) and encrypted password (SecureString) as arguments. You define user name and password as variables, convert the password to SecureString using "`ConvertTo-SecureString`" cmdlet, and then create PSCredential object.

In the following example, a series of operations is performed in interactive CLI. When writing a script code, use the parts in bold blue font.

PS command 20) Use of PSCredential object in `Connect-NaController`

```
PS C:\> $FasUser = “root”
PS C:\> $FasPasswd = “password”
PS C:\> $FasPasswd
password
* Displayed as String.

PS C:\> $SecureFasPasswd = ConvertTo-SecureString $FasPasswd -AsPlainText -Force
PS C:\> $SecureFasPasswd
System.Security.SecureString
* Encrypted as SecureString and hidden from display.

PS C:\> $FasCred = New-Object -TypeName System.Management.Automation.PSCredential ($FasUser,
$SecureFasPasswd)

PS C:\> Connect-NaController dots1 -Credential $FasCred -HTTP

Name          Address          Ontapi   Version
----          -------          ------   -------
dots1         192.168.151.101  1.14     NetApp Release 7.3.5: Mon Nov 22 08:37 PM:51 PST 2010
```

Script 1) Example of `Connect-NaController` cmdlet in a script code (1)

```
### Configure variables
$FasSystem = "dots1"
$FasUser = "root"
$FasPasswd = "password"
$SecureFasPasswd = ConvertTo-SecureString $FasPasswd -AsPlainText -Force
$FasCred = New-Object -TypeName System.Management.Automation.PSCredential ($FasUser,
$SecureFasPasswd)

### Connect to Data ONTAP
Connect-NaController $FasSystem -Credential $FasCred -HTTP
```

If you want to avoid including a password in the script and prompt the user for password input instead, you can use `Read-Host` cmdlet.

Script 2) Example of `Connect-NaController` cmdlet in a script code (2)

```
### Configure variables
$FasSystem = "dots1"
$FasUser = "root"
$FasPasswd = Read-Host "Enter password for root of dots1"
* Prompt for user input when executing the script.

$SecureFasPasswd = ConvertTo-SecureString $FasPasswd -AsPlainText -Force
$FasCred = New-Object -TypeName System.Management.Automation.PSCredential ($FasUser,
$SecureFasPasswd)

### Connect to Data ONTAP
Connect-NaController $FasSystem -Credential $FasCred -HTTP
```

5.3 Caching PSCredential Object with `Add-NaCredential`

NetApp FAS series controllers are HA-paired for redundancy in many environments, requiring you to manage two Data ONTAP systems. When managing the pair of Data ONTAP systems, it is troublesome to enter user names and passwords or create PSCredential object every time they are required.

In such a case, you can create an object and use "`Add-NaCredential`" cmdlet to cache it in the PowerShell execution environment to save trouble.

PowerShell has a concept of "Scope" for execution environment (See Appendix A.) The "`Add-NaCredential`" cmdlet caches the PSCredential object in the scope of Current User Context, a user context that is currently executing PowerShell. (This can be changed using the `-SystemScope` attribute.)

Therefore, if the same user launches PowerShell during operations to control another Data ONTAP system, the cached PSCredential object is retained. Thus, in this PowerShell environment, the user can connect to Data ONTAP without specifying credentials for the "`Connect-NaController`" cmdlet.

PS command 21) Cache PSCredential object using `Add-NaCredential` cmdlet

```
PS C:\> $FasUser = "root"
PS C:\> $FasPasswd = "password"
PS C:\> $SecureFasPasswd = ConvertTo-SecureString $FasPasswd -AsPlainText -Force
PS C:\> $FasCred = New-Object -TypeName System.Management.Automation.PSCredential ($FasUser,
$SecureFasPasswd)

PS C:\> Add-NaCredential –Controller dots1 –Credential $FasCred
* Register $FasCred for dots1 as PSCredential object.

PS C:\> Add-NaCredential –Controller dots2
* Register $FasCred for dots2 as a PSCredential object.

PS C:\> Connect-NaController dots1

Name          Address          Ontapi   Version
----          -------          ------   -------
dots1         192.168.151.101  1.14     NetApp Release 7.3.5: Mon Nov 22 08:37 PM:51 PST 2010


       <Execute in a different PowerShell environment by the same user on the same machine>


Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\> Connect-NaController dots2

Name          Address          Ontapi   Version
----          -------          ------   -------
dots1         192.168.151.101  1.14     NetApp Release 7.3.5: Mon Nov 22 08:37 PM:51 PST 2010
```

6 Data ONTAP Operations

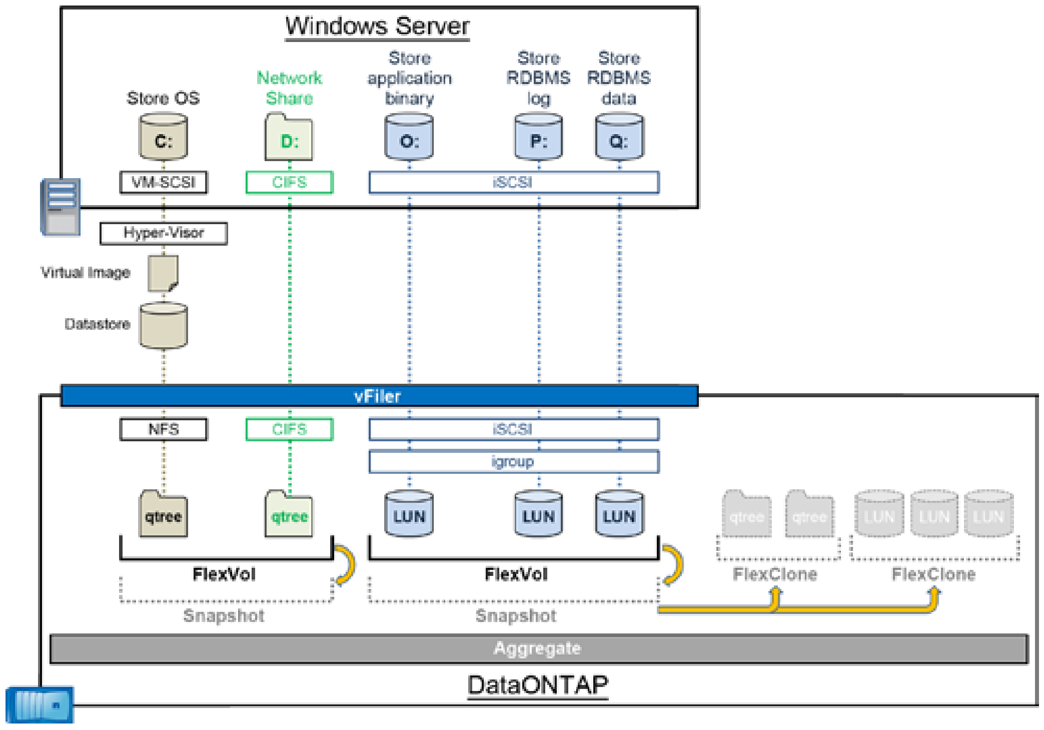
Now you have connected to Data ONTAP. Let us proceed to various Data ONTAP operations.

In this report we assume a Windows Server environment on Hyper-Visor shown below and perform a series of operations using PSTK, such as creating a volume or LUN, taking Snapshots, and then creating FlexClone volumes using the Snapshots.

Although we use a simulator for Data ONTAP, management tasks can be performed in exactly the same way in a real environment.

Note that the operations on Windows Server shown in the figure below (such as connecting to the network drive or formatting the file system) are not covered by this document.

Figure 7) Operation environment in this report



This report describes PSTK-based management tasks listed in the table below for the above configuration.

Table 14) Management tasks using PowerShell Toolkit described in this report

Section #	Description
6.1	Retrieving disk information and creating aggregate
6.2	Creating FlexVol volume
6.3	Creating vFiler, enabling protocol, and mapping FlexVol volume
6.4	Creating qtree and configuring Quota
6.5	Configuring NAS settings (NFS and CIFS)
6.6	Configuring SAN settings (creating igroup and LUN, mapping, connecting from host)
6.7	Creating and deleting Snapshot
6.8	Creating and deleting FlexClone (making off-line and deleting LUN and FlexVol volume)

6.1 Retrieving Disk Information and Creating Aggregate

First you create a new aggregate. Before that, you need to check the current information. Check the current state of existing aggregates and disks.

To get target objects or their information in PSTK, use "`Get-Na<object_name>`" for most cases.

PS command 22) Retrieve aggregate information

```
PS C:\> Get-NaAggr

Name  State  TotalSize  Used  Available  Disks RaidType RaidSize MirrorStatus
----  -----  ---------  ----  ---------  ----- -------- -------- ------------
aggr0 online 342.0 MB   47%   180.0 MB   4     raid0    8        unmirrored
```

PS command 23) Retrieve drive information

```
PS C:\> Get-NaDisk

Name   Shelf Bay Status UsedSpace PhysSpace RPM   FWModel   Pool Aggregate
----   ----- --- ------ --------- --------- ---   -------   ---- ---------
v4.16    1    0  data     121 MB    128 MB 0042  VD-100MB    0    aggr0
v4.17    1    1  data     121 MB    128 MB 0042  VD-100MB    0    aggr0
v4.18    1    2  data     121 MB    128 MB 0042  VD-100MB    0    aggr0
v4.19    1    3  data     121 MB    128 MB 0042  VD-100MB    0    aggr0
v4.20    1    4  spare    121 MB    128 MB 0042  VD-100MB    0
v4.21    1    5  spare    121 MB    128 MB 0042  VD-100MB    0
v4.22    1    6  spare    121 MB    128 MB 0042  VD-100MB    0
v4.24    1    7  spare    121 MB    128 MB 0042  VD-100MB    0
  <snip>
v4.41    2    9  spare    121 MB    128 MB 0042  VD-100MB    0
v4.42    2   10  spare    121 MB    128 MB 0042  VD-100MB    0

* Check that there are sufficient spare disks.
```

PowerShell allows you to specify conditions to filter the objects.

In this example, we use `Where-Object` to display only spare disks from the `Get-NaDisk` results.

PS command 24) Display spare disks only

```
PS C:\> Get-NaDisk | Where-Object {$_.Status -eq "spare"}

Name   Shelf Bay Status UsedSpace PhysSpace RPM  FW Model Pool Aggregate
----   ----- --- ------ --------- --------- ---  -- ----- ---- ---------
v4.25     1   4  spare     121 MB    128 MB 0042 VD-100MB   0
v4.26     1   5  spare     121 MB    128 MB 0042 VD-100MB   0
v4.27     1   6  spare     121 MB    128 MB 0042 VD-100MB   0
v4.28     1   7  spare     121 MB    128 MB 0042 VD-100MB   0
  <snip>
v4.40     2   8  spare     121 MB    128 MB 0042 VD-100MB   0
v4.41     2   9  spare     121 MB    128 MB 0042 VD-100MB   0
v4.42     2  10  spare     121 MB    128 MB 0042 VD-100MB   0
```

PS command 25) Creating aggregate

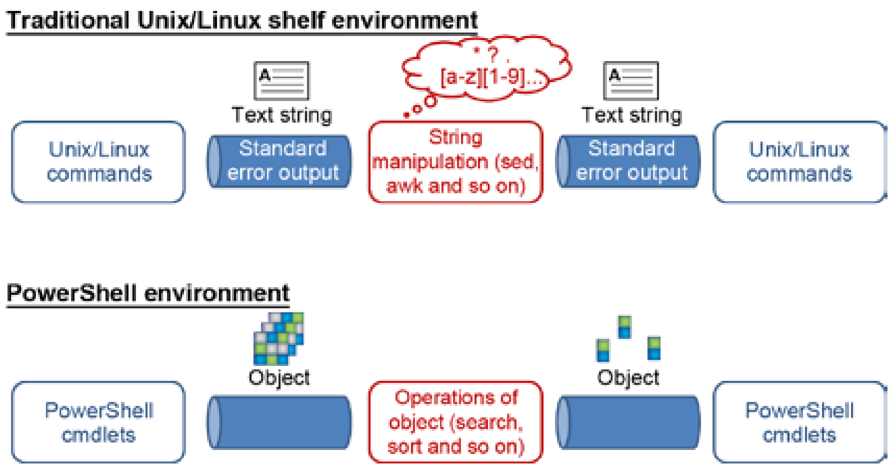
```
PS C:\> New-NaAggr aggr1 -DiskCount 8 -RaidSize 12 -RaidType raid_dp

Name  State TotalSize Used Available Disks RaidType RaidSize MirrorStatus FilesUsed FilesTotal
----  ----- --------- ---- --------- ----- -------- -------- ------------ --------- ---------
aggr1 online 513.0 MB   0%  512.9 MB   8   raid_dp     12     unmirrored      100        18k
```

(Tea Break) Pipe Lines and Objects in PowerShell

In PowerShell, like in other UNIX shells, you can connect commands using a pipe ("|").

A big difference with conventional UNIX shells is that the result of each command is output not simply as text but as an object. When creating a shell script with conventional UNIX shells, you need to resort to complex string manipulations when, for example, passing an output from one command as an attribute of another command as the output is a text string. In PowerShell, objects allow you to retrieve a specific value in an object or find only objects with a certain key value, using a single command, which significantly reduces the required time for creating scripts.

Figure 8) Pipeline expression in PowerShell



This example shows three typical cmdlets for object control.

Filter using `Where (Where-Object)` cmdlet

The `Where` cmdlet narrows down the retrieved object data according to the search conditions. In PS command 23, a list of mounted hard disk drives is retrieved using `Get-NaDisk` and narrowed down based on objects "`$_`" passed from `Get-NaDisk` with the "`Status`" property of (`-eq`) "`spare`".

This section describes some of the popular methods (logical operators) used to specify search conditions when creating a script using PSTK.

Table 15) List of operators commonly used in PSTK

Operator	Meaning	Example
`-and`	Logical AND	`{($_.Name -match "v4") -and ($_.Status -match "spare")}`
`-or`	Logical OR	`{($_.Name -match "v4") -or ($_.Status -match "spare")}`
`-not`	Logical NOT	`{$_.Bay -not 4}`
`-eq`	Equal	`{$_.Status -eq "spare"}`
`-ne`	Not Equal	`{$_.Status -ne "spare"}`
`-match`	Regex Match	`{$_.Name -match "v4"}`
`-notmatch`	Regex NotMatch	`{$_.Name -notmatch "v4"}`

Selecting properties to display using `Select (Select-Object)` cmdlet

The `Select` cmdlet selects specific properties to display for retrieved objects.

The following example shows how to display only HDD number (`Name`), status (`Status`), and physical space (`PhysSpace`) for objects retrieved by `Get-NaDisk`.

PS command 26) Use example of `Select-Object` cmdlet

```
PS C:\> Get-NaDisk | Select-Object Name, Status, PhysSpace

Name           Status                 PhysSpace
----           ------                 ---------
v4.25          spare                    128 MB
v4.26          spare                    128 MB
v4.27          spare                    128 MB
  <snip>
```

Sorting objects by specific property using `Sort(Sort-Object)` cmdlet

The `Sort` cmdlet sorts the retrieved objects by a specific property in ascending or descending order.

The following example shows how to output a list of FlexVol volumes obtained by `Get-NaVol` in descending order of available space (`Available`). The `Sort` cmdlet outputs the result in descending order by default, but you can change to ascending order by adding `-Descending` argument.

PS command 27) Example of `Sort-Object` cmdlet

```
PS C:\> Get-NaVol | Sort-Object Available -Descending

Name       State  TotalSize Used  Available Dedupe  FilesUsed FilesTotal Aggregate
----       -----  --------- ----  --------- ------  --------- ---------- ---------
vol_kelvin online    1.0 TB  15%   873.3 GB  True          9k        32M aggr0
vol_nancy  online 700.0 GB    5%   666.4 GB  True         185        22M aggr0
agate_PoC  online 700.0 GB    6%   658.9 GB  True         212        22M aggr0
vol_vf1    online 500.0 GB   43%   285.2 GB  True         316        25M aggr0
vol_suzuki online 300.0 GB   11%   266.4 GB False         115         9M aggr0
```

6.2 Creating FlexVol volumes

Next, create FlexVol volumes to store data for NAS and SAN.

If you forget arguments required for a cmdlet, run the cmdlet with the "`-?`" argument to display help for the cmdlet.

PS command 28) Create FlexVol volumes using `New-NaVol` cmdlet

```
PS C:\> New-NaVol vol_nas -Aggregate aggr1 -LanguageCode C -SpaceReserve none -Size 200MB

Name    State   TotalSize Used Available Dedupe FilesUsed FilesTotal Aggregate
----    -----   --------- ---- --------- ------ --------- ---------- ---------
vol_nas online   160.0 MB   0%  159.9 MB  False    100         6k       aggr1


PS C:\> New-NaVol vol_san -Aggregate aggr1 -LanguageCode C -SpaceReserve none -Size 200MB

Name    State   TotalSize Used Available Dedupe FilesUsed FilesTotal Aggregate
----    -----   --------- ---- --------- ------ --------- ---------- ---------
vol_san online   160.0 MB   0%  159.9 MB  False    100         6k       aggr1
```

Although FlexVol volumes have been created, the total capacity (`TotalSize`) is small because the Snapshot Reserve space is set to 20% by default.

Use the `Set-NaSnapshotReserve` cmdlet that configures the Snapshot Reserve space on a FlexVol volume to change it to 0%.

PS command 29) Configure Snapshot Reserve space using `Set-NaSnapshotReserve` cmdlet

```
PS C:\> Set-NaSnapshotReserve vol_nas -Percentage 0

Name    State  TotalSize Used Available Dedupe FilesUsed FilesTotal Aggregate
----    -----  --------- ---- --------- ------ --------- ---------- ---------
vol_nas online  200.0 MB   0%  199.9 MB  False    100         6k        aggr1


PS C:\> Set-NaSnapshotReserve vol_san -Percentage 0

Name    State  TotalSize Used Available Dedupe FilesUsed FilesTotal Aggregate
----    -----  --------- ---- --------- ------ --------- ---------- ---------
vol_san online  200.0 MB   0%  199.9 MB  False    100         6k        aggr1
```

In this example, the cmdlet is executed twice. Alternatively, you can use pipeline processing to execute the command only once as shown below.

PS command 30) Example of batch processing to configure Snapshot Reserve space

```
PS C:\> Get-NaVol |
>>        Where-Object {$_.Name -ne “vol0”} |
>>        Set-NaSnapshotReserve -Percentage 0 -Confirm:$false
```

FlexVol volume has various attributes and option values due to its multifunctionality. If you are accustomed to the operations of NetApp storage, the fact that there are only a few display items (properties) may seem odd.

Try executing the following command.

PS command 31) Output of `Get-NaVol` cmdlet results in list format

```
PS C:\> Get-NaVol vol_nas | Format-List
```

(Tea Break) Display format of output result

You must have understood by now that the execution result can be handled as an object in PowerShell.

The handling of the result as an object means that many attributes (properties) can be stored in a single object. In some cases, you may want to check the details of attribute values (properties) stored in each object instead of a familiar tabular format list.

In such a case, use one of the three formatting cmdlets (`Format-Table`, `Format-List`, and `Format-Wide`), of which `Format-List` is most common. This section describes how to use the `Format-Table` and `Format-List` cmdlets, which are frequently used.

Using `Format-Table` to display in tabular format

Most of the cmdlets in PowerShell display the execution result in tabular format by default. The tabular format displays a list of properties for each object. The functions required to facilitate the comparison of object information by an administrator are the function to easily identify items (properties) (`Select-Object`), the function to perform conditional search and sorting (`Where-Object` and `Sort-Object`), and the function to display information in a format that is easy to understand at a glance.

The `Format-Table` cmdlet is not only configured by default but also outputs information in a format easy for you to see if you specify a format in an argument. Look at the execution result of the following cmdlet:

PS command 32) Output result of `Format-Table` cmdlet according to format

```
PS C:\> $fields = "Name",@{Label = "KB"; Expression = {$_.TotalSize / 1KB}; Align = "Right"}
PS C:\> Get-NaVol | Format-Table $fields -AutoSize

Name                       KB
----                       --
ds_kelvin_san        10810064
ds_katada           524288000
ds_san_mgmt         734003200
ds_yamamoto_demo1 104857600
ds_yamamoto_demo2 104857600
ds_yamamoto_demo3 104857600
vol_fs_template     314572800
vol_reiko_vm_ds     199229440
```

In the example of PS command 32, the formatting information is registered in the `$fields` variable. The properties to be displayed are specified with commas (,) between them and, in the `@{ }` section, the TotalSize property is specified so that it is displayed with a label name of "KB" (`Label = "KB"`). Since the TotalSize property value is defined as a Byte value, it is divided by 1024=1KB (`Expression = {$_.TotalSize / 1KB}`), and the notation in the column is specified as right-aligned (`Align = "Right"`). While these object values to be displayed in tabular format can be easily identified by `Select-Object`, a combined use of the `Format-Table` cmdlet and formatting enables further customized output display.

Use of `Format-List` to display in list format

Each item in a NetApp storage system is handled as an object in PSTK. See the following cmdlet execution result to check what kinds of properties and methods are stored in objects.

PS command 33) Check object contents using `Get-Member` cmdlet

```
PS C:\> Get-NaVol | Get-Member

   TypeName: DataONTAP.Types.Volume.VolumeInfo

Name                          MemberType    Definition
----                          ----------    ----------
Aggregate                     AliasProperty Aggregate = ContainingAggregate
Available                     AliasProperty Available = SizeAvailable
```

```
Dedupe                        AliasProperty  Dedupe = DedupeEnabled
   <snip>
Equals                        Method         bool Equals(System.Object obj)
GetHashCode                   Method         int GetHashCode()
GetType                       Method         type GetType()
BlockType                     Property       System.String BlockType {get;set;}
ChecksumStyle                 Property       System.String ChecksumStyle {get;set;}
   <snip>
CloneParent                   Property       DataONTAP.Types.Volume.CloneParentInfo[]
ContainingAggregate           Property       System.String ContainingAggregate {get;set;}
DiskCount                     Property       System.Nullable`1[[System.Int32, mscorlib,
   <snip>
```

As shown in the example of PS command 33, many properties and methods are included.

For each cmdlet, the items frequently used in general are output in tabular format by `Format-Table` by default. However, you may want to display the values of these items in a list.

In such a case, use the `Format-List` cmdlet.

Look at the execution result of the following cmdlet:

PS command 34) Output FlexVol volume information using `Format-List` cmdlet

```
PS C:\> Get-NaVol vol0 | Format-List

Autosize                      : DataONTAP.Types.Volume.Autosize
BlockType                     : 64_bit
ChecksumStyle                 : block
CloneChildren                 :
CloneParent                   :
ContainingAggregate           : aggr0
DiskCount                     : 20
ExpiryDate                    :
   <snip>
```

6.3 Creating vFiler, Enabling Protocols, and Mapping FlexVol volumes

※ This section describes how to make a vFiler using PSTK for your reference.

※ A vFiler is created and a protocol is enabled according to the initial settings of the storage system.

※ Therefore, we recommend that you create a vFiler with the CLI such as the console or using OnCommand SystemManager, OnCommand Unified Manager, or the vFiler template function, etc.

In this example, a vFiler is created as a storage device to be accessed from the host.

To create a vFiler, you need to specify the FlexVol volume (vFiler vol0) that will store the vFiler configuration information and the network settings (IP Space and IP address). In this report, the following settings are used.

Table 16) vFiler configuration information in this report

Item		Value
vFiler basic configuration information	vFiler Name	vfiler1
	vFiler vol0	vfiler1_vol0
	IP Space	Default IP Space (default-ipspace)
	IP Address	192.168.151.103
CIFS configuration information	CIFS Server	VFILER1
	Active Directory Domain	spice.net
	Active Directory DC Address	192.168.151.200
	Domain Admin Username	spice\administrator
	Domain Admin Password	password

Preparation for creating vFiler

To prepare for creating a vFiler, create vFiler vol0 and check the information of default-ipspace.

PS command 35) Create vFiler vol0 and check Default IP Space

```
PS C:\> New-NaVol vfiler_vol0 -Aggregate aggr1 -LanguageCode C -SpaceReserve none -Size 200MB

Name          State   TotalSize Used Available Dedupe FilesUsed FilesTotal Aggregate
----          -----   --------- ---- --------- ------ --------- ---------- ---------
vfiler_vol0   online   160.0 MB   0%   159.9 MB  False    100         6k       aggr1


PS C:\> Get-NaNetIpspace

Ipspace                    Interfaces
-------                    ----------
default-ipspace            {e0a, e0b, e0c, e0d...}
```

Cmdlets related to IP Space are classified into the Net category. `Get / New / Remove / Set-NaNetIpspace` cmdlet)

Creating vFiler

Now create a vFiler.

PS command 36) Create vFiler using `New-NaVfiler` cmdlet

```
PS C:\> New-NaVfiler -Name vfiler1 -Addresses 192.168.151.103 -Ipspace -Storage kam_test

Name      Status     Ipspace          VfnetCount VfstoreCount AdminHost
----      ------     -------          ---------- ------------ ---------
vfiler1   running    default-ipspace           1            1
```

In a vFiler, you can configure a DNS or NIS or configure the Administration Host using cmdlets such as `Set-NaVfilerDns`.

Configuring vFiler

The next step is to connect PSTK to the vFiler and execute various cmdlets. For this purpose, you need to configure the created vFiler to enable the connection of PSTK.

Perform this step not from PSTK but from the system console.

Task	Method	vFiler	Required?
Configuration of vFiler administrator password	`passwd` command	vfiler1	Yes
Permission for administrative access via HTTP	`options httpd.admin.enable on`	vfiler1	Yes
Permission for creation of FlexClone volume via vFiler	`options vfiler.vol_clone_zapi_allow on`	vfiler0	Optional

PS command 37) Configure vFiler after it is created

```
dots> vfiler context vfiler1

vfiler1@dots> passwd
New password: netapp123
Retype new password: netapp123
Mon Jun 13 10:00 AM:00 JST [dots:passwd.changed:info]: passwd for user 'root' changed.

vfiler1@dots> options httpd.admin.enable on

vfiler1@dots> vfiler context vfiler0

dots> options vfiler.vol_clone_zapi_allow on
```

Enabling protocols

For the created vFiler, enable the NFS, CIFS, and iSCSI protocols. First, check the current protocol configurations.

Table 17) Target vFilers for enabling protocols for vFilers

Task	PSTK cmdlet	vFiler
Check the enabled protocols	`Get-NaVfilerProtocol`	vfiler0
Enable protocols	`Set-NaCifs`, etc.	vfiler1

PS command 38) Check the enabled protocol for vFiler using `Get-NaVfilerProtocol` cmdlet

```
PS C:\> Get-NaVfilerProtocol -Name vfiler1
AllowedProtocols                     DisallowedProtocols
----------------                     -------------------
{nfs, cifs, rsh, iscsi...}           {}
```

By default, the NFS protocol is automatically enabled for a vFiler after it is created.

PS command 39) Check if NFS is enabled using `Test-NaNfs` cmdlet

```
PS C:\> Test-NaNfs

   IsEnabled
   ---------
        True
```

PS command 40) Enable iSCSI protocol using `Enable-NaIscsi` cmdlet

```
PS C:\> Test-NaIscsi
False

PS C:\> Enable-NaIscsi

PS C:\> Test-NaIscsi
True
```

PS command 41) Perform initial setup of CIFS protocol (cifs setup) using `Set-NaCifs` cmdlet

```
PS C:\> Test-NaCifs
stopped

PS C:\> Set-NaCifs -CifsServer vfiler1 -AuthType AD -SecurityStyle multiprotocol
-Domain spice.net -DCAddress 192.168.151.200 -User administrator -Password password
* The -User and -Password arguments specify the administrator user and password for the AD domain
controller.

PS C:\> Test-NaCifs
started
```

Mapping FlexVol volume

Next, map the FlexVol volume created in Step 2, which will store NAS and SAN data, on the vFiler. This operation assigns the FlexVol volume owned by parent, vfiler0, to its child, vfiler1. Therefore, vfiler0 is the target of operation.

Table 18) List of operations on FlexVol volume in relation to vFiler

Task	PSTK cmdlet	vFiler
Creation of volume	`New-NaVol`	viler0
Mapping of volume	`Set-NaVfilerStorage -AddStorage`	vfiler0
Checking the volume status	`Get-NaVol`	vfiler0, vfiler1
Offlining of volume	`Set-NaVol -Offline`	vfiler0, vfiler1
Unmapping of volume	`Set-NaVfilerStorage -RemoveStorage`	vfiler0
Deletion of volume	`Remove-NaVol`	vfiler0

PS command 42) Map FlexVol volume to vFiler using `Set-NaVfilerStorage` cmtlet

```
PS C:\> Set-NaVfilerStorage -Name vfiler1 -AddStorage vol_nas
Name       Status         Ipspace              VfnetCount VfstoreCount AdminHost
----       ------         -------              ---------- ------------ ---------
vfiler1    running        default-ipspace               1            2

PS C:\> Set-NaVfilerStorage -Name vfiler1 -AddStorage vol_san
Name       Status         Ipspace              VfnetCount VfstoreCount AdminHost
----       ------         -------              ---------- ------------ ---------
vfiler1    running        default-ipspace               1            3
```

6.4 Creating qtree and Configuring Quota

Connect to a vFiler to be used, create a qtree, and then configure the capacity limit on the created qtree using the Quota function. Perform this task, being the configuration in a vFiler, after connecting to vfiler1.

PS command 43) Create qtree using `New-NaQtree` cmdlet

```
PS C:\> New-NaQtree -Path /vol/vol_nas/qt_nfs

Volume      Qtree      Status     Security     OpLocks
------      -----      ------     --------     -------
vol_nas     qt_nfs     normal     unix         enabled

PS C:\> New-NaQtree -Path /vol/vol_nas/qt_cifs

Volume      Qtree      Status     Security     OpLocks
------      -----      ------     --------     -------
vol_nas     qt_cifs     normal     unix         enabled
```

PS command 44) Configure Quota using `Set-NaQuota` cmdlet

```
PS C:\> Set-NaQuota -Type tree -Target /vol/vol_nas/qt_nfs -Volume vol_nas
-SoftDiskLimit 150m -DiskLimit 180m

Type                 : tree
Target               : {/vol/vol_nas/qt_nfs}
Volume               : vol_nas
Qtree                :
DiskLimit            : 188743680
FileLimit            : -
PerformUserMapping :
QuotaError           :
SoftDiskLimit        : 157286400
SoftFileLimit        : -
Threshold            : -

PS C:\> Enable-NaVolQuota –Volume

Name       State    TotalSize  Used  Available Dedupe  FilesUsed FilesTotal Aggregate
----       -----    ---------  ----  --------- ------  --------- ---------- ---------
vol_nas    online     196.4 MB   0%     196.4B False         151         2M aggr0

PS C:\> Get-NaQuotaReport

Volume     Qtree     QuotaType    DiskLimit    DiskUsed  FileLimit  FilesUsed
------     -----     ---------    ---------    --------  ---------  ---------
vol_nas    qt_nfs    tree            180 MB           0                     1
```

After using the `Set-NaQuota` cmdlet to configure Quota, use the `Disable-NaVolQuota` or `Enable-NaVolQuota` cmdlet to enable or disable the Quota function to reflect the setting.

6.5 Configuring NAS Settings (NFS and CIFS)

Share the created qtree between NFS and CIFS. Since the NFS and CIFS protocols have been enabled and configured, you need to add the share setting of the relevant qtree to the existing settings.

PS command 45) Configure NFS share using `Add-NaNfsExport` cmdlet

```
PS C:\> Add-NaNfsExport -Path /qt_nfs -ActualPath /vol/vol_nas/qt_nfs
-Anon 0 -ReadWrite all-hosts -NoSuid -SecurityFlavors sys;

LoadedPathnames            ExportedPathnames
---------------            -----------------
{/qt_nfs}
```

```
PS C:\> Get-NaNfsExport -Path /qt_nfs

ActualPathname       Pathname   SecurityRules
--------------       --------   -------------
/vol/vol_nas/qt_nfs  /qt_nfs    {True}

PS C:\> Get-NaNfsExport -Path /qt_nfs | Select-Object -ExpandProperty SecurityRules

Anon             : 0
Nosuid           : True
ReadOnly         :
ReadWrite        : {all-hosts}
Root             :
SecFlavor        : {sys}
NosuidSpecified : True
```

PS command 46) Configure CIFS share using `Add-NaCifsShare` cmdlet

```
PS C:\> Add-NaCifsShare -Share qt_cifs -Path /vol/vol_nas/qt_cifs -Comment "vfiler1 share"

MountPoint                ShareName     Description
----------                ---------     -----------
/vol/vol_nas/qt_cifs      qt_cifs       vfiler1 share

PS C:\> Get-NaCifsShare

MountPoint                  ShareName      Description
----------                  ---------      -----------
/vol/vfiler_vol0/etc        ETC$           Remote Administration
/vol/vfiler_vol0/home       HOME           Default Share
/                           C$             Remote Administration
/vol/vol_nas/qt_cifs        qt_cifs        vfiler1 share
```

6.6 Configuring SAN Settings (Creating igroup and LUN, Mapping, Connecting from Host)

※ This section describes how to make an igroup and a LUN using PSTK for your reference.

※ NetApp provides a SnapDrive utility to allow host administrators to create a LUN for themselves and create a backup (data still points) using Snapshot when required. Furthermore, we recommend that you use the SnapDrive utility to utilize NetApp storage in the SAN environment.

※ You can perform all the tasks related to an igroup and a LUN using SnapDrive. Commands such as sdcli and snapdrive are available for programming too. The use of these commands facilitates scripting.

Next, create a LUN to be connected via iSCSI, and create an igroup and map the LAN to accept connection from the host. This report describes an example of connecting from a Windows system.

Creating LUN

PS command 47) Create LUN using `New-NaLun` command

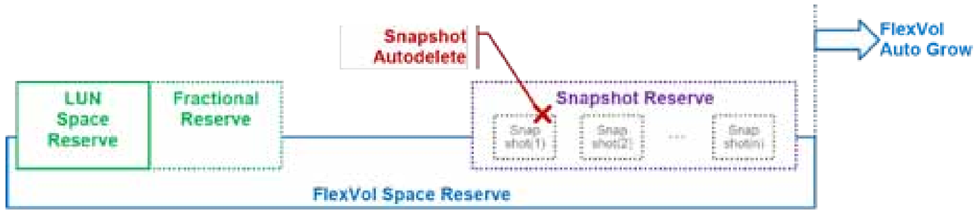
```
PS C:\> New-NaLun -Path /vol/vol_san/app_lun -Size 50m -Type windows -Unreserved

Path                    TotalSize   SizeUsed Protocol     Online Mapped  Thin  Comment
----                    ---------   -------- --------     ------ ------  ----  -------
/vol/vol_san/app_lun    50 MB              0 windows       True  False   True
```

While creating a LUN in a NetApp storage system, the configuration of the thin provisioning function is one of the important points in storage design. The following table briefly shows the configuration methods for the thin provisioning function (Space Guarantee) using PSTK.

Table 19) List of configuration methods related to thin provisioning in PSTK

Layer	Content	PSTK cmdlet	Value
LUN	Space Reserve	`New-NaLun`	`-Unreserved`: No Guarantee
		`Set-NaLunSpaceReserved`	No argument: With Guarantee `-Off`: No Guarantee
(Fractional)	Fractional Reserve	`Set-NaVolOption`	`fractional_reserve`: Specify a number (%)
FlexVol	Space Reserve	`New-NaVol`	`-SpaceReserve`: none, file, volume
		`Set-NaVolOption`	`SpaceReserve`: none, file, volume `SpaceReserveEnabled`: True / False
	Auto Grow	`Set-NaVolAutosize`	See each item value
(Snapshot)	Snapshot Reserve	`Set-NaSnapshotReserve`	`-Percentage`: Specify a number (%)
	Snapshot Autodelete	`Set-NaSnapshotAutodelete`	See each item value

Figure 9) Conceptual diagram of capacity guarantee setting related to thin provisioning function



Creating igroup and Registering iSCSI Initiator

Next, create an igroup and register an iSCSI initiator.

PS command 48) Create igroup using `New-NaIgroup` cmdlet

```
PS C:\> New-NaIgroup -Name winhost_ig -Protocol iscsi -Type windows

Name             : winhost_ig
Type             : windows
Protocol         : iscsi
PortSet          :
ALUA             : False
ThrottleBorrow   : False
ThrottleReserve  : 0
Partner          :
VSA              : False
Initiators       : {}
```

When configuring an igroup, register the IQN information of iSCSI initiator to identify an attached host.

Generally, IQN information is retrieved using a method for each supported environment during the system configuration. However, the following example shows, for your reference, how to use a cmdlet in the Host category of PSTK to get the iSCSI initiator IQN information on a Windows system.

PS command 49) Get host iSCSI initiator information using `Get-NaHostIscsiAdapter` cmdlet

```
PS C:\ > Get-NaHostIscsiAdapter | Format-List

InstanceName    : Root\ISCSIPRT\0000_0
Vendor          : Microsoft Corporation
VendorModel     : iSCSI Initiator
SerialNumber    : MSFT-05-1991
DriverName      : msiscsi.sys
FirmwareVersion : 1.5
Iqn             : iqn.1991-05.com.microsoft:winhost.spice.net
WmiPath         :
\\WINHOST\root\WMI:MSiSCSI_HBAInformation.InstanceName="Root\\ISCSIPRT\\0000_0"
```

PS command 50) Register iSCSI initiator to igroup using `Add-NaIgroupInitiator` cmdlet

```
PS C:\> Add-NaIgroupInitiator -Igroup winhost_ig
-Initiator iqn.1991-05.com.microsoft:winhost.spice.net

Name            : winhost_ig
Type            : windows
Protocol        : iscsi
PortSet         :
ALUA            : False
ThrottleBorrow  : False
ThrottleReserve : 0
Partner         :
VSA             : False
Initiators      : {iqn.1991-05.com.microsoft:winhost.spice.net}
```

Mapping LUN to igroup

Map the created LUN to an igroup so that it accepts connections from the host.

PS command 51) Map LUN to igroup using `Add-NaLunMap` cmdlet

```
PS C:\> Get-NaLun

Path                   TotalSize   SizeUsed Protocol     Online Mapped  Thin   Comment
----                   ---------   -------- --------     ------ ------  ----   -------
/vol/vol_san/app_lun       50 MB          0 windows        True  False  False


PS C:\> Add-NaLunMap -Path /vol/vol_san/app_lun -InitiatorGroup winhost_ig -ID 1

Path                   TotalSize   SizeUsed Protocol     Online Mapped  Thin   Comment
----                   ---------   -------- --------     ------ ------  ----   -------
/vol/vol_san/app_lun       50 MB          0 windows        True  True   False
```

PS command 52) Check mapping information using `Get-NaLunMap` cmdlet, etc.

```
PS C:\> Get-NaLunMap /vol/vol_san/app_lun

Name            : winhost_ig
Type            : windows
Protocol        : iscsi
PortSet         :
ALUA            : False
ThrottleBorrow  : False
ThrottleReserve : 0
Partner         :
```

```
VSA              : False
Initiators       : {iqn.1991-05.com.microsoft:winhost.spice.net}

PS C:\> Get-NaLunMapByInitiator -Initiator iqn.1991-05.com.microsoft:winhost.spice.net

InitiatorGroup        LunId Path
--------------        ----- ----
winhost_ig                1 /vol/vol_san/app_lun
```

6.7 Creating and Deleting Snapshot

The configuration that you have completed so far has enabled connection from the host and storage of user data.

In this section, try creating and deleting a Snapshot, which is the basis for the backup of stored data.

PS command 53) Create Snapshot using `New-NaSnapshot` cmdlet

```
PS C:\> New-NaSnapshot -TargetName vol_nas -SnapName Freeze_AUG2012

Name                 Created      Total Cumulative Dependency
----                 -------      ----- ---------- ----------
Freeze_AUG2012       8/27/12

PS C:\> Get-NaSnapshot -TargetName vol_nas

Name                     Created      Total Cumulative Dependency
----                     -------      ----- ---------- ----------
Freeze_AUG2012       8/27/12     80.0 KB    80.0 KB
hourly.0             8/27/12    308.0 KB   388.0 KB
hourly.1             8/27/12    340.0 KB   728.0 KB
hourly.2             8/27/12    340.0 KB     1.0 MB
hourly.3             8/26/12    268.0 KB     1.3 MB
```

Use the following cmdlets to get Snapshots automatically using the scheduler function in the NetApp storage.

PS command 54) Configure Snapshot schedule using `Set-NaSnapshotSchedule` cmdlet

```
PS C:\> Set-NaSnapshotSchedule vol_nas -Weeks 2 -Days 2 -Hours 2 -WhichHours "0,12"

Name      State       TotalSize  Used  Available Dedupe  FilesUsed FilesTotal Aggregate
----      -----       ---------  ----  --------- ------  --------- ---------- ---------
vol_nas   online        200 MB    4%    174.6 MB False         158         2M aggr1


PS C:\> Get-NaSnapshotSchedule -TargetName vol_nas

Volume     Weeks   Days   Hours   Minutes   WhichHours   WhichMinutes
------     -----   ----   -----   -------   ----------   ------------
vol_nas      2       2      2        0      0,12
```

PS command 55) Delete Snapshot using `Remove-NaSnapshot` cmdlet

```
PS C:\> Remove-NaSnapshot -TargetName vol_nas -SnapName hourly.3

Delete snapshot
Are you sure you want to delete snapshot hourly.3 from volume vol_nas?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): y
```

When the capacity runs short or you re-prepare the environment between different projects, you sometimes need to bulk-delete multiple Snapshots. In such a case, use the pipe function of PowerShell to bulk-delete multiple Snapshots as shown below.

PS command 56) Bulk-delete Snapshots using pipe

```
PS C:\> Get-NaSnapshot vol_nas | Select-Object Name

Name
----
Stable_Freeze
develop_milestone_003
develop_milestone_003
develop_milestone_003
develop_milestone_003
develop_milestone_003
daily.0
daily.1
daily.2
daily.3
training_backup_001
training_backup_002
training_backup_003


PS C:\> Get-NaSnapshot vol_nas | Where-Object {$_.Name -notmatch "Freeze"}
* Get a Snapshot other than "Stable_Freeze"

Name
----
develop_milestone_003
develop_milestone_003
develop_milestone_003
develop_milestone_003
develop_milestone_003
daily.0
daily.1
daily.2
daily.3
training_backup_001
training_backup_002
training_backup_003


PS C:\> Get-NaSnapshot vol_nas | Where-Object {$_.Name -notmatch "Freeze"} |
Remove-NaSnapshot -Confirm:$False
* Delete Snapshots other than "Stable_Freeze"

PS C:\> Get-NaSnapshot vol_nas | Select-Object Name

Name
----
Stable_Freeze
```

While creating a script file using PSTK, you sometimes need to select the latest of multiple Snapshots and use it as the starting point for FlexClone, etc.

In such a case, too, use the pipeline function described above and the `Sort-Object` cmdlet to easily get the Snapshot.

PS command 57) Get latest Snapshot using `Sort-Object` cmdlet

```
PS C:\> $LatestSnapshot = Get-NaSnapshot vol_nas | Sort-Object Created -Descending
* Use the Sort-Object cmdlet to output objects in descending order at the creation time "Created"
* Register the objects output in descending order in the $LatestSnapshot variable as an object
array

PS C:\> $LatestSnapshot | Select-Object Name, Created

Name                                              Created
----                                              -------
Freezing0002                                      8/26/12 11:28 AM:34
daily.0                                           8/25/12 03:00 AM:11
daily.1                                           8/24/12 03:00 AM:12
daily.2                                           8/23/12 03:00 AM:10

PS C:\> $LatestSnapshot[0]
* Get the first item in the $LatestSnapshot variable array

Name                                                 Created      Total Cumulative Dependency
----                                                 -------      ----- ---------- ----------
Freezing0002                                      8/26/12    216.0 KB   216.0 KB
```

6.8 Creating and Deleting FlexClone (Making Off-line and Deleting LUN and FlexVol volume)

One of the big advantages of NetApp storage is that a replica volume can be created instantaneously from a Snapshot, a static image of data, regardless of the capacity. At the end of this report, there is a description of the FlexClone function, a replication function for this virtual data volume.

Creating FlexClone

PS command 58) Create FlexClone using `New-NaVolClone` cmdlet (without Snapshot specified)

```
PS C:\> New-NaVolClone -CloneVolume vol_san_clone -ParentVolume vol_san -SpaceReserve none

Name             State       TotalSize  Used  Available Dedupe  FilesUsed FilesTotal Aggregate
----             -----       ---------  ----  --------- ------  --------- ---------- ---------
vol_san_clone    online         200 MB    4%     198 MB False         158         2M aggr1

PS C:\> Get-NaSnapshot vol_san

Name                     Created      Total Cumulative Dependency
----                     -------      ----- ---------- ----------
clone_vol_san_clone.1    8/27/12   164.0 KB   164.0 KB busy,vclone
Stable_Freeze            8/23/12   292.0 KB   456.0 KB
hourly.0                 8/27/12   308.0 KB   764.0 KB
hourly.1                 8/27/12   340.0 KB     1.1 MB
hourly.2                 8/27/12   340.0 KB     1.4 MB
* Without explicit Snapshot specification, Snapshots are automatically created.
```

PS command 59) Create FlexClone using `New-NaVolClone` cmdlet (with Snapshot specified)

```
PS C:\> New-NaVolClone -CloneVolume vol_san_clone2 -ParentVolume vol_san
-ParentSnapshot Stable_Freeze -SpaceReserve none

Name            State       TotalSize  Used  Available Dedupe  FilesUsed FilesTotal Aggregate
----            -----       ---------  ----  --------- ------  --------- ---------- ---------
vol_san_clone2  online         200 MB   4%      198 MB False         158         2M aggr1


PS C:\> Get-NaSnapshot vol_san

Name                     Created      Total Cumulative Dependency
----                     -------      ----- ---------- ----------
clone_vol_san_clone.1    8/27/12   164.0 KB   164.0 KB busy,vclone
Stable_Freeze            8/23/12   292.0 KB   456.0 KB busy,vclone
hourly.0                 8/27/12   308.0 KB   764.0 KB
hourly.1                 8/27/12   340.0 KB     1.1 MB
hourly.2                 8/27/12   340.0 KB     1.4 MB
* With explicit Snapshot specification, "Dependancy" is set in the relevant Snapshots.
```

PS command 60) Check FlexClone relationship using `Get-NaVol` command

```
PS C:\> Get-NaVol | Select Name, CloneChildren, CloneParent

Name             CloneChildren                         CloneParent
----             -------------                         -----------
vol_san          {vol_san_clone, vol_san_clone2}
vol_san_clone                                          {vol_san}
vol_san_clone2                                         {vol_san}
   <snip>
```

Making Off-line and Deleting LUN and FlexVol volume

Conceptually, a FlexClone volume can be handled in the same way as an ordinary FlexVol volume.

This section, using a FlexClone volume created earlier as an example, describes how to delete a LUN and a FlexVol volume.

PS command 61) Delete LUN using `Remove-NaLunMap` and `Set/Remove-NaLun` cmdlets

```
PS C:\> Get-NaLun

Path                          TotalSize   SizeUsed Protocol     Online Mapped  Thin  Comment
----                          ---------   -------- --------     ------ ------  ----  -------
/vol/vol_san/app_lun              50 MB          0 windows       True   True  False
/vol/vol_san_clone/app_lun        50 MB          0 windows       True   True  False
/vol/vol_san_clone2/app_lun       50 MB          0 windows       True   True  False


PS C:\> Get-NaLun | Remove-NaLunMap -InitiatorGroup winhost_ig -Confirm:$False
* Unmapping from igroup.

Path                          TotalSize   SizeUsed Protocol     Online Mapped  Thin  Comment
----                          ---------   -------- --------     ------ ------  ----  -------
/vol/vol_san/app_lun              50 MB          0 windows       True  False  False
/vol/vol_san_clone/app_lun        50 MB          0 windows       True  False  False
/vol/vol_san_clone2/app_lun       50 MB          0 windows       True  False  False


PS C:\> Get-NaLun | Set-NaLun -Offline -Confirm:$False
* Change the LUN status from "Online" to "Offline" and prepare for deletion.

Path                          TotalSize   SizeUsed Protocol     Online Mapped  Thin  Comment
----                          ---------   -------- --------     ------ ------  ----  -------
/vol/vol_san/app_lun              50 MB          0 windows      False  False  False
/vol/vol_san_clone/app_lun        50 MB          0 windows      False  False  False
/vol/vol_san_clone2/app_lun       50 MB          0 windows      False  False  False
```

```
PS C:\> Get-NaLun | Remove-NaLun -Confirm:$False
* Bulk-delete LUNs.

PS C:\> Get-NaLun
```

This series of processing can be bulk-processed as shown below.

PS command 62) Bulk-processing of LUN deletion using pipeline

```
PS C:\> Get-NaLun |
>> Remove-NaLunMap -InitiatorGroup winhost_ig -Confirm:$False |
>> Set-NaLun -Offline -Confirm:$False |
>> Remove-NaLUN -Confirm:$False
```

Table 20) How to bypass the confirmation of cmdlets

About -Confirm argument
Many cmdlets for correcting or deleting objects are implemented in a way that prompts users for confirmation of the operation. Although the confirmation process is important, it might get in the way of script execution. In such a case, specify `$False` in the `-Confirm` argument to bypass the interactive-type confirmation process.

Likewise, delete FlexVol volumes.

PS command 63) Use of pipeline to bulk-delete FlexVol volumes

```
PS C:\> Get-NaVol |
>> Where-Object {($_.Name -match "nas"} -or ($_.Name -match "san")) |
>> Set-NaVol -Offline -Confirm:$False |
>> Remove-NaVol -Confirm:$False
```

The cmdlet shown above uses the process flow shown below.

Table 21) Tasks required deleting FlexVol volumes

	Cmdlet	Description
1st line	`Get-NaVol`	Get all FlexVol volumes.
2nd line	`Where-Object`	Select FlexVol volumes with "nas" or "san" included in the FlexVol names.
3rd line	`Set-NaVol -Offline`	Change the Status of the target FlexVol volumes to "Offline".
4th line	`Remove-NaVol`	Delete the target FlexVol volumes.

7 Conclusion

As mentioned in the Introduction, this report has been written as the first-steps guide for storage administrators who perform day-to-day management tasks using native commands of Data ONTAP to allow them to perform the basic ONTAP operations using PowerShell.

PowerShell is a shell that supports a wide range of applications according to the current system infrastructure management needs. You can start with the operations as an interactive CLI and develop them easily to create interactive wizards (CUI) or a simple window-based GUI. Furthermore, you can perform full-fledged collaboration with orchestration tools or various office suite products.

The operations described in this report are only the first threshold to an extensive world of PowerShell operations. We hope that you use this report as the springboard to further pursue your interest in the programming of management tasks and collaboration with orchestration tools.

As a next step, try entering a cmdlet described in this report into a text editor and executing it in the PowerShell window. You will be able to see a batch script, a simple one, with the potential of applying it to your day-to-day tasks.

At first, you can start with a simple list of cmdlets, then change the included constants to variables, and replace repetitious syntax and processing with functions. As you make changes bit by bit, you will be able to experience converting complex manual processing into a full-fledged automatic processing program.

As one of the major features, the NetApp storage products that run Data ONTAP allow you to fully utilize the stored data in a various ways using virtualization technologies. We hope this report helps you to utilize your data management environments in a more flexible manner.

Go further, faster®

NetApp®
www.netapp.com