Full credit is given to the above companies including the OS that this PDF file was generated!

PowerShell Get-Help on command 'Save-Help'

PS C:\Users\wahid> Get-Help Save-Help

NAME

Save-Help

SYNOPSIS

Downloads and saves the newest help files to a file system directory.

SYNTAX

Save-Help [-DestinationPath] <System.String[]> [[-Module]

<System.Management.Automation.PSModuleInfo[]>] [[-UICulture]

<System.Globalization.CultureInfo[]>] [-Credential

<System.Management.Automation.PSCredential>] [-Force] [-FullyQualifiedModule

<Microsoft.PowerShell.Commands.ModuleSpecification[]>]

[-UseDefaultCredentials] [<CommonParameters>]

Save-Help [[-Module] <System.Management.Automation.PSModuleInfo[]>]

[[-UICulture] <System.Globalization.CultureInfo[]>] [-Credential

<System.Management.Automation.PSCredential>] [-Force] [-FullyQualifiedModule

<Microsoft.PowerShell.Commands.ModuleSpecification[]>] -LiteralPath

<System.String[]> [-UseDefaultCredentials] [<CommonParameters>]

DESCRIPTION

The `Save-Help` cmdlet downloads the newest help files for PowerShell modules and saves them to a directory that you specify. This feature lets you update the help files on computers that do not have access to the Internet, and makes it easier to update the help files on multiple computers.

In Windows PowerShell 3.0, `Save-Help` worked only for modules that are installed on the local computer. Although it was possible to import a module from a remote computer, or obtain a reference to a PSModuleInfo object from a remote computer by using PowerShell remoting, the HelpInfoUri property was not preserved, and `Save-Help` would not work for remote module Help.

In Windows PowerShell 4.0, the HelpInfoUri property is preserved over PowerShell remoting, which enables `Save-Help` to work for modules that are installed on remote computers. It is also possible to save a PSModuleInfo object to disk or removable media by running `Export-Clixml` on a computer that does not have Internet access, import the object on a computer that does have Internet access, and then run `Save-Help` on the PSModuleInfo object. The saved help can be transported to the remote computer by using removable storage media, such as a USB drive. The help can be installed on the remote computer by running `Update-Help`. This process can be used to install help on computers that do not have any kind of network access.

To install saved help files, run the `Update-Help` cmdlet. Add its SourcePath parameter to specify the folder in which you saved the Help files.

Without parameters, a `Save-Help` command downloads the newest help for all modules in the session and for modules that are installed on the computer in a location listed in the PSModulePath environment variable. This action skips modules that do not support Updatable Help without warning.

The `Save-Help` cmdlet checks the version of any help files in the destination

folder. If newer help files are available, this cmdlet downloads the newest help files from the Internet, and then saves them in the folder. The `Save-Help` cmdlet works just like the `Update-Help` cmdlet, except that it saves the downloaded cabinet (.cab) files, instead of extracting the help files from the cabinet files and installing them on the computer.

The saved help for each module consists of one help information (HelpInfo XML) file and one cabinet (.cab) file for the help files each UI culture. You do not have to extract the help files from the cabinet file. The `Update-Help` cmdlet extracts the help files, validates the XML for safety, and then installs the help files and the help information file in a language-specific subfolder of the module folder.

To save the help files for modules in the PowerShell installation folder (`$pshome\Modules`), start PowerShell by using the Run as administrator option. You must be a member of the Administrators group on the computer to download the help files for these modules.

This cmdlet was introduced in Windows PowerShell 3.0.

PARAMETERS

-Credential <System.Management.Automation.PSCredential>

Specifies a user credential. This cmdlet runs the command by using credentials of a user who has permission to access the file system location specified by the DestinationPath parameter. This parameter is valid only when the DestinationPath or LiteralPath parameter is used in the command.

This parameter enables you to run `Save-Help` commands that use the DestinationPath parameter on remote computers. By providing explicit credentials, you can run the command on a remote computer and access a file share on a third computer without encountering an access denied error

or using CredSSP authentication to delegate credentials.

Type a user name, such as User01 or Domain01\User01 , or enter a PSCredential object generated by the `Get-Credential` cmdlet. If you type a user name, you're prompted to enter the password.

Credentials are stored in a PSCredential (/dotnet/api/system.management.automation.pscredential)object and the password is stored as a SecureString (/dotnet/api/system.security.securestring).

> [!NOTE] > For more information about SecureString data protection, see > How secure is SecureString? (/dotnet/api/system.security.securestring#how-secure-is-securestring).

-DestinationPath <System.String[]>

Specifies the path of the folder in which the help files are saved. Do not specify a file name or file name extension.

-Force <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet does not follow the once-per-day limitation, skips version checking, and downloads files that exceed the 1 GB limit.

Without this parameter, only one `Save-Help` command for each module is permitted in each 24-hour period, downloads are limited to 1 GB of uncompressed content per module, and help files for a module are installed only when they are newer than the files on the computer.

The once-per-day limit protects the servers that host the help files, and makes it practical for you to add a `Save-Help` command to your PowerShell profile.

To save help for a module in multiple UI cultures without the Force

parameter, include all UI cultures in the same command, such as:
`Save-Help -Module PSScheduledJobs -UICulture en-US, fr-FR, pt-BR`

-FullyQualifiedModule <Microsoft.PowerShell.Commands.ModuleSpecification[]>
The value can be a module name, a full module specification, or a path to a module file.

When the value is a path, the path can be fully qualified or relative. A relative path is resolved relative to the script that contains the using statement.

When the value is a name or module specification, PowerShell searches the PSModulePath for the specified module.

A module specification is a hashtable that has the following keys.

- `ModuleName` - Required Specifies the module name. - `GUID` - Optional Specifies the GUID of the module. - It's also Required to specify at least one of the three below keys. - `ModuleVersion` - Specifies a minimum acceptable version of the module. - `MaximumVersion` - Specifies the maximum acceptable version of the module. - `RequiredVersion` - Specifies an exact, required version of the module. This can't be used with the other Version keys.

You can't specify the FullyQualifiedModule parameter in the same command as a Module parameter. the two parameters are mutually exclusive.

-LiteralPath <System.String[]>
Specifies a path of the destination folder. Unlike the value of the DestinationPath parameter, the value of the LiteralPath parameter is used exactly as it is typed. No characters are interpreted as wildcard characters. If the path includes escape characters, enclose it in single quotation marks. Single quotation marks tell PowerShell not to interpret

any characters as escape sequences.

-Module <System.Management.Automation.PSModuleInfo[]>

Specifies modules for which this cmdlet downloads help. Enter one or more module names or name patters in a comma-separated list or in a file that has one module name on each line. Wildcard characters are permitted. You can also pipe module objects from the `Get-Module` cmdlet to `Save-Help`.

By default, `Save-Help` downloads help for all modules that support Updatable Help and are installed on the local computer in a location listed in the PSModulePath environment variable.

To save help for modules that are not installed on the computer, run a `Get-Module` command on a remote computer. Then pipe the resulting module objects to the `Save-Help` cmdlet or submit the module objects as the value of the Module or InputObject parameters.

If the module that you specify is installed on the computer, you can enter the module name or a module object. If the module is not installed on the computer, you must enter a module object, such as one returned by the `Get-Module` cmdlet.

The Module parameter of the `Save-Help` cmdlet does not accept the full path of a module file or module manifest file. To save help for a module that is not in a PSModulePath location, import the module into the current session before you run the `Save-Help` command.

A value of "*" (all) attempts to update help for all modules that are installed on the computer. This includes modules that do not support Updatable Help. This value might generate errors when the command encounters modules that do not support Updatable Help.

-UICulture <System.Globalization.CultureInfo[]>

Specifies UI culture values for which this cmdlet gets updated help files. Enter one or more language codes, such as `es-ES`, a variable that contains culture objects, or a command that gets culture objects, such as a `Get-Culture` or `Get-UICulture` command.

Wildcard characters are not permitted. Do not specify a partial language code, such as "de".

By default, `Save-Help` gets help files in the UI culture set for Windows or its fallback culture. If you specify the UICulture parameter, `Save-Help` looks for help only for the specified UI culture, not in any fallback culture.

-UseDefaultCredentials <System.Management.Automation.SwitchParameter>
Indicates that this cmdlet runs the command, including the web download, with the credentials of the current user. By default, the command runs without explicit credentials.

This parameter is effective only when the web download uses NTLM, negotiate, or Kerberos-based authentication.

<CommonParameters>
This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

------ Example 1: Save the help for the DhcpServer module ------

Option 1: Run Invoke-Command to get the PSModuleInfo object for the remote DHCP Server module,
save the PSModuleInfo object in the variable $m, and then run Save-Help.

```
$m = Invoke-Command -ComputerName RemoteServer -ScriptBlock { Get-Module -Name
DhcpServer -ListAvailable }
Save-Help -Module $m -DestinationPath "C:\SavedHelp"


# Option 2: Open a PSSession--targeted at the remote computer that is running
the DhcpServer
# module--to get the PSModuleInfo object for the remote module, and then run
Save-Help.

$s = New-PSSession -ComputerName "RemoteServer"
$m = Get-Module -PSSession $s -Name "DhcpServer" -ListAvailable
Save-Help -Module $m -DestinationPath "C:\SavedHelp"


# Option 3: Open a CIM session--targeted at the remote computer that is
running the DhcpServer
# module--to get the PSModuleInfo object for the remote module, and then run
Save-Help.

$c = New-CimSession -ComputerName "RemoteServer"
$m = Get-Module -CimSession $c -Name "DhcpServer" -ListAvailable
Save-Help -Module $m -DestinationPath "C:\SavedHelp"
```

This example shows three different ways to use `Save-Help` to save the help
for the DhcpServer module from an Internet-connected client computer, without
installing the DhcpServer module or the DHCP Server role on the local computer.
------ Example 2: Install help for the DhcpServer module ------

```
# First, run Export-CliXml to export the PSModuleInfo object to a shared
folder or to removable media.

$m = Get-Module -Name "DhcpServer" -ListAvailable
```

```
Export-CliXml -Path "E:\UsbFlashDrive\DhcpModule.xml" -InputObject $m

# Next, transport the removable media to a computer that has Internet access,
and then import the
# PSModuleInfo object with Import-CliXml. Run Save-Help to save the Help for
the imported DhcpServer
# module PSModuleInfo object.

$deserialized_m = Import-CliXml "E:\UsbFlashDrive\DhcpModule.xml"
Save-Help -Module $deserialized_m -DestinationPath "E:\UsbFlashDrive\SavedHelp"

# Finally, transport the removable media back to the computer that does not
have network access, and
# then install the help by running Update-Help.

Update-Help -Module DhcpServer -SourcePath "E:\UsbFlashDrive\SavedHelp"
```

This example shows how to install help that you saved in Example 1 for the DhcpServer module on a computer that does not have Internet access.

---------- Example 3: Save help for all modules ----------

```
Save-Help -DestinationPath "\\Server01\FileShare01"
```

This command downloads the newest help files for all modules in the UI culture set for Windows on the local computer. It saves the help files in the `\Server01\Fileshare01` folder.

------ Example 4: Save help for a module on the computer ------

```
Save-Help -Module ServerManager -DestinationPath "\\Server01\FileShare01"
-Credential Domain01/Admin01
```

This command downloads the newest help files for the ServerManager module, and then saves them in the `\Server01\Fileshare01` folder.

When a module is installed on the computer, you can type the module name as the value of the Module parameter, even if the module is not imported into the current session.

The command uses the Credential parameter to supply the credentials of a user who has permission to write to the file share.
-- Example 5: Save help for a module on a different computer --

Invoke-Command -ComputerName Server02 {Get-Module -Name CustomSQL -ListAvailable} | Save-Help -DestinationPath \\Server01\FileShare01 -Credential Domain01\Admin01

These commands download the newest help files for the CustomSQL module and save them in the `\Server01\Fileshare01` folder.

Because the CustomSQL module is not installed on the computer, the sequence includes an `Invoke-Command` command that gets the module object for the CustomSQL module from the Server02 computer and then pipes the module object to the `Save-Help` cmdlet.

When a module is not installed on the computer, `Save-Help` needs the module object, which includes information about the location of the newest help files.
--- Example 6: Save help for a module in multiple languages ---

Save-Help -Module Microsoft.PowerShell* -UICulture de-DE, en-US, fr-FR, ja-JP -DestinationPath "D:\Help"

This command saves help for the core PowerShell modules in four different UI cultures. The language packs for these locales do not have to be installed on the computer.

`Save-Help` can download help files for modules in different UI cultures only

when the module owner makes the translated files available on the Internet.

------- Example 7: Save help more than one time each day -------

Save-Help -Force -DestinationPath "\\Server3\AdminShare\Help"

This command saves help for all modules that are installed on the computer. The command specifies the Force parameter to override the rule that prevents the `Save-Help` cmdlet from downloading help more than once in each 24-hour period.

The Force parameter also overrides the 1 GB restriction and circumvents version checking. Therefore, you can download files even if the version is not later than the version in the destination folder.

The command uses the `Save-Help` cmdlet to download and save the help files to the specified folder. The Force parameter is required when you have to run a `Save-Help` command more than one time each day.

REMARKS

To see the examples, type: "get-help Save-Help -examples".

For more information, type: "get-help Save-Help -detailed".

For technical information, type: "get-help Save-Help -full".

For online help, type: "get-help Save-Help -online"