PowerShell Get-Help on command 'Export-PSSession'

PS C:\Users\wahid> Get-Help Export-PSSession

NAME

    Export-PSSession

SYNOPSIS

    Exports commands from another session and saves them in a PowerShell module.

SYNTAX

```
    Export-PSSession [-Session] <System.Management.Automation.Runspaces.PSSession>
    [-OutputModule] <System.String> [[-CommandName] <System.String[]>]
    [[-FormatTypeName] <System.String[]>] [-AllowClobber] [-ArgumentList
    <System.Object[]>] [-Certificate
    <System.Security.Cryptography.X509Certificates.X509Certificate2>]
    [-CommandType {Alias | All | Application | Cmdlet | Configuration |
    ExternalScript | Filter | Function | Script | Workflow}] [-Encoding {ASCII |
    BigEndianUnicode | Default | OEM | Unicode | UTF7 | UTF8 | UTF32}] [-Force]
    [-FullyQualifiedModule <Microsoft.PowerShell.Commands.ModuleSpecification[]>]
    [-Module <System.String[]>] [<CommonParameters>]
```

DESCRIPTION

The `Export-PSSession` cmdlet gets cmdlets, functions, aliases, and other command types from another PowerShell session (PSSession) on a local or remote computer and saves them in a PowerShell module. To add the commands from the module to the current session, use the `Import-Module` cmdlet.

Unlike `Import-PSSession`, which imports commands from another PSSession into the current session, `Export-PSSession` saves the commands in a module. The commands are not imported into the current session.

To export commands, use the `New-PSSession` cmdlet to create a PSSession that has the commands that you want to export. Then use the `Export-PSSession` cmdlet to export the commands.

To prevent command name conflicts, the default for `Export-PSSession` is to export all commands, except for commands that exist in the current session. You can use the CommandName parameter to specify the commands to export.

The `Export-PSSession` cmdlet uses the implicit remoting feature of PowerShell. When you import commands into the current session, they run implicitly in the original session or in a similar session on the originating computer.

PARAMETERS

-AllowClobber <System.Management.Automation.SwitchParameter>

Exports the specified commands, even if they have the same names as commands in the current session.

If you export a command with the same name as a command in the current session, the exported command hides or replaces the original commands. For more information, see about_Command_Precedence (../Microsoft.PowerShell.Core/About/about_Command_Precedence.md).

-ArgumentList <System.Object[]>

Exports the variant of the command that results from using the specified arguments (parameter values).

For example, to export the variant of the `Get-Item` command in the certificate (Cert:) drive in the PSSession in `$S`, type `Export-PSSession -Session $S -Command Get-Item -ArgumentList cert:`.

-Certificate <System.Security.Cryptography.X509Certificates.X509Certificate2>

Specifies the client certificate that is used to sign the format files (*.Format.ps1xml) or script module files (.psm1) in the module that `Export-PSSession` creates. Enter a variable that contains a certificate or a command or expression that gets the certificate.

To find a certificate, use the `Get-PfxCertificate` cmdlet or use the `Get-ChildItem` cmdlet in the Certificate (Cert:) drive. If the certificate is not valid or does not have sufficient authority, the command fails.

-CommandName <System.String[]>

Exports only the commands with the specified names or name patterns. Wildcards are permitted. Use CommandName or its alias, Name .

By default, `Export-PSSession` exports all commands from the PSSession except for commands that have the same names as commands in the current session. This prevents commands from being hidden or replaced by commands in the current session. To export all commands, even those that hide or replace other commands, use the AllowClobber parameter.

If you use the CommandName parameter, the formatting files for the commands are not exported unless you use the FormatTypeName parameter. Similarly, if you use the FormatTypeName parameter, no commands are exported unless you use the CommandName parameter.

-CommandType <System.Management.Automation.CommandTypes>

Exports only the specified types of command objects. Use CommandType or its alias, Type .

The acceptable values for this parameter are as follows:

- `Alias`: All PowerShell aliases in the current session.

- `All`: All command types. It is the equivalent of `Get-Command -Name *`.

- `Application`: All files other than PowerShell files in paths listed in the Path environment

variable (`$env:path`), including .txt, .exe, and .dll files. - `Cmdlet`: The cmdlets in the current session. Cmdlet is the default.

- `Configuration`: A PowerShell configuration. For more information, see about_Session_Configurations (../Microsoft.PowerShell.Core/About/about_Session_Configurations.md). - `ExternalScript`: All .ps1 files in the paths listed in the Path environment variable (`$env:path`). - `Filter` and `Function`: All PowerShell functions.

- `Script` Script blocks in the current session.

- `Workflow` A PowerShell workflow. For more information, see about_Workflows (../PSWorkflow/About/about_Workflows.md).

These values are defined as a flag-based enumeration. You can combine multiple values together to set multiple flags using this parameter. The values can be passed to the CommandType parameter as an array of values or as a comma-separated string of those values. The cmdlet will combine the

values using a binary-OR operation. Passing values as an array is the
simplest option and also allows you to use tab-completion on the values.

-Encoding <System.String>

Specifies the type of encoding for the target file. The default value is
`UTF8`.

The acceptable values for this parameter are as follows:

- `ASCII`: Uses ASCII (7-bit) character set.

- `BigEndianUnicode`: Uses UTF-16 with the big-endian byte order.

- `Default`; Uses the encoding that corresponds to the system's active
code page.

- `OEM`: Uses the encoding that corresponds to the system's current OEM
code page.

- `Unicode`: Uses UTF-16 with the little-endian byte order.

- `UTF7`: Uses UTF-7.

- `UTF8`: Uses UTF-8.

- `UTF32`: Uses UTF-32 with the little-endian byte order.

-Force <System.Management.Automation.SwitchParameter>

Overwrites one or more existing output files, even if the file has the
read-only attribute.

-FormatTypeName <System.String[]>

Exports formatting instructions only for the specified Microsoft .NET

Framework types. Enter the type names. By default, `Export-PSSession`
exports formatting instructions for all .NET Framework types that are not
in the System.Management.Automation namespace.

The value of this parameter must be the name of a type that is returned by
a `Get-FormatData` command in the session from which the commands are
being imported. To get all of the formatting data in the remote session,
type `*`.

If you use the FormatTypeName parameter, no commands are exported unless
you use the CommandName parameter.

If you use the CommandName parameter, the formatting files for the
commands are not exported unless you use the FormatTypeName parameter.

-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.

-Module <System.String[]>
Exports only the commands in the specified PowerShell snap-ins and
modules. Enter the snap-in and module names. Wildcards are not permitted.

For more information, see `Import-Module` and about_PSSnapins
(../Microsoft.PowerShell.Core/About/about_PSSnapins.md).

-OutputModule <System.String>
Specifies an optional path and name for the module created by
`Export-PSSession`. The default path is
`$HOME\Documents\WindowsPowerShell\Modules`. This parameter is required.

If the module subdirectory or any of the files that `Export-PSSession`
creates already exist, the command fails. To overwrite existing files, use
the Force parameter.

-Session <System.Management.Automation.Runspaces.PSSession>
Specifies the PSSession from which the commands are exported. Enter a
variable that contains a session object or a command that gets a session
object, such as a `Get-PSSession` command. This parameter is required.

<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: Export commands from a PSSession ---------

```
$S = New-PSSession -ComputerName Server01
Export-PSSession -Session $S -OutputModule Server01
```

The `New-PSSession` command creates a PSSession on the Server01 computer. The PSSession is stored in the `$S` variable. The `Export-PSSession` command exports the `$S` variable's commands and formatting data into the Server01 module.

---------- Example 2: Export the Get and Set commands ----------

```
$S = New-PSSession -ConnectionUri https://exchange.microsoft.com/mailbox
-Credential exchangeadmin01@hotmail.com -Authentication Negotiate
Export-PSSession -Session $S -Module exch* -CommandName Get-*, Set-*
-FormatTypeName * -OutputModule $PSHOME\Modules\Exchange -Encoding ASCII
```

These commands export the `Get` and `Set` commands from a Microsoft Exchange Server snap-in on a remote computer to an Exchange module in the `$PSHOME\Modules` directory on the local computer. Placing the module in the `$PSHOME\Modules` directory makes it accessible to all users of the computer.

------ Example 3: Export commands from a remote computer ------

```
$S = New-PSSession -ComputerName Server01 -Credential Server01\User01
Export-PSSession -Session $S -OutputModule TestCmdlets -Type Cmdlet
-CommandName *test* -FormatTypeName *
Remove-PSSession $S
Import-Module TestCmdlets
Get-Help Test*
Test-Files
```

The `New-PSSession` command creates a PSSession on the Server01 computer and saves it in the `$S` variable. The `Export-PSSession` command exports the cmdlets whose names begin with Test from the PSSession in `$S` to the

TestCmdlets module on the local computer.

The `Remove-PSSession` cmdlet deletes the PSSession in `$S` from the current session. This command shows that the PSSession need not be active to use the commands that were imported from the session. The `Import-Module` cmdlet adds the cmdlets in the TestCmdlets module to the current session. The command can be run in any session at any time.

The `Get-Help` cmdlet gets help for cmdlets whose names begin with Test. After the commands in a module are added to the current session, you can use the `Get-Help` and `Get-Command` cmdlets to learn about the imported commands. The `Test-Files` cmdlet was exported from the Server01 computer and added to the session. The `Test-Files` cmdlet runs in a remote session on the computer from which the command was imported. PowerShell creates a session from information that is stored in the TestCmdlets module.

Example 4: Export and clobber commands in the current session

```
Export-PSSession -Session $S -AllowClobber -OutputModule AllCommands
```

This `Export-PSSession` command exports all commands and all formatting data from the PSSession in the `$S` variable into the current session. The AllowClobber parameter includes commands with the same names as commands in the current session.

------ Example 5: Export commands from a closed PSSession ------

```
$Options = New-PSSessionOption -NoMachineProfile
$S = New-PSSession -ComputerName Server01 -SessionOption $Options
Export-PSSession -Session $S -OutputModule Server01
Remove-PSSession $S
New-PSSession -ComputerName Server01 -SessionOption $Options
Import-Module Server01
```

The `New-PSSessionOption` cmdlet creates a PSSessionOption object, and it

saves the object in the `$Options` variable. The `New-PSSession` command
creates a PSSession on the Server01 computer. The SessionOption parameter uses
the object stored in `$Options`. The session is stored in the `$S` variable.

The `Export-PSSession` cmdlet exports commands from the PSSession in `$S` to
the Server01 module. The `Remove-PSSession` cmdlet deletes the PSSession in
the `$S` variable.

The `New-PSSession` cmdlet creates a new PSSession that connects to the
Server01 computer. The SessionOption parameter uses the object stored in
`$Options`. The `Import-Module` cmdlet imports the commands from the Server01
module. The commands in the module are run in the PSSession on the Server01
computer.

REMARKS

To see the examples, type: "get-help Export-PSSession -examples".
For more information, type: "get-help Export-PSSession -detailed".
For technical information, type: "get-help Export-PSSession -full".
For online help, type: "get-help Export-PSSession -online"