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

PowerShell Get-Help on command 'Get-Package'

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

NAME

Get-Package

SYNOPSIS

Returns a list of all software packages that were installed with PackageManagement .

SYNTAX

Get-Package [[-Name] <System.String[]>] [-AdditionalArguments <System.String[]>] [-AllVersions] [-Force] [-ForceBootstrap] [-MaximumVersion <System.String>] [-MinimumVersion <System.String>] [-ProviderName {Bootstrap | NuGet | PowerShellGet}] [-RequiredVersion <System.String>] [<CommonParameters>]

Get-Package [[-Name] <System.String[]>] [-AllowClobber] [-AllowPrereleaseVersions] [-AllVersions] [-Force] [-ForceBootstrap] [-InstallUpdate] [-MaximumVersion <System.String>] [-MinimumVersion <System.String>] [-NoPathUpdate] [-PackageManagementProvider <System.String>] [-ProviderName {Bootstrap | NuGet | PowerShellGet}] [-RequiredVersion <System.String>] [-Scope {CurrentUser | AllUsers}] [-SkipPublisherCheck] [-Type {Module | Script | All}] [<CommonParameters>]

Get-Package [[-Name] <System.String[]>] [-AllVersions] [-Destination <System.String>] [-ExcludeVersion] [-Force] [-ForceBootstrap] [-MaximumVersion <System.String>] [-MinimumVersion <System.String>] [-ProviderName {Bootstrap | NuGet | PowerShellGet}] [-RequiredVersion <System.String>] [-Scope {CurrentUser | AllUsers}] [-SkipDependencies] [<CommonParameters>]

Get-Package [[-Name] <System.String[]>] [-AllVersions] [-Force] [-ForceBootstrap] [-IncludeSystemComponent] [-IncludeWindowsInstaller] [-MaximumVersion <System.String>] [-MinimumVersion <System.String>] [-ProviderName {Bootstrap | NuGet | PowerShellGet}] [-RequiredVersion <System.String>] [<CommonParameters>]

DESCRIPTION

The `Get-Package` cmdlet returns a list of all software packages on the local computer that were installed with PackageManagement . You can run `Get-Package` on remote computers by running it as part of an `Invoke-Command` or `Enter-PSSession` command or script. !INCLUDE [nuget-module (../../includes/nuget-module.md)]

PARAMETERS

-AdditionalArguments <System.String[]>

Specifies additional arguments.

-AllowClobber <System.Management.Automation.SwitchParameter>

Overrides warning messages about conflicts with existing commands. Overwrites existing commands that have the same name as commands being installed by a module.

-AllowPrereleaseVersions <System.Management.Automation.SwitchParameter>

Includes packages marked as a prerelease in the results.

-AllVersions <System.Management.Automation.SwitchParameter>

Indicates that `Get-Package` returns all available versions of the package. By default, `Get-Package` only returns the newest available version.

-Destination <System.String>

Specifies the path to a directory that contains extracted package files.

-ExcludeVersion <System.Management.Automation.SwitchParameter>

Switch to exclude the version number in the folder path.

-Force <System.Management.Automation.SwitchParameter>

Forces the command to run without asking for user confirmation.

-ForceBootstrap <System.Management.Automation.SwitchParameter>

Indicates that `Get-Package` forces PackageManagement to automatically install the package provider.

-IncludeSystemComponent <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet includes system components in the results.

-IncludeWindowsInstaller <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet includes the Windows Installer in the results.

-InstallUpdate <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet installs updates.

-MaximumVersion <System.String>

Specifies the maximum package version that you want to find.

-MinimumVersion <System.String>

Specifies the minimum package version that you want to find. If a higher

version is available, that version is returned.

-Name <System.String[]>

Specifies one or more package names, or package names with wildcard characters. Separate multiple package names with commas.

-NoPathUpdate <System.Management.Automation.SwitchParameter>

NoPathUpdate only applies to the `Install-Script` cmdlet. NoPathUpdate is a dynamic parameter added by the provider and isn't supported by `Get-Package`.

-PackageManagementProvider <System.String>

Specifies the name of a package management provider.

-ProviderName <System.String[]>

Specifies one or more package provider names. Separate multiple package provider names with commas. Use `Get-PackageProvider` to get a list of available package providers.

-RequiredVersion <System.String>

Specifies the exact version of the package to find.

-Scope <System.String>

Specifies the search scope for the package.

-SkipDependencies <System.Management.Automation.SwitchParameter>

Switch that specifies to skip finding any package dependencies.

-SkipPublisherCheck <System.Management.Automation.SwitchParameter>

Allows you to get a package version that is newer than your installed version. For example, an installed package that is digitally signed by a trusted publisher but a new version isn't digitally signed.

-Type <System.String>
    Specifies whether to search for packages with a module, a script, or
    either.

<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: Get all installed packages ----------

Get-Package

Name   Version   Source
ProviderName
----   -------   ------
----
posh-git   0.7.3   https://www.powershellgallery.com/api/v2
PowerShellGet

Example 2: Get packages that are installed on a remote computer

PS> Invoke-Command -ComputerName Server01 -Credential CONTOSO\TestUser
-ScriptBlock {Get-Package}

`Invoke-Command` uses the ComputerName parameter to specify a remote computer,
Server01 . The Credential parameter specifies a domain and user name with
permissions to run commands on the computer. The ScriptBlock parameter runs
the `Get-Package` cmdlet on the remote computer.
------- Example 3: Get packages for a specified provider -------

```
Get-Package -ProviderName PowerShellGet -AllVersions

Name                Version      Source
ProviderName
----                -------      ------
------------
PackageManagement   1.2.2        https://www.powershellgallery.com/api/v2
PowerShellGet
PackageManagement   1.3.1        https://www.powershellgallery.com/api/v2
PowerShellGet
posh-git            0.7.3        https://www.powershellgallery.com/api/v2
PowerShellGet
PowerShellGet       2.0.1        https://www.powershellgallery.com/api/v2
PowerShellGet
```

`Get-Package` uses the ProviderName parameter to specify a specific provider, PowerShellGet . The AllVersions parameter displays each version that is installed.

---- Example 4: Get an exact version of a specific package ----

```
Get-Package -Name PackageManagement -ProviderName PowerShellGet
-RequiredVersion 1.3.1

Name                Version      Source
ProviderName
----                -------      ------
------------
PackageManagement   1.3.1        https://www.powershellgallery.com/api/v2
PowerShellGet
```

`Get-Package` uses Name parameter to specify the package name, PackageManagement . The ProviderName parameter specifies the provider, PowerShellGet . The RequiredVersion parameter specifies an installed version.

---------- Example 5: Uninstall a package ----------

Get-Package -Name posh-git -RequiredVersion 0.7.3 | Uninstall-Package

`Get-Package` uses the Name parameter to specify the package name, posh-git .

The RequiredVersion parameter is a specific version of the package. The object

is sent down the pipeline to the `Uninstall-Package` cmdlet.

`Uninstall-Package` removes the package.

REMARKS

To see the examples, type: "get-help Get-Package -examples".

For more information, type: "get-help Get-Package -detailed".

For technical information, type: "get-help Get-Package -full".

For online help, type: "get-help Get-Package -online"