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

PowerShell Get-Help on command 'Add-AppPackage'

PS C:\Users\wahid> Get-Help Add-AppPackage

NAME

Add-AppxPackage

SYNOPSIS

Adds a signed app package to a user account.

SYNTAX

Add-AppxPackage [-Path] <System.String> [-AllowUnsigned]

[-DeferRegistrationWhenPackagesAreInUse] [-DependencyPath <System.String[]>]

[-ExternalLocation <System.String>] [-ExternalPackages <System.String[]>]

[-ForceApplicationShutdown] [-ForceTargetApplicationShutdown]

[-ForceUpdateFromAnyVersion] [-InstallAllResources] [-OptionalPackages

<System.String[]>] [-RelatedPackages <System.String[]>]

[-RequiredContentGroupOnly] [-RetainFilesOnFailure] [-StubPackageOption

<StubPackageOption>] [-Volume <AppxVolume>] [-Confirm] [-WhatIf]

[<CommonParameters>]

Add-AppxPackage [-Path] <System.String> -AppInstallerFile

[-ForceTargetApplicationShutdown] [-InstallAllResources]

[-LimitToExistingPackages] [-RequiredContentGroupOnly] [-Volume <AppxVolume>]

[-Confirm] [-WhatIf] [<CommonParameters>]

Add-AppxPackage [-DependencyPackages <System.String[]>]
[-ForceApplicationShutdown] [-ForceTargetApplicationShutdown]
[-ForceUpdateFromAnyVersion] [-InstallAllResources] -MainPackage
<System.String> [-Register] [-Confirm] [-WhatIf] [<CommonParameters>]

Add-AppxPackage [-DependencyPackages <System.String[]>]
[-ForceApplicationShutdown] [-ForceTargetApplicationShutdown]
[-InstallAllResources] -MainPackage <System.String> [-OptionalPackages
<System.String[]>] -RegisterByFamilyName [-Confirm] [-WhatIf]
[<CommonParameters>]

Add-AppxPackage [-Path] <System.String> [-DependencyPath <System.String[]>]
[-DisableDevelopmentMode] [-ExternalLocation <System.String>]
[-ForceApplicationShutdown] [-ForceTargetApplicationShutdown]
[-ForceUpdateFromAnyVersion] [-InstallAllResources] -Register [-Confirm]
[-WhatIf] [<CommonParameters>]

Add-AppxPackage [-Path] <System.String> [-DependencyPath <System.String[]>]
[-ForceApplicationShutdown] [-ForceTargetApplicationShutdown]
[-ForceUpdateFromAnyVersion] [-InstallAllResources]
[-RequiredContentGroupOnly] [-RetainFilesOnFailure] -Update [-Confirm]
[-WhatIf] [<CommonParameters>]

Add-AppxPackage [-Path] <System.String> [-DependencyPath <System.String[]>]
[-ExternalLocation <System.String>] [-ExternalPackages <System.String[]>]
[-ForceUpdateFromAnyVersion] [-OptionalPackages <System.String[]>]
[-RelatedPackages <System.String[]>] [-RequiredContentGroupOnly] -Stage
[-StubPackageOption <StubPackageOption>] [-Volume <AppxVolume>] [-Confirm]
[-WhatIf] [<CommonParameters>]

DESCRIPTION

The `Add-AppxPackage` cmdlet adds a signed app package to a user account. An app package has an `.msix` or `.appx` filename extension. Use the DependencyPath parameter to add all other packages required for the installation of the app package.

You can use the Register parameter to install from a folder of unpackaged files during development of Windows Store apps.

To update an already installed package, the new package must have the same package family name.

PARAMETERS

-AllowUnsigned <System.Management.Automation.SwitchParameter>

Allows adding an unsigned package.

-AppInstallerFile <System.Management.Automation.SwitchParameter>

Runs an appinstaller file and allows the user to install all the defined packages with a single click. For more information, see Create an App Installer file manually (/windows/msix/app-installer/how-to-create-appinstaller-file).

-DeferRegistrationWhenPackagesAreInUse <System.Management.Automation.SwitchParameter>

Specifies that the app won't register for a user if currently in use. The app will update on the next launch.

-DependencyPackages <System.String[]>

Specifies the dependency package full name or dependency package bundle full name to be registered.

-DependencyPath <System.String[]>

Specifies an array of file paths of dependency packages that are required for the installation of the app package. The app package has an `.msix`, `.appx`, `.msixbundle`, or `.appxbundle` filename extension. You can specify the paths to more than one dependency package. If a package is already installed for a user, you can skip adding it to the DependencyPath.

-DisableDevelopmentMode <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet registers an existing app package installation that has been disabled, didn't register, or has become corrupted. Use the current parameter to specify that the manifest is from an existing installation, and not from a collection of files in development mode. You can also use this parameter to register an application that the Package Manager API (https://go.microsoft.com/fwlink/?LinkId=245447)has staged. Use the Register parameter to specify the location of the app package manifest `.xml` file from the installation location.

-ExternalLocation <System.String>

URI path of an external disk location outside of the MSIX package where the package manifest can reference application content.

-ExternalPackages <System.String[]>

Specifies an array of optional packages that must be installed along with the app package. It's an atomic operation, which means that if the app or its optional packages fail to install, the deployment operation will be aborted.

-ForceApplicationShutdown <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet forces all active processes associated with the package or its dependencies to shut down. If you specify this parameter, don't specify the ForceTargetApplicationShutdown parameter.

-ForceTargetApplicationShutdown <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet forces all active processes associated with the

package to shut down. If you specify this parameter, don't specify the ForceApplicationShutdown parameter.

-ForceUpdateFromAnyVersion <System.Management.Automation.SwitchParameter>
This parameter is used to force a specific version of a package to be staged or registered, regardless of whether a higher version is already staged or registered.

-InstallAllResources <System.Management.Automation.SwitchParameter>
Indicates that this cmdlet forces the deployment of all resource packages specified from a bundle argument. This overrides the resource applicability check of the deployment engine and forces staging of all resource packages, registration of all resource packages, or staging and registration of all resource packages. This parameter can only be used when specifying a resource bundle or resource bundle manifest.

-LimitToExistingPackages <System.Management.Automation.SwitchParameter>
This parameter is used to prevent missing referenced packages to be downloaded.

-MainPackage <System.String>
Specifies the main package full name or bundle full name to register.

-OptionalPackages <System.String[]>
Specifies the PackageFamilyName of the optional packages that are in a related set that need to be installed along with the app. Unlike the external packages flag, you don't need to pass in a path to the optional packages. It's an atomic operation, which means that if the app or its optional packages fail to install, the deployment operation will be aborted.

-Path <System.String>
Specifies the path to the app package file. An app package has an `.msix`,

`.appx`, `.msixbundle`, or `.appxbundle` filename extension.

-Register <System.Management.Automation.SwitchParameter>
Indicates that this cmdlet registers an application in development mode. You can use development mode to install applications from a folder of unpackaged files. You can use the current parameter to test your Windows Store apps before you deploy them as app packages. To register an existing app package installation, you must specify the DisableDevelopmentMode parameter and the Register parameter. To specify dependency packages, use the DependencyPath parameter and the DisableDevelopmentMode parameter.

-RegisterByFamilyName <System.Management.Automation.SwitchParameter>
Specifies the parameter -MainPackage that defines the family name or full name to be registered.

-RelatedPackages <System.String[]>
This is an optional element that's used to specify the other optional packages that are specified in the main app package. These packages won't be installed as part of the deployment operation.

-RequiredContentGroupOnly <System.Management.Automation.SwitchParameter>
Specifies that only the required content group that's specified in the `AppxContentGroupMap.xml` must be installed. At this point the app can be launched. Calling `Add-AppxPackage` and specifying the path to the app triggers the rest of the app to be installed in the order defined in the `AppxContentGroupMap.xml`.

-RetainFilesOnFailure <System.Management.Automation.SwitchParameter>
In case of a failed deployment, if this switch is set to `$true`, files that have been created on the target machine during the installation process aren't removed.

-Stage <System.Management.Automation.SwitchParameter>

Stages a package to the system without registering it.

-StubPackageOption <StubPackageOption>

Defines the stub behavior for an app package that's being added or staged.

The acceptable values for this parameter are:

- `Default`: Uses the default behavior

- `InstallFull`: Installs as a full app

- `InstallStub`: Installs as a stub app

- `UsePreference`: Uses the current PackageStubPreference (/uwp/api/windows.management.deployment.packagestubpreference)for the package

-Update <System.Management.Automation.SwitchParameter>

Specifies that the package being added is a dependency package update. A dependency package is removed from the user account when the parent app is removed. If you don't use this parameter, the package being added is a primary package and isn't removed from the user account if the parent app is removed. To update an already installed package, the new package must have the same package family name.

-Volume <AppxVolume>

Specifies the AppxVolume object to stage the package in. The volume also specifies the default location for user AppData .

-Confirm <System.Management.Automation.SwitchParameter>

Prompts you for confirmation before running the cmdlet.

-WhatIf <System.Management.Automation.SwitchParameter>

Shows what would happen if the cmdlet runs. The cmdlet isn't run.

<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: Add an app package ----------

Add-AppxPackage -Path '.\MyApp.msix' -DependencyPath '.\winjs.msix'

This command adds an app package that the package contains.
Example 2: Update an app, but defer registration until the app has closed

$params = @{
    Path = '.\MyApp.msix'
    DependencyPath = '.\winjs.msix'
    DeferRegistrationWhenPackagesAreInUse = $true
}
Add-AppxPackage @params

This command will register an update to an existing app, but won't do so until
the next launch of the app.
-- Example 3: Add a disabled app package in development mode --

$InstallLocation = Get-AppxPackage -Name '*WindowsCalculator*' |
    Select-Object -ExpandProperty InstallLocation
$ManifestPath = $InstallLocation + '\Appxmanifest.xml'
Add-AppxPackage -Path $ManifestPath -Register -DisableDevelopmentMode

This command gets the full path of the package manifest file of an installed
Windows Store app, and then registers that package. You can use
DisableDevelopmentMode to register an application that's staged by the

StagePackageAsync API, has been disabled, or has become corrupted during testing.

---- Example 4: Add an app along with its optional packages ----

```
Add-AppxPackage -Path '.\MyApp.msixbundle' -ExternalPackages @(
    '.\optionalpackage1.msix'
    '.\optionalpackage2.msixbundle'
)

Add-AppxPackage -Path '.\MyApp.msixbundle' -OptionalPackages
'29270sandstorm.OptionalPackage1_gah1vdar1nn7a'
```

This command adds an app package along with its optional packages. It's an atomic operation, which means that if the app or its optional packages fail to install, the deployment operation will be aborted

Example 5: Install only the required section of a streaming app

```
Add-AppxPackage -Path '.\MyApp.msixbundle' -RequiredContentGroupOnly
```

This command adds an app package but only installs the required section of a streaming app. Calling this command again without the RequiredContentGroupOnly parameter proceeds to install the rest of the application in the order defined by the `AppxContentGroupMap.xml`

---- Example 6: Install an app using the App Installer file ----

```
Add-AppxPackage -AppInstallerFile "C:\Users\user1\Desktop\MyApp.appinstaller"
```

This command adds an app package as outlined in the App Installer file with all update settings specified within the App Installer file, if any.

REMARKS

To see the examples, type: "get-help Add-AppxPackage -examples".

For more information, type: "get-help Add-AppxPackage -detailed".

For technical information, type: "get-help Add-AppxPackage -full".

For online help, type: "get-help Add-AppxPackage -online"