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

PowerShell Get-Help on command 'Compress-Archive'

PS C:\Users\wahid> Get-Help Compress-Archive

NAME

Compress-Archive

SYNOPSIS

Creates a compressed archive, or zipped file, from specified files and directories.

SYNTAX

Compress-Archive [-Path] <System.String[]> [-DestinationPath] <System.String> [-CompressionLevel {Optimal | NoCompression | Fastest}] -Force [-Confirm] [-WhatIf] [<CommonParameters>]

Compress-Archive [-DestinationPath] <System.String> [-CompressionLevel {Optimal | NoCompression | Fastest}] -Force -LiteralPath <System.String[]> [-Confirm] [-WhatIf] [<CommonParameters>]

Compress-Archive [-DestinationPath] <System.String> [-CompressionLevel {Optimal | NoCompression | Fastest}] -LiteralPath <System.String[]> -Update [-Confirm] [-WhatIf] [<CommonParameters>]

Compress-Archive [-DestinationPath] <System.String> [-CompressionLevel {Optimal | NoCompression | Fastest}] -LiteralPath <System.String[]> [-Confirm] [-WhatIf] [<CommonParameters>]

Compress-Archive [-Path] <System.String[]> [-DestinationPath] <System.String> [-CompressionLevel {Optimal | NoCompression | Fastest}] [-Confirm] [-WhatIf] [<CommonParameters>]

Compress-Archive [-Path] <System.String[]> [-DestinationPath] <System.String> [-CompressionLevel {Optimal | NoCompression | Fastest}] -Update [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `Compress-Archive` cmdlet creates a compressed, or zipped, archive file from one or more specified files or directories. An archive packages multiple files, with optional compression, into a single zipped file for easier distribution and storage. An archive file can be compressed using the compression algorithm specified by the CompressionLevel parameter.

The `Compress-Archive` cmdlet uses the System.IO.Compression.ZipArchive API to compress files. The API limits the maximum file size to 2GB. For more information, see System.IO.Compression.ZipArchive (xref:System.IO.Compression.ZipArchive).

> [!NOTE] > The `Compress-Archive` cmdlet ignores hidden files and folders when creating or updating the > archive file. > > To ensure hidden files and folders are compressed into the archive, use the .NET API instead.

Some examples use splatting to reduce the line length of the code samples. For more information, see about_Splatting (../Microsoft.PowerShell.Core/About/about_Splatting.md).

PARAMETERS

-CompressionLevel <System.String>

Specifies how much compression to apply when you're creating the archive file. Faster compression requires less time to create the file, but can result in larger file sizes.

If this parameter isn't specified, the command uses the default value, Optimal .

The following are the acceptable values for this parameter:

- Fastest . Use the fastest compression method available to reduce processing time. Faster compression can result in larger file sizes. - NoCompression . Doesn't compress the source files. - Optimal . Processing time is dependent on file size.

-DestinationPath <System.String>

This parameter is required and specifies the path to the archive output file. The DestinationPath should include the name of the zipped file, and either the absolute or relative path to the zipped file.

If the file name in DestinationPath doesn't have a `.zip` file name extension, the cmdlet adds the `.zip` file name extension.

-Force <System.Management.Automation.SwitchParameter>

Use this parameter to overwrite an existing archive file.

-LiteralPath <System.String[]>

Specifies the path or paths to the files that you want to add to the archive zipped file. Unlike the Path parameter, the value of LiteralPath is used exactly as it's typed. No characters are interpreted as wildcards. If the path includes escape characters, enclose each escape character in

single quotation marks, to instruct PowerShell not to interpret any characters as escape sequences. To specify multiple paths, and include files in multiple locations in your output zipped file, use commas to separate the paths.

-Path <System.String[]>

Specifies the path or paths to the files that you want to add to the archive zipped file. To specify multiple paths, and include files in multiple locations, use commas to separate the paths.

This parameter accepts wildcard characters. Wildcard characters allow you to add all files in a directory to your archive file.

Using wildcards with a root directory affects the archive's contents:

- To create an archive that includes the root directory, and all its files and subdirectories, specify the root directory in the Path without wildcards. For example: `-Path C:\Reference` - To create an archive that excludes the root directory, but zips all its files and subdirectories, use the asterisk (` `) wildcard. For example: `-Path C:\Reference\ ` - To create an archive that only zips the files in the root directory, use the star-dot-star (` . `) wildcard. Subdirectories of the root aren't included in the archive. For example: `-Path C:\Reference\ . `

-Update <System.Management.Automation.SwitchParameter>

Updates the specified archive by replacing older file versions in the archive with newer file versions that have the same names. You can also add this parameter to add files to an existing archive.

-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: Compress files to create an archive file -----

```
$compress = @{
  Path = "C:\Reference\Draftdoc.docx", "C:\Reference\Images\*.vsd"
  CompressionLevel = "Fastest"
  DestinationPath = "C:\Archives\Draft.zip"
}
Compress-Archive @compress
```

The Path parameter accepts specific file names and file names with wildcards,
`*.vsd`. The Path uses a comma-separated list to get files from different
directories. The compression level is Fastest to reduce processing time. The
DestinationPath parameter specifies the location for the `Draft.zip` file. The
`Draft.zip` file contains `Draftdoc.docx` and all the files with a `.vsd`
extension.

-------- Example 2: Compress files using a LiteralPath --------

```
$compress = @{
LiteralPath= "C:\Reference\Draft Doc.docx", "C:\Reference\Images\diagram2.vsd"
CompressionLevel = "Fastest"
DestinationPath = "C:\Archives\Draft.zip"
}
Compress-Archive @compress
```

Absolute path and file names are used because the LiteralPath parameter

doesn't accept wildcards. The Path uses a comma-separated list to get files from different directories. The compression level is Fastest to reduce processing time. The DestinationPath parameter specifies the location for the `Draft.zip` file. The `Draft.zip` file only contains `Draftdoc.docx` and `diagram2.vsd`.

Example 3: Compress a directory that includes the root directory

```
Compress-Archive -Path C:\Reference -DestinationPath C:\Archives\Draft.zip
```

`Compress-Archive` uses the Path parameter to specify the root directory, `C:\Reference`. The DestinationPath parameter specifies the location for the archive file. The `Draft.zip` archive includes the `Reference` root directory, and all its files and subdirectories.

Example 4: Compress a directory that excludes the root directory

```
Compress-Archive -Path C:\Reference\* -DestinationPath C:\Archives\Draft.zip
```

`Compress-Archive` uses the Path parameter to specify the root directory, `C:\Reference` with an asterisk (` `) wildcard. The DestinationPath * parameter specifies the location for the archive file. The `Draft.zip` archive contains the root directory's files and subdirectories. The `Reference` root directory is excluded from the archive.

---- Example 5: Compress only the files in a root directory ----

```
Compress-Archive -Path C:\Reference\*.* -DestinationPath C:\Archives\Draft.zip
```

`Compress-Archive` uses the Path parameter to specify the root directory, `C:\Reference` with a star-dot-star (` . `) wildcard. The DestinationPath parameter specifies the location for the archive file. The `Draft.zip` archive only contains the `Reference` root directory's files and the root directory is excluded.

--------- Example 6: Use the pipeline to archive files ---------

```
Get-ChildItem -Path C:\Reference\Afile.txt, C:\Reference\Images\Bfile.txt |
  Compress-Archive -DestinationPath C:\Archives\PipelineFiles.zip
```

`Get-ChildItem` uses the Path parameter to specify two files from different directories. Each file is represented by a FileInfo object and is sent down the pipeline to `Compress-Archive`. The two specified files are archived in `PipelineFiles.zip`.

------ Example 7: Use the pipeline to archive a directory ------

```
Get-ChildItem -Path C:\LogFiles | Compress-Archive -DestinationPath
C:\Archives\PipelineDir.zip
```

`Get-ChildItem` uses the Path parameter to specify the `C:\LogFiles` root directory. Each FileInfo and DirectoryInfo object is sent down the pipeline.

`Compress-Archive` adds each object to the `PipelineDir.zip` archive. The Path parameter isn't specified because the pipeline objects are received into parameter position 0.

--------- Example 8: How recursion can affect archives ---------

```
Get-ChildItem -Path C:\TestLog -Recurse |
  Compress-Archive -DestinationPath C:\Archives\PipelineRecurse.zip
```

The `C:\TestLog` directory doesn't contain any files. It does contain a subdirectory named `testsub` that contains the `testlog.txt` file.

`Get-ChildItem` uses the Path parameter to specify the root directory, `C:\TestLog`. The Recurse parameter processes the files and directories. A DirectoryInfo object is created for `testsub` and a FileInfo object `testlog.txt`.

Each object is sent down the pipeline to `Compress-Archive`. The DestinationPath specifies the location for the archive file. The Path

parameter isn't specified because the pipeline objects are received into parameter position 0.

The following summary describes the `PipelineRecurse.zip` archive's contents that contains a duplicate file:

- The DirectoryInfo object creates the `testsub` directory and contains the `testlog.txt` file, which reflects the original directory structure. - The FileInfo object creates a duplicate `testlog.txt` in the archive's root. The duplicate file is created because recursion sent a file object to `Compress-Archive`. This behavior is expected because each object sent down the pipeline is added to the archive.

---------- Example 9: Update an existing archive file ----------

Compress-Archive -Path C:\Reference -Update -DestinationPath C:\Archives\Draft.zip

The command updates `Draft.zip` with newer versions of existing files in the `C:\Reference` directory and its subdirectories. And, new files that were added to `C:\Reference` or its subdirectories are included in the updated `Draft.zip` archive.

REMARKS

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