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

PowerShell Get-Help on command 'Test-ModuleManifest'

PS C:\Users\wahid> Get-Help Test-ModuleManifest

NAME

Test-ModuleManifest

SYNOPSIS

Verifies that a module manifest file accurately describes the contents of a module.

SYNTAX

Test-ModuleManifest [-Path] <System.String> [<CommonParameters>]

DESCRIPTION

The `Test-ModuleManifest` cmdlet verifies that the files that are listed in the module manifest (`.psd1`) file are actually in the specified paths.

This cmdlet is designed to help module authors test their manifest files. Module users can also use this cmdlet in scripts and commands to detect errors before they run scripts that depend on the module.

`Test-ModuleManifest` returns an object that represents the module. This is

the same type of object that `Get-Module` returns. If any files are not in the locations specified in the manifest, the cmdlet also generates an error for each missing file.

PARAMETERS

-Path <System.String>

Specifies a path and file name for the manifest file. Enter an optional path and name of the module manifest file that has the `.psd1` file name extension. The default location is the current directory. Wildcard characters are supported, but must resolve to a single module manifest file. This parameter is required. You can also pipe a path to `Test-ModuleManifest`.

<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: Test a manifest ----------

Test-ModuleManifest -Path "$pshome\Modules\TestModule.psd1"

This command tests the `TestModule.psd1` module manifest.

------- Example 2: Test a manifest by using the pipeline -------

"$pshome\Modules\TestModule.psd1" | test-modulemanifest

Test-ModuleManifest : The specified type data file 'C:\Windows\System32\WindowsPowerShell\v1.0\Modules\TestModule\TestTypes.ps1xml' could not be processed because the file was not found. Please correct the path and try again.

```
At line:1 char:34
+ "$pshome\Modules\TestModule.psd1" | test-modulemanifest <<<<
+ CategoryInfo          : ResourceUnavailable: (C:\Windows\System32\WindowsPowe
rShell\v1.0\Modules\TestModule\TestTypes.ps1xml:String) [Test-ModuleManifest],
FileNotFoundException
+ FullyQualifiedErrorId : Modules_TypeDataFileNotFound,Microsoft.PowerShell.Com
mands.TestModuleManifestCommandName

Name              : TestModule
Path              :
C:\Windows\system32\WindowsPowerShell\v1.0\Modules\TestModule\TestModule.psd1
Description       :
Guid              : 6f0f1387-cd25-4902-b7b4-22cff6aefa7b
Version           : 1.0
ModuleBase        :
C:\Windows\system32\WindowsPowerShell\v1.0\Modules\TestModule
ModuleType        : Manifest
PrivateData       :
AccessMode        : ReadWrite
ExportedAliases   : {}
ExportedCmdlets   : {}
ExportedFunctions : {}
ExportedVariables : {}
NestedModules     : {}
```

This command uses a pipeline operator (`|`) to send a path string to
`Test-ModuleManifest`.

The command output shows that the test failed, because the TestTypes.ps1xml
file, which was listed in the manifest, was not found.

---- Example 3: Write a function to test a module manifest ----

```
function Test-ManifestBool ($path)
```

{$a = dir $path | Test-ModuleManifest -ErrorAction SilentlyContinue; $?}

This function is like `Test-ModuleManifest`, but it returns a Boolean value. The function returns `$True` if the manifest passed the test and `$False` otherwise.

The function uses the Get-ChildItem cmdlet, alias = dir, to get the module manifest specified by the `$path` variable. The command uses a pipeline operator (`|`) to pass the file object to `Test-ModuleManifest`.

`Test-ModuleManifest` uses the ErrorAction common parameter with a value of SilentlyContinue to suppress the display of any errors that the command generates. It also saves the PSModuleInfo object that `Test-ModuleManifest` returns in the $a variable. Therefore, the object is not displayed.

Then, in a separate command, the function displays the value of the `$?` automatic variable. If the previous command generates no error, the command displays `$True`, and `$False` otherwise.

You can use this function in conditional statements, such as those that might precede an `Import-Module` command or a command that uses the module.

REMARKS

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