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

PowerShell Get-Help on command 'Import-LocalizedData'

PS C:\Users\wahid> Get-Help Import-LocalizedData

NAME

Import-LocalizedData

SYNOPSIS

Imports language-specific data into scripts and functions based on the UI culture that's selected for the operating system.

SYNTAX

Import-LocalizedData [[-BindingVariable] <System.String>] [[-UICulture] <System.String>] [-BaseDirectory <System.String>] [-FileName <System.String>] [-SupportedCommand <System.String[]>] [<CommonParameters>]

DESCRIPTION

The `Import-LocalizedData` cmdlet dynamically retrieves strings from a subdirectory whose name matches the UI language set for the current user of the operating system. It's designed to enable scripts to display user messages in the UI language selected by the current user.

`Import-LocalizedData` imports data from `.psd1` files in language-specific

subdirectories of the script directory and saves them in a local variable that's specified in the command. The cmdlet selects the subdirectory and file based on the value of the `$PSUICulture` automatic variable. When you use the local variable in the script to display a user message, the message appears in the user's UI language.

You can use the parameters of `Import-LocalizedData` to specify an alternate UI culture, path, and filename, to add supported commands, and to suppress the error message that appears if the `.psd1` files aren't found.

The `Import-LocalizedData` cmdlet supports the script internationalization initiative that was introduced in Windows PowerShell 2.0. This initiative aims to better serve users worldwide by making it easy for scripts to display user messages in the UI language of the current user. For more information about this and about the format of the `.psd1` files, see about_Script_Internationalization (../Microsoft.PowerShell.Core/About/about_Script_Internationalization.md).

PARAMETERS

-BaseDirectory <System.String>

Specifies the base directory where the `.psd1` files are located. The default is the directory where the script is located. `Import-LocalizedData` searches for the `.psd1` file for the script in a language-specific subdirectory of the base directory.

-BindingVariable <System.String>

Specifies the variable into which the text strings are imported. Enter a variable name without a dollar sign (`$`).

In Windows PowerShell 2.0, this parameter is required. In Windows PowerShell 3.0, this parameter is optional. If you omit this parameter, `Import-LocalizedData` returns a hashtable of the text strings. The

hashtable is passed down the pipeline or displayed at the command line.

When using `Import-LocalizedData` to replace default text strings specified in the DATA section of a script, assign the DATA section to a variable and enter the name of the DATA section variable in the value of the BindingVariable parameter. Then, when `Import-LocalizedData` saves the imported content in the BindingVariable , the imported data will replace the default text strings. If you aren't specifying default text strings, you can select any variable name.

-FileName <System.String>

Specifies the name of the data file (`.psd1)` to be imported. Enter a filename. You can specify a filename that doesn't include its `.psd1` filename extension, or you can specify the filename including the `.psd1` filename extension. Data files should be saved as Unicode or UTF-8.

The FileName parameter is required when `Import-LocalizedData` isn't used in a script. Otherwise, the parameter is optional and the default value is the base name of the script. You can use this parameter to direct `Import-LocalizedData` to search for a different `.psd1` file.

For example, if the FileName is omitted and the script name is `FindFiles.ps1`, `Import-LocalizedData` searches for the `FindFiles.psd1` data file.

-SupportedCommand <System.String[]>

Specifies cmdlets and functions that generate only data.

Use this parameter to include cmdlets and functions that you have written or tested. For more information, see about_Script_Internationalization (../Microsoft.PowerShell.Core/About/about_Script_Internationalization.md).

-UICulture <System.String>

Specifies an alternate UI culture. The default is the value of the `$PsUICulture` automatic variable. Enter a UI culture in `<language>-<region>` format, such as `en-US`, `de-DE`, or `ar-SA`.

The value of the UICulture parameter determines the language-specific subdirectory (within the base directory) from which `Import-LocalizedData` gets the `.psd1` file for the script.

The cmdlet searches for a subdirectory with the same name as the value of the UICulture parameter or the `$PsUICulture` automatic variable, such as `de-DE` or `ar-SA`. If it can't find the directory, or the directory doesn't contain a `.psd1` file for the script, it searches for a subdirectory with the name of the language code, such as de or ar. If it can't find the subdirectory or `.psd1` file, the command fails and the data is displayed in the default language specified in the script.

<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: Import text strings ----------

```
Import-LocalizedData -BindingVariable "Messages"
```

If the command is included in the Archives.ps1 script in the `C:\Test` directory, and the value of the `$PsUICulture` automatic variable is zh-CN, `Import-LocalizedData` imports the `Archives.psd1` file in the `C:\test\zh-CN` directory into the `$Messages` variable.

---------- Example 2: Import localized data strings ----------

```
Import-LocalizedData -FileName "Test.psd1" -UICulture "en-US"
```

Name Value

---- -----

Msg3 "Use $_ to represent the object that's being processed."

Msg2 "This command requires the credentials of a member of the

Administrators group on the...

Msg1 "The Name parameter is missing from the command."

`Import-LocalizedData` returns a hashtable that contains the localized data

strings.

---------- Example 3: Import UI culture strings ----------

Import-LocalizedData -BindingVariable "MsgTbl" -UICulture "ar-SA" -FileName

"Simple" -BaseDirectory "C:\Data\Localized"

This command imports text strings into the `$MsgTbl` variable of a script.

It uses the UICulture parameter to direct the cmdlet to import data from the

`Simple.psd1` file in the `ar-SA` subdirectory of `C:\Data\Localized`.

-------- Example 4: Import localized data into a script --------

PS C:\> # In C:\Test\en-US\Test.psd1:

ConvertFrom-StringData @'

English strings

Msg1 = "The Name parameter is missing from the command."

Msg2 = "This command requires the credentials of a member of the

Administrators group on the computer."

Msg3 = "Use $_ to represent the object that's being processed."

'@

```
# In C:\Test\Test.ps1

Import-LocalizedData -BindingVariable "Messages"
Write-Host $Messages.Msg2

# In Windows PowerShell

PS C:\> .\Test.ps1
```

This command requires the credentials of a member of the Administrators group on the computer.

The first part of the example shows the contents of the `Test.psd1` file. It contains a `ConvertFrom-StringData` command that converts a series of named text strings into a hashtable. The `Test.psd1` file is located in the en-US subdirectory of the `C:\Test` directory that contains the script.

The second part of the example shows the contents of the `Test.ps1` script. It contains an `Import-LocalizedData` command that imports the data from the matching `.psd1` file into the `$Messages` variable and a `Write-Host` command that writes one of the messages in the `$Messages` variable to the host program.

The last part of the example runs the script. The output shows that it displays the correct user message in the UI language set for the current user of the operating system.

----- Example 5: Replace default text strings in a script -----

```
PS C:\> # In TestScript.ps1
$UserMessages = DATA

{    ConvertFrom-StringData @'
```

```
    # English strings

        Msg1 = "Enter a name."
        Msg2 = "Enter your employee ID."
        Msg3 = "Enter your building number."
'@
}

Import-LocalizedData -BindingVariable "UserMessages"
$UserMessages.Msg1...
```

In this example, the DATA section of the TestScript.ps1 script contains a `ConvertFrom-StringData` command that converts the contents of the DATA section to a hashtable and stores in the value of the `$UserMessages` variable.

The script also includes an `Import-LocalizedData` command, which imports a hashtable of translated text strings from the TestScript.psd1 file in the subdirectory specified by the value of the `$PsUICulture` variable. If the command finds the `.psd1` file, it saves the translated strings from the file in the value of the same `$UserMessages` variable, overwriting the hashtable saved by the DATA section logic.

The third command displays the first message in the `$UserMessages` variable.

If the `Import-LocalizedData` command finds a `.psd1` file for the `$PsUICulture` language, the value of the `$UserMessages` variable contains the translated text strings. If the command fails for any reason, the command displays the default text strings defined in the DATA section of the script.

Example 6: Suppress error messages if the UI culture isn't found

```
PS C:\> # In Day1.ps1

Import-LocalizedData -BindingVariable "Day"
```

```
# In Day2.ps1

Import-LocalizedData -BindingVariable "Day" -ErrorAction:SilentlyContinue

PS C:\> .\Day1.ps1
Import-LocalizedData : Can't find PowerShell data file 'Day1.psd1' in
directory 'C:\ps-test\fr-BE\'
or any parent culture directories.
At C:\ps-test\Day1.ps1:17 char:21+ Import-LocalizedData <<<<  Day
Today is Tuesday

PS C:\> .\Day2.ps1
Today is Tuesday
```

You can use the ErrorAction common parameter with a value of SilentlyContinue to suppress the error message. This is especially useful when you have provided user messages in a default or fallback language, and no error message is needed.

This example compares two scripts, `Day1.ps1` and Day2.ps1, that include an `Import-LocalizedData` command. The scripts are identical, except that Day2 uses the ErrorAction common parameter with a value of `SilentlyContinue`.

The sample output shows the results of running both scripts when the UI culture is set to `fr-BE` and there are no matching files or directories for that UI culture. `Day1.ps1` displays an error message and English output. `Day2.ps1` just displays the English output.

REMARKS

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