PowerShell

FPDF Library
PDF generator

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

PowerShell Get-Help on command 'Start-Process'

PS C:\Users\wahid> Get-Help Start-Process

NAME

Start-Process

SYNOPSIS

Starts one or more processes on the local computer.

SYNTAX

Start-Process [-FilePath] <System.String> [[-ArgumentList] <System.String[]>]
[-Credential <System.Management.Automation.PSCredential>] [-LoadUserProfile]
[-NoNewWindow] [-PassThru] [-RedirectStandardError <System.String>]
[-RedirectStandardInput <System.String>] [-RedirectStandardOutput
<System.String>] [-UseNewEnvironment] [-Wait] [-WindowStyle {Normal | Hidden |
Minimized | Maximized}] [-WorkingDirectory <System.String>]
[<CommonParameters>]

Start-Process [-FilePath] <System.String> [[-ArgumentList] <System.String[]>]
[-PassThru] [-Verb <System.String>] [-Wait] [-WindowStyle {Normal | Hidden |
Minimized | Maximized}] [-WorkingDirectory <System.String>]
[<CommonParameters>]

DESCRIPTION

The `Start-Process` cmdlet starts one or more processes on the local computer. By default, `Start-Process` creates a new process that inherits all the environment variables that are defined in the current process.

To specify the program that runs in the process, enter an executable file or script file, or a file that can be opened using a program on the computer. If you specify a non-executable file, `Start-Process` starts the program that's associated with the file, similar to the `Invoke-Item` cmdlet.

You can use the parameters of `Start-Process` to specify options, such as loading a user profile, starting the process in a new window, or using alternate credentials.

PARAMETERS

-ArgumentList <System.String[]>

Specifies parameters or parameter values to use when this cmdlet starts the process. Arguments can be accepted as a single string with the arguments separated by spaces, or as an array of strings separated by commas. The cmdlet joins the array into a single string with each element of the array separated by a single space.

The outer quotes of the PowerShell strings aren't included when the ArgumentList values are passed to the new process. If parameters or parameter values contain a space or quotes, they need to be surrounded with escaped double quotes. For more information, see about_Quoting_Rules (../Microsoft.PowerShell.Core/About/about_Quoting_Rules.md).

For the best results, use a single ArgumentList value containing all the arguments and any needed quote characters.

-Credential <System.Management.Automation.PSCredential>

Specifies a user account that has permission to perform this action. By default, the cmdlet uses the credentials of the current user.

Type a user name, such as User01 or Domain01\User01 , or enter a PSCredential object generated by the `Get-Credential` cmdlet. If you type a user name, you're prompted to enter the password.

Credentials are stored in a PSCredential (/dotnet/api/system.management.automation.pscredential)object and the password is stored as a SecureString (/dotnet/api/system.security.securestring).

> [!NOTE] > For more information about SecureString data protection, see > How secure is SecureString? (/dotnet/api/system.security.securestring#how-secure-is-securestring).

-FilePath <System.String>

Specifies the optional path and filename of the program that runs in the process. Enter the name of an executable file or of a document, such as a `.txt` or `.doc` file, that's associated with a program on the computer. This parameter is required.

If you specify only a filename, use the WorkingDirectory parameter to specify the path.

-LoadUserProfile <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet loads the Windows user profile stored in the `HKEY_USERS` registry key for the current user.

This parameter doesn't affect the PowerShell profiles. For more information, see about_Profiles (../Microsoft.PowerShell.Core/About/about_Profiles.md).

-NoNewWindow <System.Management.Automation.SwitchParameter>

Start the new process in the current console window. By default on Windows, PowerShell opens a new window.

You can't use the NoNewWindow and WindowStyle parameters in the same command.

-PassThru <System.Management.Automation.SwitchParameter>

Returns a process object for each process that the cmdlet started. By default, this cmdlet doesn't generate any output.

-RedirectStandardError <System.String>

Specifies a file. This cmdlet sends any errors generated by the process to a file that you specify. Enter the path and filename. By default, the errors are displayed in the console.

-RedirectStandardInput <System.String>

Specifies a file. This cmdlet reads input from the specified file. Enter the path and filename of the input file. By default, the process gets its input from the keyboard.

-RedirectStandardOutput <System.String>

Specifies a file. This cmdlet sends the output generated by the process to a file that you specify. Enter the path and filename. By default, the output is displayed in the console.

-UseNewEnvironment <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet uses new environment variables specified for the process. By default, the started process runs with the environment variables inherited from the parent process.

-Verb <System.String>

Specifies a verb to use when this cmdlet starts the process. The verbs that are available are determined by the filename extension of the file that runs in the process.

The following table shows the verbs for some common process file types.

| File type | Verbs | | --- | --- | | .cmd | `Edit`, `Open`, `Print`, `RunAs`, `RunAsUser` | | .exe | `Open`, `RunAs`, `RunAsUser` | | .txt | `Open`, `Print`, `PrintTo` | | .wav | `Open`, `Play` |

To find the verbs that can be used with the file that runs in a process, use the `New-Object` cmdlet to create a System.Diagnostics.ProcessStartInfo object for the file. The available verbs are in the Verbs property of the ProcessStartInfo object. For details, see the examples.

-Wait <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet waits for the specified process and its descendants to complete before accepting more input. This parameter suppresses the command prompt or retains the window until the processes finish.

-WindowStyle <System.Diagnostics.ProcessWindowStyle>

Specifies the state of the window that's used for the new process. The default value is `Normal`. The acceptable values for this parameter are:

- `Normal`

- `Hidden`

- `Minimized`

- `Maximized`

You can't use the WindowStyle and NoNewWindow parameters in the same command.

-WorkingDirectory <System.String>
Specifies the location that the new process should start in. The default is the location of the executable file or document being started.
Wildcards aren't supported. The path must not contain characters that would be interpreted as wildcards.

<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: Start a process that uses default values -----

```
Start-Process -FilePath "sort.exe"
```

---------- Example 2: Print a text file ----------

```
Start-Process -FilePath "myfile.txt" -WorkingDirectory "C:\PS-Test" -Verb Print
```

---- Example 3: Start a process to sort items to a new file ----

```
$processOptions = @{
    FilePath = "sort.exe"
    RedirectStandardInput = "TestSort.txt"
    RedirectStandardOutput = "Sorted.txt"
    RedirectStandardError = "SortError.txt"
    UseNewEnvironment = $true
}
Start-Process @processOptions
```

This example uses splatting to pass parameters to the cmdlet. For more
information, see about_Splatting
(../microsoft.powershell.core/about/about_splatting.md).

------- Example 4: Start a process in a maximized window -------

```
Start-Process -FilePath "notepad" -Wait -WindowStyle Maximized
```

------- Example 5: Start PowerShell as an administrator -------

```
Start-Process -FilePath "powershell" -Verb RunAs
```

----- Example 6: Using different verbs to start a process -----

```
$startExe = New-Object System.Diagnostics.ProcessStartInfo -Args powershell.exe
$startExe.verbs
```

```
open
runas
runasuser
```

The example uses `New-Object` to create a System.Diagnostics.ProcessStartInfo
object for `powershell.exe`, the file that runs in the PowerShell process. The

Verbs property of the ProcessStartInfo object shows that you can use the Open and `RunAs` verbs with `powershell.exe`, or with any process that runs a `.exe` file.

-------- Example 7: Specifying arguments to the process --------

Start-Process -FilePath "$env:comspec" -ArgumentList "/c dir `"%SystemDrive%\Program Files`""

Start-Process -FilePath "$env:comspec" -ArgumentList "/c","dir","`"%SystemDrive%\Program Files`""

REMARKS

To see the examples, type: "get-help Start-Process -examples".

For more information, type: "get-help Start-Process -detailed".

For technical information, type: "get-help Start-Process -full".

For online help, type: "get-help Start-Process -online"