Start-Service

Module: Microsoft.PowerShell.Management

Starts one or more stopped services.

In this article

PowerShell Copy

```
Start-Service
     [-InputObject] <ServiceController[]>
     [-PassThru]
     [-Include <String[]>]
     [-Exclude <String[]>]
     [-WhatIf]
     [-Confirm]
     [<CommonParameters>]
```

PowerShell Copy

```
Start-Service
     [-Name] <String[]>
     [-PassThru]
     [-Include <String[]>]
     [-Exclude <String[]>]
     [-WhatIf]
     [-Confirm]
     [<CommonParameters>]
```

```powershell
Start-Service
      [-PassThru]
      -DisplayName <String[]>
      [-Include <String[]>]
      [-Exclude <String[]>]
      [-WhatIf]
      [-Confirm]
      [<CommonParameters>]
```

Description

The **Start-Service** cmdlet sends a start message to the Windows Service Controller for each of the specified services. If a service is already running, the message is ignored without error. You can specify the services by their service names or display names, or you can use the *InputObject* parameter to supply a service object that represents the services that you want to start.

Examples

Example 1: Start a service by using its name

```powershell
PS C:\> Start-Service -Name "eventlog"
```

This command starts the EventLog service on the local computer. It uses the *Name* parameter to identify the service by its service name.

Example 2: Display information without starting a service

```powershell
PS C:\> Start-Service -DisplayName *remote* -WhatIf
```

This command tells what would occur if you started the services that have a display name that includes remote. It uses the *DisplayName* parameter to specify the services by their display name instead of by their service name. And, the command uses the *WhatIf* parameter. That parameter means that this command displays what would occur if you run the command without making changes.

Example 3: Start a service and record the action in a text file

PowerShell

```
PS C:\> $s = Get-Service wmi
PS C:\> Start-Service -InputObject $s -PassThru | Format-List >> services.txt
```

These commands start the Windows Management Instrumentation (WMI) service on the computer and add a record of the action to the services.txt file. The first command uses **Get-Service** to get an object that represent the WMI service and store it in the $s variable.

The second command starts the WMI service. It identifies the service by using the *InputObject* parameter to pass the $s variable that contains the WMI service object to **Start-Service**. Then, it uses *PassThru* to create an object that represents the starting of the service. Without *PassThru*, **Start-Service** does not create any output.

The pipeline operator (|) passes the object that **Start-Service** creates to the Format-List cmdlet, which formats the object as a list of its properties. The append redirection operator (>>) redirects the output to the services.txt file, where it is added to the end of the existing file.

Example 4: Start a disabled service

PowerShell Copy

```
PS C:\> Start-Service tlntsvr
Start-Service : Service 'Telnet (TlntSvr)' cannot be started due to the    following error:
Cannot start service TlntSvr on computer '.'.
At line:1 char:14
+ start-service  <<<< tlntsvr PS C:\> Get-WMIObject win32_service | Where-Object {$_.Name -
eq "tlntsvr"}
ExitCode  : 0
Name      : TlntSvr
ProcessId : 0
StartMode : Disabled
State     : Stopped
Status    : OK PS C:\> Set-Service tlntsvr -StartupType manual PS C:\> start-service
tlntsvr
```

This series of commands shows how to start a service when the start type of the service is Disabled. The first command, which attempts to start the Telnet service (tlntsvr), fails.

The second command uses **Get-WmiObject** to get the Tlntsvr service. This command retrieves an object that has the start type property in the **StartMode** field. The resulting display reveals that the start type of the Tlntsvr service is Disabled.

The next command uses **Set-Service** to change the start type of the Tlntsvr service to "Manual".

Now, we can resubmit the **Start-Service** command. This time, the command succeeds.

To verify that the command succeeded, run **Get-Service**.

Required Parameters

`-DisplayName`

Specifies the display names of the services to start. Wildcard characters are permitted.

Type:
String[]

Position:
Named

Default value:
None

Accept pipeline input:
False

Accept wildcard characters:
False

`-InputObject`

Specifies **ServiceController** objects representing the services to be started. Enter a variable that contains the objects, or type a command or expression that gets the objects.

Type:
ServiceController[]

Position:
0

Default value:
None

Accept pipeline input:
True (ByValue)

Accept wildcard characters:
False

`-Name`

Specifies the service names for the service to be started.

The parameter name is optional. You can use *Name* or its alias, *ServiceName*, or you can omit the parameter name.

Type:
String[]

Aliases:
ServiceName

Position:
0

Default value:
None

Accept pipeline input:
True (ByPropertyName, ByValue)

Accept wildcard characters:
False

Optional Parameters

`-Confirm`

Prompts you for confirmation before running the cmdlet.

Type:
SwitchParameter

Aliases:
cf

Position:
Named

Default value:
False

Accept pipeline input:
False

Accept wildcard characters:
False

`-Exclude`

Specifies services that this cmdlet omits. The value of this parameter qualifies the *Name* parameter. Enter a name element or pattern, such as "s*". Wildcard characters are permitted.

Type:
String[]

Position:
Named

Default value:
None

Accept pipeline input:
False

Accept wildcard characters:
False

`-Include`

Specifies services that this cmdlet starts. The value of this parameter qualifies the *Name* parameter. Enter a name element or pattern, such as "s*". Wildcard characters are permitted.

Type:
String[]

Position:
Named

Default value:
None

Accept pipeline input:
False

Accept wildcard characters:
False

`-PassThru`

Returns an object that represents the service. By default, this cmdlet does not generate any output.

Type:
SwitchParameter

Position:
Named

Default value:
None

Accept pipeline input:
False

Accept wildcard characters:
False

`-WhatIf`

Shows what would happen if the cmdlet runs. The cmdlet is not run.

Type:
SwitchParameter

Aliases:
wi

Position:
Named

Default value:
False

Accept pipeline input:
False

Accept wildcard characters:
False

Inputs

System.ServiceProcess.ServiceController, System.String

You can pipe objects that represent the services or strings that contain the service names to this cmdlet.

Outputs

None, System.ServiceProcess.ServiceController

This cmdlet generates a **System.ServiceProcess.ServiceController** object that represents the service, if you specify *PassThru*. Otherwise, this cmdlet does not generate any output.

Notes

- You can also refer to **Start-Service** by its built-in alias, **sasv**. For more information, see about_Aliases.
- **Start-Service** can control services only if the current user has permission to do this. If a command does not work correctly, you might not have the required permissions.
- To find the service names and display names of the services on your system, type `Get-Service`. The service names appear in the **Name** column, and the display names appear in the **DisplayName** column.
- You can start only the services that have a start type of Manual or Automatic. You cannot start the services that have a start type of Disabled. If a **Start-Service** command fails with the message `Cannot start service \<service-name\> on computer`, use Get-WmiObject to find the start type of the service and, if you have to, use the Set-Service cmdlet to change the start type of the service.
- Some services, such as Performance Logs and Alerts (SysmonLog) stop automatically if they have no work to do. When Windows PowerShell starts a service that stops itself almost immediately, it displays the following message: `Service \<display-name\> start failed.`

Related Links

- Get-Service
- New-Service
- Restart-Service
- Resume-Service
- Set-Service
- Stop-Service
- Suspend-Service