Windows 11 and Server 2022 Services

Configuration and Troubleshooting

Lowell Vanderpool & Nathan Vanderpool
TECH SAVVY PRODUCTIONS

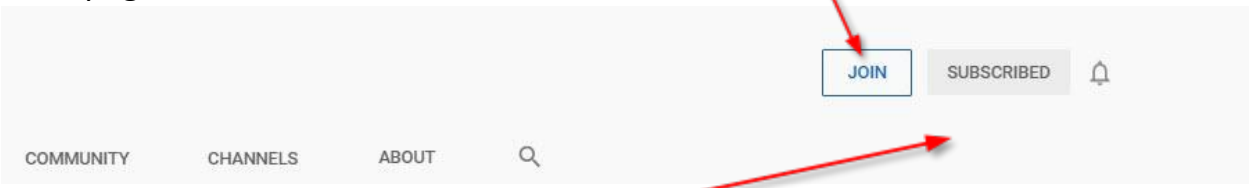
CONTACT US:

mrvanderpool@techsavvyproductions.com

nathan@techsavvyproductions.com

SUPPORT US:

If you would like to support the channel, Join our channel membership, it's $2.99/month (less than a Starbucks coffee); see the "Join" button on our channel homepage.



OR

Subscribe to the channel as it helps our channel perform better on YouTube's algorithm.

SOCIAL MEDIA AND WEBSITE:

Check out our **Website**: https://www.techsavvyproductions.com

Follow us on **Twitter:** @_TechSavvyTeam

Like us on **Facebook:**
https://www.facebook.com/Tech-Savvy-Productions-105287381500897

Mr.V **Linkedin**: https://www.linkedin.com/in/lowell-vanderpool-57970623/

Nathan **Linkedin**: _https://www.linkedin.com/in/nathan-vanderpool-50a27822/

Follow on **Instagram**: techsavvyproductions
https://www.instagram.com/techsavvyproductions/

We translate subtitles on our videos into the following languages: عربى, български, 简

Tech Savvy Productions

体中文), 中國傳統的）, Nederlands, Suomalainen, Pilipino, français, Deutsche, हिंदी, Magyar, bahasa Indonesia, 日本語, 한국어, norsk, Polskie, português, Română, русский, Española, Kiswahili, Svenska, and Tiếng Việt

Social media logos and “Tech Savvy Productions” teaser created by The 11th Hour: https://www.youtube.com/user/The11thH0UR/featured

Understanding and Managing Windows Services

How to disable all third-party services in one click on Windows 11

How to start and stop services manually on Windows 10

How to Start, Stop, Restart, Enable, and Disable Services in Windows 10

[PowerShell Commands for Services]

[PsService v2.25]

[Fix Windows Service will not start]

[Monitoring Services with PowerShell]

[Managing Windows Services via PowerShell]

[Troubleshooting with Windows Logs]

[Easily find out who started, stopped or updated your Windows Services with Windows Service Auditor]

[Essential Tools for Windows Services: SC.EXE]

[List of Windows 10 Services]

[Windows 10 Service Defaults]

[Hands off my Windows Services: Learn how to harden]

[Empirically Assessing Windows Service Hardening]

[Understanding Windows Service Hardening]

[How to Run Any Program as a Background Service in Windows]

Understanding and Managing Windows Services

https://www.howtogeek.com/school/using-windows-admin-tools-like-a-pro/lesson8/

author: LOWELL HEDDINGS

In today's Geek School lesson, we're going to teach you about Windows Services and how to manage them using the built-in utilities.

Over the years, people have spent a lot of time disabling and tweaking the configuration of Windows Services, and entire web sites have been devoted to understanding which services you can disable.

Thankfully modern versions of Windows have greatly streamlined the things that run as services, added the ability to delay them from starting until later, and allowed them to run only when triggered rather than all the time. The overall footprint of Windows has even decreased due to all this work.

But people still are determined to disable services. So today's lesson is going to teach you about services, how to analyze them, remove them, or disable them. What we're not going to do is give you an exact list of services to disable, because for the most part, you should leave the built-in services alone.

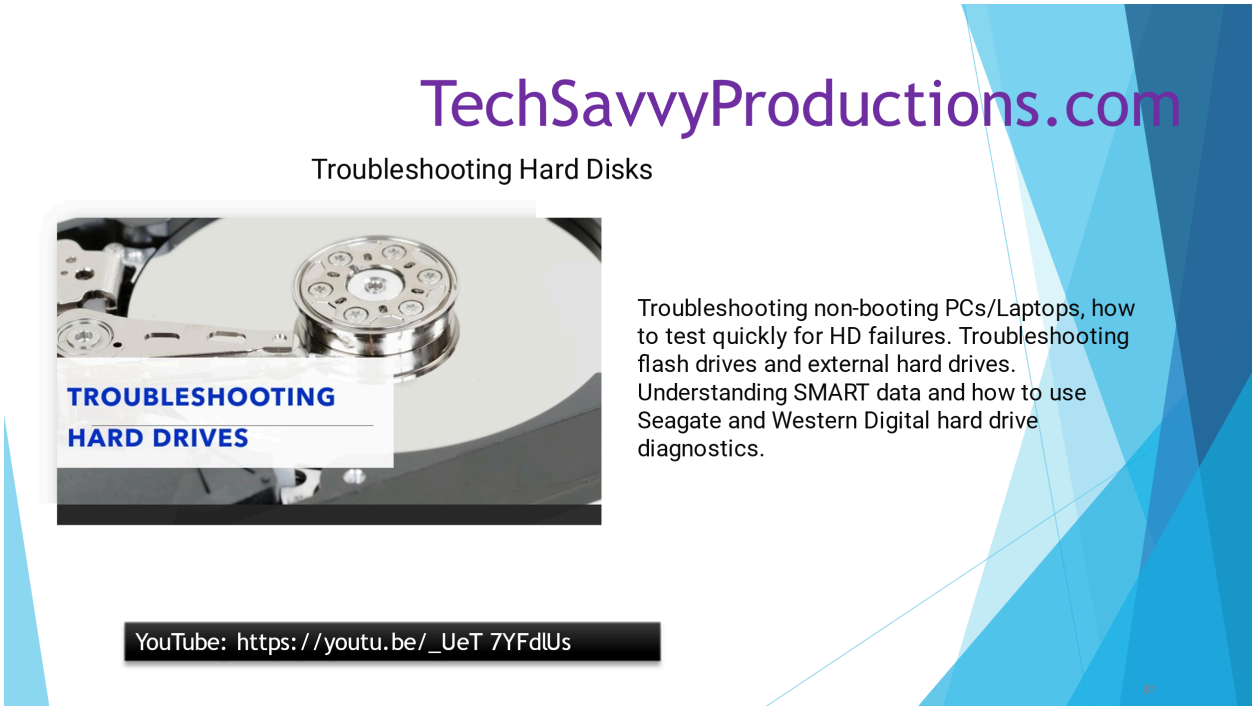
What Are Services Exactly?

Windows services are a special type of application that is configured to launch and run in the background, in some cases before the user has even logged in. They can be configured to run as the local system account. Services are designed to run continuously in the background and perform system tasks, like backing up your computer or running a server process that listens on a network port.

Back in the Windows XP days, services could be configured to run interactively and run alongside the rest of your applications, but since Vista, all services are forced to run in a special window session that can't interact with your local desktop. So a service that tries to open a dialog box or show you a message won't be allowed to do so.

Unlike regular applications, which can be simply launched and run under your user account, a service must be installed and registered with Windows, which requires an administrator account, and usually a User Account Control prompt

before that happens. So if you don't allow an application to run as administrator, it cannot just create a service to run in the background.

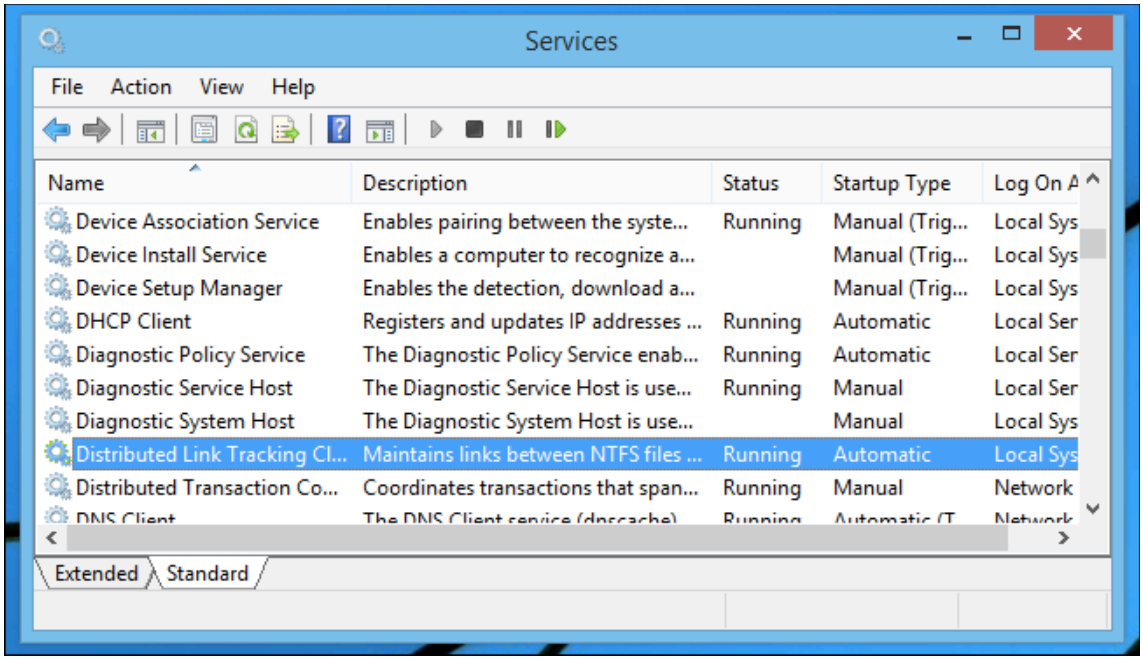


https://youtu.be/_UeT7YFdlUs

The Services Panel

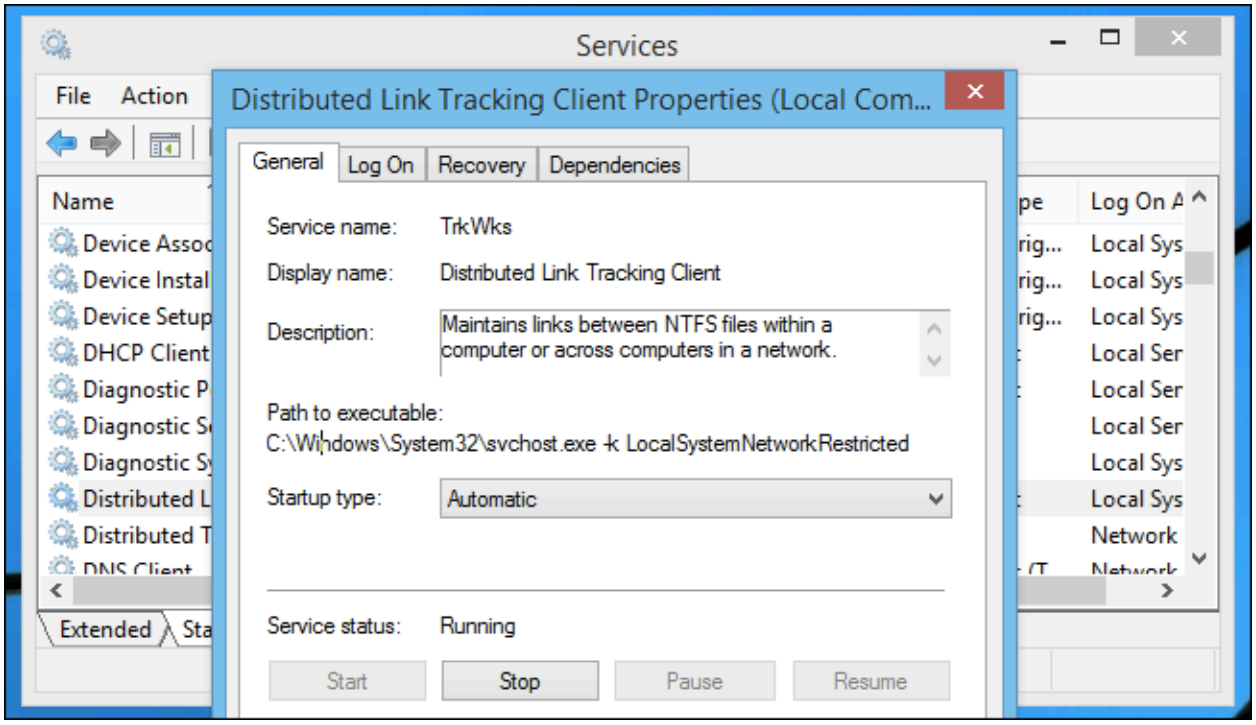
Windows has always used the Services panel as a way to manage the services that are running on your computer. You can easily get there at any point by simply hitting WIN + R on your keyboard to open the Run dialog, and typing in *services.msc*.

The Services panel is fairly simple: there are a list of services, a status column to show whether it is running or not, and more information like name, description, and the startup type of the service. You'll notice that not every service is running all the time.



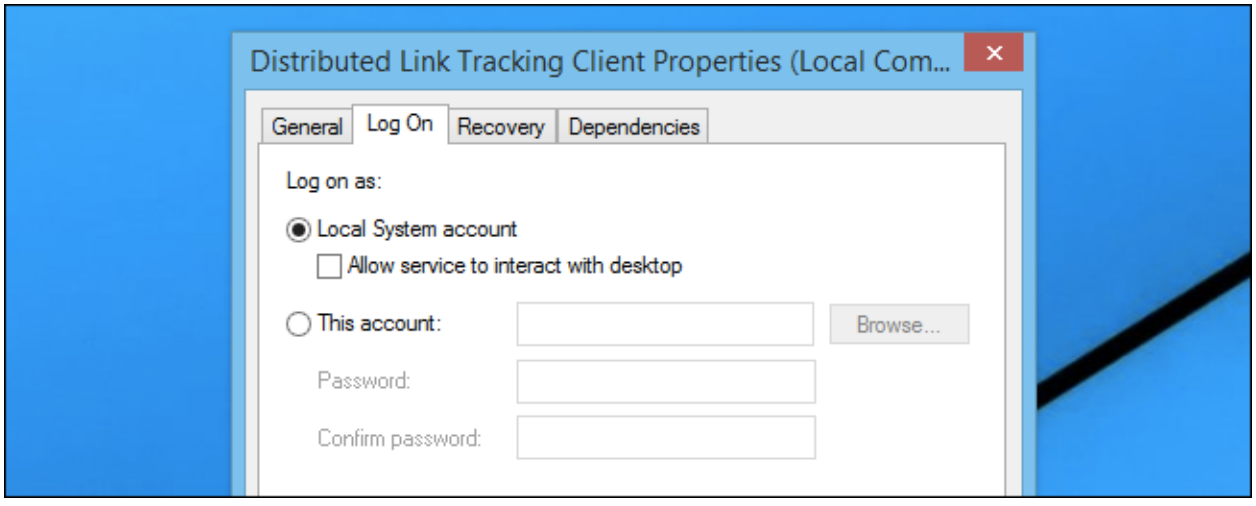
While you can select a service and either right-click it or click the toolbar buttons to start, stop, or restart it, you can also double-click to open up the properties view and get more information.

Disabling the service is as simple as changing the Startup type drop-down to disabled and choosing Apply, although you can also change it to Manual or automatic with a delayed start. From this dialog you can see the full path to the executable as well, which can help in many cases when you want to see what exactly the service is running.



The **Log On tab** allows you to choose whether the service is logged on as the local system account or under another account. This is mostly useful in a server environment where you might want to run a service account from the domain that has access to resources on other servers.

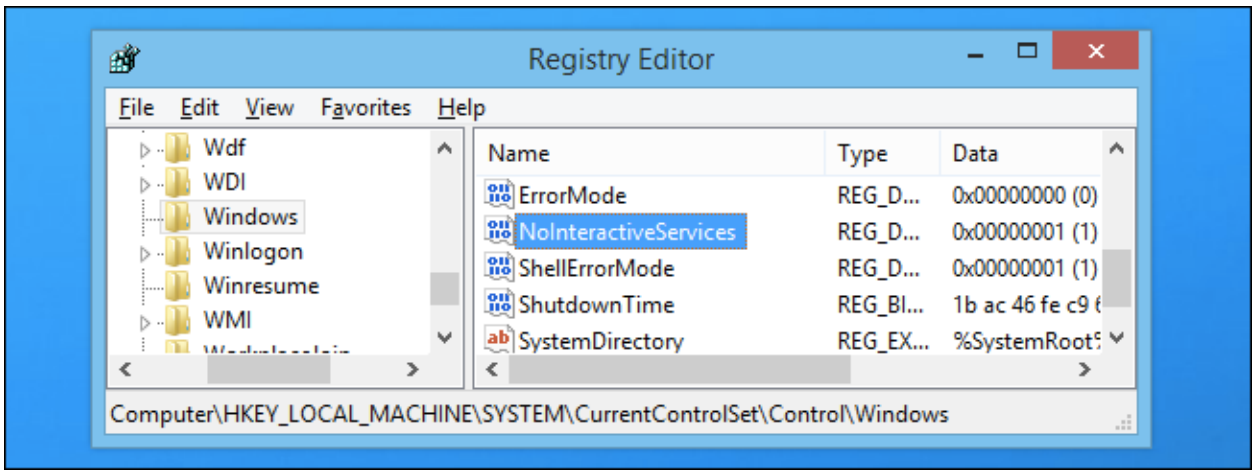
https://youtu.be/Hon2XIJ2lNk



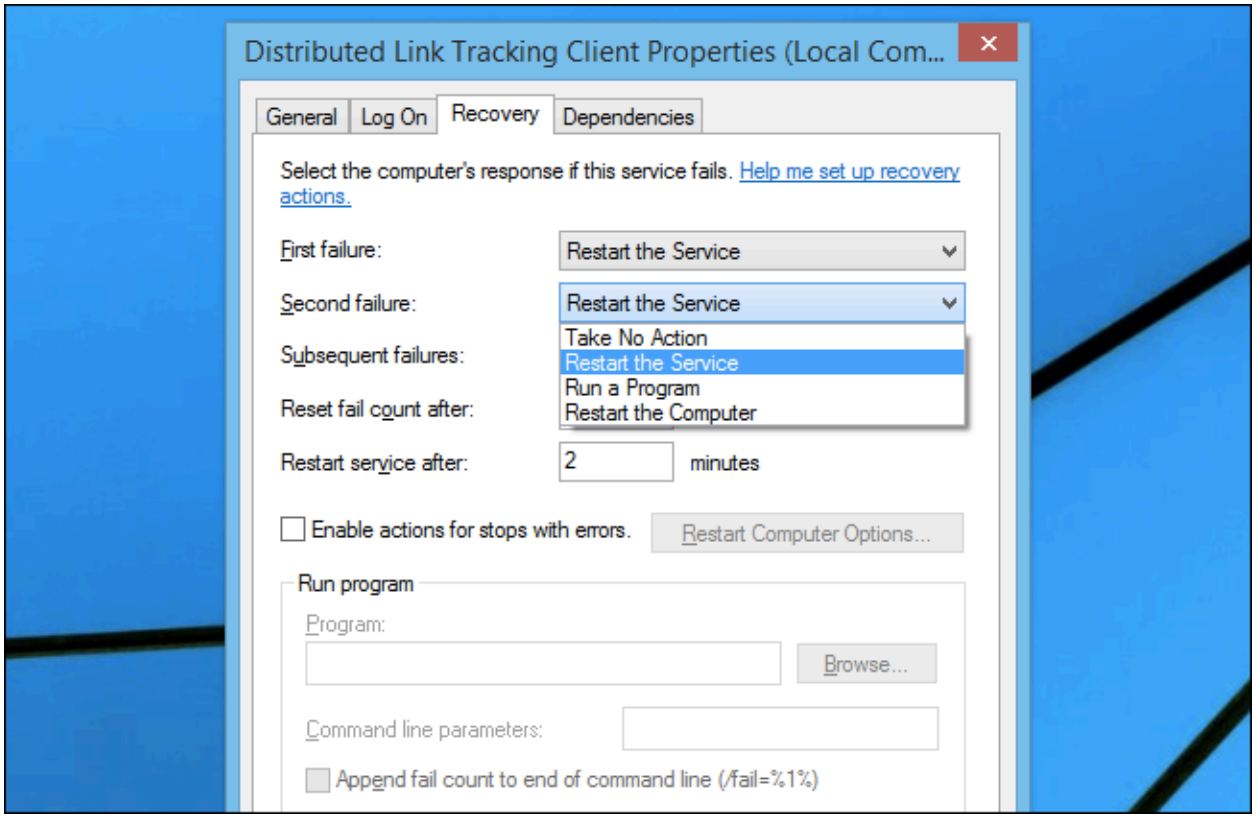
You might notice the option for "Allow service to interact with desktop", which we mentioned earlier – by default, services are not allowed to access your desktop unless this box is checked, and this checkbox is really only there for legacy support.

But just checking that box doesn't immediately give them access – you would also need to make sure that the NoInteractiveServices value in the registry is set to 0.

because when it is set to 1, that checkbox is ignored and services can't interact with the desktop at all. *Note:* in Windows 8, the value is set to 1, and interactive services are prohibited.

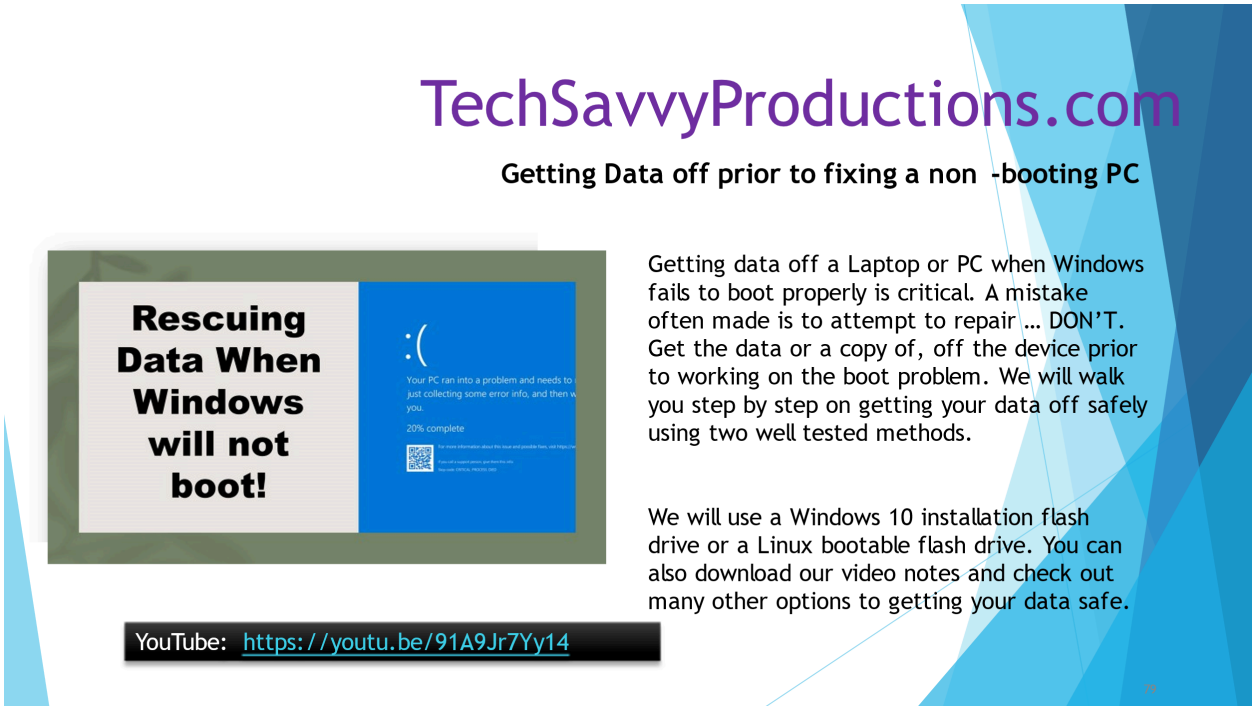


Services aren't supposed to be interactive because all windows exist in the same user terminal with access to common resources like the clipboard, and if they are running along with other processes there could be an issue where a malicious application running in a normal user process could attempt to gain more access through a service, and considering that services run as the local system account, that probably isn't a good thing.

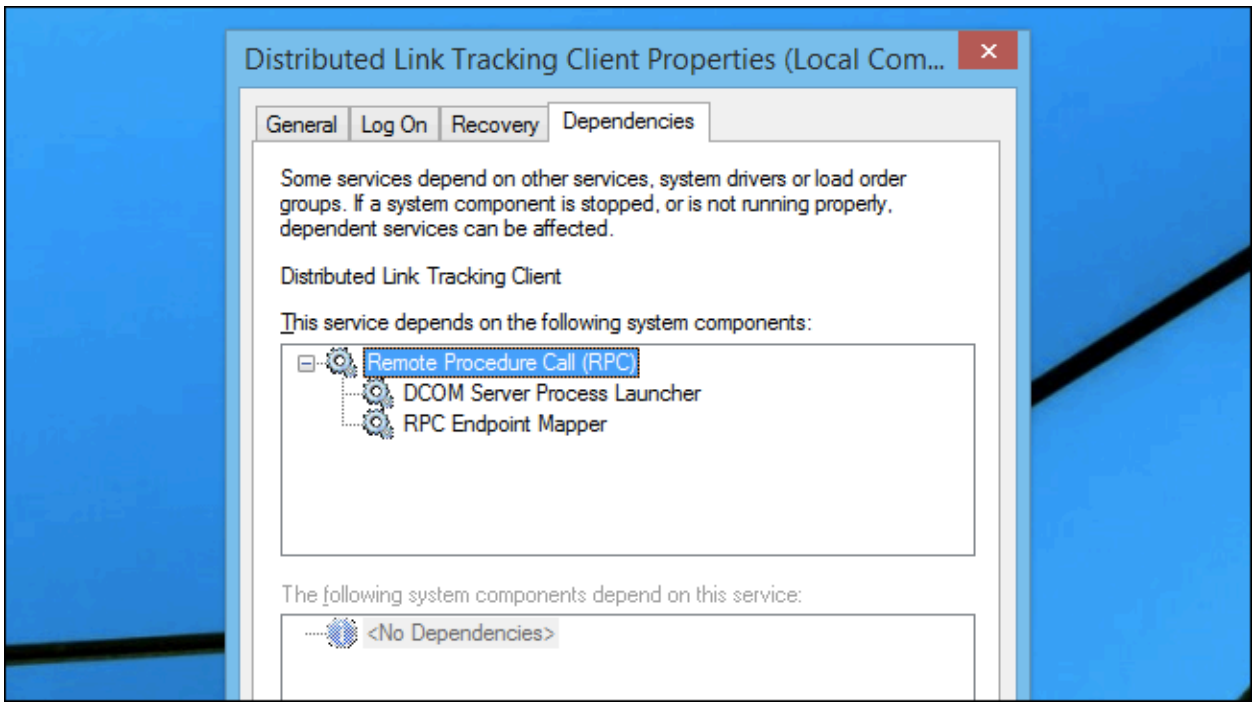


The **Recovery tab** allows you to choose options for what happens when the service fails. You can choose to automatically restart the service, which is generally the default option, or you can run a program or restart the computer.

The Run a program option is probably the most useful, since you could set Windows to automatically send out an email if the service fails more than once – a helpful option in a server environment. It's definitely much less helpful on a regular desktop.



https://youtu.be/91A9Jr7Yy14



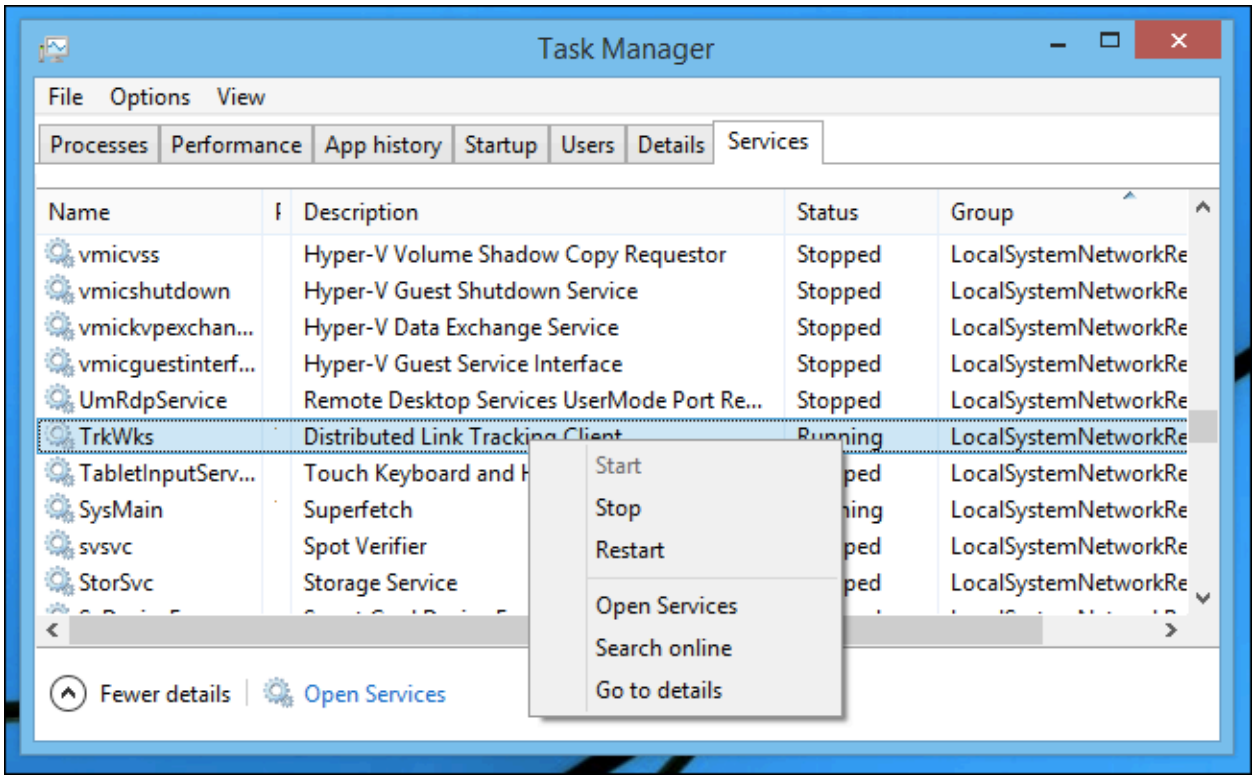
The **dependencies tab** shows which services depend on a particular service, and which services depend on the one you are looking at. If you are planning on

disabling a service, you should probably consult this section first to make sure nothing else requires that service.

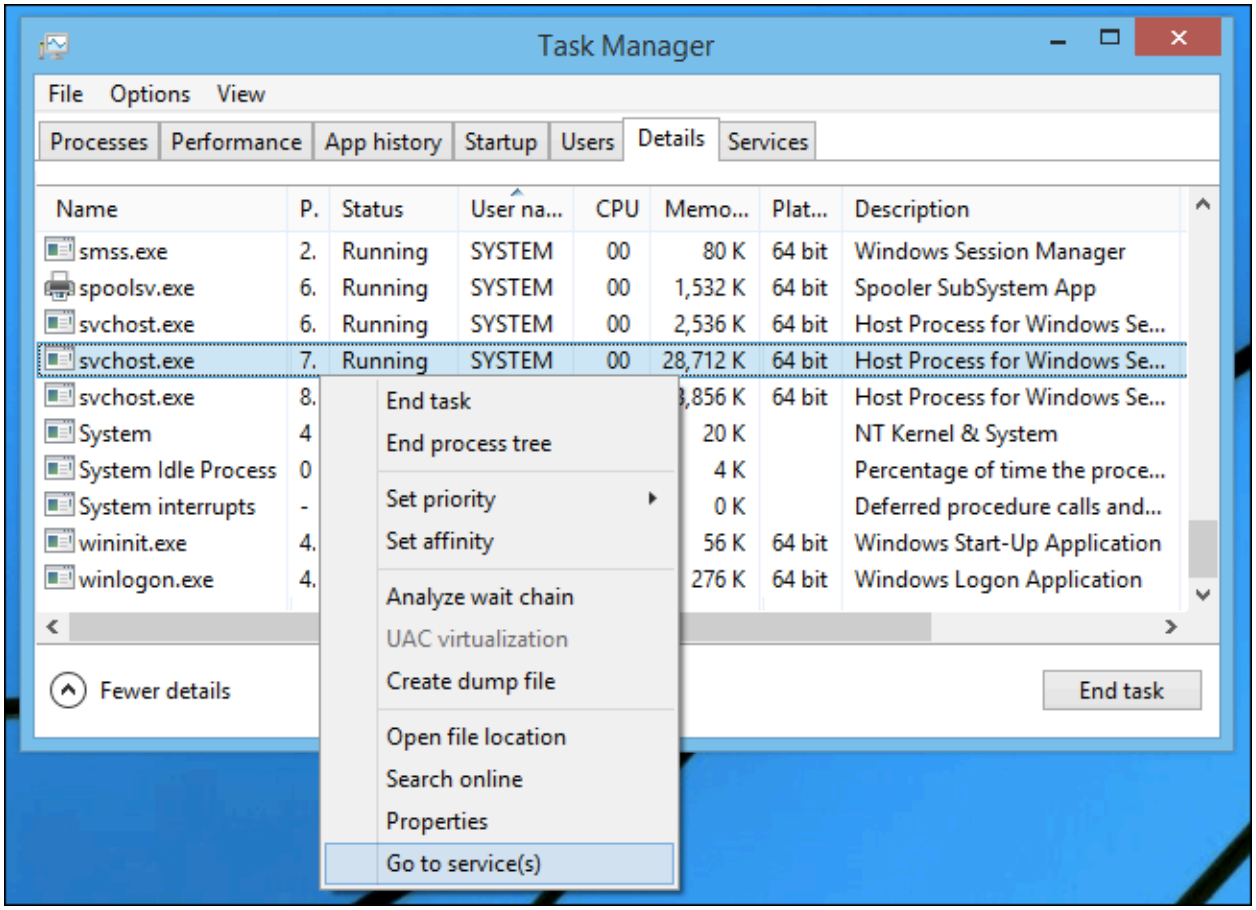
Looking at Services in Task Manager for Windows 8.x

The regular services panel hasn't changed much in years, but thankfully there is a much better way to look at which services are running, and which of those services are using a lot of resources.

Task manager in Windows 8 has a new Services tab, which allows you to stop and start services, but also comes with a "Search online" option, and even more useful, the "Go to details" option.



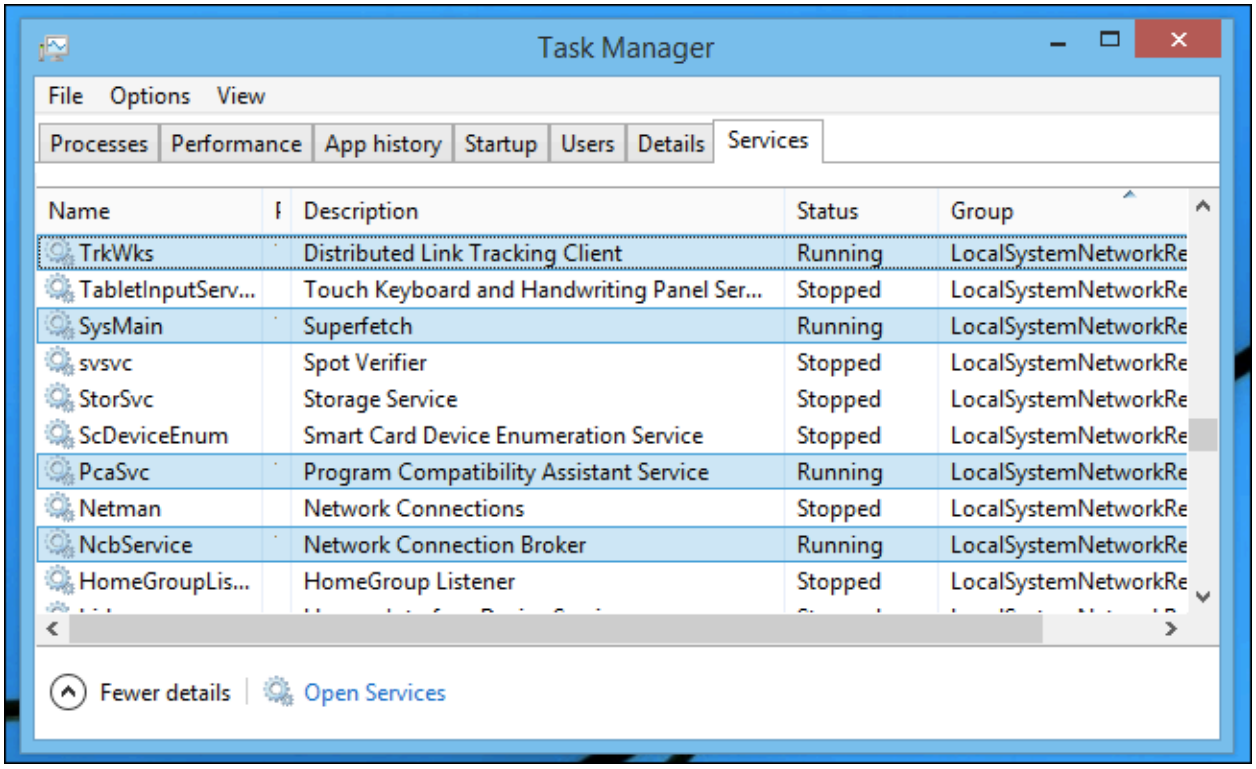
Once you've selected Go to details from the menu, you'll be switched over to the Details tab, and the process that is responsible for that service will be highlighted.



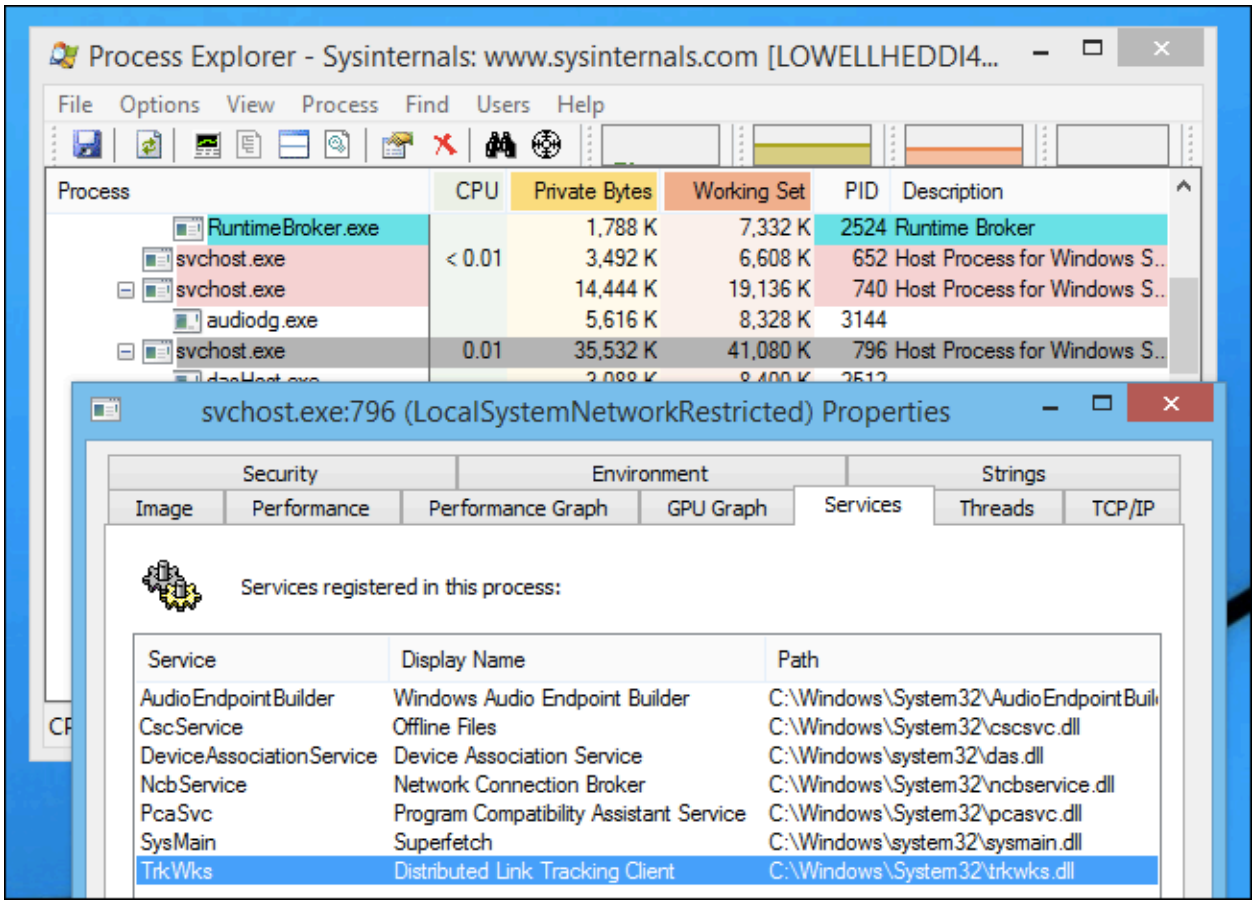
As you can see, the process responsible for the Distributed Link Tracking is taking up 28,712 K of memory, which seems like a lot, until you realize that the particular svchost.exe process is actually responsible for a whole bunch of services.

Right-click it again, and then select Go to Services, and you'll see what we're talking about. Now many services are selected in the Services window, and you'll notice they are all in the LocalSystemNetworkRestricted group, and they are all currently running.

So that 28 MB of memory is actually being used for a whole set of services, which makes it more understandable why it is using all that memory.

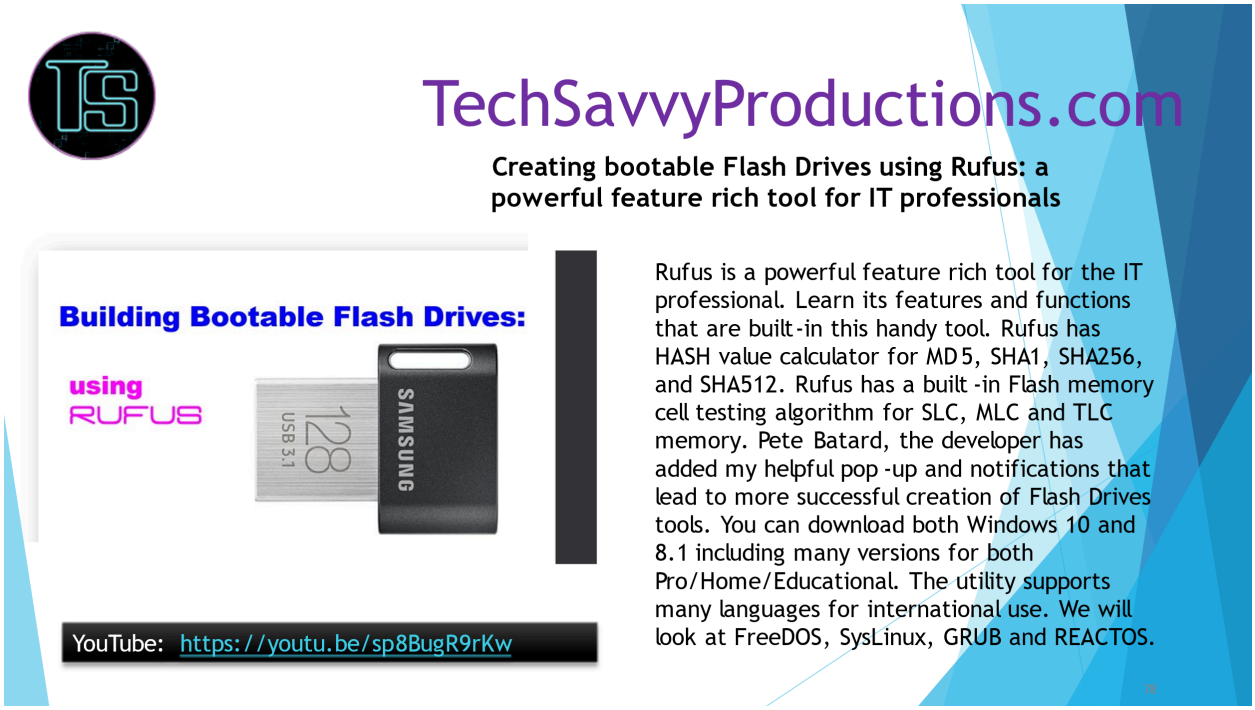


Using Process Explorer to Look at Services



If you want a much clearer view of what services are running under each process, your best bet is to pull out Process Explorer, find the service in the list, double-click it, and then go to the Services tab. This method works on any version of Windows.

Hint: in Process Explorer all the services should be in the tree underneath services.exe.



https://youtu.be/sp8BugR9rKw

Should You Disable Services?

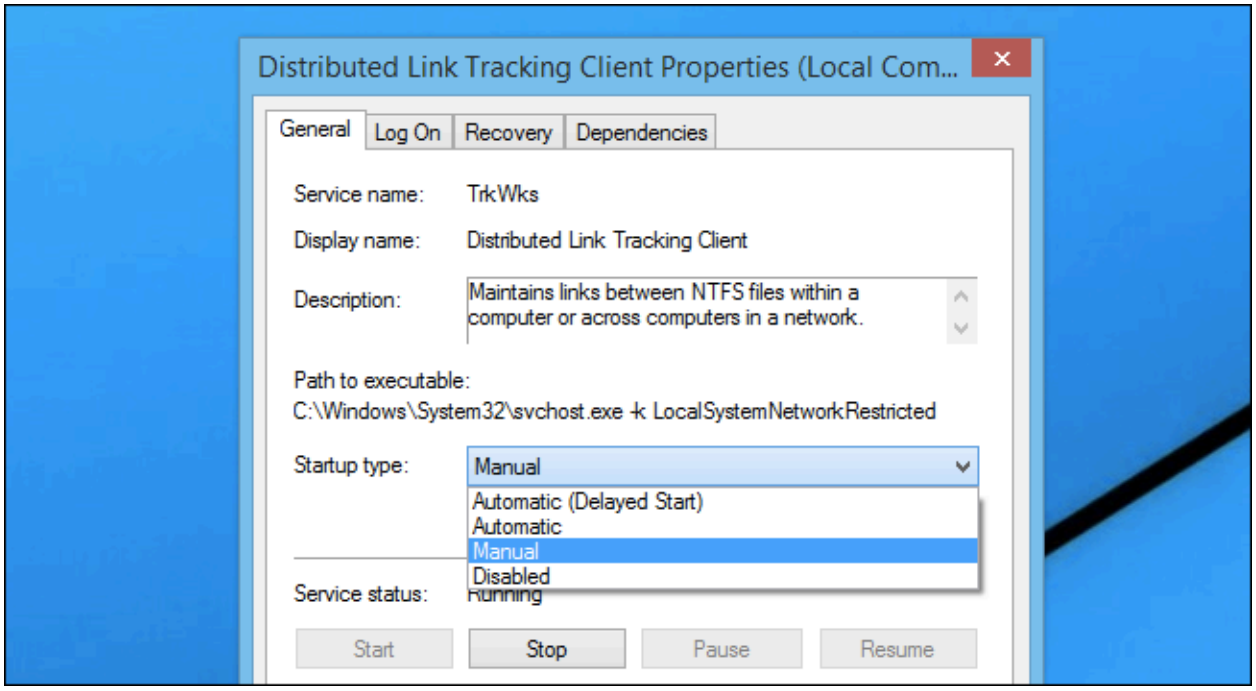
Unfortunately, many crapware applications install Windows Services during their installation process, and use them to keep their nonsense running in the background and re-launching repeatedly. Other applications implement a Windows Service to provide functionality that you might not need. These are the services that you should disable.

Our general rule is that Microsoft's built-in Windows services should be left alone – Windows 8 or even Windows 7 has done a good job of cutting down the services to just really important functionality, and you won't gain much in the way of resources by disabling those services.

What you should definitely do, however, is look for any services that are not part of Windows, and try to deal with them instead. If you don't have any idea what the service is, or it is for an application that you don't want running all the time, you should do some research and decide whether to disable it.

Don’t Disable, Set to Manual

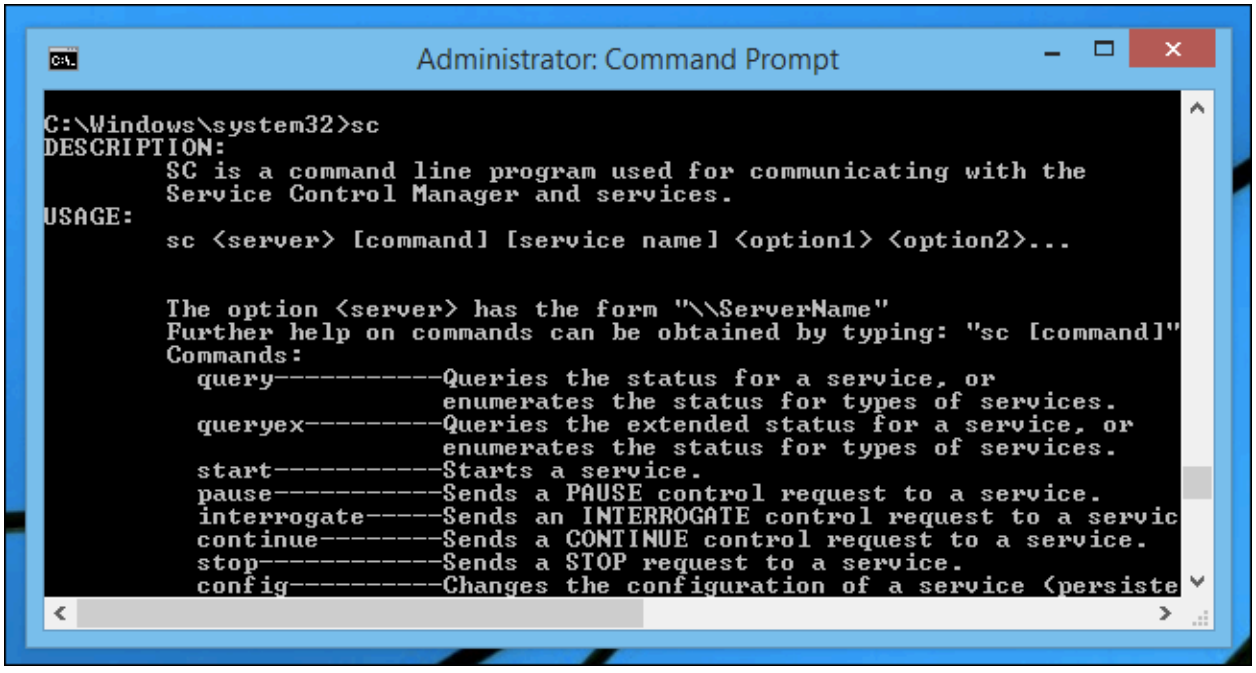
One of the rules that we like to follow is to avoid disabling services, since that can cause problems and errors. Instead, just try setting the service to Manual start.



If you find that a particular service needs to be running, but maybe doesn’t need to be running immediately, you can also change it to Automatic (Delayed Start) instead, which will delay starting until the system calms down after boot.

Administering Services from the Command Prompt

https://youtu.be/aceGQ36aDsA

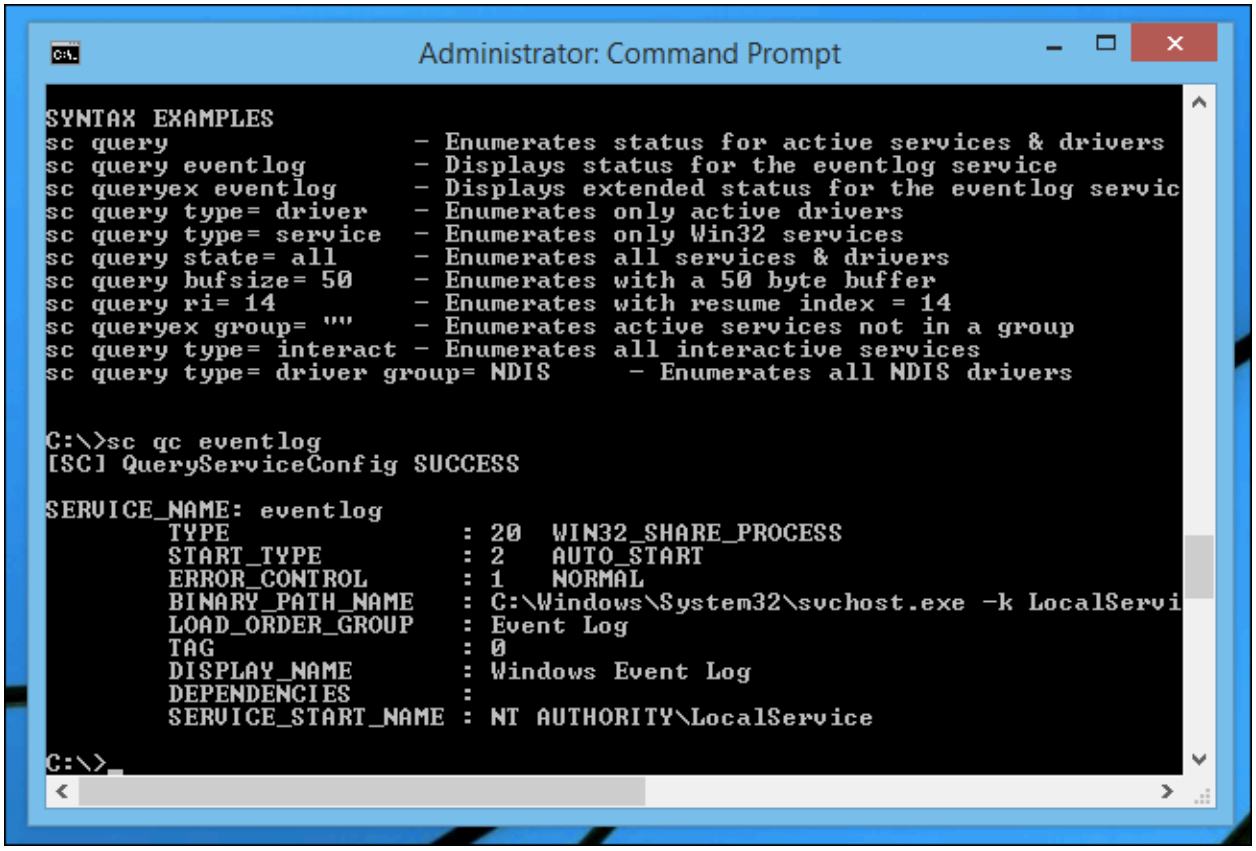


Some operations just can't be done through the graphical user interface. If you want to delete a service, for example, you can only do that through the command line.

Note: please do NOT delete services.

You can query the status of a service using the sc command, like this:

sc qc eventlog



There are many other commands and operations that you can perform, including deleting a service, which we would only recommend if you have malware on your system that is running as a service.

sc delete <malwareservicename>

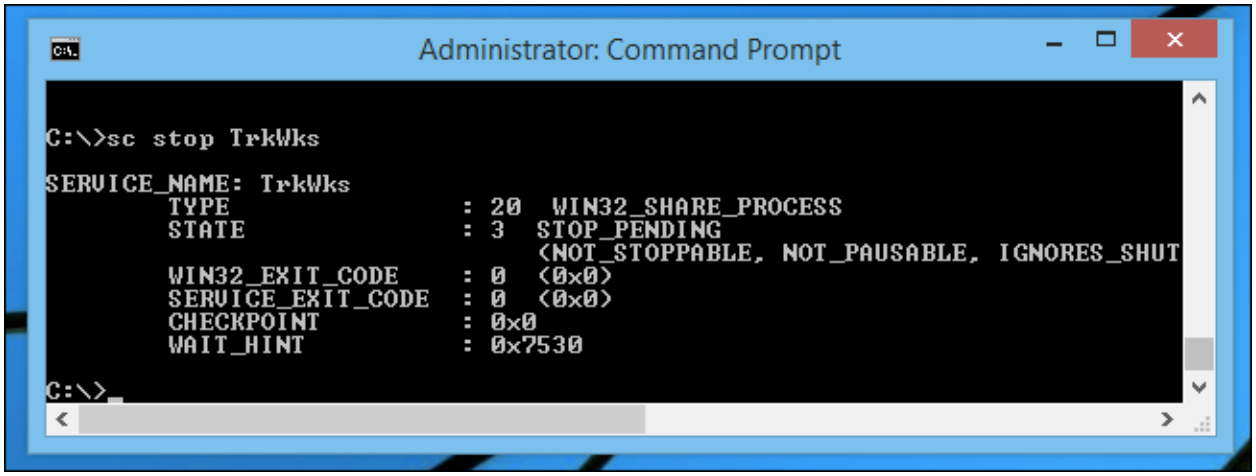
https://youtu.be/-bsLmDfvF1Y

Do not delete services.

You can also do other things, like stopping and restarting services from the command prompt using the sc utility. For example, to stop the distributed link tracking client, use this command:

sc stop TrkWks

To start it again, use sc start <servicename>.



Final Thoughts

If you have services running that are wasting resources and slowing your computer down, you should simply uninstall and remove the applications that put them there. There's really no reason to delete services, disable them, or anything else.

Because why disable something that needs to be uninstalled?

https://youtu.be/4xdVcWvtUnc

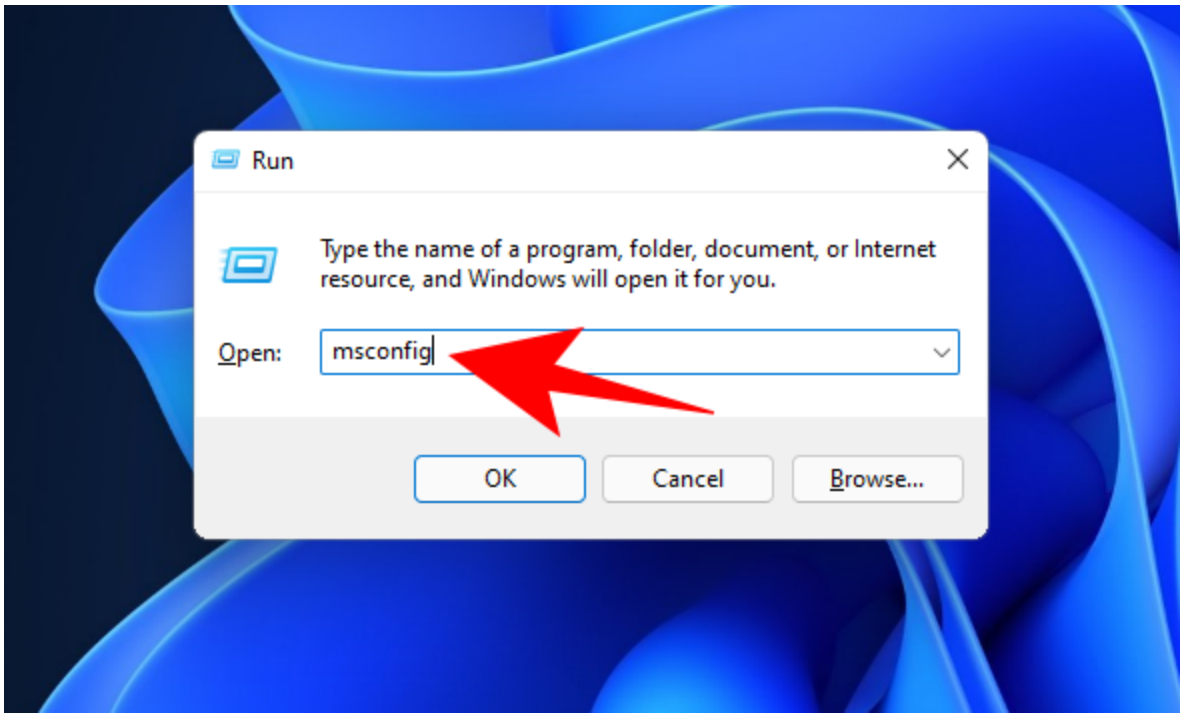
If you are troubleshooting:

1. bootup problems
2. shutdown problems
3. OS unstable after log o
4.

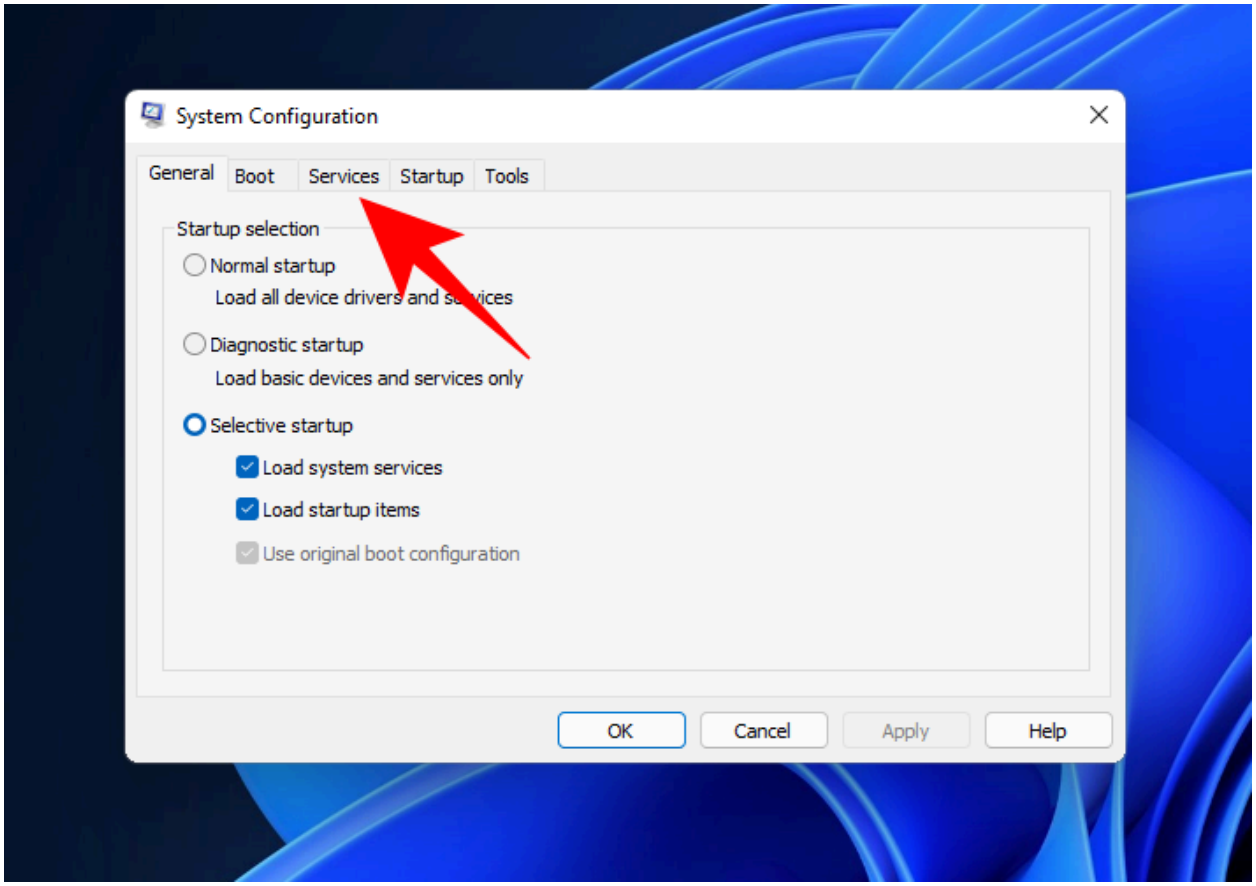
How to disable all third-party services in one click on Windows 11

There's a quick hack that lets you disable all non-Microsoft services in one go. If you want to disable all third-party services and improve your system performance drastically, here's how you can do so:

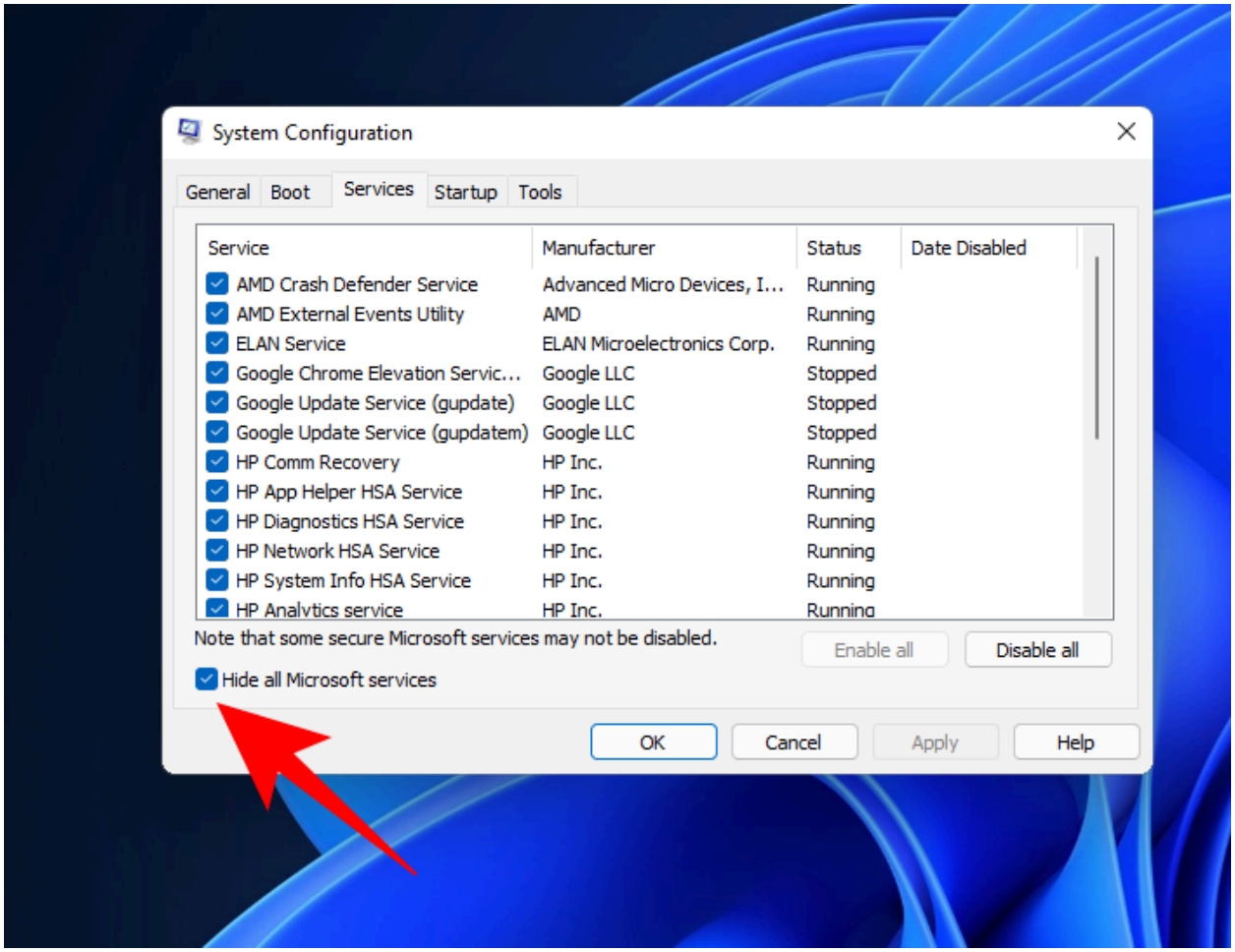
Press Win + R to open the RUN box, type msconfig, and press Enter.



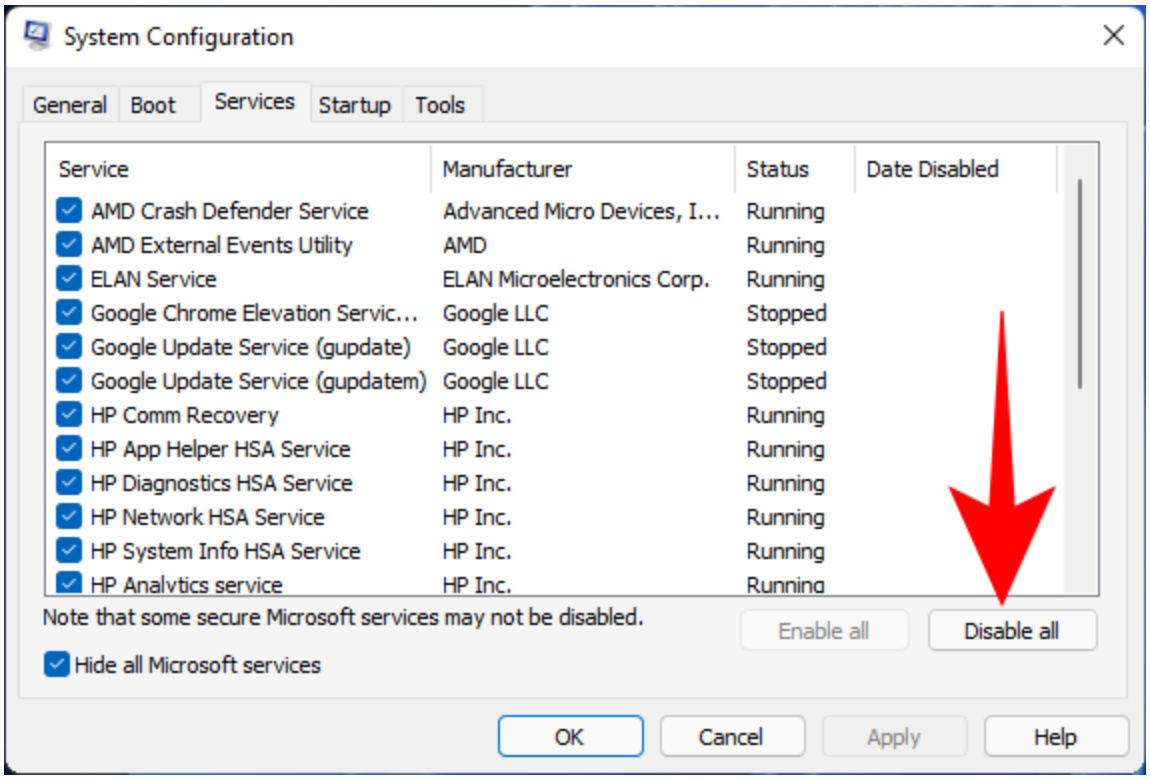
Click on the 'Services' tab to select it.



Then click on Hide all Microsoft services so that they're not shown on the list.



What's left are all third-party apps that you can safely turn off without affecting your system negatively. Now click on Disable all to turn them off.



Click OK.

When prompted, click on Restart to do so.

FAQs

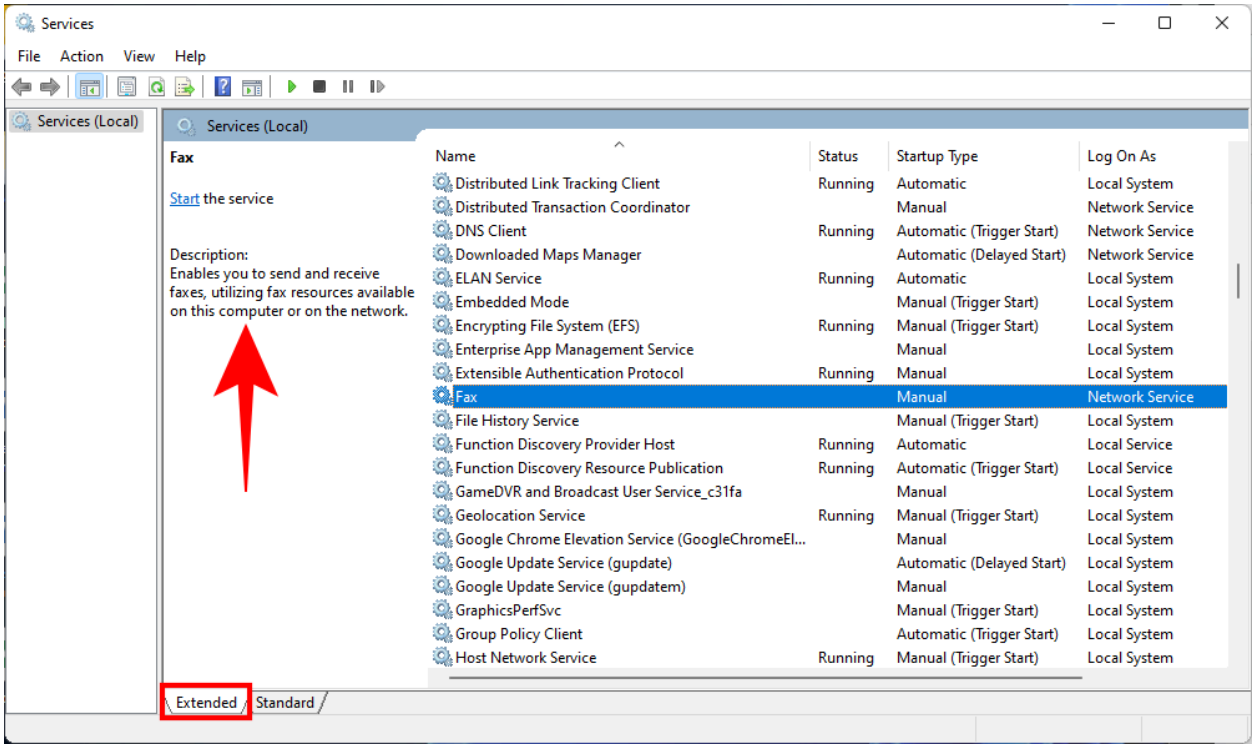
The Services window can seem too daunting a place to make changes. One never knows what one's getting into. These are, after all, some of the services that you might need to keep Windows and its features functioning properly. With that in mind, let's take a look at some of the commonly asked questions and the things that you should know about these background services so that you're a little more confident about disabling the ones that you don't need.

Why should you disable some Windows 11 services?

If you want to make Windows 11 faster on your system hardware, disabling inconsequential services is one of the things that you can do. Services that run in the background without actually having any effect on your daily use are nothing

more than resource hoggers. Regardless of whether they're native services or not, you can go ahead and disable them if you know you're never going to use that service or the services related to it.

When in doubt, you can always click on a service and read its description. By default, the Services window will display the selected service's description in the left panel. If you don't see it, click on a service to get its description.



What happens when you disable Windows 11 services?

When a service is disabled on Windows 11, it won't turn on unless manually specified. This will ensure that services that you don't need do not run unnecessarily in the background and your precious system resources are put to better use and improve system performance on the whole.

We hope you were able to determine which services are important and which are not, and were able to disable the unimportant ones to improve your system's performance on Windows 11.

PowerShell Commands for Services

1. Show all the running services on the server.

```
Get-Service | Where {$_.status –eq 'running'}
```

2. Get the service from (SERVICE-NAME) in SERVER1, SERVER2 with formate table.

```
Get-Service -ServiceName *SERVICE-NAME* -ComputerName SERVER1, SERVER2 | select name, status, machinename | sort machinename | format-table -AutoSize
```

3. Get the service status form the list of server name store in MyServers.txt.

```
Get-Service -computername (get-content c:\folder\Servers_list.txt) | Select servername,status |sort name | format-table -autosize<span id="mce_marker" data-mce-type="bookmark" data-mce-fragment="1"></span>
```

4. Get server from wuauserv from listed VM's.

```
'vm1','vm2','vm3'| foreach {get-service

wuauserv -computername $_} | Format-Table Name,Status,Machinename -AutoSize
```

5. Run the service from the listed remote server.

```
Get-Service -Name express* -Computer vm1, vm2, vm3 |

? { $_.Status -eq "Running" } |

format-table -auto MachineName, ServiceName, DisplayName, Status
```

6. Find out from the list of server in which one SQL is installed and export in csv file.

```
$Machines = Get-Content -Path "C:\machines.txt"
```

```
foreach ($computer in $machines){

Write-host "SQL Service chacking on $computer" -b "green" -foregroundcolor
"red"

Get-Service -Displayname "*SQl Server*" -ComputerName "$computer"

}

 ps >.\sql_service.ps1 | Export-Csv C:sql_install_server.csv -encoding "unicode"
```

Running Multiple different services on different servers

7. Starting Multiple services on different servers.

Create a 2 txt file and save one as services.txt which is the list of service one by one and in machines.txt list server one by one

services.txt >

```
wuauserv
MpsSvc

W32Time

W3SVC

wmiApSrv
```

machines.txt >

```
vm1

vm2
```

```
vm3

vm4

vm5
```

Next, download PsServices.exe and save to C : drive from where you are running your script.

```
 $Services = Get-Content -Path "C:\services.txt"

$Machines = Get-Content -Path "C:\machines.txt"

foreach ($computer in $machines){

Write-host "Service is going to start on $computer" -b "green" -foregroundcolor "white"

foreach ($service in $services) {

PsService.exe \$computer start $service

}

}
```

8. The same script can be performed well by using Get-Service

```
$Services = Get-Content -Path "C:service_list.txt"

$Machines = Get-Content -Path "C:computers_list.txt"

foreach ($computer in $machines){
```

```
Write-host "Service is going to start on $computer" -b "green" -foregroundcolor
"red"

foreach ($service in $services) {

Get-Service -ComputerName $computer -Name $service

}

}
```

9. Run script from the list.csv file :

list.csv >

```
 Hostname Services

vm1 VMTools

vm2 VMTools

vm2 wuauserv

Import-Csv C:\list.csv -Delimiter "`t" | % {

$services = $_.Services -split ','

echo -Computer $_.Hostname -Name $services

Start-Service -Computer $_.Hostname -Name $services

}
```

10. Multiple unique services Stop and Start to a different server.

Save text file server.txt with below details:

server.txt >

```
 ComputerName,Service

vm1,"VMTools,wuauserv"

vm2,""MpsSvc,Schedule"

$List = Import-CSV c:\servers.csv



ForEach ($Server in $List)

{ Invoke-Command -ScriptBlock { Stop-Service $args[0] } -ComputerName
$Server.ComputerName -ArgumentList $Server.Service.Split(",")

}

Start-Sleep -Seconds 6



ForEach ($Index in (($List.Count - 1)..0))

{ Invoke-Command -ScriptBlock { Stop-Service $args[0] } -ComputerName
$List[$Index].ComputerName -ArgumentList $List[$Index].Service.Split(",")
```

PsService v2.25

 **Download PsTools (2.7 MB)**

Introduction

PsService is a service viewer and controller for Windows. Like the SC utility that's included in the Windows NT and Windows 2000 Resource Kits, *PsService* displays the status, configuration, and dependencies of a service, and allows you to start, stop, pause, resume and restart them. Unlike the SC utility, *PsService* enables you to logon to a remote system using a different account, for cases when the account from which you run it doesn't have required permissions on the remote system. *PsService* includes a unique service-search capability, which identifies active instances of a service on your network. You would use the search feature if you wanted to locate systems running DHCP servers, for instance.

Finally, *PsService* works on both NT 4, Windows 2000 and Windows Vista, whereas the Windows 2000 Resource Kit version of SC requires Windows 2000, and *PsService* doesn't require you to manually enter a "resume index" in order to obtain a complete listing of service information.>

Installation

Just copy *PsService* onto your executable path, and type "psservice".

Using PsService

The default behavior of *PsService* is to display the configured services (both running and stopped) on the local system. Entering a command on the command-line invokes a particular feature, and some commands accept options. Typing a command followed by "- " displays information on the syntax for the command.

Usage: psservice [\\computer [-u username] [-p password]] <command> <options>

USING PSSERVICE

Parameter	Description
query	Displays the status of a service.
config	Displays the configuration of a service.
setconfig	Sets the start type (disabled, auto, demand) of a service.
start	Starts a service.
stop	Stops a service.
restart	Stops and then restarts a service.

pause	Pauses a service
cont	Resumes a paused service.
depend	Lists the services dependent on the one specified.
security	Dumps the service's security descriptor.
find	Searches the network for the specified service.
\\computer	Targets the NT/Win2K system specified. Include the -u switch with a username and password to login to the remote system if your security credentials do not permit you to obtain performance counter information from the remote system. If you specify the -u option, but not a password with the -p option, *PsService* will prompt you to enter the password and will not echo it to the screen.

How it Works

PsService uses the Service Control Manager APIs that are documented in the Platform SDK.

 **Download PsTools (2.7 MB)**

PsTools

PsService is part of a growing kit of Sysinternals command-line tools that aid in the administration of local and remote systems named *PsTools*.

Runs on:

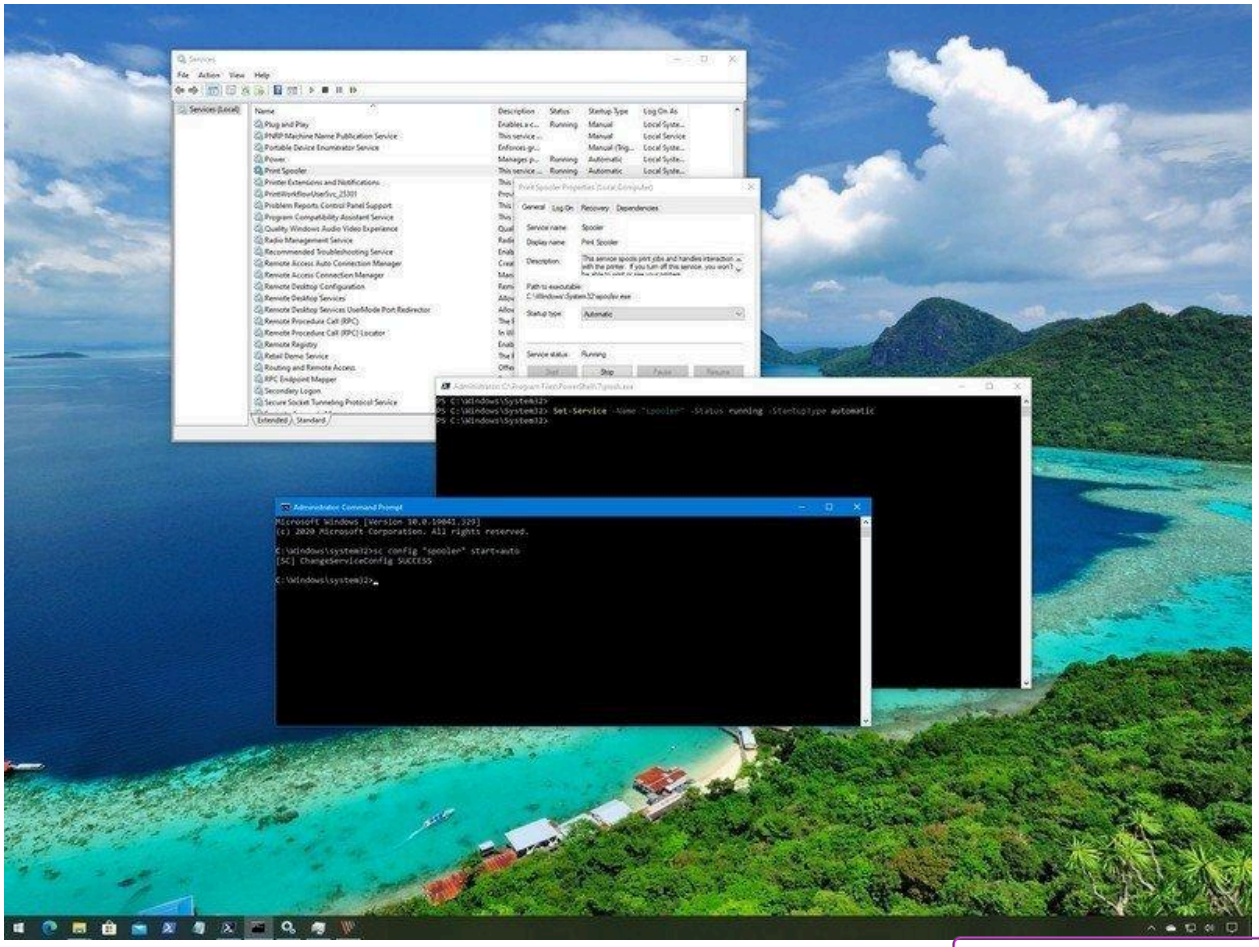
- Client: Windows Vista and higher.
- Server: Windows Server 2008 and higher.

How to start and stop services manually on Windows 10

https://www.windowscentral.com/how-start-and-stop-services-windows-10

Do you need to control a Windows 10 or app service? In this guide, we'll show you four different ways to accomplish this task.

MAURO HUCULAK



On Windows 10, services are programs that run in the background without a user interface and enable system features (such as printing, networking, remote access, File Explorer, Windows Search, updates, etc.) and apps to operate as intended.

Although the system does a pretty good job managing background services, sometimes, you may need to control them manually when a feature or app isn't working correctly, or an app requires you to manage its services manually.

Whatever the case it might be, Windows 10 includes at least four methods to stop, start, disable, or enable services using the Services console, Task Manager, Command Prompt, and PowerShell.

In this Windows 10 guide, we'll walk you through the steps to manage system and apps services on your computer.

How to manage services using Services console

How to manage services using Task Manager

How to manage services using PowerShell

How to manage services using Command Prompt

How to manage services using Services console

Using the Services consoles is perhaps the simplest method to stop, start, disable, or enable one or multiple services on Windows 10.

Stop service

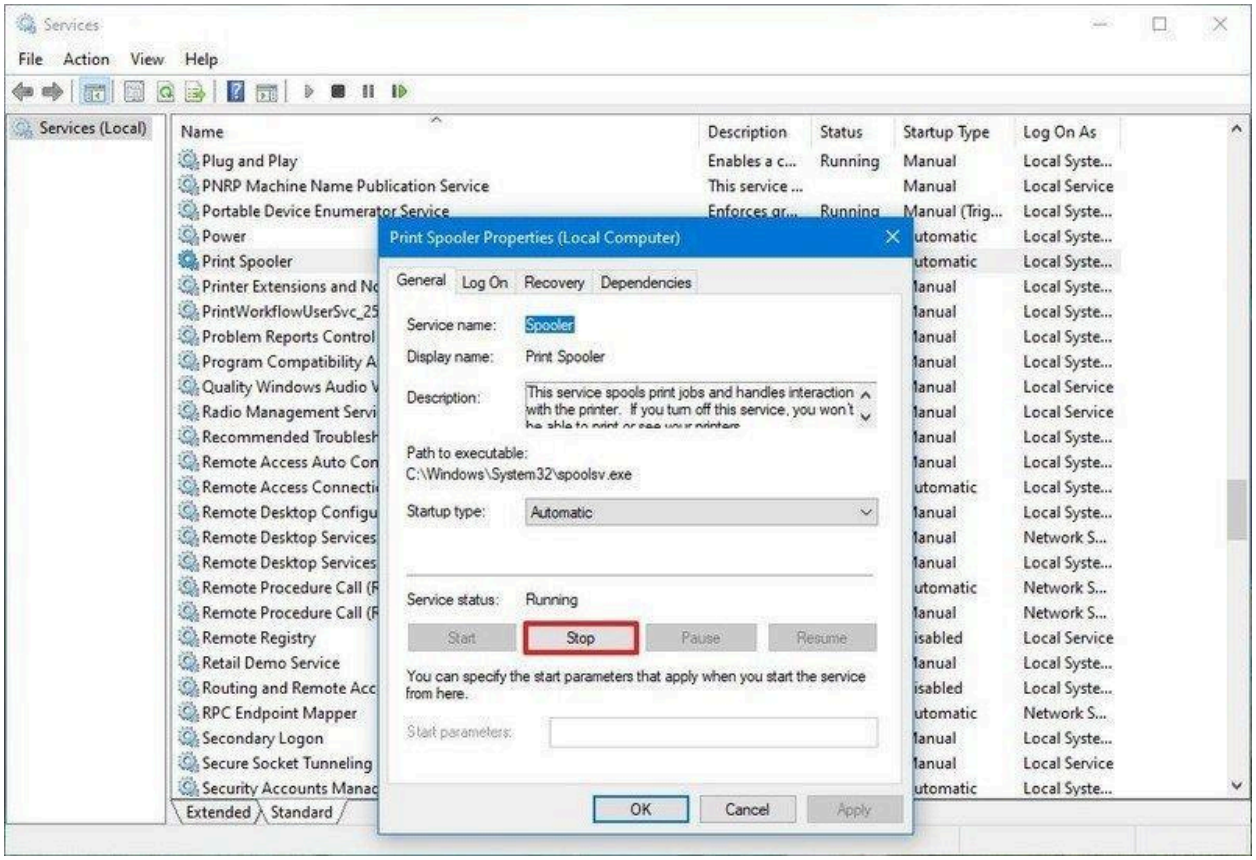
To stop a running service using Services, use these steps:

Open Start.

Search for Services and click the top result to open the console.

Double-click the service that you intend to stop.

Click the Stop button.



Quick tip: You can also manage the state by right-clicking the service and selecting the option. Or you can select the service and then use the controls at the top to start, stop, pause, or restart.

Click the Apply button.

Click the OK button.

Once you complete the steps, the service will stop running on the device.

If you're unable to stop a system service, consider that some services are required for the operation of Windows 10, and they can't be stopped.

Start service

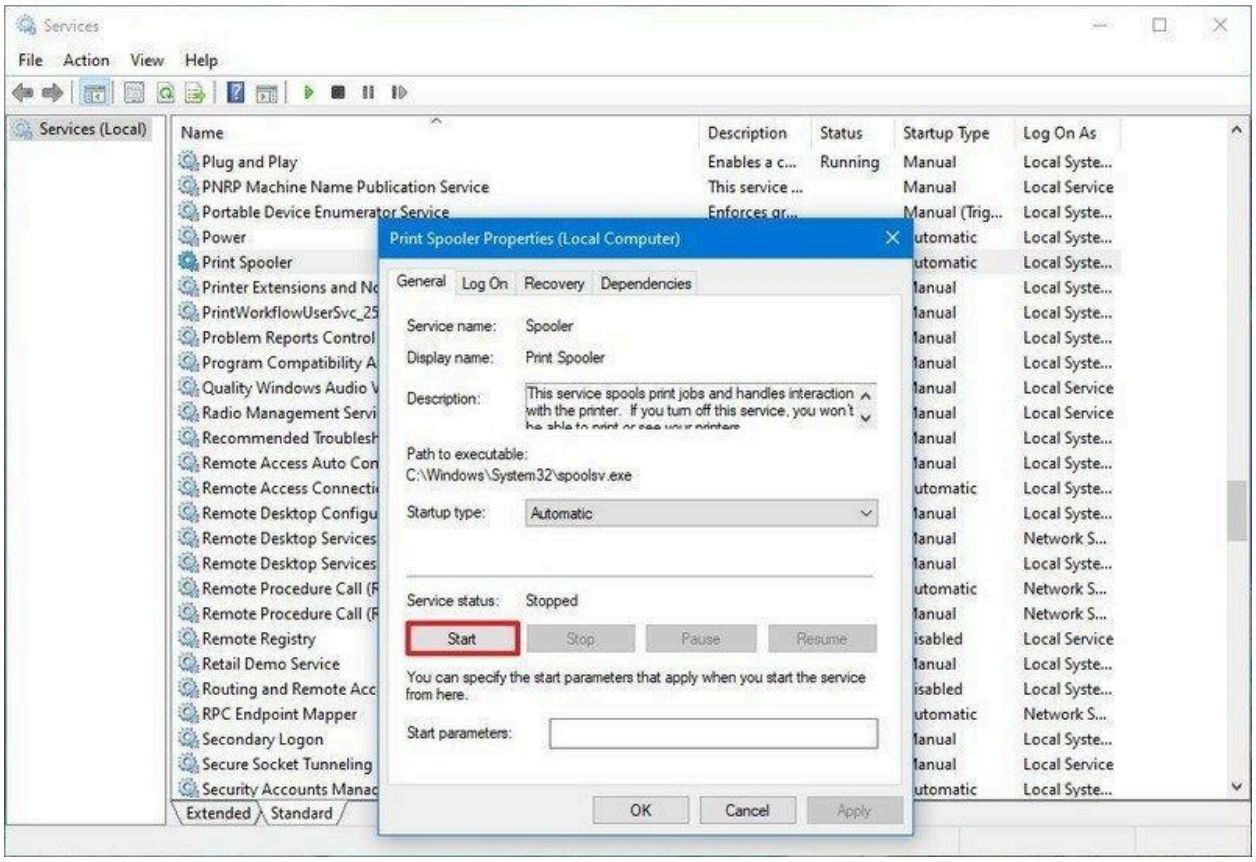
To start a service on Windows 10, use these steps:

Open Start.

Search for Services and click the top result to open the console.

Double-click the service that you intend to stop.

Click the Start button.



Click the Apply button.

Click the OK button.

After you complete the steps, the service you specified will start for the current session.

Disable service

To set a service a disabled, use these steps:

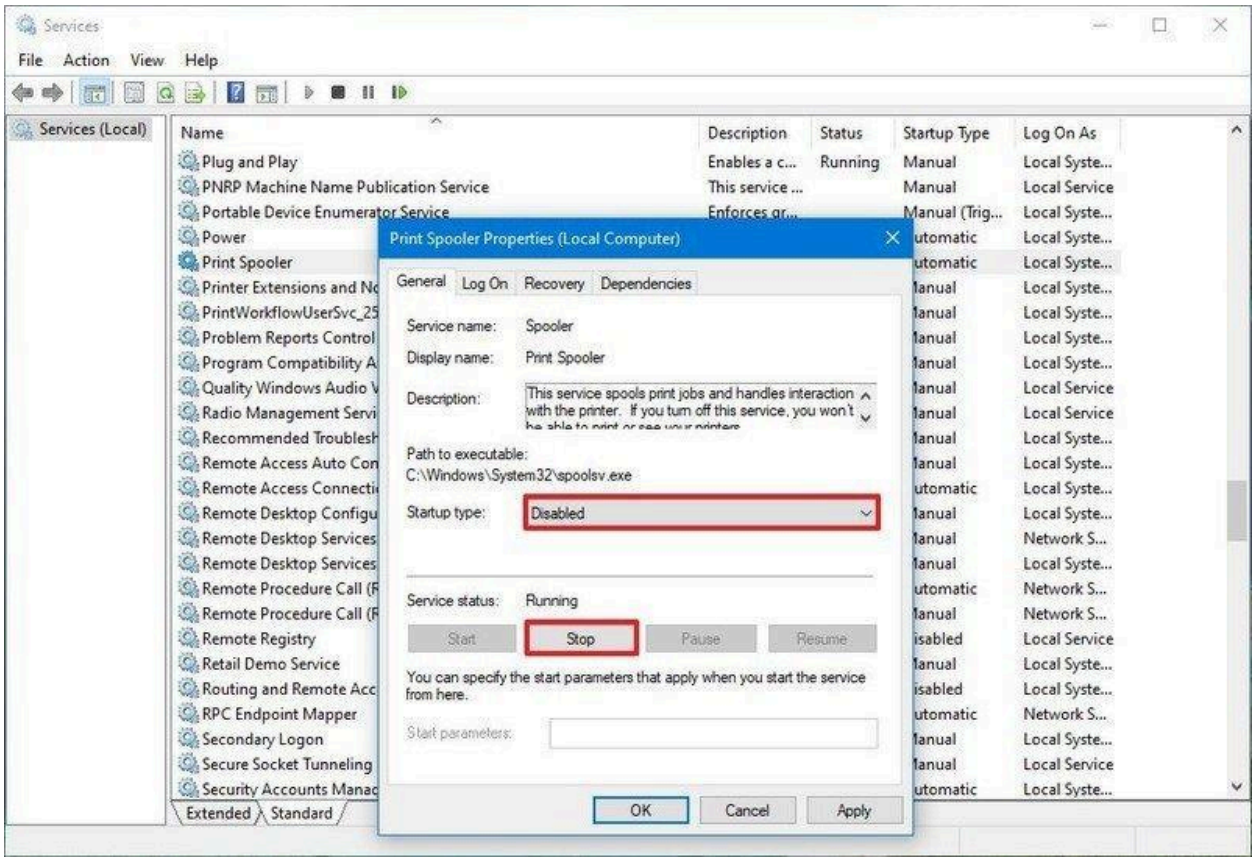
Open Start.

Search for Services and click the top result to open the console.

Double-click the service that you intend to stop.

Click the Stop button.

Use the "Start type" drop-down menu and select the Disabled option.



Click the Apply button.

Click the OK button.

Once you complete the steps, the service will no longer start automatically after restarting your device.

Enable service

To enable a specific service, use these steps:

Open Start.

Search for Services and click the top result to open the console.

Double-click the service that you intend to stop.

Click the Start button.

Use the "Start type" drop-down menu and select the Automatic option.

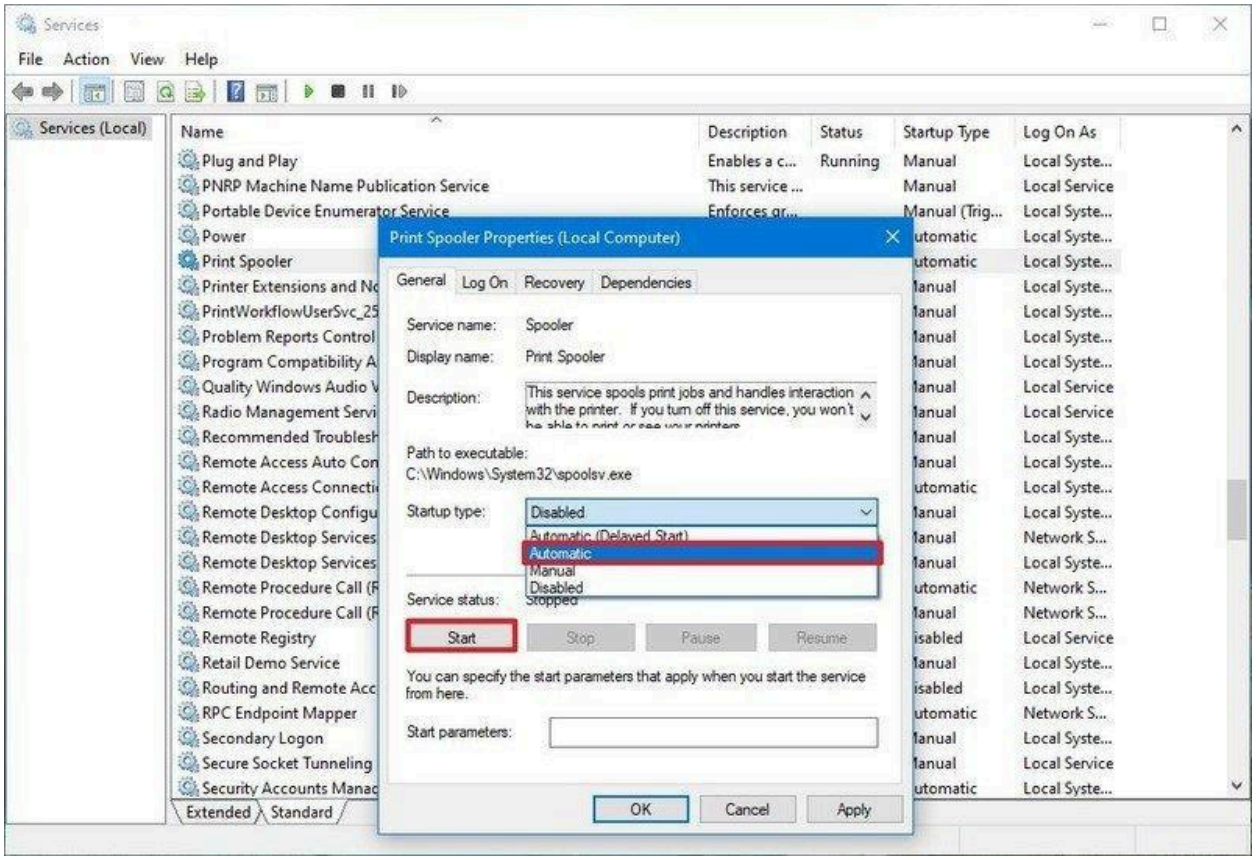
These are the Startup types on Windows 10:

Automatic – service starts at boot.

Automatic (Delayed Start) – service start after boot.

Manual – starts service manually as needed.

Disabled – stops service from running.



Source: Windows Central

Click the Apply button.

Click the OK button.

After you complete the steps, the Windows 10 or app service will enable, but if it was in a stopped state, you'd need to start it manually or restart the device for the service to run.

How to manage services using Task Manager

Task Manager also includes a section to quickly manage services for Windows 10 and apps.

To stop, start, or restart a service using Task Manager, use these steps:

Open Start.

Search for Task Manager and click the top result to open the app.

Quick tip: Windows 10 includes many other ways to open the experience, including right-clicking the taskbar and selecting the Task Manager option and using the Ctrl + Shift + ESC keyboard shortcut.

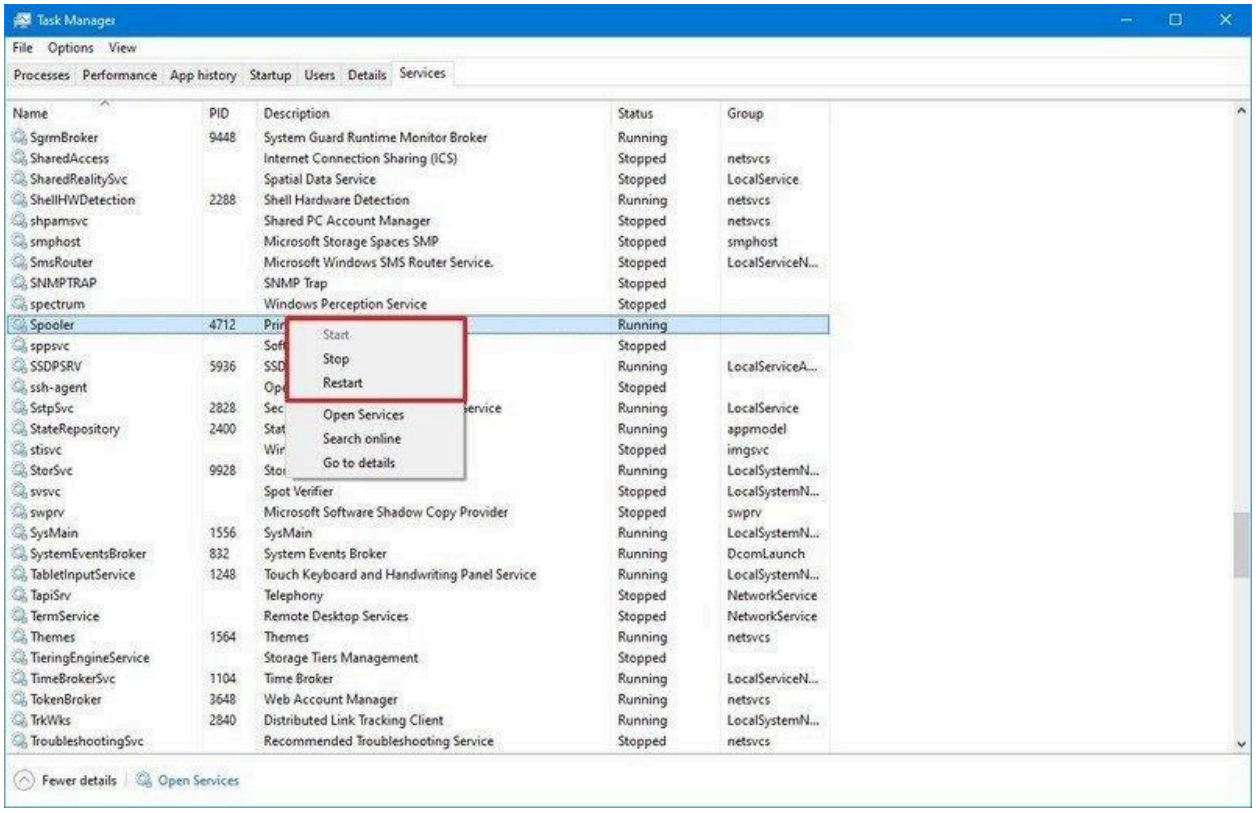
Click the Services tab.

Right-click the service name, and select one of the options:

Stop.

Start.

Restart.



Source: Windows Central

Quick note: Task Manager only displays the service name, not the display name. For example, if you're using this method, you'll the "Print Spooler" defined as "Spooler."

Once you complete the steps, the service will respond to the option you selected.

How to manage services using PowerShell

You can also use PowerShell commands to manage background services for Windows 10 and apps.

Stop service

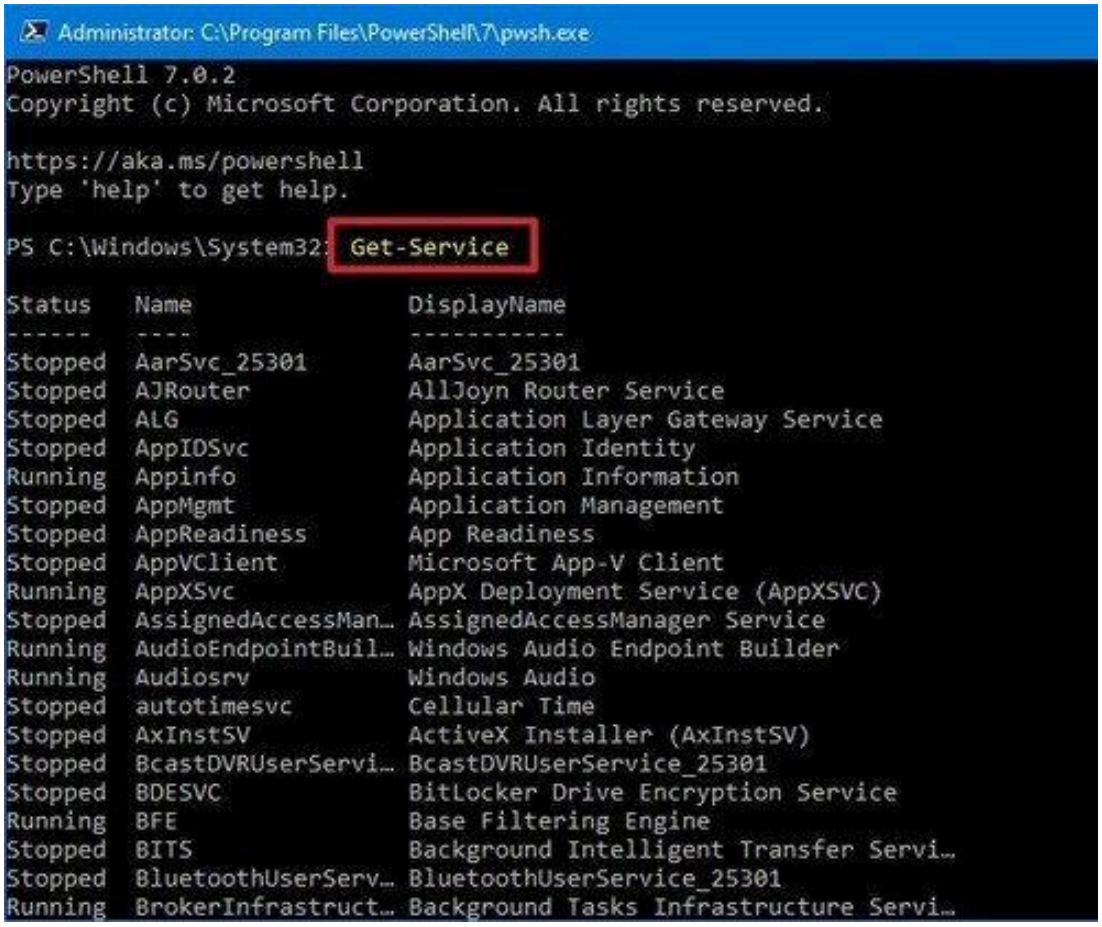
To stop a specific service with PowerShell, use these steps:

Open Start.

Search for PowerShell, right-click the top result, and select the Run as administrator option.

(Optional) Type the following command to view a list of all the services and press Enter:

Get-Service



Type the following command to stop a service and press Enter:

Stop-Service -Name "SERVICE-NAME"

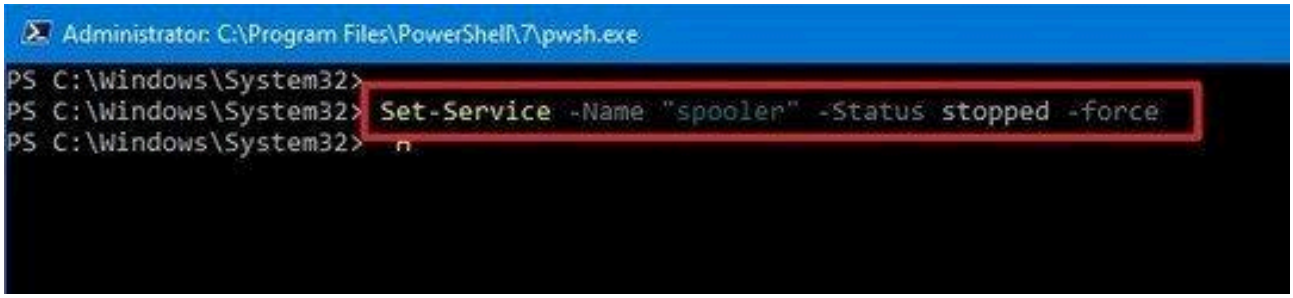
For example, this command stops the printer spooler service on Windows 10:

Stop-Service -Name "spooler"

In the command, replace "SERVICE-NAME" for the name of the service that you intend to stop. If you want to use the display name,
replace -Name for -DisplayName and then specify the display name of the service.

Alternatively, you can also use this variant of the command to stop the service:

Set-Service -Name "SERVICE-NAME" -Status stopped



In the command, replace "SERVICE-NAME" for the name of the service that you intend to stop. If you want to use the display name,
replace -Name for -DisplayName and then specify the display name of the service. You only need the quotation marks if there's a space within the name.

Quick tip: If you're getting a dependency error, you can append the -force option in either of the commands to stop the service. For example, Stop-Service -Name "SERVICE-NAME" -Force.

After you complete the steps, the PowerShell command will stop the service on your device.

Start service

To start a Windows 10 or app service with PowerShell, use these steps:

Open Start.

Search for PowerShell, right-click the top result, and select the Run as administrator option.

Type the following command to start a service and press Enter:

Start-Service -Name "SERVICE-NAME"

For example, this command starts the printer spooler service on Windows 10:

Start-Service -Name "spooler"

In the command, replace "SERVICE-NAME" for the name of the service. Using the display name is supported, replacing -Name for -DisplayName and then specify the display name of the service.

Alternatively, you can also use this variant of the command to start a service:

Set-Service -Name "SERVICE-NAME" -Status running

In the command, replace "SERVICE-NAME" for the name of the service. If you want to use the display name, replace -Name for -DisplayName and then specify the display name of the service.

Once you complete the steps, the service will start on your computer.

Disable service

To disable a service using a PowerShell command, use these steps:

Open Start.

Search for PowerShell, right-click the top result, and select the Run as administrator option.

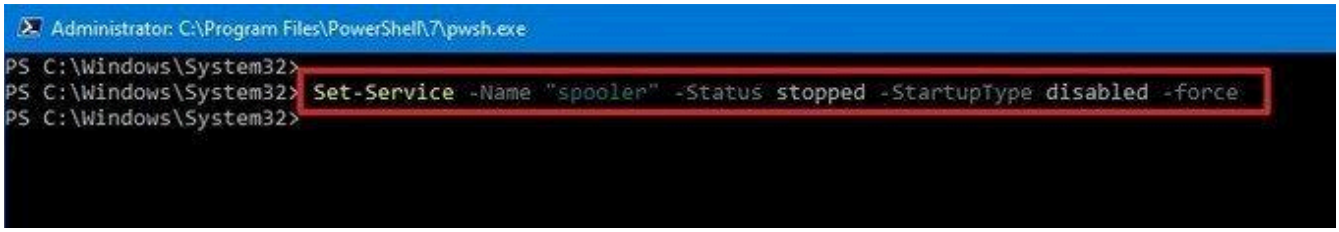
Type the following command to disable a service and press Enter:

Set-Service -Name "SERVICE-NAME" -Status stopped -StartupType disabled

In the command, update "SERVICE-NAME" for the name of the service. If you want to use the display name of the service, then replace -Name for -DisplayName and specify the service name. If you want to disable the service without stopping it immediately, you can remove the -Status stopped portion of the command.

For example, this command disables the printer spooler service on Windows 10:

Set-Service -Name "spooler" -Status stopped -StartupType disabled



After you complete the steps, the PowerShell command will disable the specified service.

Enable service

To enable a specific background service with PowerShell, use these steps:

Open Start.

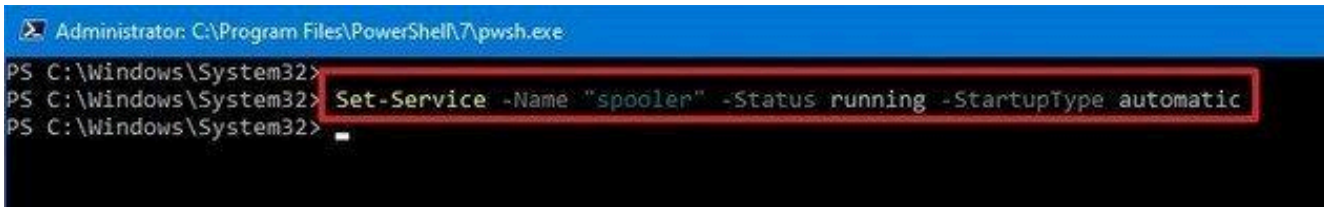
Search for PowerShell, right-click the top result, and select the Run as administrator option.

Type the following command to enable a service and press Enter:

Set-Service -Name "SERVICE-NAME" -Status running -StartupType automatic

For example, this command enables the printer spooler service using PowerShell:

Set-Service -Name "spooler" -Status running -StartupType automatic



Quick note: You may able to use the display -DisplayName option, but the command may also prompt you to supply the name of the service, adding an extra step to the process. If you want to enable the service without starting it immediately, you can remove the -Status running portion of the command.

Once you complete the steps, PowerShell will enable the service specified with the command.

How to manage services using Command Prompt

If you're comfortable using the command line, Command Prompt offers the "net" command (older) to stop or start, or the "sc" command (newer) to stop, start, disable, or enable services on Windows 10.

Stop service

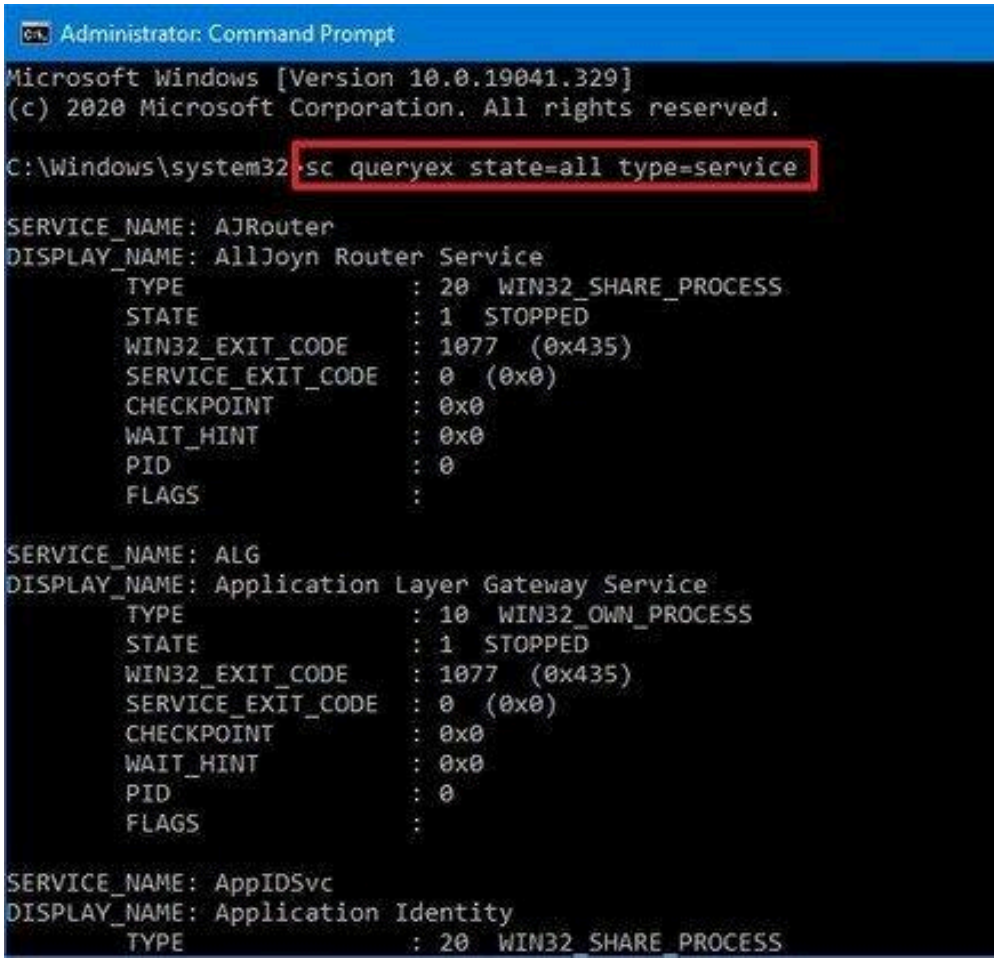
To stop a Windows 10 or app service with Command Prompt, use these steps:

Open Start.

Search for Command Prompt, right-click the top result, and select the Run as administrator option.

(Optional) Type the following command to view a list of all the services and press Enter:

sc queryex state=all type=service



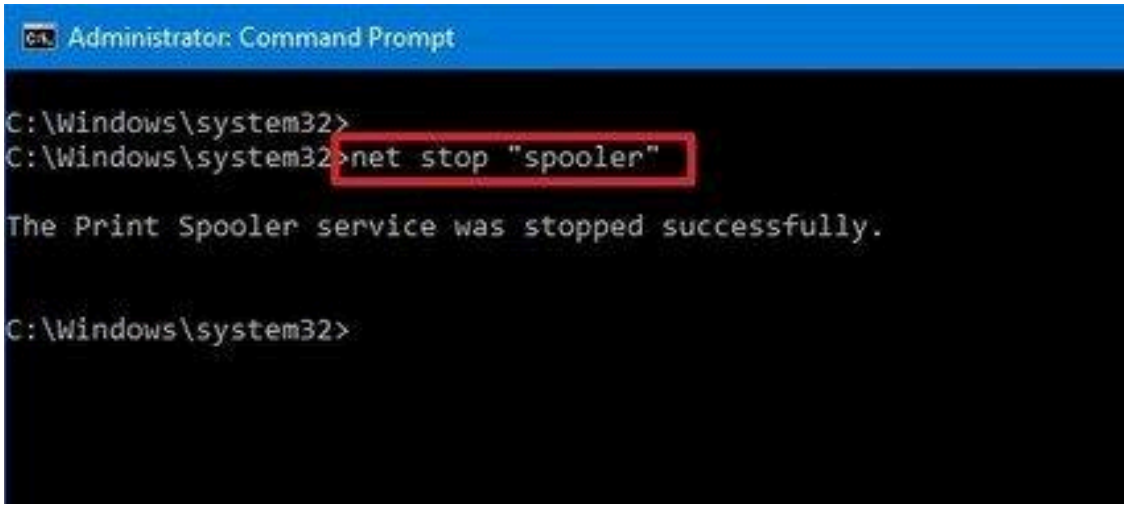
Type the following command to stop a service and press Enter:

net stop "SERVICE-NAME"

In the command, replace "SERVICE-NAME" for the name or display name of the service. You only need the quotation marks if there's a space within the name.

For example, this command stops the printer spooler using the service name:

net stop "spooler"

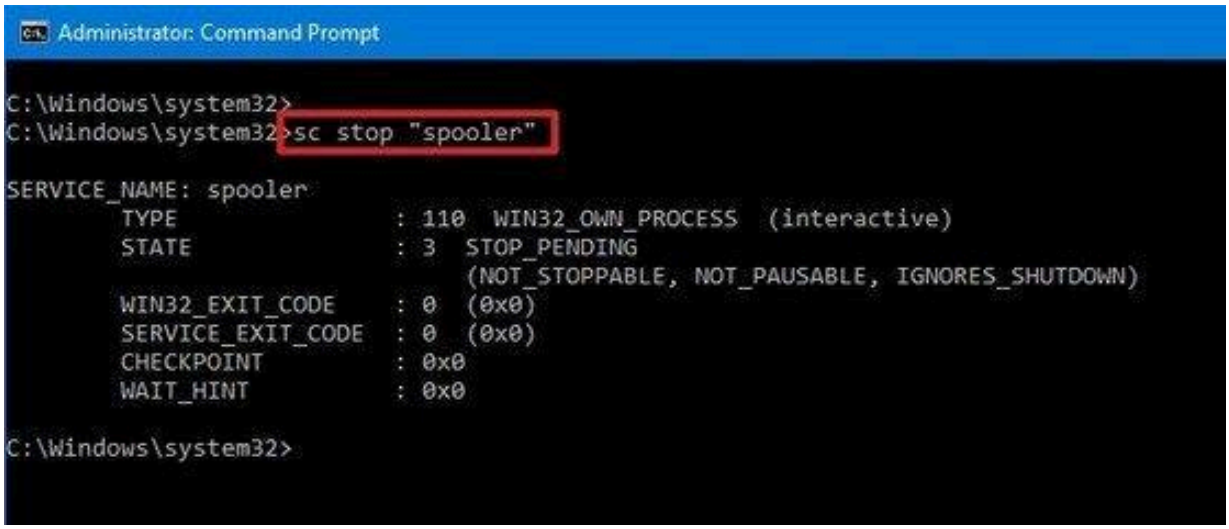


Alternatively, you can also use the more advanced "sc" command:

sc stop "SERVICE-NAME"

For example, this command stops the printer spooler using the service name:

sc stop "spooler"



After you complete the steps, the command will stop the specified service on Windows 10.

Start service

To start a service with the command line, use these steps:

Open Start.

Search for Command Prompt, right-click the top result, and select the Run as administrator option.

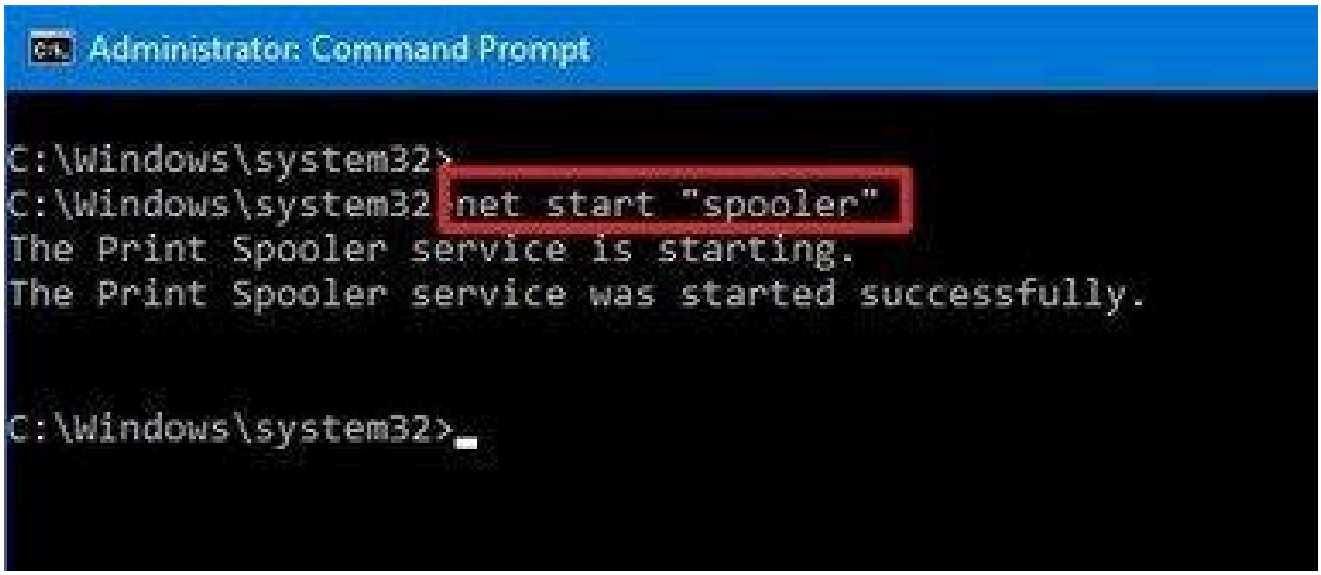
Type the following command to start a service and press Enter:

net start "SERVICE-NAME"

In the command, replace "SERVICE-NAME" for the name or display name of the service. You only need the quotation marks if there's a space within the name.

For example, this command starts the printer spooler using the service name:

net start "spooler"

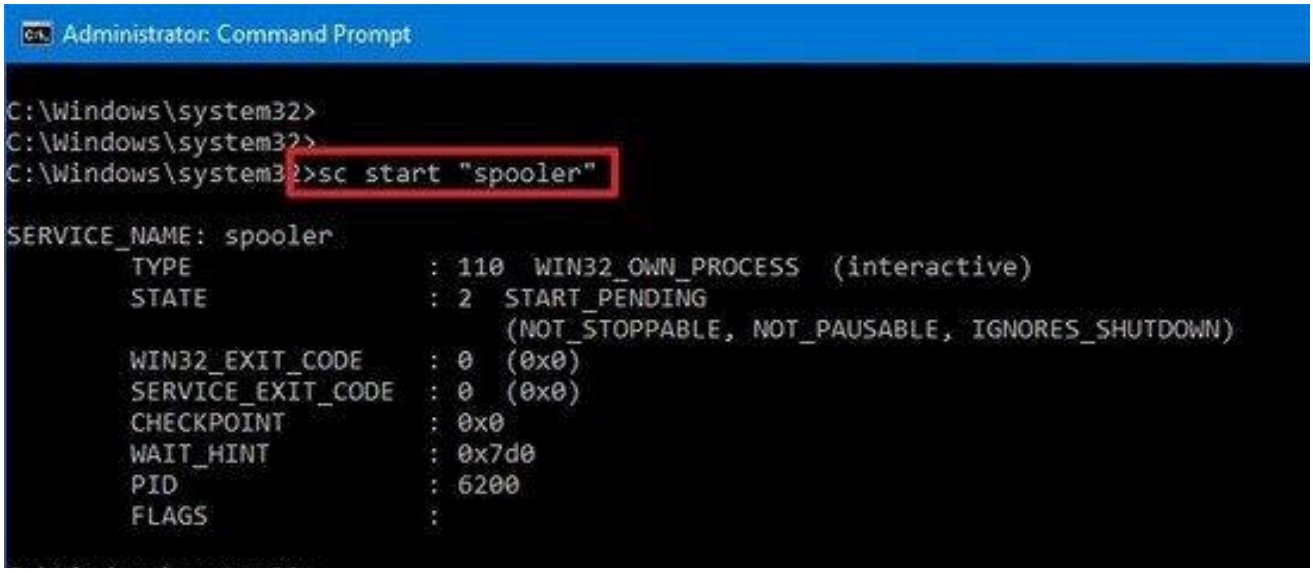


Alternatively, you can also use the "sc" command:

sc start "SERVICE-NAME"

For example, this command starts the printer spooler using the service name:

sc start "spooler"



Once you complete the steps, the command will execute and start the service you specified.

Disable service

To disable a service with Command Prompt, use these steps:

Open Start.

Search for Command Prompt, right-click the top result, and select the Run as administrator option.

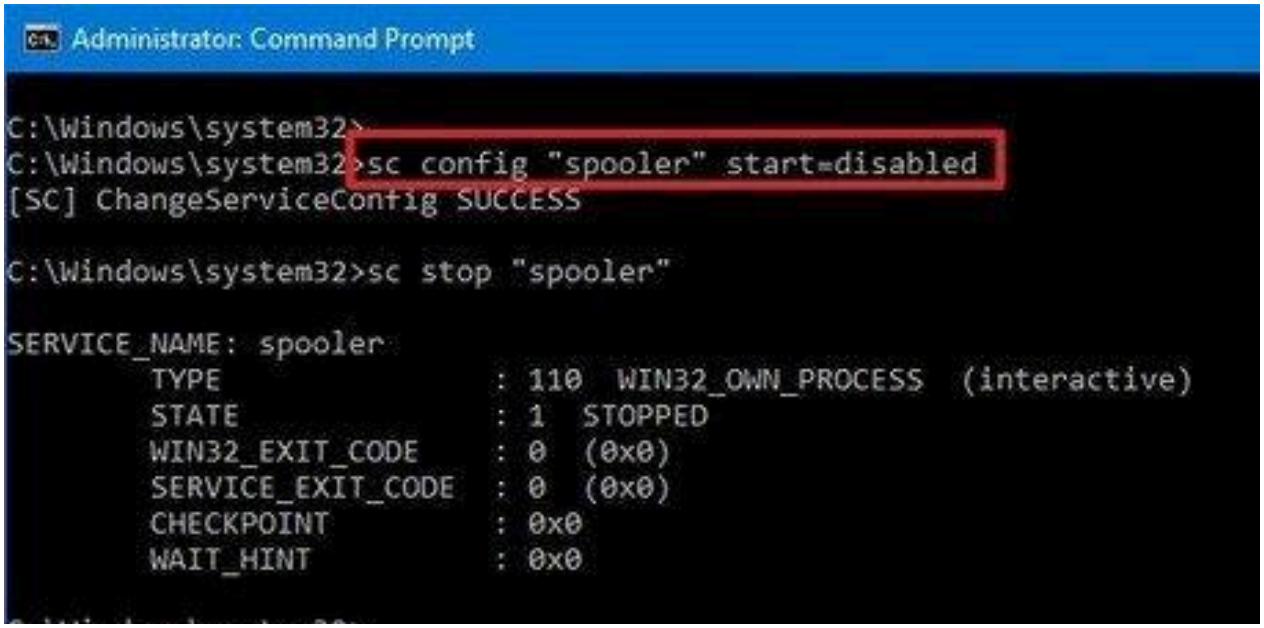
Type the following command to disable a service and press Enter:

sc config "SERVICE-NAME" start=disabled

In the command, replace "SERVICE-NAME" for the name of the service that you want to disable.

For example, this command disables printer spooler using the service name:

sc config "spooler" start=disabled



(Optional) Type the following command to stop the service and press Enter:

sc stop "SERVICE-NAME"

Quick note: When you disable a service, it doesn't stop the current state of the service. You can either restart your computer or stop the service using the above command.

After you complete the steps, the sc command will run disabling the Windows 10 or app service you specified.

Enable service

To enable a service with a command, use these steps:

Open Start.

Search for Command Prompt, right-click the top result, and select the Run as administrator option.

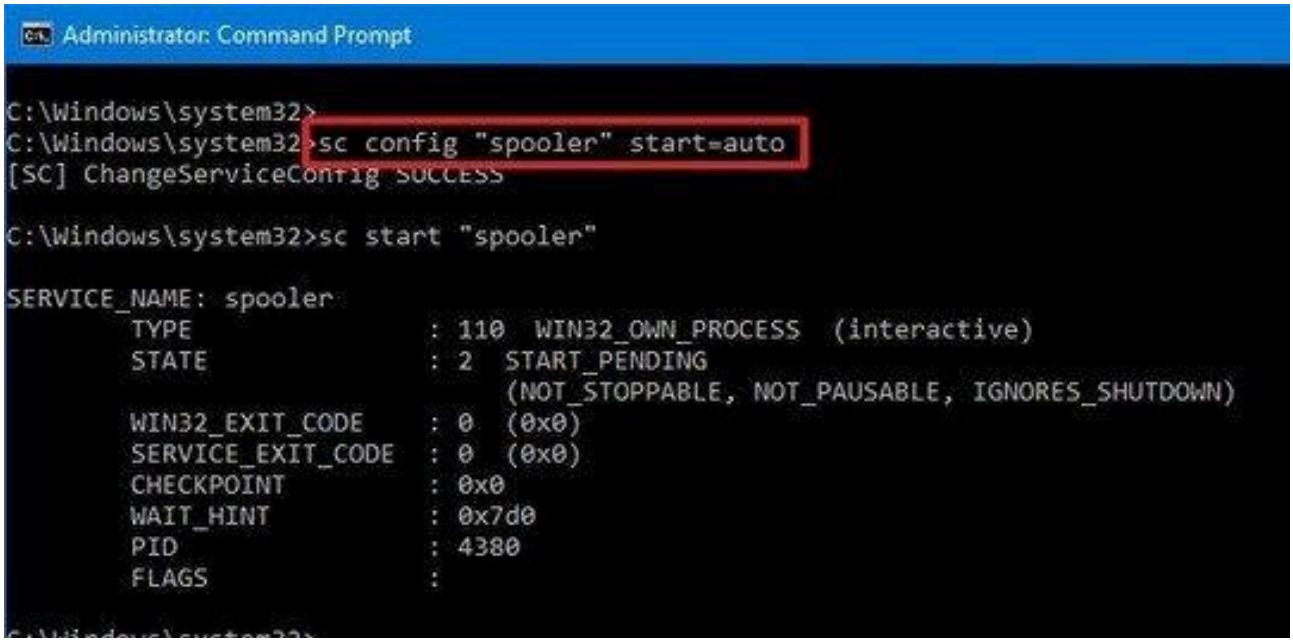
Type the following command to enable a service and press Enter:

sc config "SERVICE-NAME" start=auto

In the command, replace "SERVICE-NAME" for the name of the service that you want to enable.

For example, this command enables the printer spooler automatically using the service name:

sc config "spooler" start=auto



These are alternative commands to enable a particular service:

Manual: sc config "SERVICE-NAME" start=demand

Automatic Delayed: sc config "SERVICE-NAME" start=delayed-auto

(Optional) Type the following command to start the service and press Enter:

sc start "SERVICE-NAME"

Once you complete the steps, the service will enable and start automatically on reboot according to the command you used.

You can only use the "net" command to start or stop services. The "sc" command allows you to perform more tasks, including start, stop, enable, or disable services, among other options. If you're choosing to manage services with command lines, then, in either case, it's best to use the service name instead of the display name.

Also, when using any of the methods outlined above, consider that making modifications to the default settings can alter the operation of one or more features that depend on that service negatively affecting the experience.

Furthermore, if you restart a service, you might be required to start its dependencies manually as well to make the app or feature operational again.

We're focusing this guide on Windows 10, but the ability to manage services has been available for several years, which means that you can refer to this guide if you're still running Windows 8.1, Windows 7, and older versions.

How to Start, Stop, Restart, Enable, and Disable Services in Windows 10

https://www.tenforums.com/tutorials/4499-start-stop-disable-services-windows-10-a.html

A service is an application type that runs in the system background without a user interface and is similar to a UNIX daemon process. Services provide core operating system features, such as Web serving, event logging, file serving, printing, cryptography, and error reporting.

This tutorial will show you how to start, stop, restart, enable, and disable **services** in **Windows 10**.

You must be signed in as an **administrator** to be able to do the steps in this tutorial.

If you stop, start, or restart a service, any dependent services are also affected. Starting a service does not automatically restart its dependent services.

Changing the default service settings may prevent key services from running correctly. It is especially important to use caution when changing the **Startup type** setting of services that are configured to start automatically.

Some services, such as **Remote Procedure Call (RPC)**, **Event Log**, and **Plug and Play**, cannot be stopped by using the **Services** snap-in window or the **net**

stop command. These services are required for the operating system to function properly.

It is highly recommended that you **create a restore point** before making changes to the services. This way if you make a mistake that cripples your computer, you will be able to do a **System Restore** using the restore point to undo the changes.

If you disabled the wrong service and lost access to the computer, then try booting into **Safe Mode** to change the service back.

Contents

OPTION ONE

To Start, Stop, and Disable Services in Services Window

1 Do **step 2** or **step 3** below for how you would like to open the **Services** snap-in window.

2 Press the Win + R keys to open the Run dialog, type **services.msc** into Run, press Enter, and go to **step 4** below.

3 Open the **Control Panel (icons view)**, click/tap on the **Administrative Tools** icon, double click/tap on **Services** shortcut, close Administrative Tools, and go to **step 4** below.

4 Do **step 5** (stop), **step 6** (enable/start), or **step 7** (disable) below for what you would like to do. (see screenshots below)

Note

"Startup Type" for Service

Automatic - With a service in this state, it will start at boot time. Some services, when no longer required, will also automatically stop when not needed. If you find you do not need a service, place it into Manual or Disabled.

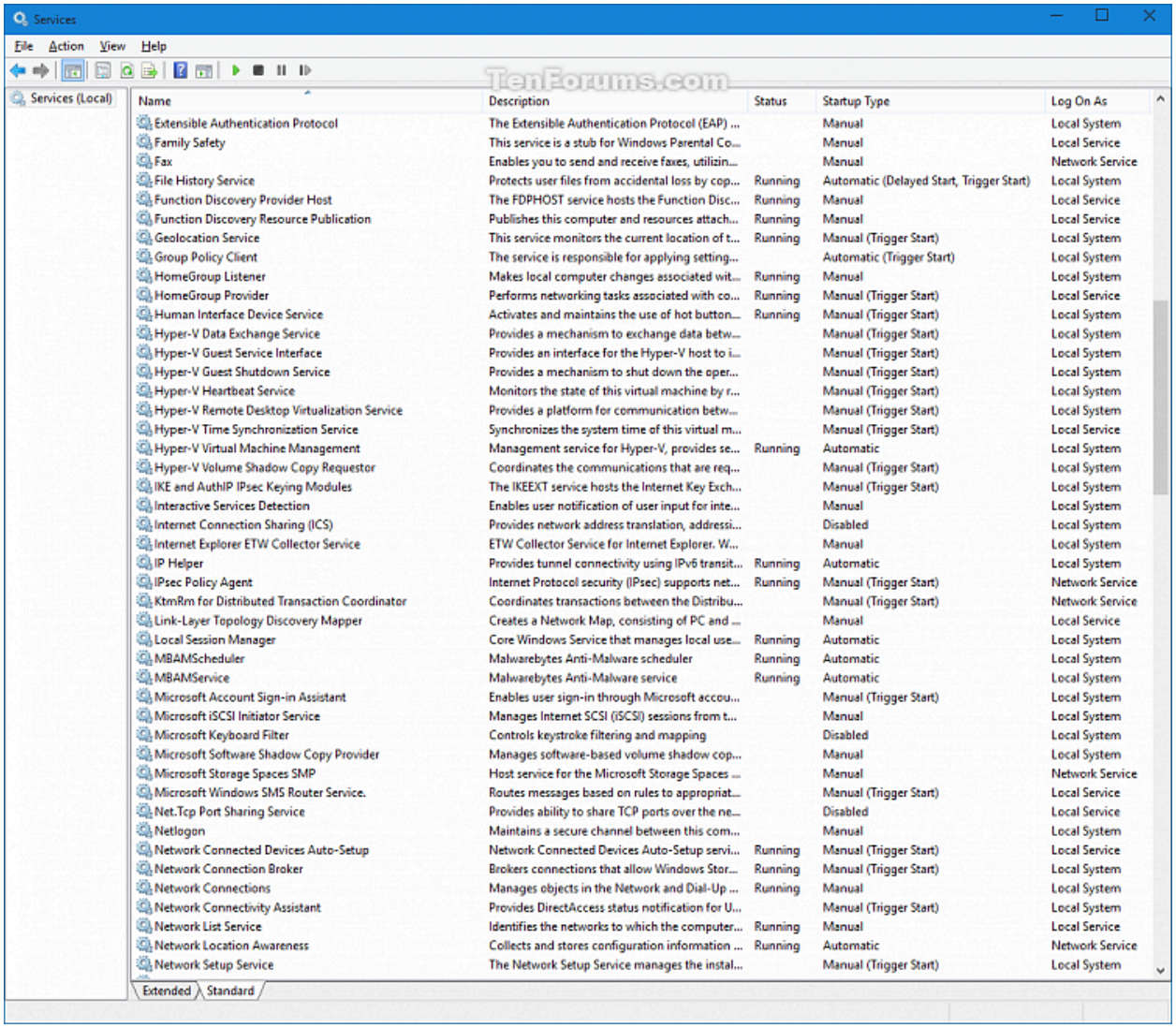
Automatic (Delayed Start) - With a service in this state, it will start just after boot time. Some services, when no longer required, will also automatically stop when not needed. If you find you do not need a service, place it into Manual or Disabled.

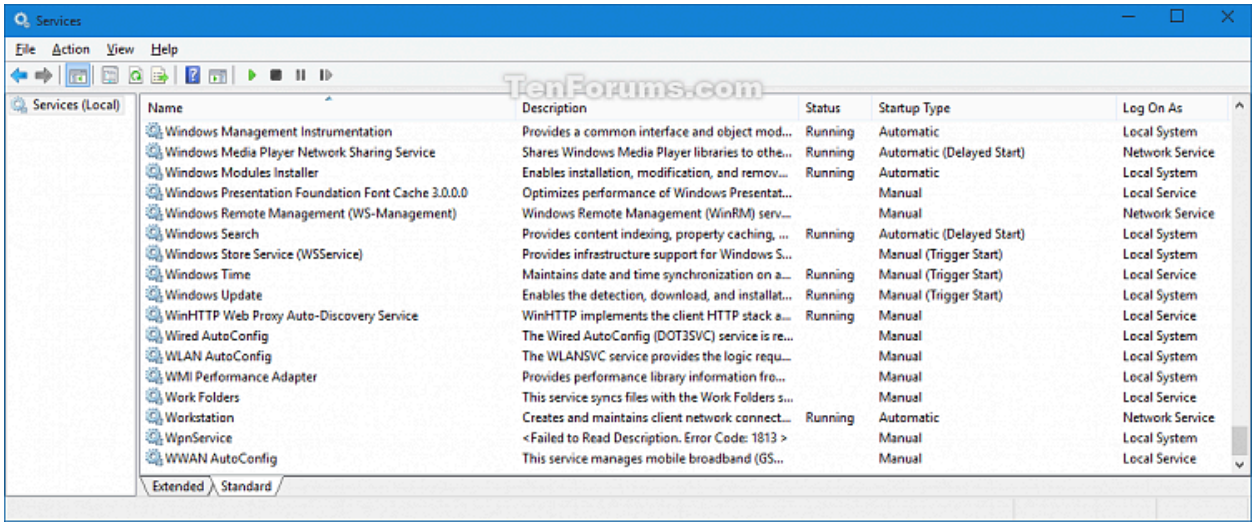
Automatic (Delayed Start, Trigger Start) - With a service in this state, it will start just after boot when specifically called.

Manual (Trigger Start) - This is a version of Manual mode that allows Windows to start a service when specifically called and Microsoft's answer to "too many services running all the time".

Manual - Manual mode allows Windows to start a service when needed. However, very few services will start up when required in Manual mode. If you find you need a service, place it into Automatic.

Disabled - This setting will stop a service from starting, even if needed. Errors in the Event Viewer will show up complaining of that fact. Some services, while Disabled, will constantly complain. However, this situation is taken care of if placed in Manual. The service descriptions identifies those that should be in Manual vice Disabled.





5. To Stop a Service

A) Double click/tap on a service with a status of running that you want to stop. (see screenshot below **step 4**)

B) Click/tap on the **Stop** button, wait until the service status shows as stopped, and go to **step 8** below. (see left screenshot below **step 8**)

6. To Enable/Start a Service

A) Double click/tap on a service with no status that you want to start. (see screenshot below **step 4**)

B) If the **Startup type** of the service is set to **Disabled**, then you will need to change it to either **Manual**, **Automatic**, or **Automatic (Delayed Start)** first, and click/tap on **Apply**. (see right screenshot below **step 8**)

C) Click/tap on the **Start** button, and wait until the service status shows as running, and go to **step 8** below. (see left screenshot below **step 8**)

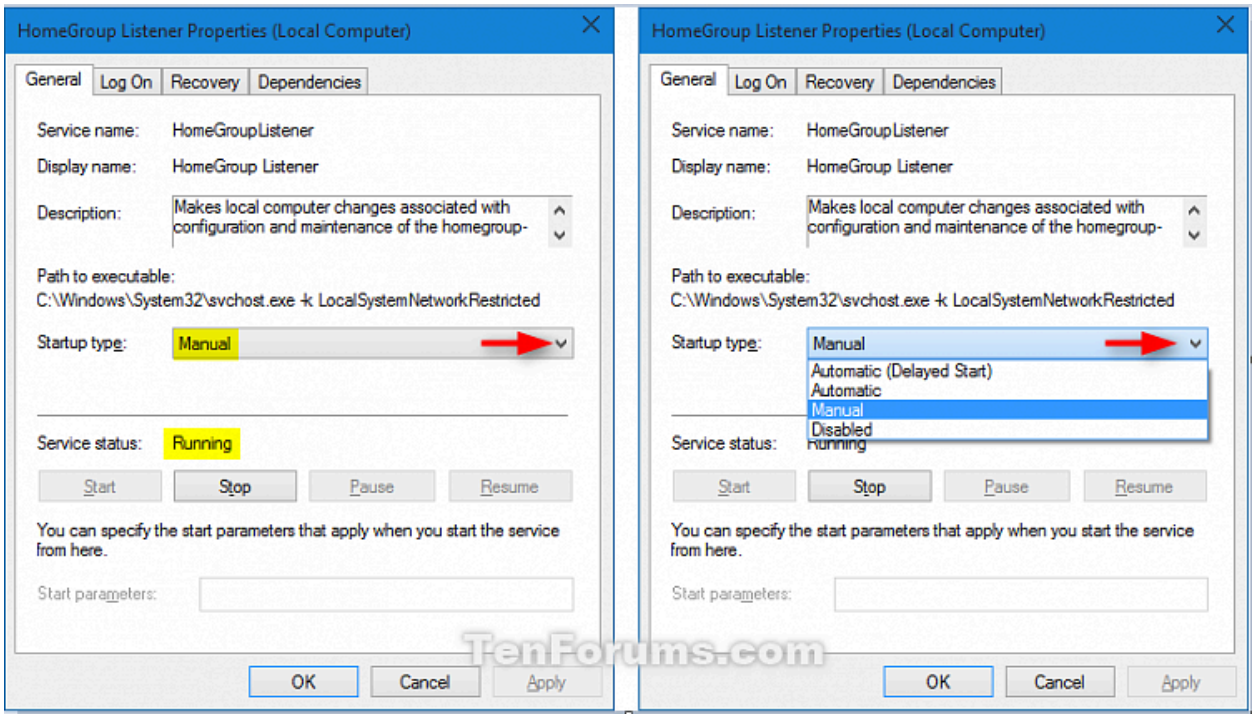
7. To Disable a Service

A) Double click/tap on a service with that you want to disable. (see screenshot below **step 4**)

B) If the service shows a status of running, then click/tap on the **Stop** button, and wait until the service status shows as stopped. (see left screenshot below **step 8**)

C) Change the **Startup type** to **Disabled**, and go to **step 8** below. (see right screenshot below **step 8**)

8 When finished, click/tap on **OK**, and close the Services window. (see screenshots below)



OPTION TWO

To Start and Stop Services using net Command

1 Open an **elevated command prompt**, and do **step 2** (stop) or **step 3** (start) below for what you would like to do.

2. To Stop a Service using "Net Stop" Command in Command Prompt

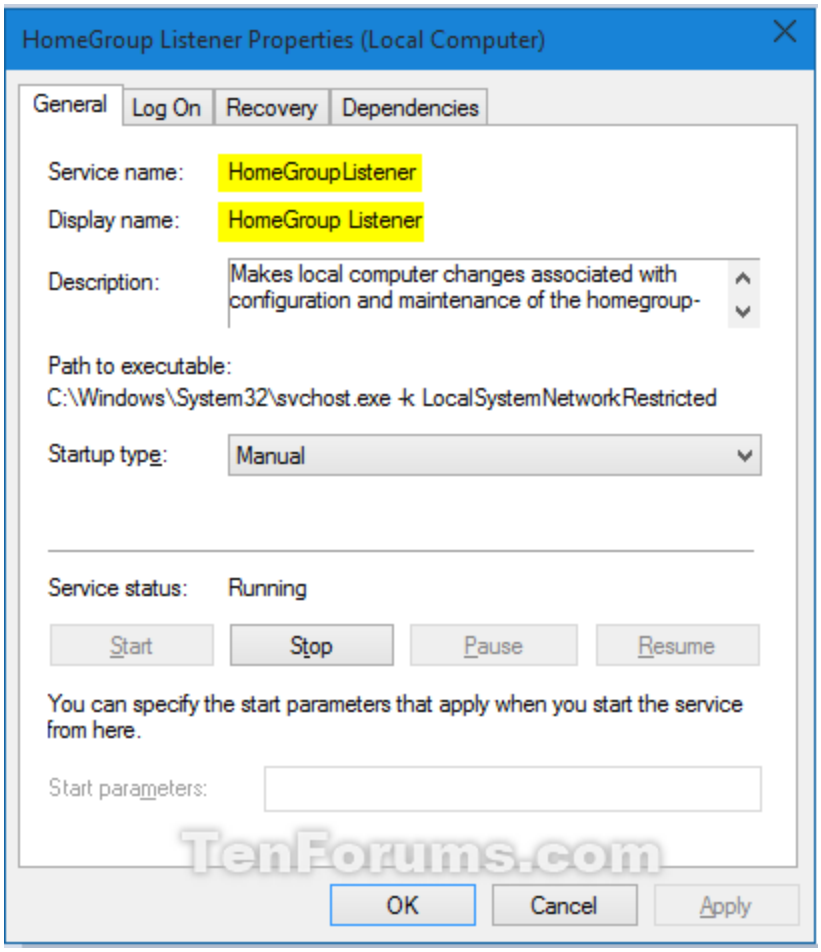
A) Type either command below into the elevated command prompt, press Enter, and go to **step 4** below. (see screenshots below)

The **Display name** of a service is the name displayed in the **Services** snap-in window, and in the service's properties.

net stop "service name"

OR

net stop "display name of service"



For example:

If I wanted to stop the **HomeGroup Listener** (display name) or **HomeGroupListener** (service name) service, I would type either command below exactly in the command prompt, and press Enter.

net stop "HomeGroup Listener"

OR

net stop "HomeGroupListener"

3. To Start a Service using "Net Start" Command in Command Prompt

A) Type either command below into the elevated command prompt, press Enter, and go to **step 4** below. (see screenshot below)

The **Display name** of a service is the name displayed in the **Services** snap-in window, and in the service's properties.

net start "service name"

OR

net start "display name of service"

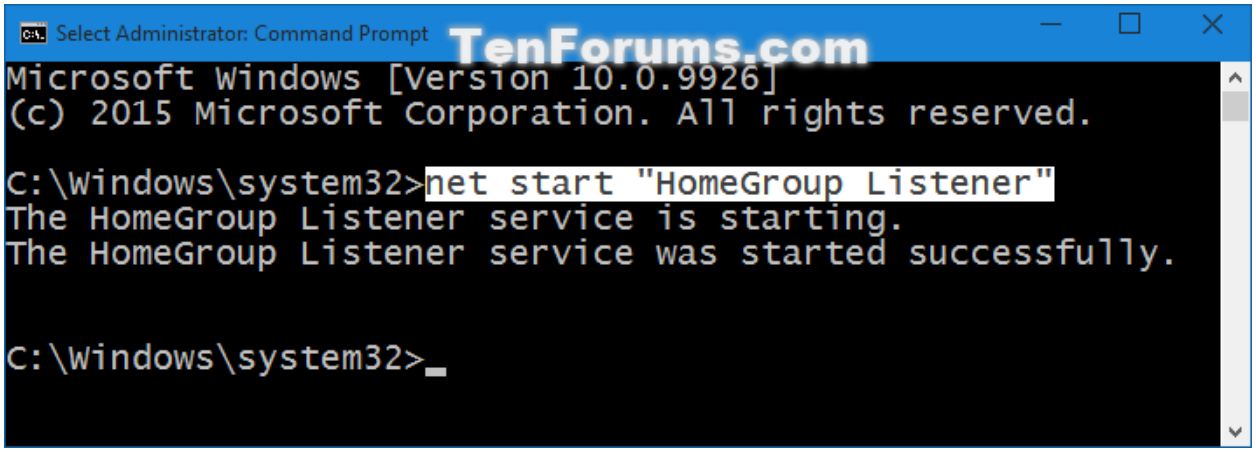
For example:

If I wanted to start the **HomeGroup Listener** (display name) or **HomeGroupListener** (service name) service, I would type either command below exactly in the command prompt, and press Enter.

net start "HomeGroup Listener"

OR

net start "HomeGroupListener"



4 When finished, you can close the elevated command prompt.

OPTION THREE

To Start, Stop, and Disable Services using Sc Command

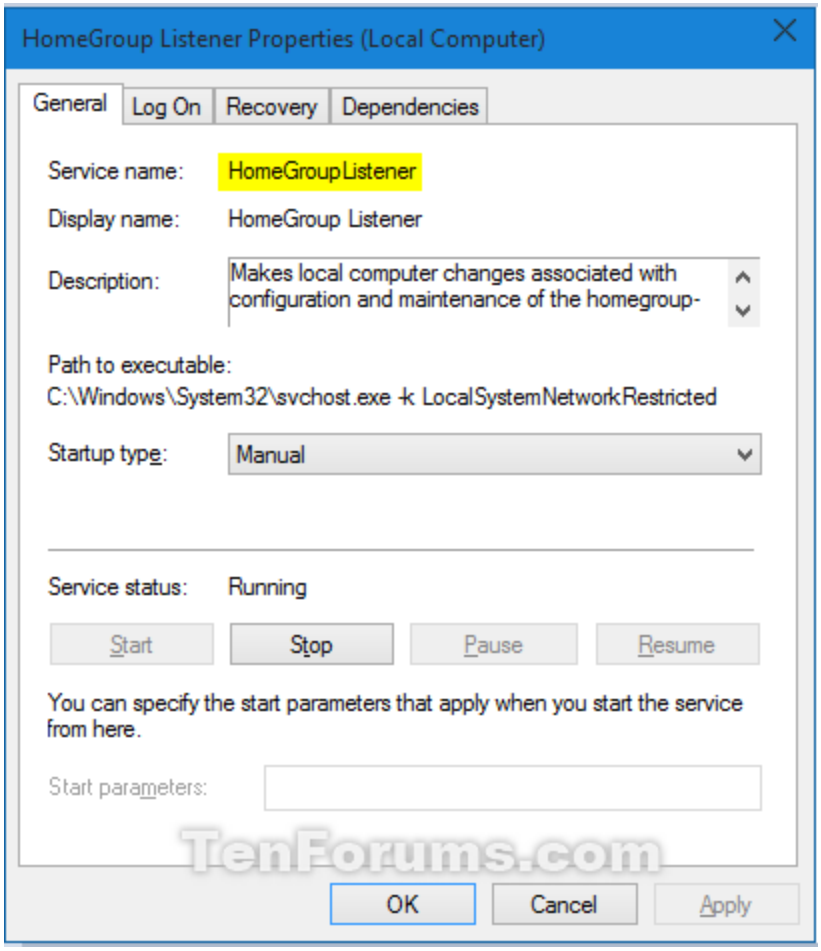
1 Open an **elevated command prompt**, and do **step 2** (stop), **step 3** (disable), **step 4** (enable), or **step 5** (start) below for what you would like to do.

2. To Stop a Service using "Sc Stop" Command in Command Prompt

A) Type the command below into the elevated command prompt, press Enter, and go to **step 6** below.

The **Service name** of a service is displayed in the service's properties.

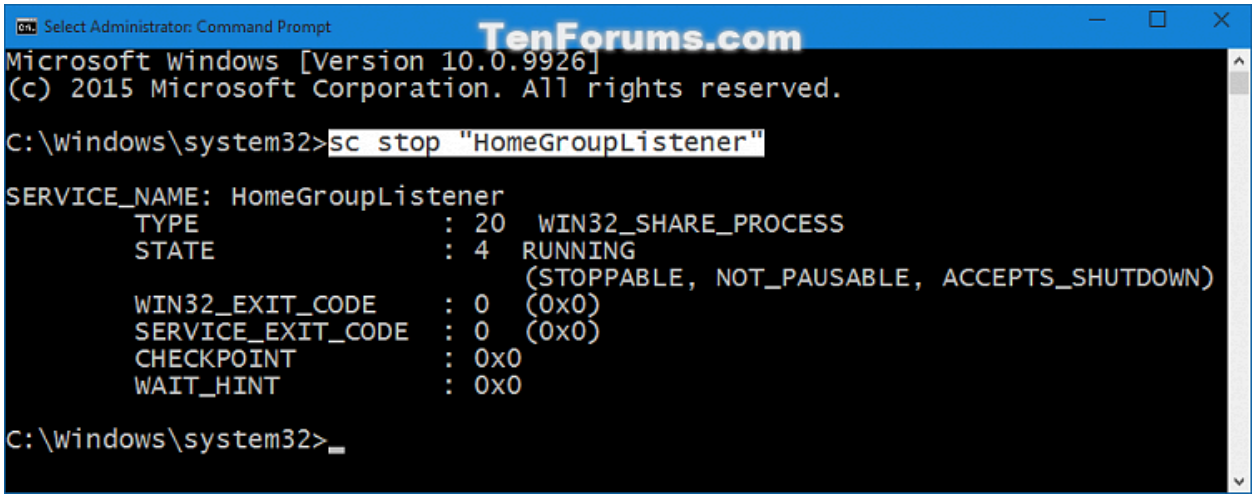
sc stop "service name"



For example:

If I wanted to stop the **HomeGroup Listener** service, I would type the command below using the **HomeGroupListener** (service name) exactly in the command prompt, and press Enter.

sc stop "HomeGroupListener"



3. To Disable a Service using "Sc Config" Command in Command Prompt

A) Do **step 2** above to stop the service, and return to continue with **step 3B** below.

B) Type the command below into the elevated command prompt, press Enter, and go to **step 6** below.

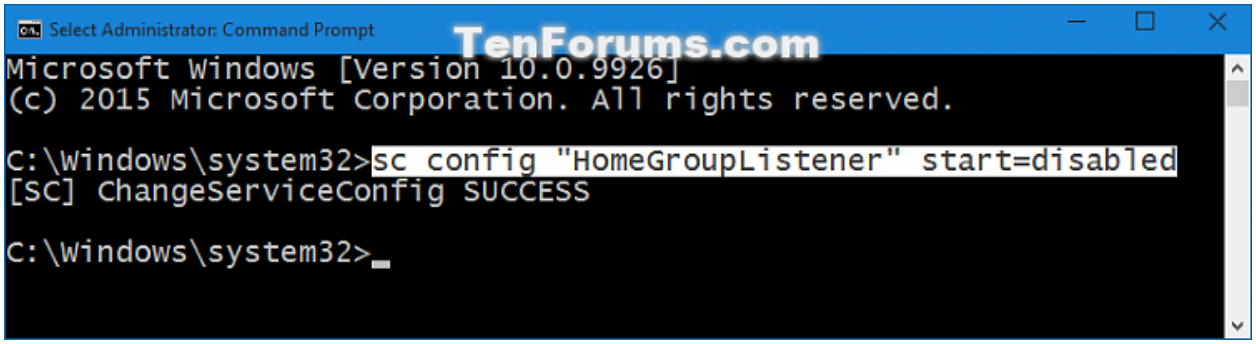
The **Service name** of a service is displayed in the service's properties.

sc config "service name" start=disabled

For example:

If I wanted to disable the **HomeGroup Listener** service, I would type the command below using the **HomeGroupListener** (service name) exactly in the command prompt, and press Enter.

sc config "HomeGroupListener" start=disabled



4. To Enable a Service using "Sc Config" Commands

A) If the **Startup type** of the service is set to **Disabled**, then in the elevated command prompt, type the command below using the startup type you want to set instead, and press Enter.

Note

The **Service name** of a service is displayed in the service's properties.

"Startup Type" for Service

Manual (demand) - Manual mode allows Windows to start a service when needed. However, very few services will start up when required in Manual mode. If you find you need a service, place it into Automatic.

Automatic (auto) - With a service in this state, it will start at boot time. Some services, when no longer required, will also automatically stop when not needed. If you find you do not need a service, place it into Manual or Disabled.

Automatic (Delayed Start) (delayed-auto) - With a service in this state, it will start just after boot time. Some services, when no longer required, will also automatically stop when not needed. If you find you do not need a service, place it into Manual or Disabled.

sc config "service name" start=demand

OR

sc config "service name" start=auto

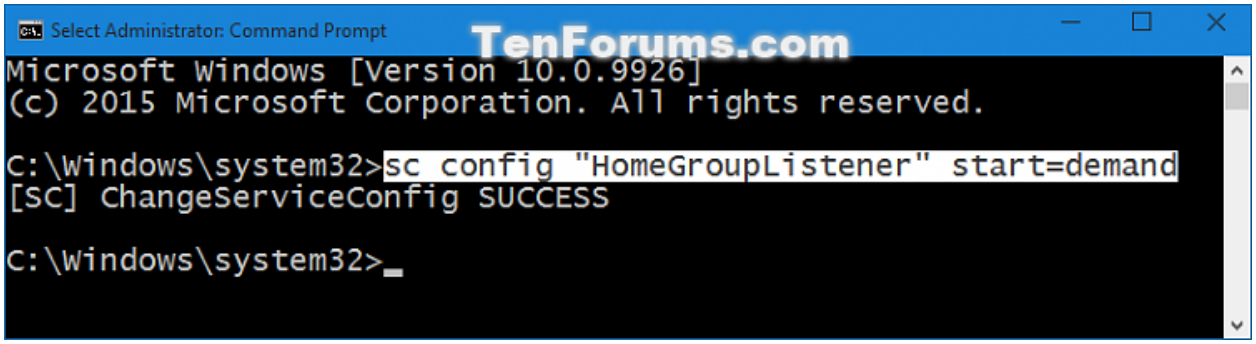
OR

sc config "service name" start=delayed-auto

For example:

If I wanted to set the startup type for the **HomeGroup Listener** service to **Manual**, I would type the command below using the **HomeGroupListener** (service name) exactly in the command prompt, and press Enter.

sc config "HomeGroupListener" start=demand



B) If you would like to start the service, then go to **step 5** below. Other wise go to **step 6** below.

5. To Start a Service using "Sc Start" Command

A) If not already, you will need to enable the service using **step 4** above first.

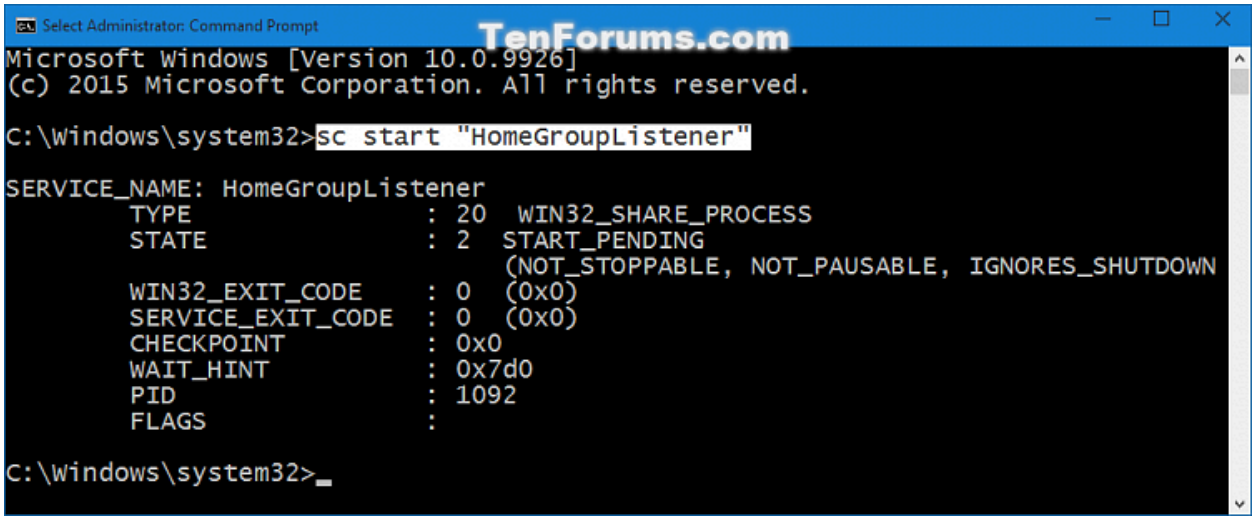
B) Type the command below into the elevated command prompt, press Enter, and go to **step 6** below.

sc start "service name"

For example:

If I wanted to start the **HomeGroup Listener** service, I would type the command below using the **HomeGroupListener** (service name) exactly in the command prompt, and press Enter.

sc start "HomeGroupListener"



6 When finished, you can close the elevated command prompt.

OPTION FOUR

To Start, Stop, and Restart Services in Task Manager

1 Open **Task Manager**, and click/tap on the **Services** tab. (see screenshot below)

2 Do **step 3**, **step 4**, or **step 5** below for what you would like to do.

3. To Start a Service

A) Right click or press and hold on a service (ex: HomeGroupListener), click/tap on **Start**, and go to **step 6** below.

4. To Stop a Service

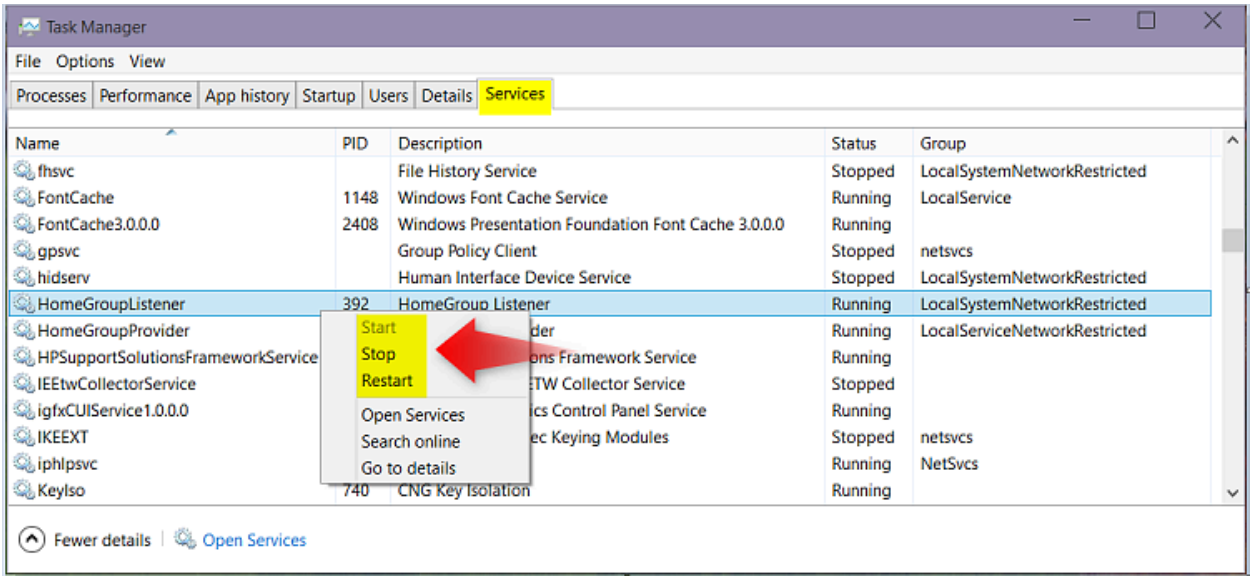
A) Right click or press and hold on a service (ex: HomeGroupListener), click/tap on **Stop**, and go to **step 6** below.

5. To Restart a Service

You will not be able to restart a service if it is stopped. The service's status needs to show as "running" (start) before you will be able to restart it.

A) Right click or press and hold on a service (ex: HomeGroupListener), click/tap on **Restart**, and go to **step 6** below.

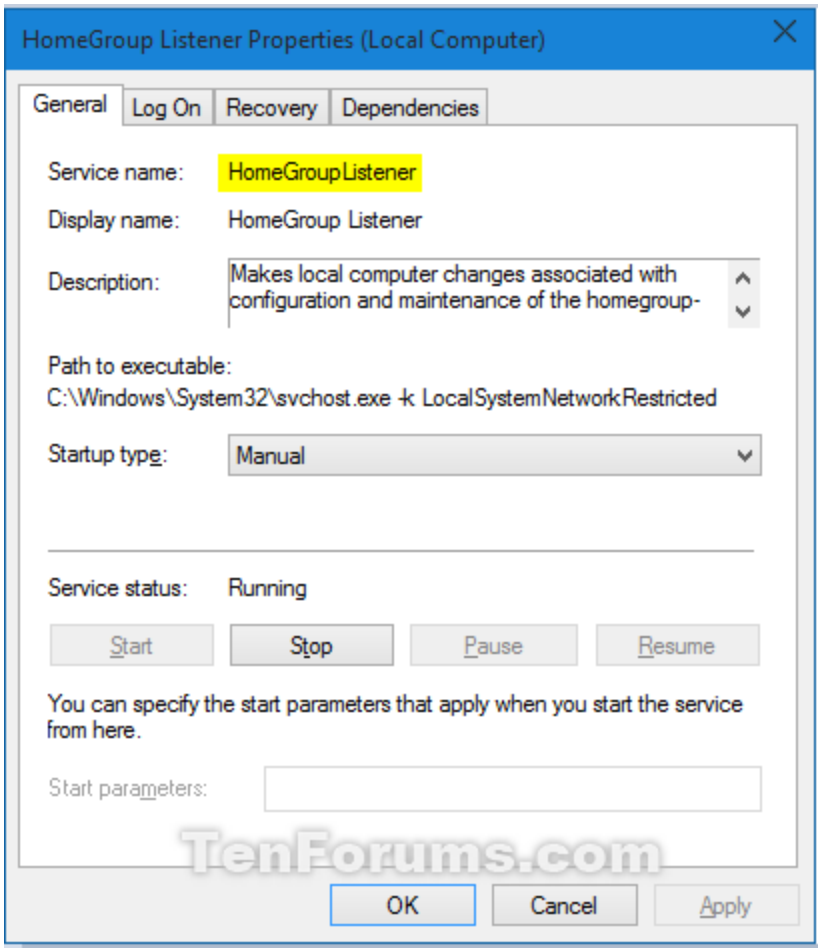
6 When finished, you can close Task Manager if you like.



OPTION FIVE

To Start, Stop, and Disable Services in Registry Editor

1 First, open **services.msc** and double click/tap on the service to see what the "**Service name**" is for the service. This will be the **Service name** you will need to use for the registry key in **step 4** below.



2 Press the Win + R keys to open the Run dialog, type **regedit** into Run, and click/tap on **OK** to open Registry Editor.

3 If prompted by **UAC**, then click/tap on **Yes**.

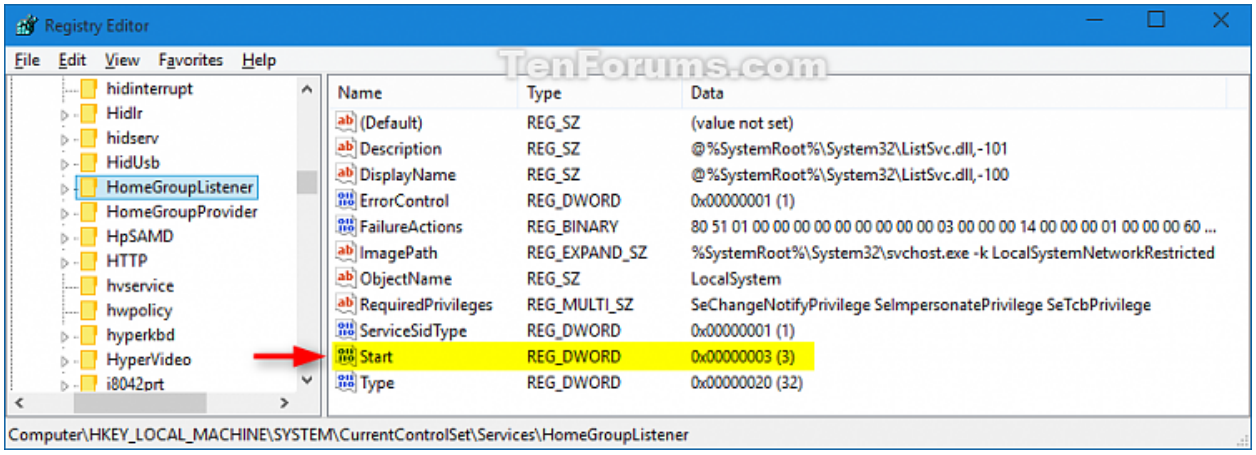
4 In Registry Editor, go to the location below: (see screenshot below)

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Service name

Substitute **Service name** in the location above with the actual service name from **step 1** above.

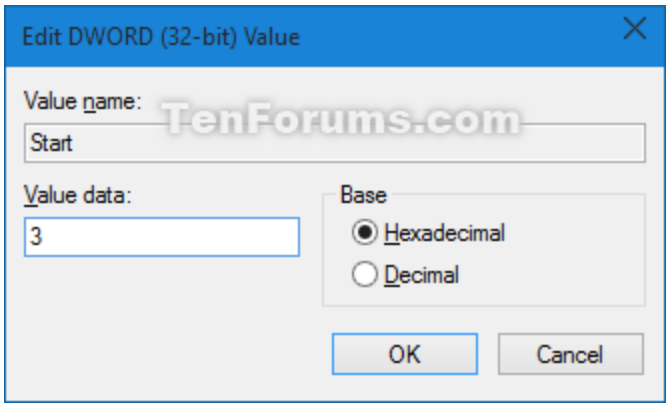
For example: The "Service name" for the HomeGroup Listener service is **HomeGroupListener**, so I would go to this location in the registry.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\HomeGroupListener



5 In the left pane of the **Service name** (ex: HomeGroupListener), double click on the **Start** DWORD to modify it. (see screenshot below)

6 Type in a **data** value from the table below for **Start** for what you would like, then click/tap on **OK**. (see screenshot below)

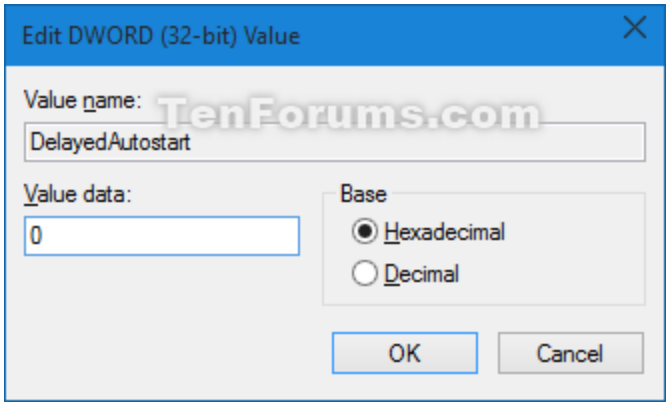


Startup Type	DWORD "Data" Value

Automatic (Delayed Start)	DelayedAutostart = **1** Start = **2**
Automatic	DelayedAutostart = **0** Start = **2**
Manual	DelayedAutostart = **0** Start = **3**
Disabled	DelayedAutostart = **0** Start = **4**

7 If needed, double click/tap on the **DelayedAutostart** DWORD to modify it's data value (0 or 1) to what's in the table above for what you want it set as, and click/tap on **OK**. (see screenshot below)

If the **DelayedAutostart** DWORD is not there, then you can right click on an empty area in the right pane, click on **New** and **DWORD (32-bit) Value**, type **DelayedAutostart**, and press Enter to add it. If **DelayedAutostart** is not there, then it will be the same as it being set to **0** (zero).



8 When finished, close Registry Editor, and **restart the computer** to apply.

OPTION SIX

To Check Status of Services in PowerShell

To see more usage options for the **Get-Service** command, see: **Get-Service - Microsoft Developer Network**

1 Open **PowerShell**.

2 Do **step 3** (specific service) or **step 4** (all services) below for what you would like to do.

3. To Check Status of a Specific Service

A) Type the command below you want to use into PowerShell, press Enter, and go to **step 5**. (see screenshot below)

Get-Service -Name "Service name" | Format-Table -Auto

OR

Get-Service -DisplayName "Display name" | Format-Table -Auto

For example:

If I wanted to check the current status of the **HomeGroup Listener** (Display name) or **HomeGroupListener** (Service name) service, I would type either command below exactly in the command prompt, and press Enter.

Get-Service -Name "HomeGroupListener" | Format-Table -Auto

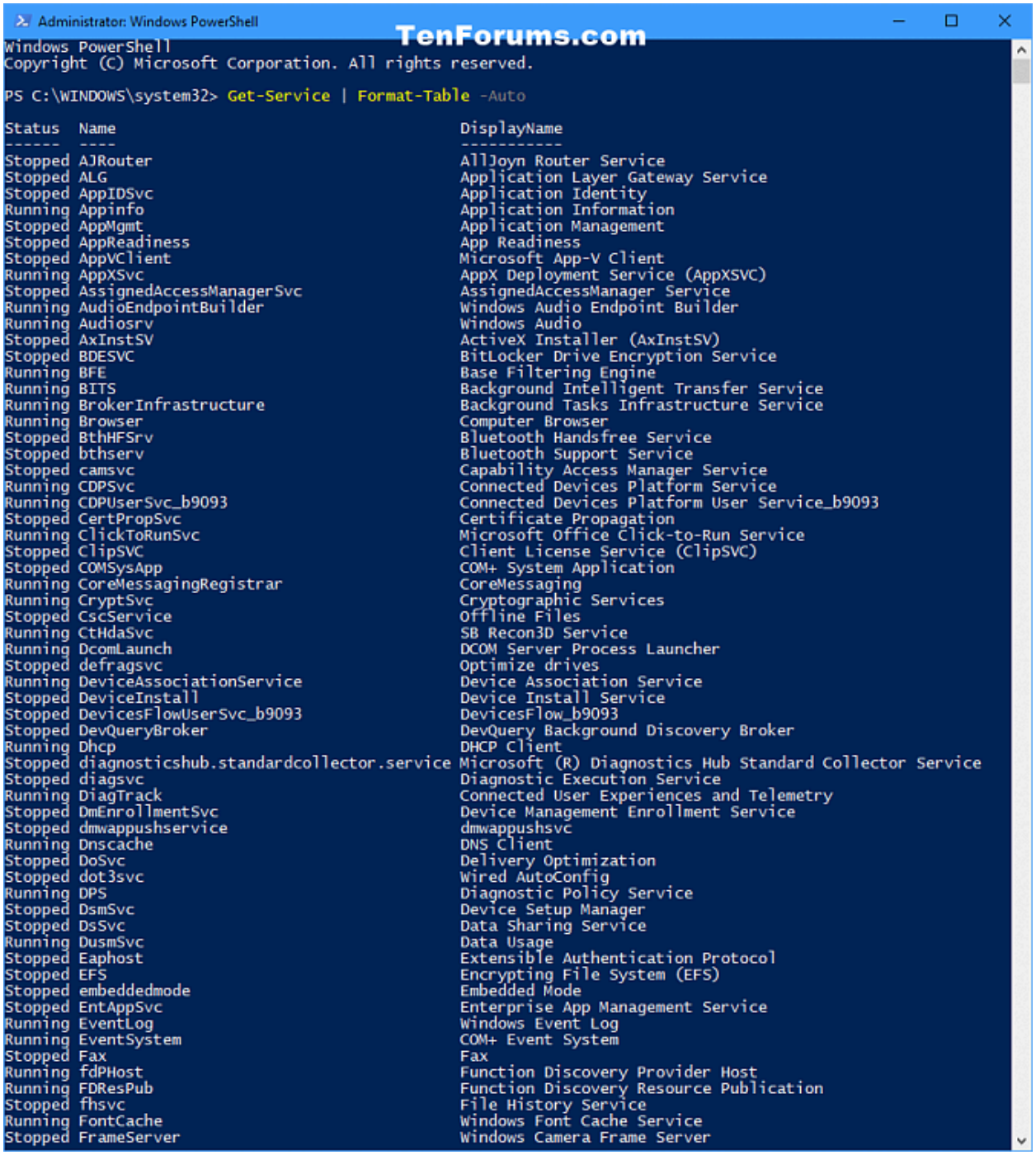
OR

Get-Service -DisplayName "HomeGroup Listener" | Format-Table -Auto

4. To Check Status of All Services

A) Type the command below into PowerShell, press Enter, and go to **step 5**. (see screenshot below)

Get-Service | Format-Table -Auto



5 When finished, you can close PowerShell if you like.

OPTION SEVEN

To Start, Stop, Restart, Disable, and Enable Services in PowerShell

1 Open an **elevated PowerShell**.

2 Do **step 3** (start), **step 4** (stop), **step 4** (restart), **step 6** (disable), or **step 7** (enable) below for what you would like to do.

3. To Start a Service

To see more usage options for the **Set-Service** command, see: **Set-Service - Microsoft Developer Network**

To see more usage options for the **Start-Service** command, see: **Start-Service - Microsoft Developer Network**

A) Type the command below you want to use into PowerShell, press Enter, and go to **step 8**. (see screenshot below)

Set-Service -Name "Service name" -Status Running

Set-Service -DisplayName "Display name" -Status Running

OR

Start-Service -Name "Service name"

Start-Service -DisplayName "Display name"

For example:

If I wanted to start the **HomeGroup Listener** (Display name)
or **HomeGroupListener** (Service name) service, I would type either command below exactly in the command prompt, and press Enter.

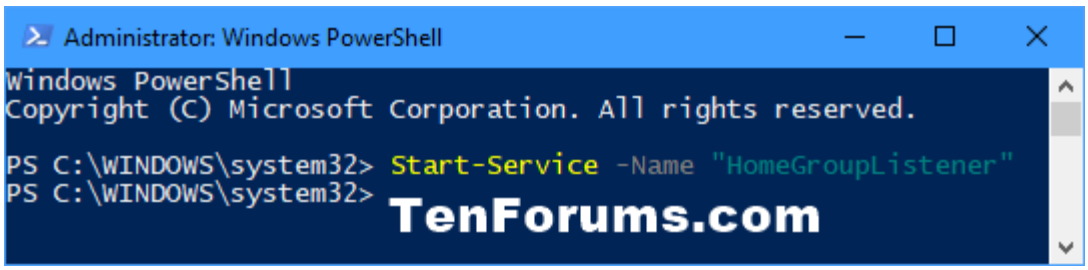
Set-Service -Name "HomeGroupListener" -Status Running

Set-Service -DisplayName "HomeGroup Listener" -Status Running

OR

Start-Service -Name "HomeGroupListener"

Start-Service -DisplayName "HomeGroup Listener"



4. To Stop a Service

To see more usage options for the **Set-Service** command, see: **Set-Service - Microsoft Developer Network**

To see more usage options for the **Stop-Service** command, see: **Stop-Service - Microsoft Developer Network**

A) Type the command below you want to use into PowerShell, press Enter, and go to **step 8**. (see screenshot below)

Set-Service -Name "Service name" -Status Stopped

Set-Service -DisplayName "Display name" -Status Stopped

OR

Stop-Service -Force -Name "Service name"

Stop-Service -Force -DisplayName "Display name"

For example:

If I wanted to stop the **HomeGroup Listener** (Display name)
or **HomeGroupListener** (Service name) service, I would type either command below exactly in the command prompt, and press Enter.

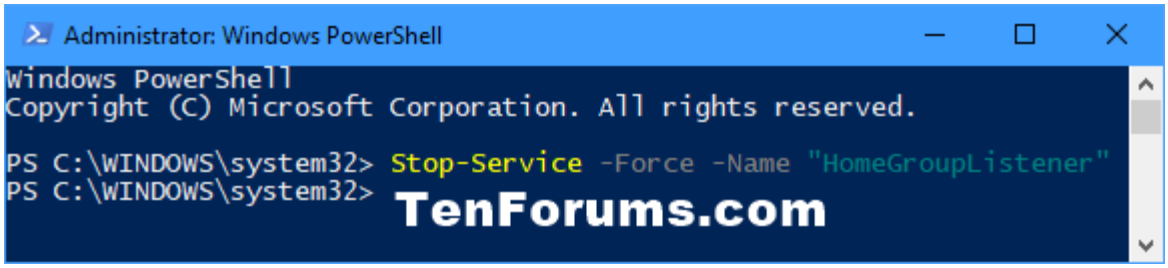
Set-Service -Name "HomeGroupListener" -Status Stopped

Set-Service -DisplayName "HomeGroup Listener" -Status Stopped

OR

Stop-Service -Force -Name "HomeGroupListener"

Stop-Service -Force -DisplayName "HomeGroup Listener"



5. To Restart a Service

To see more usage options for the **Restart-Service** command, see: **Restart-Service - Microsoft Developer Network**

A) Type the command below you want to use into PowerShell, press Enter, and go to **step 8**. (see screenshot below)

Restart-Service -Force -Name "Service name"

OR

Restart-Service -Force -DisplayName "Display name"

For example:

If I wanted to restart the **HomeGroup Listener** (Display name) or **HomeGroupListener** (Service name) service, I would type either command below exactly in the command prompt, and press Enter.

Restart-Service -Force -Name "HomeGroupListener"

OR

Restart-Service -Force -DisplayName "HomeGroup Listener"

6. To Stop and Disable a Service

To see more usage options for the **Set-Service** command, see: **Set-Service - Microsoft Developer Network**

A) Type the command below you want to use into PowerShell, press Enter, and go to **step 8**. (see screenshot below)

Set-Service -Name "Service name" -StartupType Disabled -Status Stopped

OR

Set-Service -DisplayName "Display name" -StartupType Disabled -Status Stopped

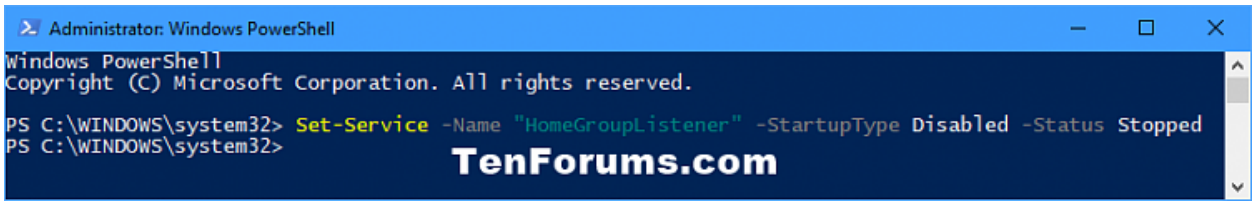
For example:

If I wanted to stop and disable the **HomeGroup Listener** (Display name) or **HomeGroupListener** (Service name) service, I would type either command below exactly in the command prompt, and press Enter.

Set-Service -Name "HomeGroupListener" -StartupType Disabled -Status Stopped

OR

Set-Service -DisplayName "HomeGroup Listener" -StartupType Disabled -Status Stopped



7. To Enable a Service

To see more usage options for the **Set-Service** command, see: **Set-Service - Microsoft Developer Network**

A) Type the command below you want to use into PowerShell, press Enter, and go to **step 8**. (see screenshot below)

Set-Service -Name "Service name" -StartupType Manual

Set-Service -DisplayName "Display name" -StartupType Manual

OR

Set-Service -Name "Service name" -StartupType Automatic

Set-Service -DisplayName "Display name" -StartupType Automatic

Note

"Startup Type" for Service

Manual - Manual mode allows Windows to start a service when needed. However, very few services will start up when required in Manual mode. If you find you need a service, place it into Automatic.

Automatic - With a service in this state, it will start at boot time. Some services, when no longer required, will also automatically stop when not needed. If you find you do not need a service, place it into Manual or Disabled.

For example:

If I wanted to enable the **HomeGroup Listener** (Display name) or **HomeGroupListener** (Service name) service, I would type either command below exactly in the command prompt, and press Enter.

Set-Service -Name "HomeGroupListener" -StartupType Manual

Set-Service -DisplayName "HomeGroup Listener" -StartupType Manual

OR

Set-Service -Name "HomeGroupListener" -StartupType Automatic

Set-Service -DisplayName "HomeGroup Listener" -StartupType Automatic

8 When finished, you can close PowerShell if you like.

That's it,
Shawn

Related Tutorials

How to Enable or Disable Scheduled Task in Windows 10

How to Restore Default Services in Windows 10

How to Delete a Service in Windows 7, Windows 8, and Windows 10

How to Export List of Running and Stopped Services in Windows

How to Add Services to Control Panel in Windows 7, 8, and 10

Fix Windows Service will not start

Windows Services are applications that typically start when the computer is booted and run quietly in the background until it is shut down. Strictly speaking, a service is any Windows application that is implemented with the services API. However, services normally handle low-level tasks that require little or no user interaction. Here are some suggestions. But before you start, create a System Restore point manually.

1. Check Services Startup type
2. Troubleshoot in Clean Boot State
3. Run SFC and DISM
4. Troubleshoot specific Services thus
5. Try this Hotfix
6. Try this Fix It
7. Use system restore
8. Reset Windows 10.

1] Check Services Startup type

To manage Windows Services, you have to open the Run box, type *services.msc* and hit Enter to open the Services Manager. Here you can set its startup type to Automatic, Delayed, Manual or Disabled. Check if the specific service with whom you are facing problems is not set to **Disabled**. See if you can start it manually by clicking on the **Start** button.

2] Troubleshoot in Clean Boot State

Boot in Safe Mode and see if the Service is starting. Many times, non-Microsoft services or Drivers can interfere with the proper functioning of System Services. Alternatively, you could also execute a Clean Boot and check.

3] Run SFC and DISM

Run the System File Checker ie. Run *sfc /scannow* from an elevated command prompt. Reboot on completion and check. Windows 10/8.1 users may repair their Windows System Image and see if it helps.

4] Troubleshoot specific Services thus

If you are facing problems in starting some specific Services, check if any of these posts can help you:

- Windows Time, Windows Firewall, Windows Event Log, services fail to start
- Windows could not start the Windows Update service on Local Computer
- Windows Time Service not working
- Windows Firewall service does not start
- Windows Event Log Service not starting
- Windows Security Center service can't be started
- Windows could not start the WLAN AutoConfig service
- Windows Search service stops
- Windows Defender Service Couldn't Be Started
- User Profile Service failed the logon
- Group Policy Client Service failed to start
- Problem uploading to the Windows Error Reporting service
- Background Intelligent Transfer Service giving problems
- Failed to connect to a Windows service
- Cryptographic Service Provider reported an error
- Windows Wireless Service is not running on this computer.

5] Try this Hotfix

If you are facing a problem with your Windows 7 or Windows Server 2008 R2 SP1 system – where you experience a long delay before all services are ready after you install an application, then visit KB2839217 and request for a hotfix. This can

typically happen when the application creates a file whose filename is longer than 127 characters.

6] Try this Fix It

If you receive an error Windows could not start the Windows Firewall, DHCP client, or Diagnostic Policy on Local Computer on Windows 7 or Windows Vista, then apply this Fix It from KB943996.

7] Use system restore

See if restoring your Windows, using a prior good system restore point helps you.

8] Reset Windows 11/10

If nothing helps, you may have using Refresh or Reset PC in Windows 11/10.

Monitoring Services with PowerShell

How to Monitor Windows Services via Powershell & Alert if Down/Up!

https://www.pcwdld.com/monitor-windows-services-via-powershell#wbounce-modal

Marc Wilson

There are many ways to monitor Windows services but one of the most flexible methods is via PowerShell (PS).

The advanced PS scripts referenced in this post will help you detect when a Windows service is stopped, when it is back up again, and will ultimately send an email alert with details.

To help automate monitoring, you can configure PS to schedule tasks to run scripts daily, at different intervals.

The few critical Windows OS services are started when the server boots and are stopped when the server shuts down.

These services need to be continuously running and they don't have a reason to stop their operations while the server is up.

But on the other hand, there are applications such as IIS, Exchange, Active Directory that are running on a Windows server.

These applications need their services to be up and running but, if any of these services stop, the whole application is disrupted.

A Windows Services monitoring solution should be able to keep track of these services and send alerts when one of these services stops or is down.

There are some Windows Services monitoring solutions that can help you automate this process.

These tools include PRTG Network Monitor, SolarWinds SAM, ManageEngine, and more.

But if you have basic Powershell scripting skills, you can set up your own monitoring solution with the help of basic functions and cmdlets.

Managing Services with Cmdlets

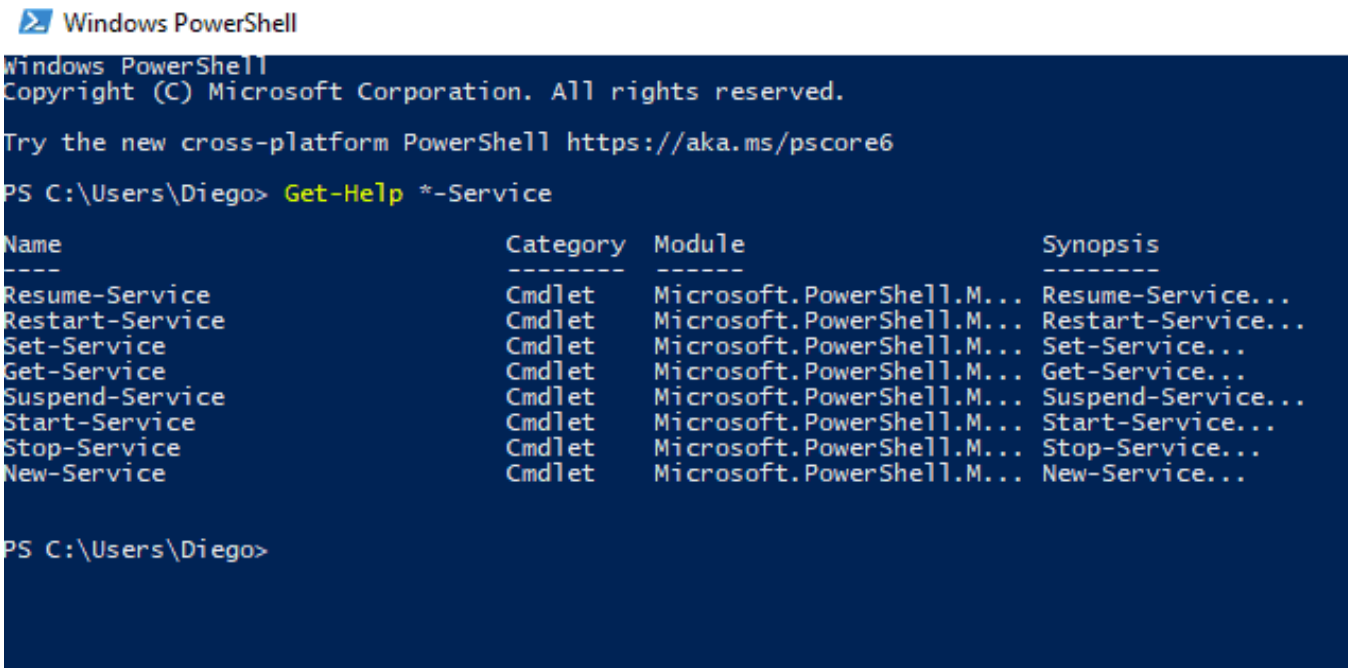
A cmdlet is a lightweight Windows PowerShell (PS) script that helps interact with the PS platform and the scripting language.

There are eight service cmdlets that let you perform an extensive number of tasks on Windows services.

These can let you query, reset, start, stop, and set services.

To get a list of the cmdlets, you can open PowerShell and use the command:

Get-Help *-Service

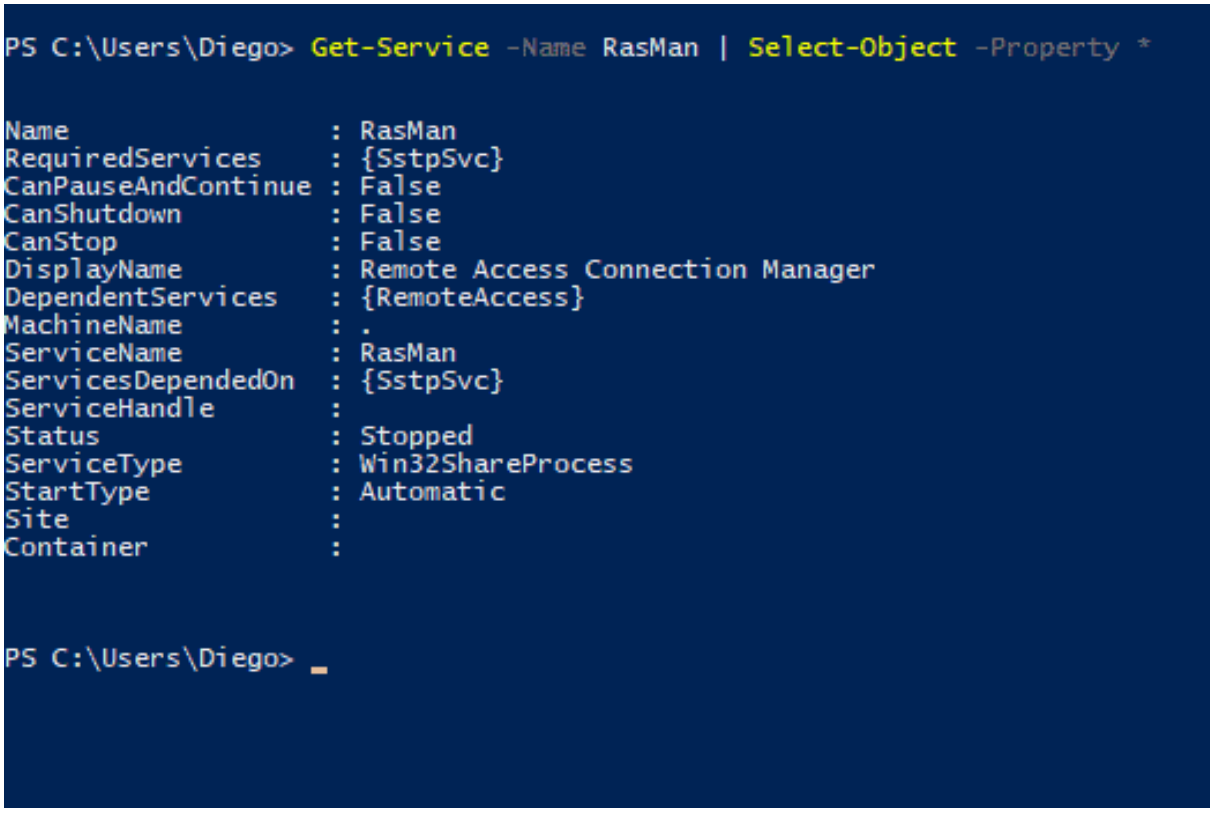


We will be using the “Get-Service” cmdlet to display the status of a service named Microsoft Remote Access Service (RasMan).

For example, let’s say that currently, RasMan is down.

Run the following command:

Get-Service -Name RasMan | Select-Object -Property *



As you can see from the output, the current “Status” of the service is “Stopped.”

From this output, you can also see other details like whether you can start it, stop it, reset it, its dependent services, type, etc.

The Get-Service is the right cmdlet to monitor services, but you can also use others to push actions such as the “Start-Service” or “Restart-Service.”

Let’s go ahead and attempt to start the service using the “Start-Service” cmdlet.

Run the following command:

Start-Service -Name RasMan

Windows PowerShell

```
PS C:\Users\Diego> Get-Service -Name RasMan | Select-Object -Property *


Name                  : RasMan
RequiredServices      : {SstpSvc}
CanPauseAndContinue   : False
CanShutdown           : True
CanStop               : True
DisplayName           : Remote Access Connection Manager
DependentServices     : {RemoteAccess}
MachineName           : .
ServiceName           : RasMan
ServicesDependedOn    : {SstpSvc}
ServiceHandle         :
Status                : Running
ServiceType           : Win32ShareProcess
StartType             : Automatic
Site                  :
Container             :
```

And now from the results shown above, you can see that the service is up and running.

Now, the good thing with PowerShell is that you can combine the results of a “Get-Service” with the actions of the “Start-Service”.

You can use this combination of both to query and then start a service if it is equal to “stopped.”

To start a service if it is not running run the following line:

Get-Service RasMan | %{if ($_.Status -eq "Stopped") { Start-Service RasMan }}

```
PS C:\Users\Diego> Get-Service RasMan | %{if ($_.Status -eq "Stopped") { Start-Service RasMan }}
PS C:\Users\Diego> _
```

Cmdlets are amazing lightweight tools that can help monitor services running on a local or remote server.

But these tools will not be capable of notifying you when a service status changes or for automating specific tasks.

The next advanced PowerShell functions will take this simple script to a new level.

Advanced PowerShell Functions

The functions shown here are able to keep track of a Windows service, attempt to restart, and even send an alert via email.

The Get-PSServiceStatus Function:

The open-source PowerShell code Get-PSServiceStatus developed by dfranciscus, suggests using the Send-MailMessage function, along with the cmdlets.

This script is capable of sending alerts via email when the service changes.

The Get-PSServiceStatus can only keep track of the last service status polled and notify you when there is a change.

To accomplish this, the script uses a text file.

If a service is down, the code creates the text file with detailed information and sends an alert.

When the service is back up, it deletes the text file.

The following is the overall flow of the function:

Check for a text file and the services.

If there is a text file, it means that the service was not running before.

If the service is not running and there is no text file, create a text file.

Send an email alert saying that the service is down.

If the service is running again, remove the text file.

Send an email saying that the service is up.

To run the function you'll need to specify the computer name (remote or local), the service name, and path name where the text files will be stored.

Other important parameters to specify are, the source email and destination email.

For example:

C:\> Get-PSServiceStatus -ComputerName test-01, -ServiceName ‘RasMan' -Path ‘C:\PSServiceStatus\' -FromAddress ‘alerts@emailsrv.com' -ToAddress ‘ITadmin@emailsrv.com' -SmtpServer ‘smtp.domain.com'

Running this script will check the computer [test-01], the service [RasMan], and notify via email and the console if the service is not running.

Download the Source Code:

The Get-PSServiceStatus.ps1 can be downloaded here: https://github.com/dfranciscus/Get-PSServiceStatus/blob/master/Get-PSServiceStatus.ps1

Other Functions

Below are two other PowerShell functions that can help you monitor Windows services, from the Microsoft web portal TechNet:

Check-Svc Function
Another great script used to monitor Windows Services via PowerShell is Microsoft’s check-Svc. The script finds a stopped service with Get-Service cmdlet, and attempts to restart it. If after the third time, it is unable to start, it will send an alert via email.

CheckServices Function
The Checkservices is another PS script that allows you to monitor the Windows Services of local and remote computers. This script will check the service status and report back in HTML all services that are running or stopped. It will also send an alert via email if the services are stopped.

Scheduling a Task to Run these PowerShell Functions

To automate the monitoring even more, you can use the Microsoft Windows Task Scheduler to automatically launch a PowerShell script, such as Get-PSServiceStatus function, at a certain time or when specific conditions are met.

For example to create a new task called “MonitorServ,” you can run the following command.

C:\Windows\System32\schtasks.exe /create /TN MonitorServ /ST 12:30 /SC DAILY /RI 10 /TR "powershell.exe C:\ Get-PSServiceStatus"

This newly created Windows Task Schedule will run the Get-PSServiceStatus function daily starting at 12:30 and every 10 minutes.

Final Words & Conclusion

A simple PowerShell script with two cmdlets can help you start a Windows service if it is stopped, much like the built-in Windows Services "Automatic Startup" function.

Throw in a scheduling task and you'll have a simple polling system.

But when you start to combine it with other functions such as the "Send-MailMessage," and advanced polling mechanisms such as the one in Get-PSServiceStatus, you can take your Windows Services monitoring to the next level.

Managing Windows Services via PowerShell

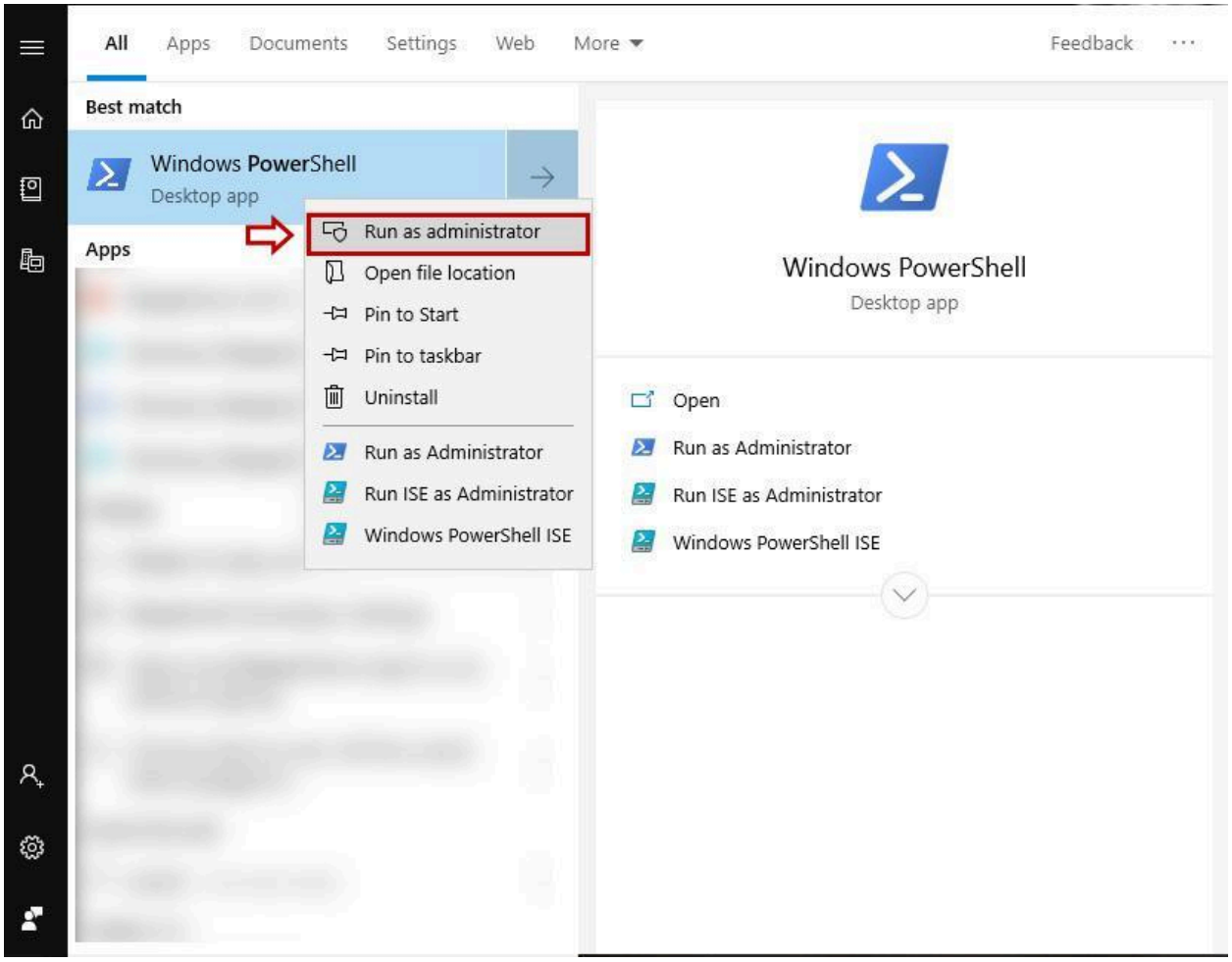
https://monovm.com/blog/management-windows-services-with-powershell/

PowerShell is a task automation and configuration management framework from Microsoft. Today we will familiarize you with how to **manage Windows services** through *PowerShell* as it is much quicker and more efficient than other methods.

As you may know, one of the most important parts of each **operating system** is the service which runs through it, and in general, it can be said that every part of the operating system that starts has a specific service that can be managed and controlled.

Here's a detailed tutorial on how to **use PowerShell to manage Windows services**.

Run a PowerShell terminal with Administrator access



At first, you must get a **list of available services in powershell** using the following command (**powershell list services**):

Get-Service

This is a sample of the output you'll receive.

PS C:\Users\Administrator> Get-Service

Status Name Display Name

------ ---- -----------

Running AcrSch2Svc Acronis Scheduler2 Service

```
Running  AdobeUpdateService AdobeUpdateService
Running  afcdpsrv          Acronis Nonstop Backup Service
Running  AGMService        Adobe Genuine Monitor Service
Running  AGSService        Adobe Genuine Software Integrity Se...
Stopped  AJRouter          AllJoyn Router Service
Stopped  ALG               Application Layer Gateway Service
Running  AMD External Ev... AMD External Events Utility
Stopped  AppIDSvc          Application Identity
Stopped  Appinfo           Application Information
Stopped  AppMgmt           Application Management
Stopped  AppReadiness      App Readiness
Stopped  AppVClient        Microsoft App-V Client
Stopped  AppXSvc           AppX Deployment Service (AppXSVC)
Stopped  AssignedAccessM... AssignedAccessManager Service
Running  AudioEndpointBu... Windows Audio Endpoint Builder
Running  Audiosrv          Windows Audio
Stopped  AxInstSV          ActiveX Installer (AxInstSV)
Stopped  BcastDVRUserSer... GameDVR and Broadcast User Service_...
Stopped  BDESVC            BitLocker Drive Encryption Service
Running  BFE               Base Filtering Engine
Running  BITS              Background Intelligent Transfer Ser...
Stopped  BluetoothUserSe... Bluetooth User Support Service_24d986
```

In the output, by default, you will see 3 main sections: Status, Name and Display Name.

Now, if you want to search and list a particular service, you can filter out any of the parameters.

For example:

Show all services whose names start with **wi**:

Get-Service -Name wi*

Show all services whose display names start with **win**:

Get-Service -DisplayName win*

Note: If you want to access another computer through the network, you can see the list of services for that system with this command:

Get-Service -ComputerName Server1

An important part of **service management** is management of Dependent Services.

To access the Dependency Services list for a specific service we can use the following command:

Get-Service -Name WinDefend -DependentServices

You can also use the **Required Services** parameter to get a list of service pre-requisites.

Get-Service -Name WinDefend - RequiredServices

So with the above commands, we can find the name of the service we are looking for, see the status and related services or its prerequisites. Now we want to explain the services' management commands.

For **stopping a service with PowerShell** you can use the following command:

Stop-Service -Name Windefend

For **starting a service in PowerShell** you can use this command:

Start-Service -Name Windefend

One of the most commonly used commands to work with services is **service restarting in PowerShell** command. The structure of service restarting command is as such:

Restart-Service -Name Windefend

And lastly, the following command is used to temporarily **suspend a service in PowerShell**.

Suspend-Service -Name Windefend

These are the most commonly used command for service management in PowerShell. For more information about PowerShell commands and how they work, use the **Get-Help** command.

Windows servers can be run with Powershell, you can manage and run your services in Windows Server with PowerShell.

Managing Services

https://docs.microsoft.com/en-us/powershell/scripting/samples/managing-services?view=powershell-7.1

There are eight core Service cmdlets, designed for a wide range of service tasks . We will look only at listing and changing running state for services, but you can get a list of Service cmdlets by using Get-Help *-Service, and you can find information about each Service cmdlet by using Get-Help <Cmdlet-Name>, such as Get-Help New-Service.

Getting Services

You can get the services on a local or remote computer by using the Get-Service cmdlet. As with Get-Process, using the Get-Service command without parameters returns all services. You can filter by name, even using an asterisk as a wildcard:

PS> Get-Service -Name se*

Status Name DisplayName

------ ---- -----------

Running seclogon Secondary Logon

Running SENS System Event Notification

Stopped ServiceLayer ServiceLayer

Because it is not always obvious what the real name for the service is, you may find you need to find services by display name. You can do this by specific name, using wildcards, or using a list of display names:

PS> Get-Service -DisplayName se*

Status Name DisplayName

------ ---- -----------

Running lanmanserver Server

Running SamSs Security Accounts Manager

Running seclogon Secondary Logon

Stopped ServiceLayer ServiceLayer

Running wscsvc Security Center

PS> Get-Service -DisplayName ServiceLayer,Server

Status Name DisplayName

------ ---- -----------

Running lanmanserver Server

Stopped ServiceLayer ServiceLayer

You can use the ComputerName parameter of the Get-Service cmdlet to get the services on remote computers. The ComputerName parameter accepts multiple values and wildcard characters, so you can get the services on multiple computers

with a single command. For example, the following command gets the services on the Server01 remote computer.

Get-Service -ComputerName Server01

Getting Required and Dependent Services

The Get-Service cmdlet has two parameters that are very useful in service administration. The DependentServices parameter gets services that depend on the service. The RequiredServices parameter gets services upon which this service depends.

These parameters just display the values of the DependentServices and ServicesDependedOn (alias=RequiredServices) properties of the System.ServiceProcess.ServiceController object that Get-Service returns, but they simplify commands and make getting this information much simpler.

The following command gets the services that the LanmanWorkstation service requires.

PS> Get-Service -Name LanmanWorkstation -RequiredServices

Status Name DisplayName

------ ---- -----------

Running MRxSmb20 SMB 2.0 MiniRedirector

Running bowser Bowser

Running MRxSmb10 SMB 1.x MiniRedirector

Running NSI Network Store Interface Service

The following command gets the services that require the LanmanWorkstation service.

PS> Get-Service -Name LanmanWorkstation -DependentServices

Status Name DisplayName

------ ---- -----------

Running SessionEnv Terminal Services Configuration

Running Netlogon Netlogon

Stopped Browser Computer Browser

Running BITS Background Intelligent Transfer Ser...

You can even get all services that have dependencies. The following command does just that, and then it uses the Format-Table cmdlet to display the Status, Name, RequiredServices and DependentServices properties of the services on the computer.

```
Get-Service -Name * | Where-Object {$_.RequiredServices -or
$_.DependentServices} |

  Format-Table -Property Status, Name, RequiredServices, DependentServices
-auto
```

Stopping, Starting, Suspending, and Restarting Services

The Service cmdlets all have the same general form. Services can be specified by common name or display name, and take lists and wildcards as values. To stop the print spooler, use:

```
Stop-Service -Name spooler
```

To start the print spooler after it is stopped, use:

```
Start-Service -Name spooler
```

To suspend the print spooler, use:

```
Suspend-Service -Name spooler
```

The Restart-Service cmdlet works in the same manner as the other Service cmdlets, but we will show some more complex examples for it. In the simplest use, you specify the name of the service:

```
PS> Restart-Service -Name spooler
```

WARNING: Waiting for service 'Print Spooler (Spooler)' to finish starting...

WARNING: Waiting for service 'Print Spooler (Spooler)' to finish starting...

PS>

You will notice that you get a repeated warning message about the Print Spooler starting up. When you perform a service operation that takes some time, Windows PowerShell will notify you that it is still attempting to perform the task.

If you want to restart multiple services, you can get a list of services, filter them, and then perform the restart:

PS> Get-Service | Where-Object -FilterScript {$_.CanStop} | Restart-Service

WARNING: Waiting for service 'Computer Browser (Browser)' to finish stopping...

WARNING: Waiting for service 'Computer Browser (Browser)' to finish stopping...

Restart-Service : Cannot stop service 'Logical Disk Manager (dmserver)' because

it has dependent services. It can only be stopped if the Force flag is set.

At line:1 char:57

+ Get-Service | Where-Object -FilterScript {$_.CanStop} | Restart-Service <<<<

WARNING: Waiting for service 'Print Spooler (Spooler)' to finish starting...

WARNING: Waiting for service 'Print Spooler (Spooler)' to finish starting...

These Service cmdlets do not have a ComputerName parameter, but you can run them on a remote computer by using the Invoke-Command cmdlet. For example, the following command restarts the Spooler service on the Server01 remote computer.

Invoke-Command -ComputerName Server01 {Restart-Service Spooler}

Setting Service Properties

The Set-Service cmdlet changes the properties of a service on a local or remote computer. Because the service status is a property, you can use this cmdlet to

start, stop, and suspend a service. The Set-Service cmdlet also has a StartupType parameter that lets you change the service startup type.

To use Set-Service on Windows Vista and later versions of Windows, open Windows PowerShell with the "Run as administrator" option.

For more information, see Set-Service

See Also

Get-Service

Set-Service

Restart-Service

Suspend-Service

Recommended content

Managing Processes with Process Cmdlets - PowerShell

PowerShell provides several cmdlets that help manage processes on local and remote computers.

Performing Networking Tasks - PowerShell

This article shows how to use WMI classes in PowerShell to manage network configuration setting in Windows.

Managing Windows PowerShell Drives - PowerShell

A PowerShell drive is a data store location that you can access like a file system drive in PowerShell. By default, PowerShell includes providers that support the file system, the registry, certificate stores, and others.

about PSSessions - PowerShell

Describes PowerShell sessions (PSSessions) and explains how to establish a persistent connection to a remote computer.

Show more

New-Service PowerShell

Module:

Microsoft.PowerShell.Management

Creates a new Windows service.

Syntax

PowerShellCopy

New-Service

[-Name] <String>

[-BinaryPathName] <String>

[-DisplayName <String>]

[-Description <String>]

[-SecurityDescriptorSddl <String>]

[-StartupType <ServiceStartupType>]

[-Credential <PSCredential>]

[-DependsOn <String[]>]

[-WhatIf]

[-Confirm]

[<CommonParameters>]

Description

The New-Service cmdlet creates a new entry for a Windows service in the registry and in the service database. A new service requires an executable file that runs during the service.

The parameters of this cmdlet let you set the display name, description, startup type, and dependencies of the service.

Examples

Example 1: Create a service

PowerShellCopy

```
New-Service -Name "TestService" -BinaryPathName
'"C:\WINDOWS\System32\svchost.exe -k netsvcs"'
```

This command creates a service named TestService.

Example 2: Create a service that includes description, startup type, and display name

PowerShellCopy

```
$params = @{
  Name = "TestService"
  BinaryPathName = '"C:\WINDOWS\System32\svchost.exe -k netsvcs"'
  DependsOn = "NetLogon"
  DisplayName = "Test Service"
  StartupType = "Manual"
  Description = "This is a test service."
}
New-Service @params
```

This command creates a service named TestService. It uses the parameters of New-Service to specify a description, startup type, and display name for the new service.

Example 3: View the new service

PowerShellCopy

```
Get-CimInstance -ClassName Win32_Service -Filter "Name='testservice'"


ExitCode  : 0
Name      : testservice
ProcessId : 0
StartMode : Auto
State     : Stopped
Status    : OK
```

This command uses Get-CimInstance to get the **Win32_Service** object for the new service. This object includes the start mode and the service description.

Example 4: Set the SecurityDescriptor of a service when creating.

This example adds the **SecurityDescriptor** of the service being created.

PowerShellCopy

```
$SDDL =
"D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;B
A)(A;;CCLCSWLOCRRC;;;SU)"

$params = @{
  BinaryPathName = '"C:\WINDOWS\System32\svchost.exe -k netsvcs"'
  DependsOn = "NetLogon"
  DisplayName "Test Service"
  StartupType = "Manual"
```

```
  Description = "This is a test service."
  SecurityDescriptorSddl = $SDDL
}
New-Service @params
```

The **SecurityDescriptor** is stored in the $SDDLToSet variable.
The **SecurityDescriptorSddl** parameter uses $SDDL to set
the **SecurityDescriptor** of the new service.

Parameters

-BinaryPathName

Specifies the path of the executable file for the service. This parameter is required.

The fully qualified path to the service binary file. If the path contains a space, it must be quoted so that it is correctly interpreted. For example, d:\my share\myservice.exe should be specified as '"d:\my share\myservice.exe"'.

The path can also include arguments for an auto-start service. For example, '"d:\myshare\myservice.exe arg1 arg2"'. These arguments are passed to the service entry point.

TABLE 1

Type:	String
Aliases:	Path
Position:	1
Default value:	None
Accept pipeline input:	False
Accept wildcard characters:	False

For more information, see the **lpBinaryPathName** parameter of CreateServiceW API.

-Confirm

Prompts you for confirmation before running the cmdlet.

TABLE 2

Type:	SwitchParameter
Aliases:	cf
Position:	Named
Default value:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Credential

Specifies the account used by the service as the Service Logon Account.

Type a user name, such as **User01** or **Domain01\User01**, or enter a **PSCredential** object, such as one generated by the Get-Credential cmdlet. If you type a user name, this cmdlet prompts you for a password.

Credentials are stored in a PSCredential object and the password is stored as a SecureString.

How to use the Event Viewer to troubleshoot Windows Services

The Core Technologies Blog

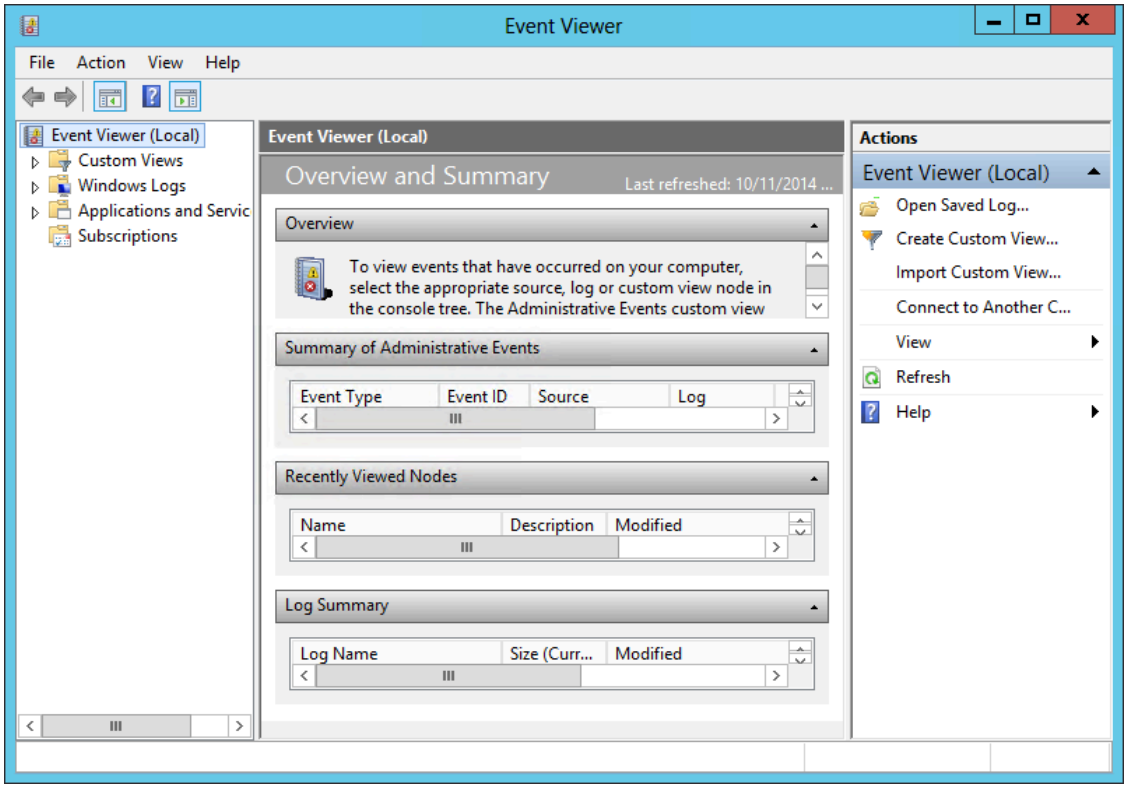
https://www.coretechnologies.com/blog/windows-services/event-viewer-troubleshoot-windows-services/

How to use the Event Viewer to troubleshoot problems with a Windows Service

A windows service, designed to run “headless” and unattended in the background, cannot easily employ conventional popup windows to report its activities as a user may not even be logged on. Instead, a service is encouraged to send important communication to the **Windows Event Log** – an administrative utility that collects and stores messages and events. Once recorded, these messages can be very helpful in troubleshooting problems, for example when a service stops unexpectedly or when it fails to start at all.

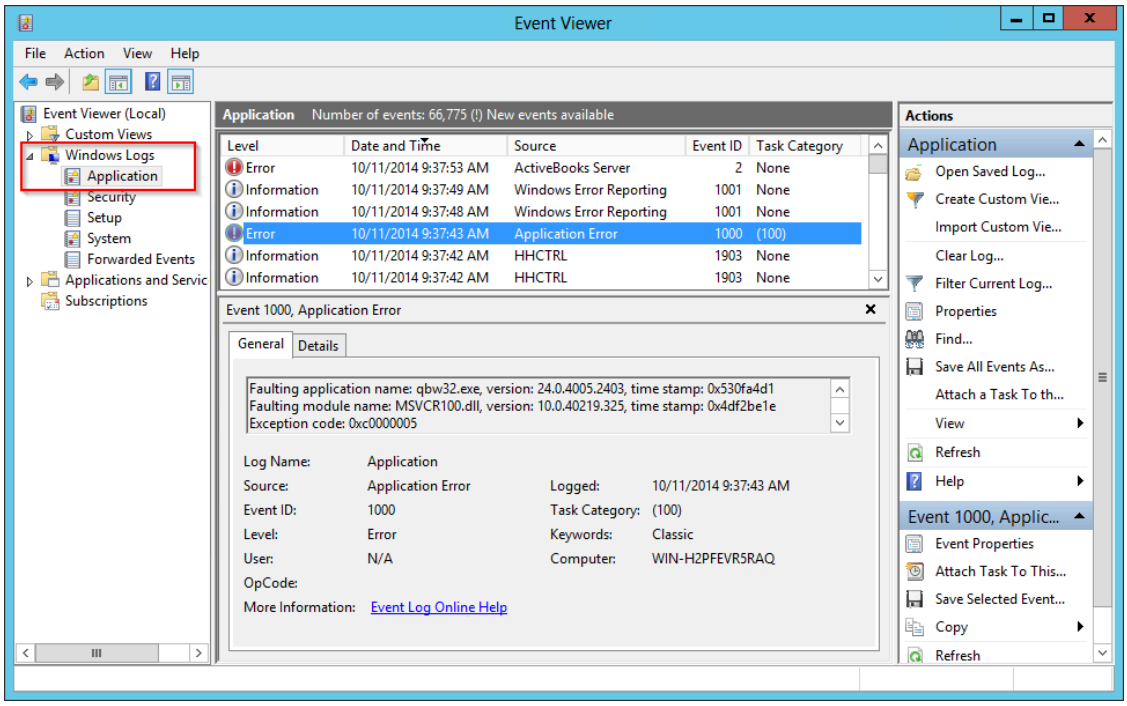
Viewing Events from Windows Services

Use Microsoft’s Event Viewer to see messages written to the Event Log. Start the application by clicking on the Start button and typing in **Event Viewer**, or from the Control Panel (search for it by name). The somewhat cluttered window should come up after a few seconds:



The left hand side shows a tree grouping the various logs captured on your machine. The events from Windows Services (and other applications running on

your PC) are filed under **Windows Logs > Application**. Navigate to that section to load the events in the center of the window, with the entire list in the top and details of the highlighted event underneath:



Messages from your windows service will have the **display name of the service in the Source column**.

Important Components of an Event

The Event Viewer shows over 10 pieces of information associated with each event, including:

Level – How important is this event?

Each event is classified into one of three categories:

Information: An informative yet unimportant event. You will probably see a lot of these, and they can be safely ignored unless you are digging into a specific issue from an application or service.

⚠ **Warning**: A moderately important event. These don't necessarily signify a failure, and your software will probably limp along, but they should be reviewed regularly to see if anything mentioned can be resolved.

Error: Indicates a critical problem or failure that may deserve your immediate attention!

Date and Time – When did this event occur?

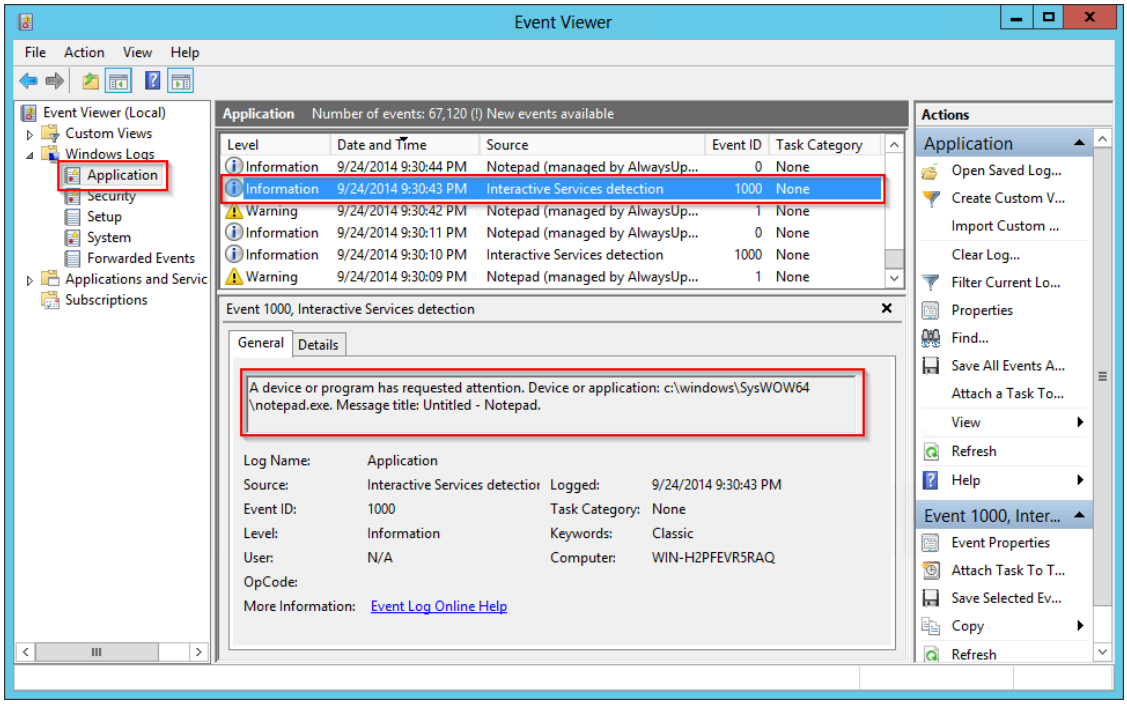
Source – Which application reported this event?

As mentioned before, an event written by a Windows Service will contain the service's display name as the Source.

Description – Which happened?

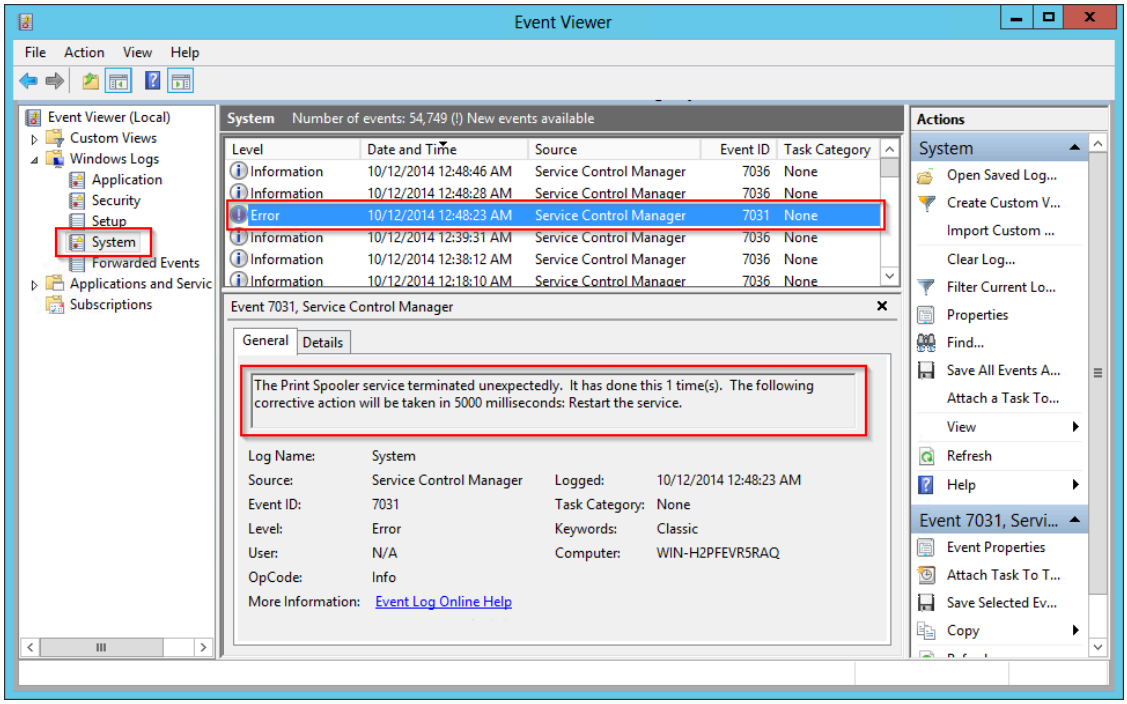
The full description shown prominently in the lower pane will (hopefully) provide the relevant details of the event.

For example, this information event is from the Interactive Services detection service ("UI0Detect") reporting that Notepad is showing itself in Session 0:



Viewing Events *about* Windows Services

While the Application log keeps track of events from a running service, the **Windows Logs > System** area records when services are started, stopped, crash or fail to start. Look for events with the Source set to **Service Control Manager** (SCM). For example, here is the SCM telling us that the Windows Print Spooler service has crashed:



Viewing Events from AlwaysUp and Service Protector

Both AlwaysUp and Service Protector write messages to the Application section of the event logs (**Windows Logs > Application**).

For AlwaysUp, events from your application named “My Application” will be logged with Source set to **My Application (managed by AlwaysUpService)**. The Event Log Messages Page lists and explains the events reported.

For Service Protector, events related to your service named “MyService” will have a Source of **ServiceProtector: MyService**.

And for both applications, events related to the starting and stopping of the underlying services themselves appear in the **Windows Logs > System** section. Look there if you have a problem with AlwaysUp itself failing to start at boot.

Troubleshooting with Windows Logs

The most common reason people look at Windows logs is to troubleshoot a problem with their systems or applications and services.

This article presents common troubleshooting use cases for security, crashes, and failed services. Examples demonstrate diagnosing the root cause of the problem using the events in your logs. Remember to check warnings and errors proceeding a critical event to see the bigger picture.

Application Failed to Log On

Well-written applications also log authentication failure events. Here's an example of a failed logon attempt in SQL Server. It includes information about who attempted to log on and why the attempt failed.

Log Name: Application

Source: MSSQLSERVER

Date: 6/25/2019 4:42:52 AM

Event ID: 18456

Task Category: Logon

Level: Information

Keywords: Classic,Audit Failure

User: N/A

Computer: PSQ-Serv-1

Description:

logon failed for user 'sa'. Reason: Password did not match that for the logon provided. [CLIENT: <local machine>]

Why Did My Server or Application Crash?

If you're investigating why your server or application crashed, the Event log is a great place to start looking. The **Application** and **System** logs can tell you when and why a crash occurred. For example, it can give you a clue if this was due to a system or application problem.

Almost all critical errors generate more than one event log entry. There are usually a number of previous warnings or errors prior to the final critical error. When troubleshooting, be sure to look at the messages preceding the crash error.

To find these events, filter your log data for a particular application name, then by critical or error events, and finally sort them by date. These are three of the most common events when troubleshooting a crash:

- Unexpected Reboot
- Application Hang
- Application Fault

Unexpected Reboot

An unexpected reboot error appears in the log when the system fails to shut down and restart gracefully. A likely cause of this error is the operating system stopped responding and crashed, or the server lost power. Look for events preceding the reboot to see a possible root cause.

Here is an excerpt from an unexpected reboot event:

Log Name: System

Source: Microsoft-Windows-Kernel-Power

Date: 25-06-2019 01:13:56

Event ID: 41

Task Category: (63)

Level: Critical

Keywords: (2)

User: SYSTEM

Computer: PSQ-Serv-1

Description:

The system has rebooted without cleanly shutting down first. This error could be caused if the system stopped responding, crashed, or lost power unexpectedly.

Application Hang

An application hang error appears in the Event log when a program running in your server stops responding. In this case, your server's hardware and the OS were functioning properly, but the application was either stuck in a loop or waiting for a resource that was not available.

This example shows how an application stopped responding to Windows and Windows shut it down.

Log Name: Application

Source: Application Hang

Date: 6/19/2019 8:31:53 PM

Event ID: 1002

Task Category: (101)

Level: Error

Keywords: Classic

User: N/A

Computer: WIN-AOTBQV71KQP

Description:

The program YourPhone.exe version 1.19053.13.0 stopped interacting with Windows and was closed. To see if more information about the problem is available, check the problem history in the Security and Maintenance control panel.

Process ID: 1428

Start Time: 01d529e6311325cf

Termination Time: 4294967295

Application Path: C:\Program Files\WindowsApps\Microsoft.YourPhone_1.19053.13.0_x64__8wekyb3d8bbwe\YourPhone.exe

Report Id: 454e89c9-d311-4a7e-8650-da9016851456

Faulting package full name: Microsoft.YourPhone_1.19053.13.0_x64__8wekyb3d8bbwe

Faulting package-relative application ID: App

Hang type: Quiesce

Application Fault

An application fault error appears in Event log when a program running in your server encounters a critical error. This error is generally a bug in the application code or an issue with memory running out. Here's an example of a faulty DLL for **svchost.exe**.

Log Name: Application

Source: Application Error

Date: 6/24/2019 8:35:59 AM

Event ID: 1000

Task Category: (100)

Level: Error

Keywords: Classic

User: N/A

Computer: WIN-AOTBQV71KQP

Description:

Faulting application name: svchost.exe_SysMain, version: 10.0.17763.1, time stamp: 0xb900eeff

Faulting module name: sysmain.dll, version: 10.0.17763.503, time stamp: 0x572b556e

Exception code: 0xc0000005

Fault offset: 0x000000000004a21c

Faulting process id: 0x798

Faulting application start time: 0x01d529e58e7ac36d

Faulting application path: C:\WINDOWS\system32\svchost.exe

Faulting module path: c:\windows\system32\sysmain.dll

Report Id: 1d7447f6-a0bb-40e8-8905-47e79dff220e

Faulting package full name:

Faulting package-relative application ID:

Finding the Root Cause of a Failed Service

A Windows service is a special kind of application that runs in the background and has its own Windows session. People often want to know why a particular service did not start or run successfully.

You can find service failures in the Application log by filtering on **Service Control Manager** source and then filtering for critical or error events. Here are common examples of failed service events.

- Service Failed to Start

- Service Timeout
- Windows Update Failure
- Scheduled Task Delayed or Failed

Service Failed to Start

This error is logged when a service fails to start normally. In this example, the **Group Policy Client** did not start in a timely fashion. The event and its message indicate when the problem happened. Check the preceding messages to track down the root cause.

Log Name: System

Source: Service Control Manager

Date: 06-21-2019 10:49:27

Event ID: 7000

Task Category: None

Level: Error

Keywords: Classic

User: N/A

Computer: PSQ-Serv-1

Description:

The Group Policy Client service failed to start due to the following error:

The service did not respond to the start or control request in a timely fashion.

Service Timeout

A service timeout error appears when a service does not start within the expected period of time (default is three seconds). Normally services are designed to start quickly and run continuously to spread out processing load. This error could be

due to the service waiting for a resource that was not available. This example shows an event generated from the **Windows Error Reporting Service**.

Log Name: System

Source: Service Control Manager

Date: 6/23/2019 11:04:00 AM

Event ID: 7009

Task Category: None

Level: Error

Keywords: Classic

User: N/A

Computer: PSQ-Serv-1

Description:

A timeout was reached (30000 milliseconds) while waiting for the Windows Error Reporting Service service to connect.

Essential Tools for Windows Services: SC.EXE

Posted on April 30, 2014

While the useful NET.EXE utility is great for starting an stopping windows services, it cannot do much beyond that. Enter Microsoft's SC.EXE – a versatile command-line utility built into Windows that can help you start, stop, restart or configure any Windows Service.

Type **SC** at a command prompt to see the extensive set of options available:

DESCRIPTION:

SC is a command line program used for communicating with the

Service Control Manager and services.

USAGE:

sc <server> [command] [service name] <option1> <option2>...

The option has the form "\\ServerName"

Further help on commands can be obtained by typing: "sc [command]"

Commands:

query----------Queries the status for a service, or

enumerates the status for types of services.

queryex---------Queries the extended status for a service, or

enumerates the status for types of services.

start----------Starts a service.

pause----------Sends a PAUSE control request to a service.

interrogate-----Sends an INTERROGATE control request to a service.

continue--------Sends a CONTINUE control request to a service.

stop------------Sends a STOP request to a service.

config----------Changes the configuration of a service (persistent).

description-----Changes the description of a service.

failure---------Changes the actions taken by a service upon failure.

failureflag-----Changes the failure actions flag of a service.

sidtype---------Changes the service SID type of a service.

privs-----------Changes the required privileges of a service.

qc--------------Queries the configuration information for a service.

qdescription----Queries the description for a service.

qfailure--------Queries the actions taken by a service upon failure.

qfailureflag----Queries the failure actions flag of a service.

qsidtype--------Queries the service SID type of a service.

qprivs----------Queries the required privileges of a service.

qtriggerinfo----Queries the trigger parameters of a service.

qpreferrednode--Queries the preferred NUMA node of a service.

delete----------Deletes a service (from the registry).

create----------Creates a service. (adds it to the registry).

control---------Sends a control to a service.

sdshow----------Displays a service's security descriptor.

sdset-----------Sets a service's security descriptor.

showsid---------Displays the service SID string corresponding to

an arbitrary name.

triggerinfo-----Configures the trigger parameters of a service.

preferrednode---Sets the preferred NUMA node of a service.

GetDisplayName--Gets the DisplayName for a service.

GetKeyName------Gets the ServiceKeyName for a service.

EnumDepend------Enumerates Service Dependencies.

The following commands don't require a service name:

sc <server> <command> <option>

boot----------(ok | bad) Indicates whether the last boot should be saved as the last-known-good boot configuration

Lock----------Locks the Service Database

QueryLock-------Queries the LockStatus for the SCManager Database

Stopping/Starting a Service with SC

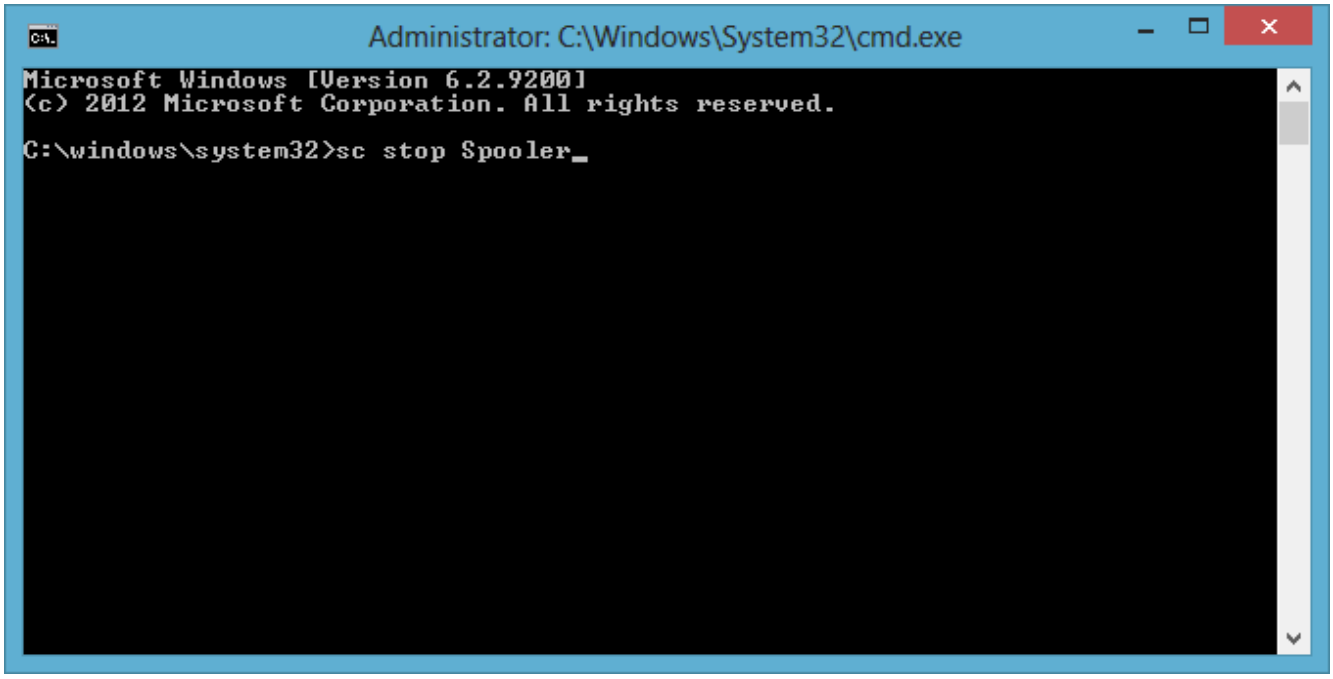
To stop a windows service from an elevated DOS prompt, run:

SC STOP <Service-Name>

where <Service-Name> is the name of the service. Be sure to enclose the name in quotes if it contains a space!

For example, to stop the Print Spooler service (named “Spooler”), run:

SC STOP Spooler



Notice that the SC command will simply **make a request for the service to stop and return immediately, before the service has actually stopped**. This is evidenced by the STOP_PENDING state (which means that the service is in the process of winding down) returned in the screenshot above. If you plan to use this command in a batch file, you may need to add a sleep/pause after calling SC to give the service some time to respond. (The NET.EXE command, which will wait/block until the service has completely stopped, may be a better choice in this respect.)

Similarly, to start a windows service, use:

SC START <Service-Name>

Again, the request will be made but SC will not wait for the service to complete its startup before returning.

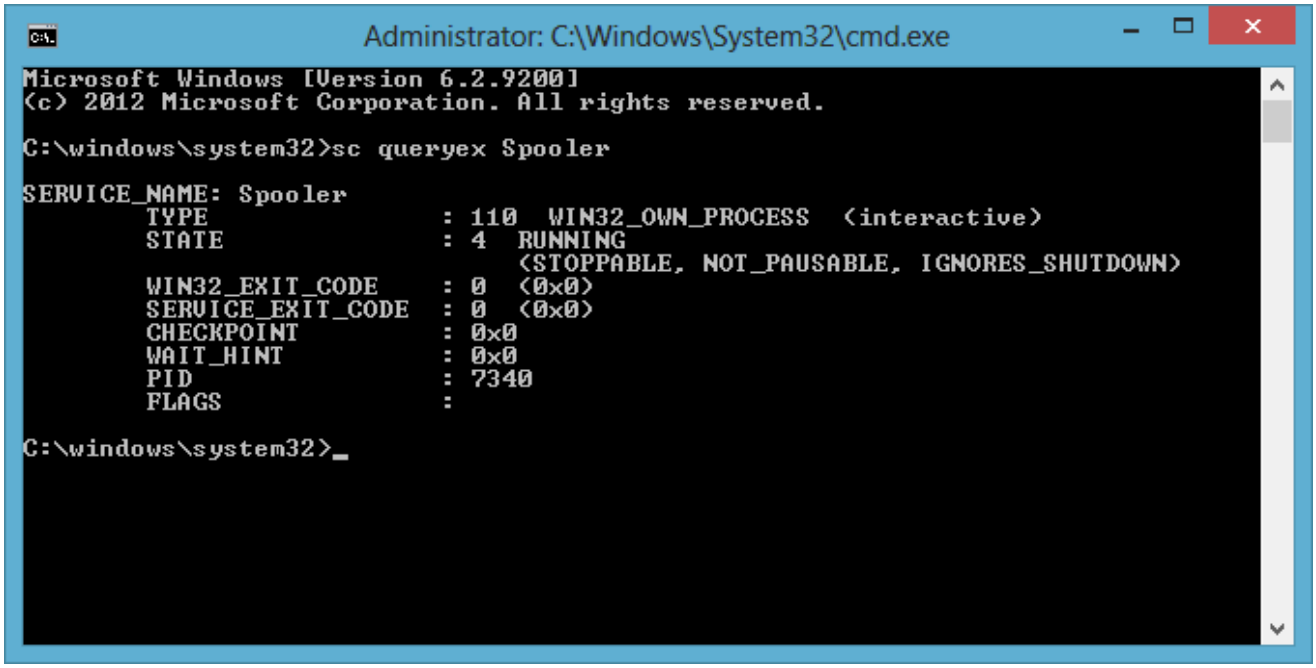
Using SC to Check the Status of a Service

To discover the state of your service, run SC with the QUERYEX option:

SC QUERYEX <Service-Name>

Check on the Spooler service like this:

SC QUERYEX Spooler



If the service is running, SC will return the underlying process identifier ("PID") which can be used to manipulate the service's process. This is very handy information when a service is stuck or unresponsive and must be forcibly terminated!

Disabling a Service

The **CONFIG** option enables you to modify a service's settings. If you wish to disable a naughty service, preventing anyone from starting it, type:

SC CONFIG <Service-Name> start= disabled

For example, this command disables the infamous Interactive Services Detection Service (named "UI0Detect"):

SC CONFIG UI0Detect start= disabled

Note that the space in between "start=" and "disabled" is required!

How to Create a New Service with SC

SC can be used to create a new service as well. Type “SC CREATE” to see the many settings that can be applied but at a minimum you must specify:

the name of the service,

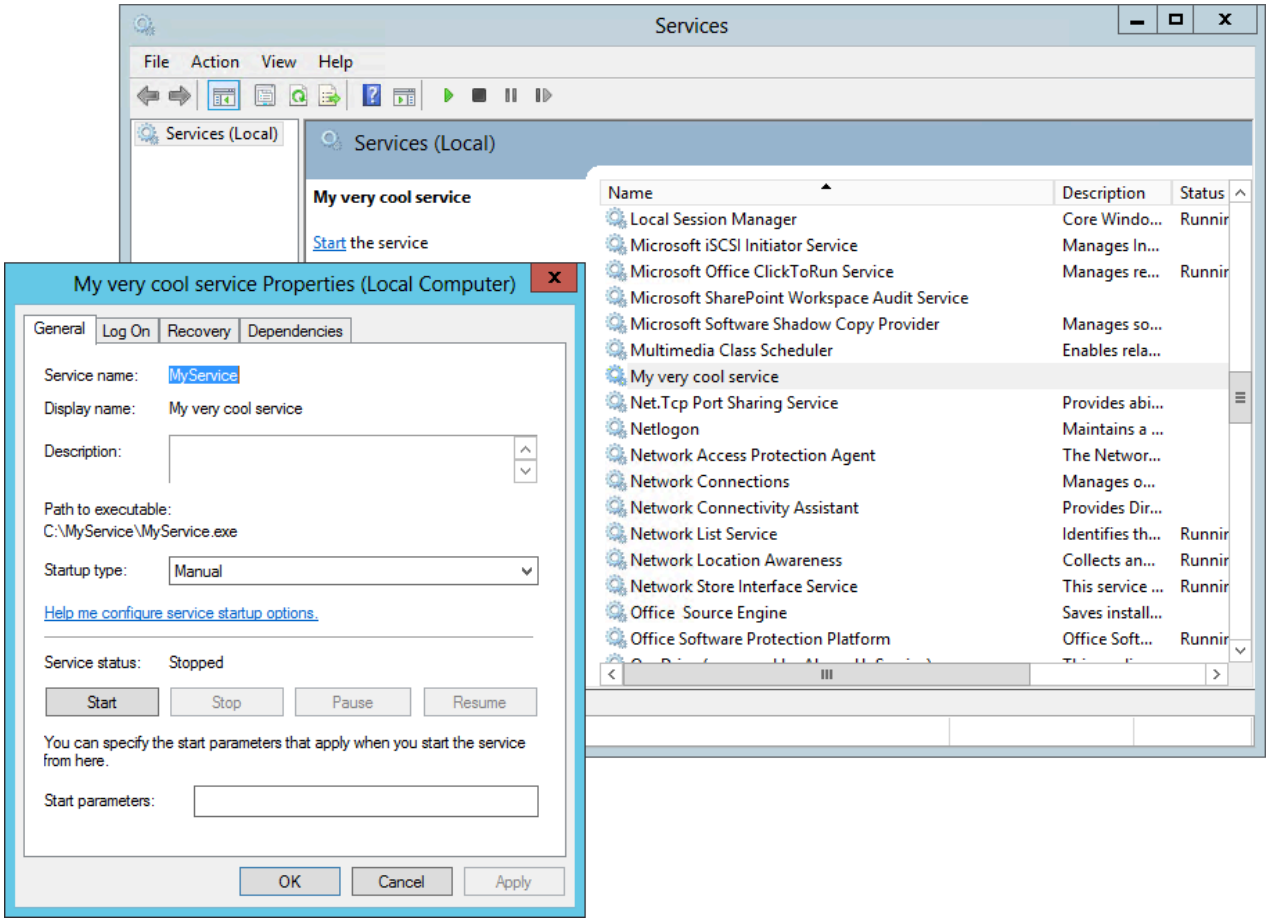
the display name of the service (a more descriptive moniker),

the full path to the executable hosting the service

For example, the following command creates a service called “MyService” with an executable located in “C:\MyService\MyService.exe”:

SC CREATE MyService binPath= “C:\MyService\MyService.exe” DisplayName= “My very cool service”

Once installed, you can work with the new service as normal in the Services application:



Note that only executables explicitly written to interface with the Windows Service Control Manager should be installed this way. While SC will happily accept a regular, non-service binary, you will receive the fatal Error 1053 when you attempt to start the service. Employ a service wrapper like AlwaysUp in that situation.

Using SC to Delete a Service

The command to remove a service with SC is straightforward:

SC DELETE <Service-Name>

To discard the service named “MyService” that we installed above, use:

SC DELETE MyService

Needless to say, please use this command with caution!. Once a service is deleted it cannot be easily re-instated. **Removing the wrong service can render your computer unusable!**

SC is the Complete Command Line Utility for Windows Services

So whenever you need to work with a service via a batch file or from a DOS command prompt, look to SC for support. This versatile, essential tool has earned its reputation as the "Swiss Army Knife" for Windows Services!

https://www.coretechnologies.com/products/

Easily find out who started, stopped or updated your Windows Services with Windows Service Auditor

Our free, portable utility enables advanced auditing and probes the Windows Event Logs to **help you investigate** your important services

It can be very difficult to figure out who (or what) keeps **messing with your essential Windows Services**.

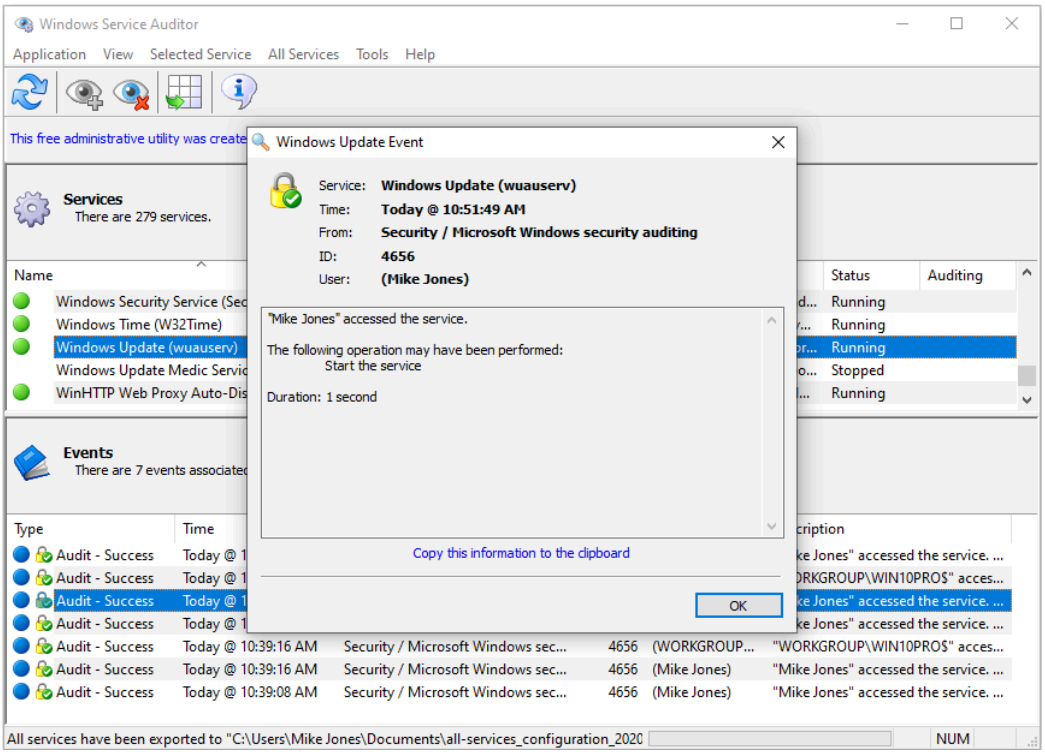
Microsoft has provided a few administrative tools to help (such as auditpol and the Event Viewer) but they are poorly documented and can be tricky to configure.

So we created Windows Service Auditor — a **free, easy-to-use application** that shines a light on your services.

Use Windows Service Auditor to help you answer burning questions, such as:

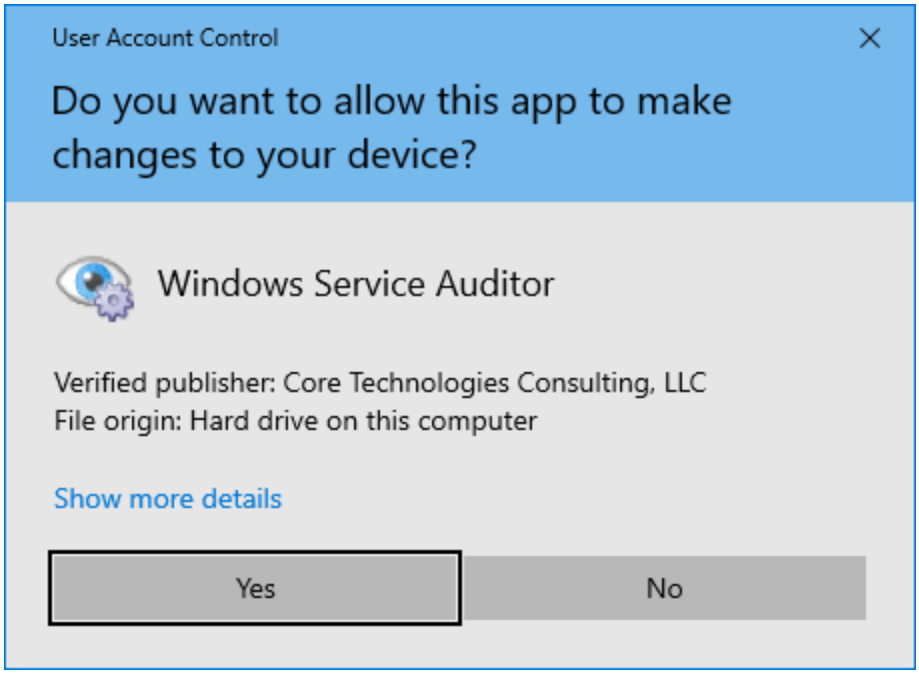
- **Who stopped my Windows Service?**
- **When was my service started?**
- **Who deleted my service?**
- **At what time did my service start?**
- **Did my service encounter any errors after it was started?**
- **Have any Windows Services been added or modified?**

The intuitive interface makes it super easy to perform your detective work:



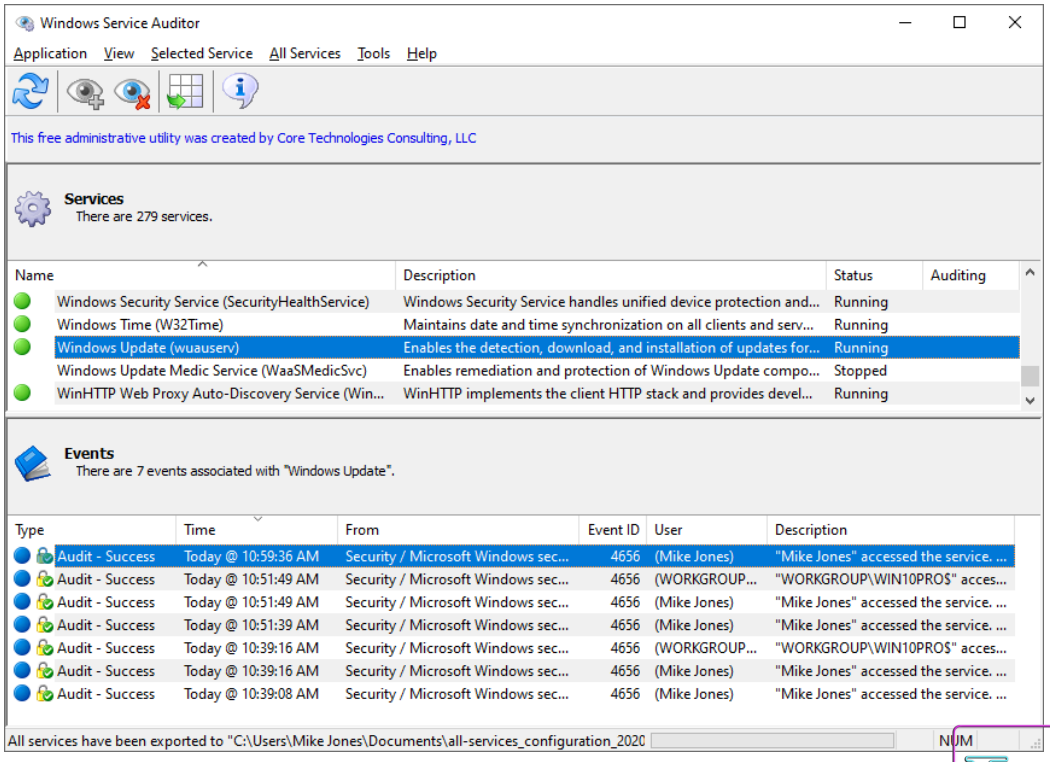
How to use Windows Service Auditor

1. Download Windows Service Auditor. Save the executable file on your desktop, or to another well-known location on your computer.
2. Double-click the **WindowsServiceAuditor.exe** file to launch the program on your desktop. If necessary, confirm the standard User Account Control (UAC) security prompt to proceed:

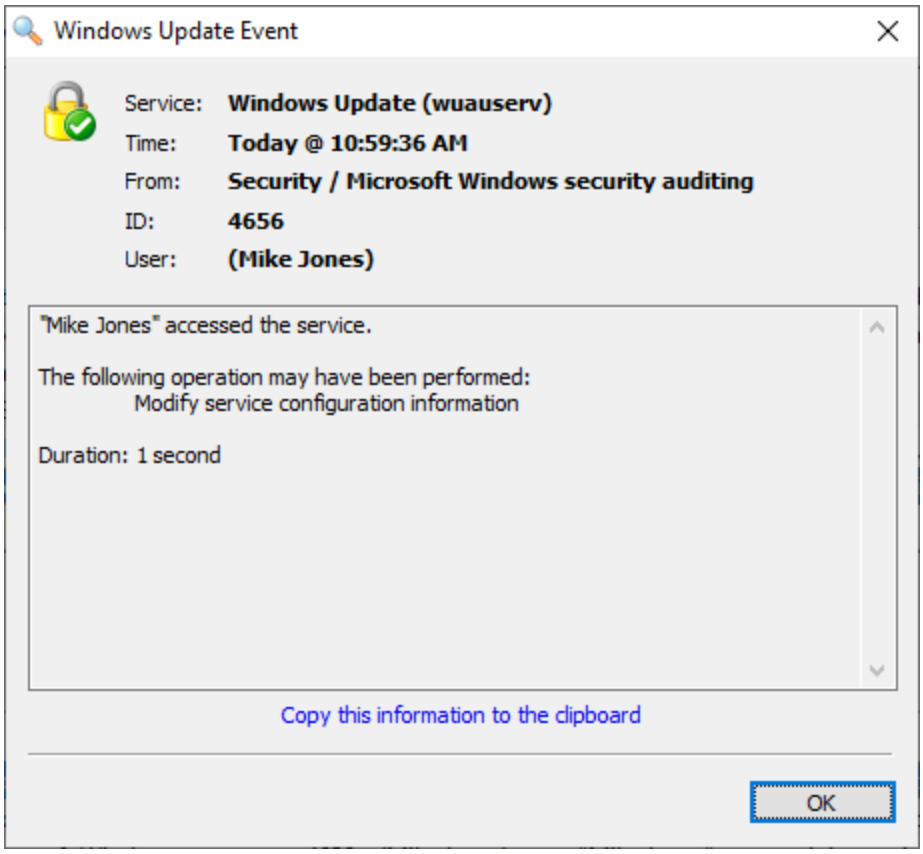


3. The window that comes up is divided into two parts. The upper pane lists every service installed on your computer while the lower panel shows the events associated with the service selected in the upper pane.

For example, here you can see the **Windows Update** service selected:

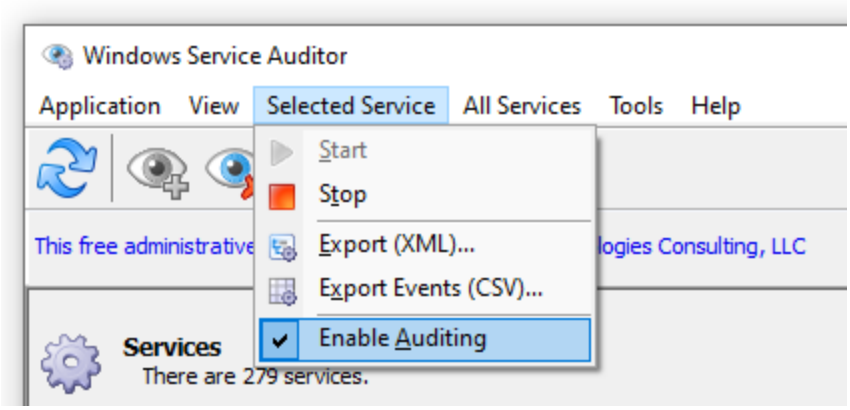


Double-click a row in the lower panel to see the event's details:

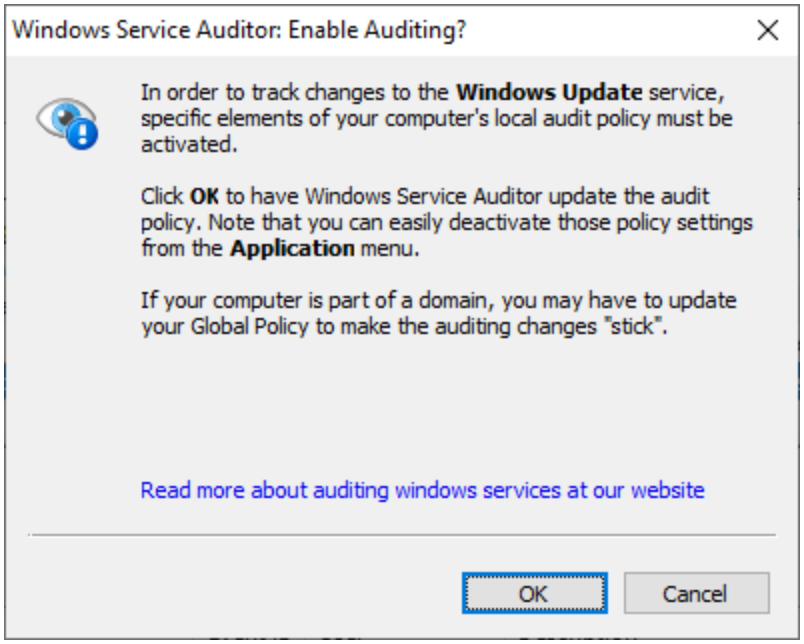


4. Unfortunately the majority of the service events will not show the account that performed the operation. That is because Windows does not keep track of user information by default. You must enable advanced security auditing to capture that level of detail.

Windows Service Auditor makes it easy to enable auditing for your service. Simply check **Enable Auditing** from the **Service** menu:

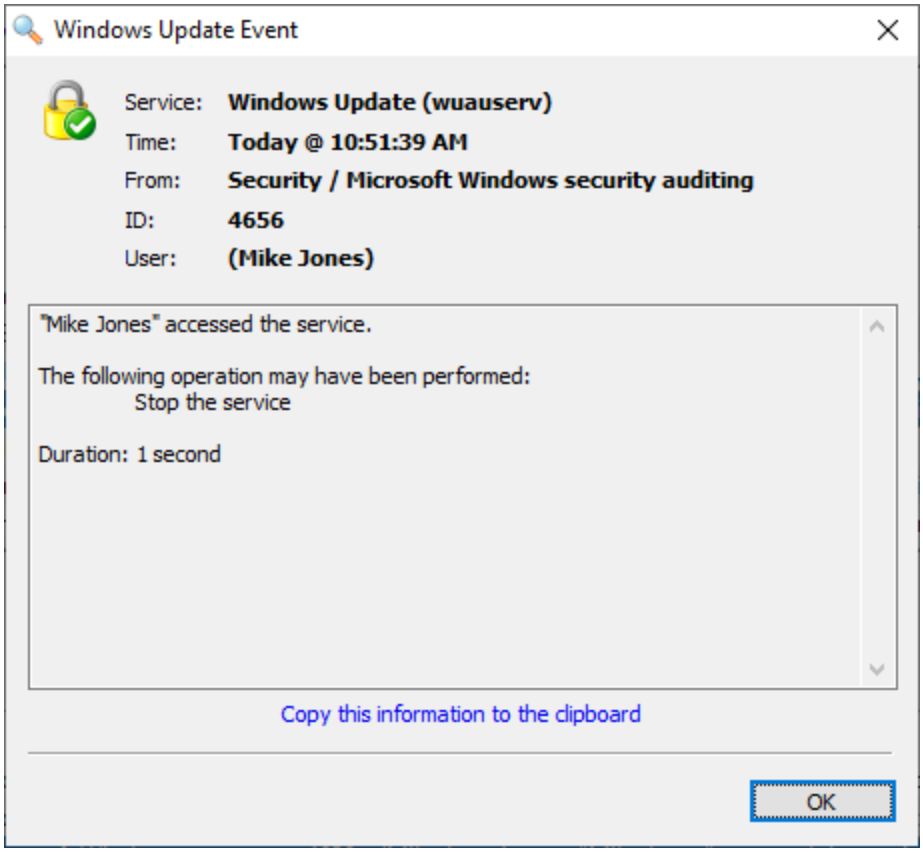


You will be prompted if your computer's Local Audit Policy must be updated. Click OK to proceed:



5. With the audit policy in place, Windows will capture detailed audit events whenever anyone tries to start, stop or update your service.

For example, this event tells us that "Mike Jones" stopped the Windows Update service today at 10:51 AM:



Exporting your services

Would you like to keep a record of the Windows Services installed on your machine? If so, select **All Services > Export (XML)** to have Windows Service Auditor save all your services to an XML file:

The XML will contain an entry for each service, including dependencies, recovery/failure options, triggers and more.

Note that you can select **Selected Service > Export (XML)** to save a single service instead.

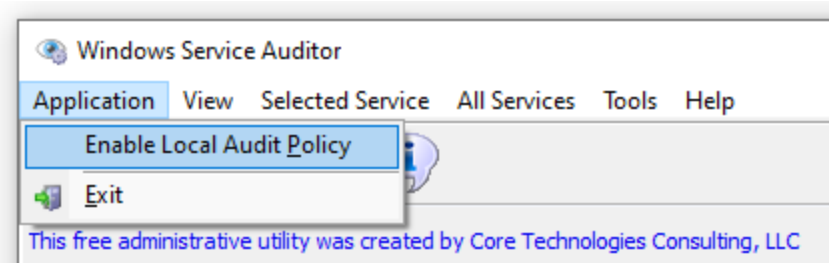
Working with Local & Global Audit Policies

In order to track users who are starting, stopping or updating a Windows Service, several advanced security audit policies must be enabled. These include:

- Other Object Access
- Handle Manipulation
- Security System Extension

Windows Service Auditor will automatically update your computer's local audit policy the first time that you enable auditing for a service. From that point on, the Event Logs will capture detailed records related to your service.

Note: At any time, you can disable advanced auditing in the areas above by simply un-checking **Enable Local Audit Policy**, available from the **Application** menu:



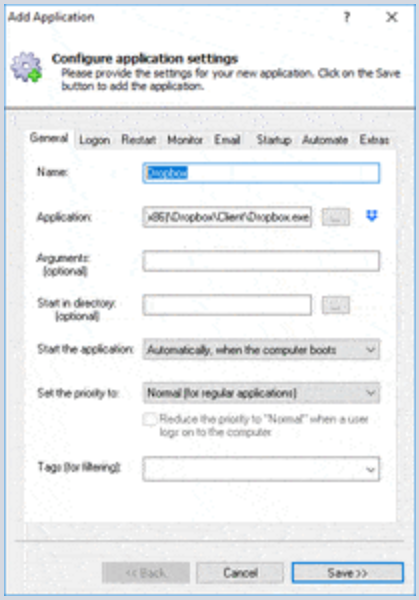
Of course, that will disable auditing for all services.

Domain Computers: Update the Global Audit Policy

If your computer is part of a domain, any changes made by Windows Service Auditor will be overwritten the next time the policy is refreshed by the server. You will have to update the Global Audit Policy yourself to enable advanced auditing.

This guide describes how to update the Global Audit Policy. Configure the system to audit success events in the **Other Object Access**, **Handle Manipulation** and **Security System Extension** areas, which can all be found in the **Security Settings / Advanced Audit Policy Configuration / Audit Policies / Object Access** section.

is **available 24/7**. And regular, detailed **email alerts** from AlwaysUp will keep you informed of performance, crashes, scheduled restarts and other relevant events.



(AlwaysUp configuration - click to enlarge)

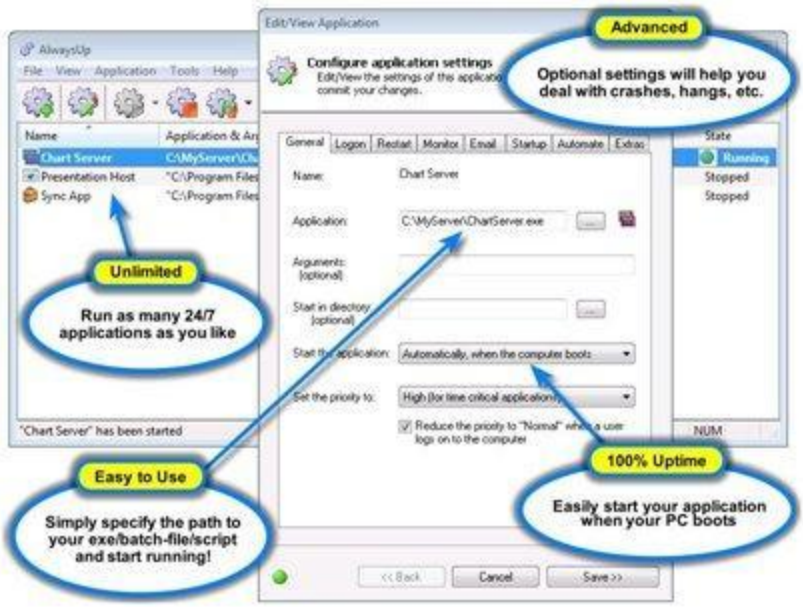
When run as a Windows Service, your application can start automatically without someone having to log on, survive user logons/logoffs, and **run entirely without user intervention**. No programming is required.

And best of all, after years of constant refinement based on client feedback and many thousands of installations on a variety of PCs, **AlwaysUp remains a trusted, reliable solution for many large and small businesses worldwide**.

Take advantage of our **risk-free 30-day trial** and find out for yourself!

An easy to use Interface to run any Program as a Windows Service

AlwaysUp's intuitive GUI makes it **very easy** to configure any application as a Windows Service. Just specify two key items (a name and the path to your application) and you are on your way! Come back for the advanced features later.



Key Features & Benefits of Running an Application as a Windows Service with AlwaysUp

- Ensure that any application **(.exe, batch file, .com, .pif, script, shortcut, perl script, java app, php, delphi, vb, etc.)** runs 24x7
- **Very easy to use** — install your application as a Windows Service in just a few seconds!
- Able to **automatically start your application when your computer boots**, to run when no user is logged in and to run **despite logon/logoffs** — all to guarantee uptime without manual user intervention
- Converts both **GUI and non-GUI** applications to run as Services
- Constantly monitors your application and **restarts it whenever it fails**
- **Secure** — no need to auto-logon to Windows
- Able to detect and **restart "misbehaving" applications** that hang, hog the CPU, or consume too much memory

- Able to restart your application (or **reboot the computer**) at a **scheduled time**
- Able to **boost your application's priority** to ensure that it gets preferential treatment on the host computer
- **Emails you** with details of crashes, restarts and other problems
- Supports the integration of your own **custom** "sanity check" utilities, executed regularly to test if your application is functioning normally or not
- Automatically **dismisses common "Application error" dialog boxes** that prevent crashed applications from fully exiting
- The intuitive GUI makes it easy to set up your application as a Windows Service, but no GUI is necessary once your application has been configured
- Supports powerful automation, to **automatically dismiss custom dialogs, fill in forms**, etc. from your program running as a Windows Service
- Reports all activities to the **Windows Event Log**
- Works flawlessly in all virtual environments (**VMWare, Virtual PC**, etc.)
- **Very efficient**; demands minimal CPU & memory resources
- Compatible with Session 0 Isolation in Windows Server 2019/2016/2012 and Windows 10/8
- Control AlwaysUp applications from your **web browser** using our free add-on program, AlwaysUp Web Service
- **Free, courteous technical support** via email or phone
- Site, OEM and Volume licensing available to suit your needs
- Branding for Enterprise, Site and OEM licensees
- **No programming** required
- **Future-proof**; tailored for Windows 10 & Server 2019 and has earned the "Certified for Windows Server 2012" logo (and others) from Microsoft.
- **Trusted** by many of the world's most recognizable companies

- Robust and stable, with an **installed base in the tens of thousands**

But perhaps most important of all, AlwaysUp was designed and implemented by **Senior Software Engineers with over 25 years of real-world experience** developing robust, mission-critical applications. Our software is of the **highest quality**, and we stand by it without reservation.

List of Windows 10 Services

AllJoyn Router Service – AllJoyn is a component of the AllSeen Alliance, run by the Linux Foundation, to enable the "Internet of Things" with devices talking to each other. Windows 10 certainly wants a future in that, so the service allows it to talk to other AllJoyn devices for home automation and many other future tasks embracing the IoT.

Andrea RT Filters Service – Sigmatel uses the Andrea ST Filters service mostly for noise cancellation. (Mostly HP Computers)

App Readiness – Gets apps ready for use the first time a user signs in to this PC, and when adding new apps.

AppX Deployment Service – Provides infrastructure support for deploying Store applications. This service is started on demand and if disabled Store applications will not be deployed to the system, and may not function properly.

Auto Time Zone Updater – While you can set the time zone manually, by using the Time zone drop-down options, you can now instead let Windows 10 do so automatically.

Bluetooth Handsfree Server – Converges Bluetooth stack excluding Hands-free profile (HFP), and Audio/Video Remote Control Profile. Migrates Bluetooth drivers for updating Windows 7 / 8.1 to 10 and allows/blocks connections and pairings based off of profiles supported.

Client License Service – Part of the Windows Store since Windows 8.1, required for all windows store services

Connected Device Platform Service – Related to Xbox and other media related services such as touch devices.

Connected User Experiences and Telemetry – Formerly named Diagnostic Tracking Service (As in spying on user info.), now renamed to CUE&T. Sends user information back to Microsoft.

Contact Data – Manages your contact information for such apps as Mail, Skype, Etc.

Core Messaging – Enables sending and receiving of SMS/MMS messages to cell phones, integrates phone apps, and Skype.

Data Sharing Service – Shares personal information from calendars, events, or other such contact information on Microsoft accounts.

Data Collection Publishing Service – Much like Data Sharing Service, however, it collects information from local and remote accounts.

Delivery Optimization – Downloads and Updates applications and windows from local network PC's and on the Internet.

Device Association Service – Serves as a framework for connecting and pairing both wired and wireless devices with Windows.

Device Install Service – Installs and Removes device driver information automatically with little or no user input.

Device Management Enrollment Service – Part of the Active Directory Service computer account for machine certificates, enrolls and registers devices.

Device Setup Manager – Checks Windows Update every night for device driver updates, and keeps Windows updated.

DevQuery Background Discovery Broker – Runs in the background enables applications to discover devices connected.

Download Maps Manager – Used to update maps through the integrated maps application and relay location information.

Embedded Mode – Access apps when using the device, also enables other functionality such as background tasks that can run forever. (IoT)

Enterprise App Management Service – Manages both Store and non-Store apps as part of the native (MDM) Mobile Device Management capabilities. New in Windows 10 is the ability to take inventory of all your apps.

File History Service – Creates backups of the Documents, Music, Pictures, Videos, and Desktop folders, along with OneDrive for offline use.

HV Host Service – Part of Hypervisor services for virtual machine hosting.

Hyper-V – Hypervisor controls all aspects of Virtual Machines on Windows 10 computers. (I paraphrase here there are many Hyper-V services.)

Infrared Monitor Service – Detects other Infrared devices that are in range and launches the file transfer application.

Local Session Manager – Core Windows Service that manages local user sessions.

Messaging Service – Enables applications running at different times to communicate across heterogeneous networks and systems that may be temporarily offline.

Microsoft (R) Diagnostic Hub Standard Collector Service – Event tracing for Windows provides performance capacity and analysis of applications by logging application, user, and kernel event information.

Microsoft Account Sign-In Assistant – Provides and end user sign-in service for Microsoft enabled accounts.

Microsoft Passport – Management and verification for user and business account information using Two-Factor authentication.

Microsoft Passport Container – Manages local user identity keys used to authenticate users to identity providers.

Microsoft Storage Spaces – Helps protect your data from drive failures and extend storage over time as you add drives to your PC. You can use Storage Spaces to group two or more drives together in a storage pool and then use capacity from that pool to create virtual drives called storage spaces.

Microsoft Windows SMS Router Service – The SMS activates a workflow service host, which creates an instance store (such as SQL Workflow Instance Store) and invokes a method on the store to load the workflow service instance from the instance store into memory.

Network Connected Devices Auto-Setup – Automatically connects to local networks using network discovery and installs qualified devices. Can be accessed in Change advanced sharing settings.

Network Connection Broker – Brokers connections that allow Windows Store Apps to receive notifications

Optimize Drives – Modern renamed version of Disk Defragmenter.

Phone Service – Allows Cortana for Android users to get missed-call notifications on their PC

Printer Extensions and Notifications – This service opens custom printer dialog boxes and handles notifications from a remote print server or a printer.

Radio Management Service – This is additional Bluetooth management support for things like airplane mode in Windows 10

Retail Demo Service – Is used during in-store sales for promotional purposes only.

Sensor Data Service – Unified sensor interface and universal driver for Windows 10 that will support a slew of environmental, biometric, proximity, health and motion sensors

Sensor Monitoring Service – Monitors various sensors in order to expose data and adapt to system and user state.

Sensor Service – Management for different sensors functionality and Simple Device Orientation.

Shared PC Account Manager – Optimizes Windows 10 for shared use scenarios, such as touchdown spaces in an enterprise and temporary customer use in retail.

Shell Hardware Detection – Service is in charge of Autoplay functions.

Smart Card Enumeration Service – Creates software device nodes for all smart card readers accessible

Spot Verifier – Service Provides Disk Corruption Spot Verification Capabilities

State Repository Service – Provides required infrastructure support for the application model. (API)

Still, Image Acquisition Events – Applications can be initiated from the scanner scan button via push *events*

Storage Service – Provides real information you can use, and tools that let you see what is using up space, and even allowing you to reclaim some of it.

Storage Tiers Management – Automated progression or demotion of data across different **tiers** of **storage** devices and media

Sync Host – This service synchronizes mail, contacts, calendar and various other user data

System Events Broker – Coordinates execution of background work for WinRT application

Tile Data Model Server – Controls and coordinates the tiles in Windows Start Menu

Time Broker – Updated version of the runtime broker service checks if an app is declaring all of its permissions (like accessing your Photos) and informing the user whether or not its being allowed.

Touch Keyboard and Handwriting Panel Service – Enables Touch Keyboard and Handwriting Panel pen and ink functionality.

Update Orchestrator Service for Windows Update – Runs Windows updating services and is responsible for rebooting after updates

User Data Access – Provides apps *access* to structured *user data*, including contact info, calendars, messages, and other content.

User Data Storage – Handles storage of structured user data, including contact info, calendars, messages, and other content.

User Manager – Windows user account and group management system

WalletService – Houses sensitive information for electronic payments such as credit card information

Windows Camera Frame Server – Connects to the data streams from webcams, the frame server supports multiple connections from applications and shares the video data from the camera to every connected app

Windows Connection Manager – Automatic connection management makes connection decisions by looking at Ethernet, Wi-Fi, and mobile broadband interfaces. These decisions lead to automatic connect and disconnect actions on Wi-Fi and mobile broadband interfaces

Windows Insider Service – Enables Windows Insiders to consume and deploy preproduction code to their test machines

Windows License Manager Service – Provides management and storage of the Licenses associated with purchased apps from the windows store.

Windows Mobile Hotspot Service – Provides the ability to share a cellular data connection with another device

Windows Push Notification System Service – Enables third-party developers to send toast, tile, badge, and raw updates from their own cloud service.

Work Folders – For file servers running Windows Server that provides a consistent way for users to access their work files from their PCs and devices

Xbox Live Auth Manager – Provides authentication and authorization services for interacting with Xbox Live.

Xbox Live Game Save – If you have more than one Xbox 360 console or you want to play *games* at a friend's house, you can store your *saved games* in the cloud

Xbox Live Networking Service – Networking protocol that's used to establish secure communications between clients and servers, and to facilitate connectivity between devices behind routers that use network address translation (NAT).

Windows 10 Service Defaults

http://revertservice.com/10/

In the table below you can see the complete list of Windows 10 service default startup configurations.

Click on a service name to learn how it works in Windows 10, get a fix for it and view its additional default settings such as dependencies, description, registry key, error control and so on.

Service name	Startup type
1394 OHCI Compliant Host Controller	Manual
3ware	Manual
ACPI Devices driver	Manual
ACPI Power Meter Driver	Manual
ACPI Processor Aggregator Driver	Manual
ACPI Wake Alarm Driver	Manual
ActiveX Installer (AxInstSV)	Manual
Acx01000	Manual
Adaptec SAS/SATA-II RAID Storport's Miniport Driver	Manual
ADP80XX	Manual

Afunix	System
Agent Activation Runtime	Manual
AllJoyn Router Service	Manual
AMD AGP Bus Filter Driver	Manual
AMD K8 Processor Driver	Manual
AMD Processor Driver	Manual
Amdsata	Manual
Amdsbs	Manual
Amdxata	Manual
Ancillary Function Driver for Winsock	System
App Readiness	Manual
AppID Driver	Manual
Application Compatibility Cache	System
Application Identity	Manual
Application Information	Manual
Application Layer Gateway Service	Manual
Application Management	Manual
AppvStrm	Manual
AppvVemgr	Manual

AppvVfs	Manual
AppX Deployment Service (AppXSVC)	Manual
AssignedAccessManager Service	Manual
Auto Time Zone Updater	Manual, Disabled
AVCTP service	Manual
Background Activity Moderator Driver	System
Background Intelligent Transfer Service	Automatic
Background Tasks Infrastructure Service	Automatic
Base Filtering Engine	Automatic
BasicDisplay	System
BasicRender	System
Bcmfn Service	Manual
Bcmfn2 Service	Manual
Beep	System
BitLocker Drive Encryption Filter Driver	Boot
BitLocker Drive Encryption Service	Manual
Block Level Backup Engine Service	Manual
Bluetooth Audio Gateway Service	Manual
Bluetooth Audio/Video Remote Control HID	Manual

Bluetooth Device (RFCOMM Protocol TDI)	Manual
Bluetooth Enumerator Service	Manual
Bluetooth Hands-Free Call Control HID	Manual
Bluetooth Handsfree Service	Manual
Bluetooth Low Energy Driver	Manual
Bluetooth Modem Communications Driver	Manual
Bluetooth Port Driver	Manual
Bluetooth Radio Driver	Manual
Bluetooth Radio USB Driver	Manual
Bluetooth Support Service	Manual
Bluetooth User Support Service	Manual
BranchCache	Manual
Browser	Manual
Capability Access Manager Service	Manual
CaptureService	Manual
CD-ROM Driver	System
CD/DVD File System Reader	Disabled
Cellular Time	Manual
Certificate Propagation	Manual

Charge Arbitration Driver	Manual
Chipidea USB Role-Switch Driver	Manual
CimFS	System
Client License Service (ClipSVC)	Manual
Clipboard User Service	Manual
CNG	Boot
CNG Hardware Assist algorithm provider	Disabled
CNG Key Isolation	Manual
COM+ Event System	Automatic
COM+ System Application	Manual
Common Driver for Buttons, DockMode and Laptop/Slate Indicator	Manual
Common Driver for HID Buttons implemented with interrupts	Manual
Common Log (CLFS)	Boot
Composite Bus Enumerator Driver	Manual
Computer Browser	Manual
Connected Devices Platform Service	Manual, Disabled, Automatic
Connected Devices Platform User Service	Automatic

Connected User Experiences and Telemetry	Automatic
ConsentUX	Manual
Console Driver	Manual
Consumer IR Devices	Manual
Contact Data	Manual
CoreMessaging	Automatic
Credential Manager	Manual
CredentialEnrollmentManagerUserSvc	Manual
Cryptographic Services	Automatic
Data Sharing Service	Manual
Data Usage	Automatic
DataCollectionPublishingService	Manual
DCOM Server Process Launcher	Automatic
Delivery Optimization	Automatic, Manual
Desktop Activity Moderator Driver	System
Device Association Service	Manual
Device Install Service	Manual
Device Management Enrollment Service	Manual

Device Management Wireless Application Protocol (WAP) Push message Routing Service	Automatic, Manual
Device Setup Manager	Manual
DeviceAssociationBroker	Manual
DevicePicker	Manual
DevicesFlow	Manual
DevQuery Background Discovery Broker	Manual
DFS Namespace Client Driver	System
DHCP Client	Automatic
Diagnostic Execution Service	Manual
Diagnostic Policy Service	Automatic
Diagnostic Service Host	Manual
Diagnostic System Host	Manual
DialogBlockingService	Disabled
Disk Driver	Boot
Disk I/O Rate Filter Driver	Boot
Display Enhancement Service	Manual
Display Policy Service	Automatic
Distributed Link Tracking Client	Automatic

Distributed Transaction Coordinator	Manual
Dmvsc	Manual
DNS Client	Automatic
Downloaded Maps Manager	Automatic
Driver Verifier Extension	Manual, Disabled
Dynamic Volume Manager	Boot
EHome Infrared Receiver (USBCIR)	Manual
Embedded Mode	Manual
Encrypting File System (EFS)	Manual
Enhanced Storage Filter Driver	Boot
Enterprise App Management Service	Manual
ExFAT File System Driver	Manual
Extensible Authentication Protocol	Manual
Family Safety Filter Driver	Manual
FAT12/16/32 File System Driver	Manual
Fax	Manual
Fcvsc	Manual
File History Service	Manual
File Information FS MiniFilter	Boot

File System Dependency Minifilter	Manual
FileCrypt	System
Filetrace	Manual
Floppy Disk Controller Driver	Manual
Floppy Disk Driver	Manual
FltMgr	Boot
Function Discovery Provider Host	Manual
Function Discovery Resource Publication	Manual
GameDVR and Broadcast User Service	Manual
Generic USB Function Class	Manual
Geolocation Service	Manual
GPU Energy Driver	System
GraphicsPerfSvc	Manual
Group Policy Client	Automatic
Hardware Policy Driver	Boot
HID driver for CapImg touch screen	Manual
HID UPS Battery Driver	Manual
High-Capacity Floppy Disk Drive	Manual
HomeGroup Listener	Manual

HomeGroup Provider	Manual
HpSAMD	Manual
HTTP Service	Manual
Human Interface Device Service	Manual
Hyper-V Data Exchange Service	Manual
Hyper-V Guest Service Interface	Manual
Hyper-V Guest Shutdown Service	Manual
Hyper-V Heartbeat Service	Manual
Hyper-V PowerShell Direct Service	Manual
Hyper-V Remote Desktop Virtualization Service	Manual
Hyper-V Time Synchronization Service	Manual
Hyper-V Volume Shadow Copy Requestor	Manual
Hyperkbd	Manual
HyperVideo	Manual
I8042 Keyboard and PS/2 Mouse Port Driver	Manual
IDE Channel	Boot
IKE and AuthIP IPsec Keying Modules	Manual
Indirect Displays Kernel-Mode Driver	Manual
Infrared monitor service	Manual

Intel AGP Bus Filter	Manual
Intel Chipset SATA RAID Controller	Manual
Intel Processor Driver	Manual
Intel RAID Controller Windows 7	Manual
Intel Serial IO GPIO Controller Driver	Manual
Intel SoC GPIO Controller Driver	Manual
Intel(R) Atom(TM) Processor I2C Controller Service	Manual
Intel(R) Dynamic Device Peak Power Manager Driver	Manual
Intel(R) Power Engine Plug-in Driver	Manual, Boot
Intel(R) SATA RAID Controller Windows	Manual
Intel(R) Serial IO I2C Host Controller	Manual
Intel(R) Telemetry Service	Boot
Intelide	Boot
Interactive Services Detection	Manual
Internet Connection Sharing (ICS)	Manual
Internet Explorer ETW Collector Service	Manual
IoQos	Manual
IP Helper	Automatic
IP Network Address Translator	Manual

IP Traffic Filter Driver	Manual
IP Translation Configuration Service	Manual
IPMIDRV	Manual
IPsec Policy Agent	Manual
IPT	Manual
IR Bus Enumerator	Manual
Irda	Manual
Isapnp	Manual
IScsiPort Driver	Manual
ItSas35i	Manual
Kbldfltr	Manual
Kernel Mode Driver Frameworks service	Boot
Keyboard Class Driver	Manual
Keyboard HID Driver	Manual
KSecDD	Boot
KSecPkg	Boot
KtmRm for Distributed Transaction Coordinator	Manual
Language Experience Service	Manual
LDDM Graphics Subsystem	Manual, System

Link-Layer Topology Discovery Mapper	Manual
Link-Layer Topology Discovery Mapper I/O Driver	Automatic
Link-Layer Topology Discovery Responder	Automatic
Local Profile Assistant Service	Manual
Local Session Manager	Automatic
LSI_SAS	Manual
LSI_SAS2i	Manual
LSI_SAS3i	Manual
LSI_SSS	Manual
MA-USB Host Controller Driver	Manual
MA-USB IP Filter Driver	Manual
MBB Network Adapter Class Extension	Manual
Megasas	Manual
Megasas2i	Manual
Megasas35i	Manual
Megasr	Manual
MessagingService	Manual
Microsoft (R) Diagnostics Hub Standard Collector Service	Manual

Microsoft 1.1 UAA Function Driver for High Definition Audio Service	Manual
Microsoft Account Sign-in Assistant	Manual
Microsoft ACPI Control Method Battery Driver	Manual
Microsoft ACPI Driver	Boot
Microsoft ACPIEx Driver	Boot
Microsoft AGPv3.5 Filter	Manual
Microsoft App-V Client	Disabled
Microsoft Bluetooth A2dp driver	Manual
Microsoft Bluetooth Avrcp Transport Driver	Manual
Microsoft Bluetooth Hands-Free Profile driver	Manual
Microsoft Bluetooth HID Miniport	Manual
Microsoft Defender Antivirus Boot Driver	Boot
Microsoft Defender Antivirus Mini-Filter Driver	Boot
Microsoft Defender Antivirus Network Inspection Service	Manual
Microsoft Defender Antivirus Network Inspection System Driver	Automatic, Manual
Microsoft Defender Antivirus Service	Automatic
Microsoft driver for storage devices supporting IEEE 1667 and TCG protocols	Manual

Microsoft Edge Elevation Service (MicrosoftEdgeElevationService)	Manual
Microsoft Edge Update Service (edgeupdate)	Automatic
Microsoft Edge Update Service (edgeupdatem)	Manual
Microsoft Generic AGPv3.0 Filter for K8 Processor Platforms	Manual
Microsoft GPIO Class Extension Driver	Manual
Microsoft Hardware Error Device Driver	Manual
Microsoft Hardware Notifications Class Extension Driver	Manual
Microsoft HID Class Driver	Manual
Microsoft Hyper-V Generation Counter	Manual
Microsoft Hyper-V Guest Infrastructure Driver	Manual
Microsoft Hyper-V Storage Accelerator	Manual
Microsoft I2C HID Miniport Driver	Manual
Microsoft Infrared HID Driver	Manual
Microsoft Input Configuration Driver	Manual
Microsoft IPv6 Protocol Driver	Manual
Microsoft iSCSI Initiator Service	Manual
Microsoft Kernel Debug Network Miniport (NDIS 6.20)	Manual
Microsoft Keyboard Filter	Disabled

Microsoft Link-Layer Discovery Protocol	Automatic
Microsoft MAC Bridge	Manual
Microsoft Memory Module Driver	Manual
Microsoft Monitor Class Function Driver Service	Manual
Microsoft NDIS Capture	Manual, System
Microsoft Network Adapter Multiplexor Protocol	Manual
Microsoft Passport	Manual
Microsoft Passport Container	Manual
Microsoft Remote Desktop Input Driver	Manual
Microsoft Security Events Component Minifilter	Manual, Boot
Microsoft Software Shadow Copy Provider	Manual
Microsoft SPI HID Miniport Driver	Manual
Microsoft Standard NVM Express Driver	Manual
Microsoft Standard SATA AHCI Driver	Manual
Microsoft Storage Spaces SMP	Manual
Microsoft Store Install Service	Manual
Microsoft Streaming Clock Proxy	Manual
Microsoft Streaming Quality Manager Proxy	Manual
Microsoft Streaming Service Proxy	Manual

Microsoft Streaming Tee/Sink-to-Sink Converter	Manual
Microsoft System Management BIOS Driver	System
Microsoft Trusted Audio Drivers	Manual
Microsoft Tunnel Miniport Adapter Driver	Manual
Microsoft UAA Bus Driver for High Definition Audio	Manual
Microsoft UEFI Driver	Manual
Microsoft UMPass Driver	Manual
Microsoft Universal Flash Storage (UFS) Driver	Manual
Microsoft USB 2.0 Enhanced Host Controller Miniport Driver	Manual
Microsoft USB Generic Parent Driver	Manual
Microsoft USB Open Host Controller Miniport Driver	Manual
Microsoft USB PRINTER Class	Manual
Microsoft USB Serial Driver	Manual
Microsoft USB Standard Hub Driver	Manual
Microsoft USB Universal Host Controller Miniport Driver	Manual
Microsoft Virtual Drive Enumerator	Boot
Microsoft Virtual Network Adapter Enumerator	Manual
Microsoft WFP Message Capture	Manual

Microsoft Windows Filtering Platform	Boot
Microsoft Windows Management Interface for ACPI	Manual
Microsoft Windows SMS Router Service.	Manual
Microsoft Windows Trusted Runtime Secure Service	Boot
Modem	Manual
Mount Point Manager	Boot
Mouse Class Driver	Manual
Mouse HID Driver	Manual
Msfs	System
Msisadrv	Boot
MsRPC	Manual
Multimedia Class Scheduler	Automatic
Mup	Boot
Mvumis	Manual
Named pipe service trigger provider	System
NativeWiFi Filter	Manual
Natural Authentication	Manual
NDIS Proxy Driver	Manual
NDIS System Driver	Boot

NDIS Usermode I/O Protocol	Manual
Net.Tcp Port Sharing Service	Disabled
NetBIOS Interface	System
NetBT	System
NetIO Legacy TDI Support Driver	System
Netlogon	Manual
Netvsc	Manual
Network Adapter Wdf Class Extension Library	Manual
Network Connected Devices Auto-Setup	Manual
Network Connection Broker	Manual
Network Connections	Manual
Network Connectivity Assistant	Manual
Network List Service	Manual
Network Location Awareness	Automatic
Network Setup Service	Manual
Network Store Interface Service	Automatic
Npfs	System
NSI Proxy Service Driver	System
Ntfs	Manual

Null	System
NVIDIA nForce AGP Bus Filter	Manual
Nvraid	Manual
Nvstor	Manual
Offline Files	Manual
Offline Files Driver	System
OpenSSH Authentication Agent	Manual, Disabled
Optimize drives	Manual
Packet Monitor Driver	Manual
Parallel port driver	Manual
Parental Controls	Manual
Partition driver	Boot
Parvdm	Automatic
Pass-through HID to KMDF Filter Driver	Manual
Pass-through HID to UMDF Driver	Manual
Payments and NFC/SE Manager	Manual
PCI Bus Driver	Boot
Pciide	Manual
Pcmcia	Manual

Pdc	Boot
PEAUTH	Automatic
Peer Name Resolution Protocol	Manual
Peer Networking Grouping	Manual
Peer Networking Identity Manager	Manual
Percsas2i	Manual
Percsas3i	Manual
Performance Counters for Windows Driver	Boot
Performance Logs & Alerts	Manual
Phone Service	Manual
Plug and Play	Manual
PNRP Machine Name Publication Service	Manual
Portable Device Enumerator Service	Manual
Portcfg	Manual
Power	Automatic
Print Spooler	Automatic
Printer Extensions and Notifications	Manual
PrintWorkflow	Manual
Problem Reports Control Panel Support	Manual

Processor Driver	Manual
Program Compatibility Assistant Service	Manual
QoS Packet Scheduler	System
Quality Windows Audio Video Experience	Manual
QWAVE driver	Manual
Radio Management Service	Manual
RAS Asynchronous Media Driver	Manual
ReadyBoost	Boot
Realtek RT640 NT Driver	Manual
Recommended Troubleshooting Service	Manual
Redirected Buffering Sub System	System
Remote Access Auto Connection Driver	Manual
Remote Access Auto Connection Manager	Manual
Remote Access Connection Manager	Manual, Automatic
Remote Access IP ARP Driver	Manual, Automatic
Remote Access IPv6 ARP Driver	Manual
Remote Access LEGACY NDIS WAN Driver	Manual
Remote Access NDIS TAPI Driver	Manual

Remote Access NDIS WAN Driver	Manual
Remote Access PPPOE Driver	Manual
Remote Desktop Configuration	Manual
Remote Desktop Device Redirector Bus Driver	Manual
Remote Desktop Device Redirector Driver	Manual
Remote Desktop Generic USB Device	Manual
Remote Desktop Services	Manual
Remote Desktop Services UserMode Port Redirector	Manual
Remote Desktop USB Hub	Manual
Remote Desktop USB Hub Class Filter Driver	Manual
Remote Desktop Video Miniport Driver	Manual
Remote Procedure Call (RPC)	Automatic
Remote Procedure Call (RPC) Locator	Manual
Remote Registry	Disabled
Resource Hub proxy driver	Manual
Retail Demo Service	Manual
Routing and Remote Access	Disabled
RPC Endpoint Mapper	Automatic
S3cap	Manual

SBP-2 Transport/Protocol Bus Driver	Manual
SD Storage Port Driver	Manual
Sdbus	Manual
Secondary Logon	Manual
Secure Socket Tunneling Protocol Service	Manual
Security Accounts Manager	Automatic
Security Center	Automatic
Sensor Data Service	Manual
Sensor Monitoring Service	Manual
Sensor Service	Manual
Serenum Filter Driver	Manual
Serial Mouse Driver	Manual
Serial port driver	Manual
Serial UART Support Library	Manual
Serial UART Support Library	Manual
Server	Automatic
Server SMB 1.xxx Driver	Automatic
Server SMB 2.xxx Driver	Manual
Service for Portable Device Control devices	Manual

Shared PC Account Manager	Disabled
Shell Hardware Detection	Automatic
Simple Peripheral Bus Support Library	Manual
SIS AGP Bus Filter	Manual
SiSRaid2	Manual
SiSRaid4	Manual
Smart Card	Disabled, Manual
Smart Card Device Enumeration Service	Manual
Smart card PnP Class Filter Driver	Manual
Smart Card Removal Policy	Manual
Smartlocker Filter Driver	Manual
SMB 1.x MiniRedirector	Automatic
SMB 2.0 MiniRedirector	Manual
SMB MiniRedirector Wrapper and Engine	Manual
SNMP Trap	Manual
Software Bus Driver	Manual
Software Protection	Automatic
Spatial Data Service	Manual
Spot Verifier	Manual

Srvnet	Manual
SSDP Discovery	Manual
State Repository Service	Manual
Stexstor	Manual
Still Image Acquisition Events	Manual
Storage QoS Filter Driver	Automatic
Storage Service	Manual, Automatic
Storage Spaces Driver	Boot
Storage Tiers Management	Manual
Storvsc	Manual
SuperSpeed Hub	Manual
Sync Host	Automatic
Synopsys USB Role-Switch Driver	Manual
Synth3dVsc	Manual
SysMain	Automatic
System Event Notification Service	Automatic
System Events Broker	Automatic
Task Scheduler	Automatic

TCP/IP NetBIOS Helper	Manual
TCP/IP Protocol Driver	Boot
TCP/IP Registry Compatibility	Automatic
Telephony	Manual
Themes	Automatic
Tile Data model server	Automatic, Manual
Time Broker	Manual
Time Broker	Manual
Touch Keyboard and Handwriting Panel Service	Manual
TPM	Manual
UAC File Virtualization	Automatic
UCM-TCPCI KMDF Class Extension	Manual
UCM-UCSI ACPI Client	Manual
UCM-UCSI KMDF Class Extension	Manual
Udfs	Disabled
Udk User Service	Manual
UevAgentDriver	Disabled
Uli AGP Bus Filter	Manual

UMBus Enumerator Driver	Manual
Update Orchestrator Service	Manual, Automatic
UPnP Device Host	Manual
USB Attached SCSI (UAS) Driver	Manual
USB Audio 2.0 Service	Manual
USB Audio Driver (WDM)	Manual
USB Chipidea Controller	Manual
USB Connector Manager KMDF Class Extension	Manual
USB Connector Manager UCSI Client	Manual
USB Device Emulation Support Library	Manual
USB Function Class Extension	Manual
USB Host Support Library	Manual
USB Mass Storage Driver	Manual
USB Role-Switch Support Library	Manual
USB Synopsys Controller	Manual
USB xHCI Compliant Host Controller	Manual
User Data Access	Manual
User Data Storage	Manual

User Experience Virtualization Service	Disabled
User Manager	Automatic
User Mode Driver Frameworks Platform Driver	Manual
User Profile Service	Automatic
Vhdmp	Manual
VIA AGP Bus Filter	Manual
VIA C7 Processor Driver	Manual
VIA StorX Storage RAID Controller Windows Driver	Manual
Virtual Disk	Manual
Virtual HID Framework (VHF) Driver	Manual
Virtual Machine Bus	Manual
Virtual Registry for Containers	Automatic
Virtual WiFi Filter Driver	System
Virtual Wireless Bus Driver	Manual
VMBusHID	Manual
Volume driver	Boot
Volume Manager Driver	Boot
Volume Shadow Copy	Manual
Volume Shadow Copy driver	Boot

Volumetric Audio Compositor Service	Manual
Vsmraid	Manual
Wacom Serial Pen HID Driver	Manual
WalletService	Manual
WAN Miniport (IKEv2)	Manual
WAN Miniport (L2TP)	Manual
WAN Miniport (PPTP)	Manual
WAN Miniport (SSTP)	Manual
WarpJITSvc	Manual
WDI Driver Framework	Manual
WdmCompanionFilter	Manual
Web Account Manager	Manual
WebClient	Manual
WebDav Client Redirector Driver	Manual
Wi-Fi Direct Services Connection Manager Service	Manual
WIMMount	Manual
Windows Audio	Automatic
Windows Audio Endpoint Builder	Automatic
Windows Backup	Manual

Windows Bind Filter Driver	Manual, Automatic
Windows Biometric Service	Manual
Windows Camera Frame Server	Manual
Windows Cloud Files Filter Driver	Automatic
Windows Color System	Manual
Windows Connect Now - Config Registrar	Manual
Windows Connection Manager	Automatic
Windows Container Isolation	Automatic
Windows Container Name Virtualization	Automatic, Manual
Windows Defender Advanced Threat Protection Service	Manual
Windows Defender Firewall	Automatic
Windows Defender Firewall Authorization Driver	Manual
Windows Defender Network Stream Filter Driver	Manual
Windows Driver Foundation - User-mode Driver Framework	Manual
Windows Driver Foundation - User-mode Driver Framework Reflector	Manual
Windows Encryption Provider Host Service	Manual
Windows Error Reporting Service	Manual

Windows Event Collector	Manual
Windows Event Log	Automatic
Windows Font Cache Service	Automatic
Windows Image Acquisition (WIA)	Manual
Windows Insider Service	Manual
Windows Installer	Manual
Windows License Manager Service	Manual
Windows Licensing Monitoring Service	Automatic
Windows Management Instrumentation	Automatic
Windows Management Service	Manual
Windows Media Player Network Sharing Service	Manual
Windows Mobile Hotspot Service	Manual
Windows Modules Installer	Manual
Windows NAT Driver	Manual
Windows Network Data Usage Monitoring Driver	Automatic
Windows Overlay File System Filter Driver	Boot
Windows Perception Service	Manual
Windows Perception Simulation Service	Manual

Windows Push Notifications System Service	Manual, Automatic
Windows Push Notifications User Service	Manual, Automatic
Windows PushToInstall Service	Manual
Windows RAM Disk Driver	Manual
Windows Remote Management (WS-Management)	Manual
Windows Search	Automatic
Windows Security Service	Automatic, Manual
Windows Store Service (WSService)	Manual
Windows Time	Manual
Windows Trusted Execution Environment Class Extension	Boot
Windows Update	Manual
Windows Update Medic Service	Manual
WinHTTP Web Proxy Auto-Discovery Service	Manual
Winsock IFS Driver	Disabled
WinUsb Driver	Manual
Wired AutoConfig	Manual
WLAN AutoConfig	Manual

WMI Performance Adapter	Manual
Work Folders	Manual
Workstation	Automatic
WPD File System driver	Manual
WPD Upper Class Filter Driver	Manual
WWAN AutoConfig	Manual
Xbox Accessory Management Service	Manual
Xbox Game Input Protocol Driver	Manual
Xbox Game Monitoring	Manual
Xbox Live Auth Manager	Manual
Xbox Live Game Save	Manual
XINPUT HID Filter Driver	Manual

Services can you safely turn off on Windows 11

In general I do not recommend messing with services unless you are technical training and understand the impact. Mr.V

There are a few services that you just shouldn't touch at all. These are important to run basic functions, security features, and make the Windows experience seamless. We shall not mention these.

The only ones that you should look out for are the ones that we've mentioned below. However, even amongst the following, there are a few that may be required in specific cases. For a general understanding of the services, do read

their basic descriptions to know when a service may be needed and which ones you can turn off without a problem.

1. **FAX** — As its name suggests, this is a service needed only if you want to send and receive faxes. If you're not going to use, which may be the case for most people, disable it.
2. **AllJoyn Router Service** — This is a service that lets you connect Windows to the Internet of Things and communicate with devices such as smart TVs, refrigerators, light bulbs, thermostats, etc. If you're not using these or don't connect Windows to them, go ahead and turn it off.
3. **Secondary logon** — This service lets you log on to a standard account with admin privileges and run specific applications. It is triggered to start when a program is set to 'Run as different user' from the extended context menu. But if you are the sole user of your PC, then go ahead and disable this.
4. **Connected User Experiences and Telemetry** — If you're concerned with privacy and don't want to send usage data to Microsoft for analysis, then this service is one to go. Though some would say that such assessment of data is important to improve Windows on the whole, disabling it doesn't affect normal usage and, frankly, one less data bundle wouldn't bring the house down.
5. **Program Compatibility Assistant Service** — Unless you're still using legacy software on your Windows 11 PC, you can easily turn off this service. This service lets you detect software incompatibility issues for old games and software. But if you're using programs and apps built for Windows 10 or 11, go ahead and disable it.
6. **Device Management Wireless Application Protocol (WAP) Push message Routing Service** — This service is another service that helps to collect and send user data to Microsoft. Strengthen your privacy by disabling it, it is recommended that you do so.
7. **Windows Mobile Hotspot Service** — As the name suggests, this service is needed if you're sharing your mobile's internet connection with your PC. But if you don't remember the last time you connected to a mobile hotspot service, you may look to disable it entirely.

8. **Remote Desktop Configuration and Remote Desktop Services** — These two services let you connect to other PCs in the vicinity. If you don't need remote connectivity, disable these two services.
9. **Remote Registry** — This service lets any user access and modify the Windows registry. It is highly recommended that you disable this service for security purposes. Your ability to edit the registry locally (or as admin) won't be affected.
10. **Touch Keyboard and Handwriting Panel Service** — As the name tells, this service facilitates touch keyboard and handwriting input for touch-enabled screens. So unless you have one of those, go ahead and disable it.
11. **Windows Insider Service** — Disable this service only if you're not in the Windows Insider program. Currently, as Windows 11 is only available through it, you shouldn't disable it. But if you're on the final and stable version of Windows and are not testing upcoming features, disabling it shouldn't be a problem.
12. **Windows Image Acquisition** — This service is important for people who connect scanners and digital cameras to their PC. But if you don't have one of those, or are never planning on getting one, disable it by all means.
13. **Windows Connect Now** — This service is mainly meant for laptops and computers that need to connect to wireless networks and devices (camera, printers, and other PCs). But if you have a desktop setup without a wireless card, you won't need this service and can disable it safely.
14. **Windows Defender** — This may raise some eyebrows, but we're only recommending turning this off only and only if you have an antivirus that's protecting your system. If that's the case, Windows Defender would virtually be inactive anyway, as the third-party antivirus would act as your primary threat protection. Disabling Windows Defender at that point would help you free up valuable resources, without compromising the security of your device.
15. **Downloaded Maps Manager** — Do you use Bing Maps? Chances are that most of you rely on Google Maps built within your favorite browser and

can't care for Bing Maps. So find this unnecessary service and make sure that it's disabled.

16. **Parental Control** — Again, the name says it all – this service allows parents to put restrictions on what their kids are accessing on the internet. But, as with many things brought with Vista, this is obsolete if you know how to filter content for your kids on the browser itself. Also, if you don't have any kids around, that's an obvious reason to keep this service disabled.

17. **Xbox Services** — Do you use the Xbox app to play games? If not, then you don't need any of the Xbox services. These include the 'Xbox Accessory Management Service', 'Xbox Live Auth Manager', 'Xbox Live Game Save', and 'Xbox Live Networking Service'. These won't affect your daily use unless you do use the Xbox app on your PC. In that case, don't touch these.

18. **Security Center** — This is another one of those services that only advanced users should turn off. The functionalities of this service are crucial – it scans the system for issues and keeps you posted about the system's health, including pending updates, whether or not an antivirus is installed, UAC notifications, and other such messages you receive in the system tray. If you know how to check for these issues on your own, you can disable the service without any problems. If, however, you're not sure how to check your system's health, leave this one alone.

19. **Print Spooler** — Connected to the printer in the past few months? If not, then this service is useless to you. Go ahead and disable it if you're not planning on using a printer anytime soon.

20. **Portable Device Enumerator Service** — This service is needed for making group policy changes for removable drives and to synchronize content for applications like Windows Media Player and Image Import Wizard on the removable drive. If these don't mean anything to you, go ahead and disable it. Rest assured, it won't affect your regular thumb drive use.

21. **Retail Demo Service** — Finally, this service is only meant for vendors and retailers who have to showcase the PC and Windows features for customers. Of course, a regular user would never need to use such a service, and so can disable it without any consequences.

Do note that some of these services may either be disabled or will be set to run manually by default. Nevertheless, it is good to ensure that that is actually the case in order to free up system resources and speed up your PC's performance considerably.

Hands off my Windows Services: Learn how to harden

https://decoder.cloud/2020/11/05/hands-off-my-service-account/

Windows service accounts are one of the preferred attack surface for privilege escalation. If you are able to compromise such an account, it is quite easy to get the highest privileges, mainly due to the powerful impersonation privileges that are granted by default to services by the operating system.

Even if Microsoft introduced WSH (Windows Service Hardening), there isn't much you can do to "lower" the power of standard Windows services, but if you need to build or deploy a third-party service you can definitely strengthen it by using WSH.

In this post I will show you some useful tips.

Make use of Virtual accounts

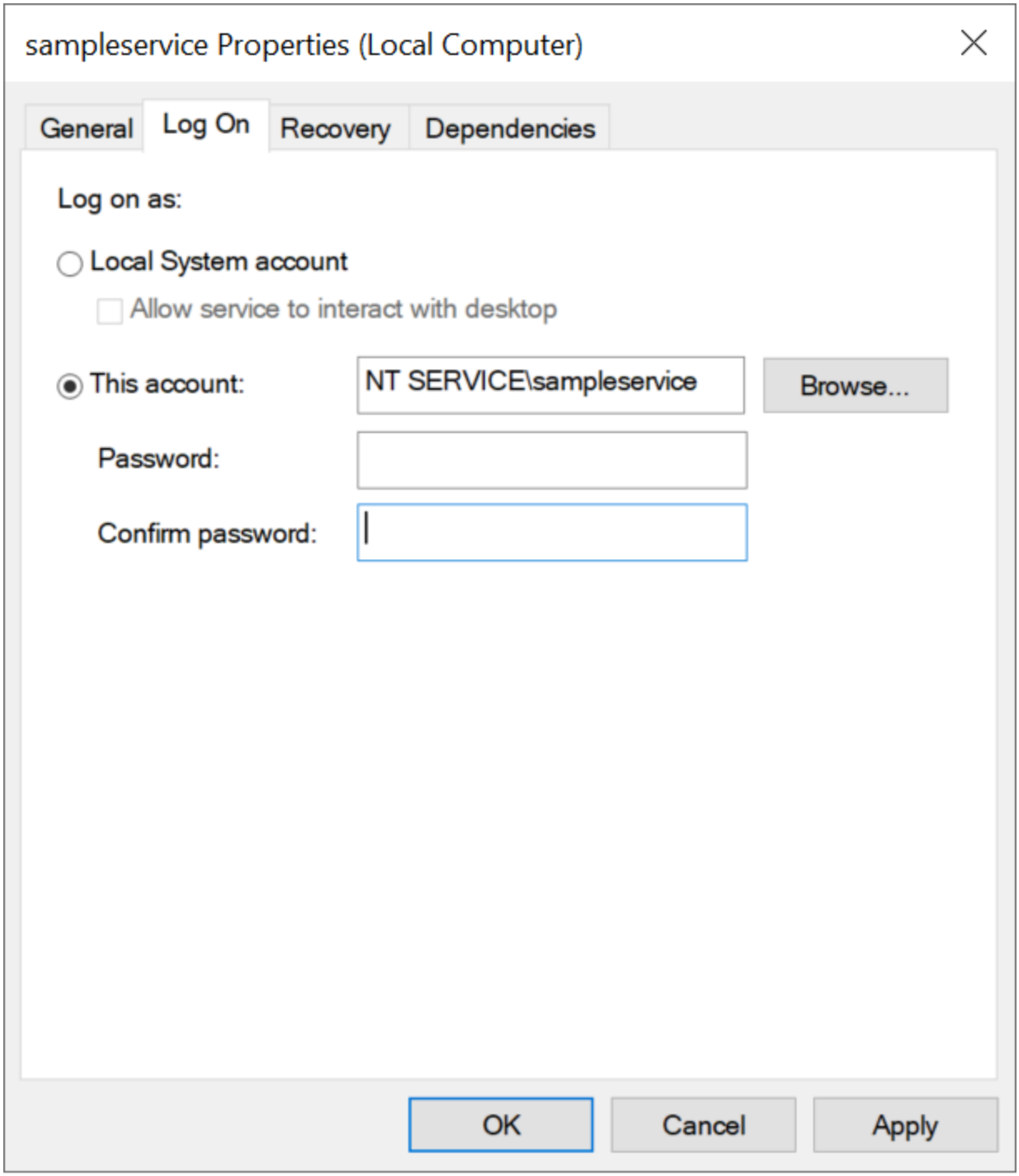
Obviously a service must be run with a specific account. There are some built-in Windows accounts like SYSTEM (oh no !!), Local Service, and Network Service. But there is also the ability to use Virtual Accounts (I won't write about "*Group Managed Service Accounts*" in this post) which are self-managed (no need to deal with passwords) and you can grant specific permissions by setting the correct ACL's on the resources. This allows you to isolate the service and in case of compromise, they can only access the resources you have allowed. Virtual accounts can also access network resources, but in this case they will impersonate the computer account (COMPUTERNAME$)

Configuring Services with Virtual Accounts

First of all, you don't need to create VSAs, when the service is installed, a matching account is automatically created for you in the form:

NT SERVICE\<service name>

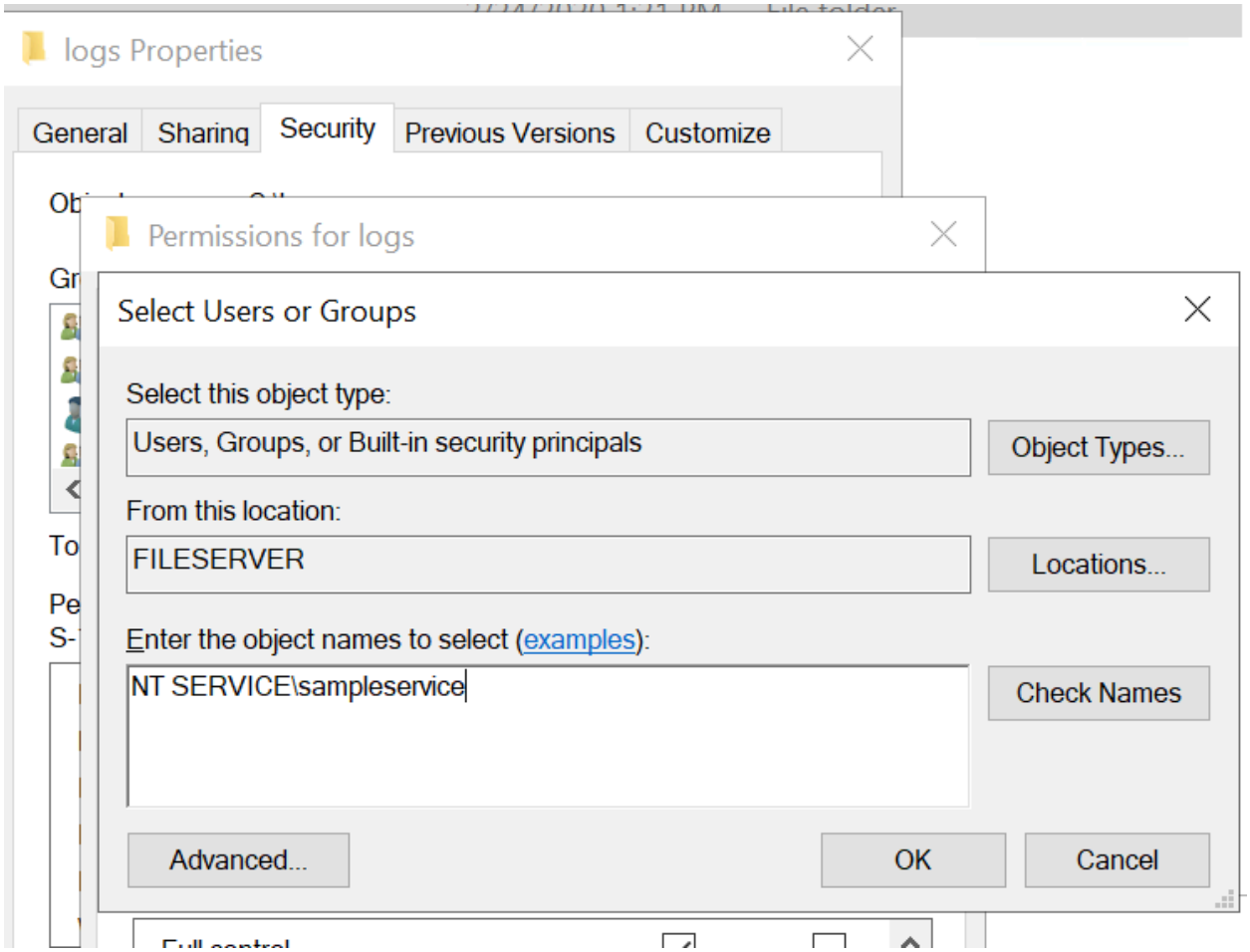
all you need is to assign the account to your service:

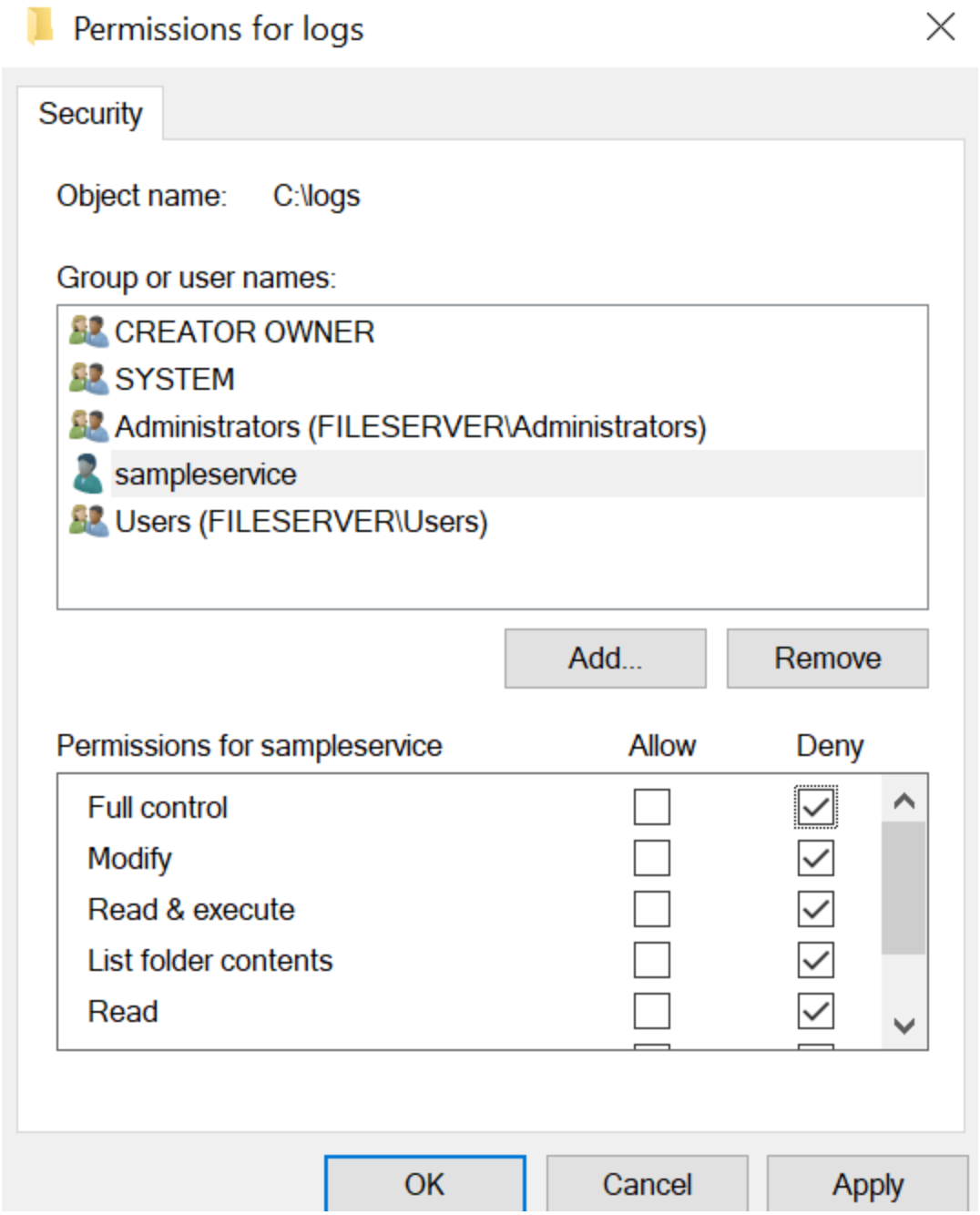


Don’t forget to leave the password field blank!

Restricting access to Virtual Accounts

Now that your service runs under a specific account and not a generic one like Local Service or Network Service , you can implement fine-grained access control on resources like files and directories:





```
C:\>whoami
whoami
nt service\sampleservice

C:\>cd logs
cd logs
Access is denied.

C:\>_
```

Removing unnecessary privileges

Impersonation privileges are a nightmare, so if your service *doesn't need to impersonate*, why should we grant them to this service user?

```
C:\>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

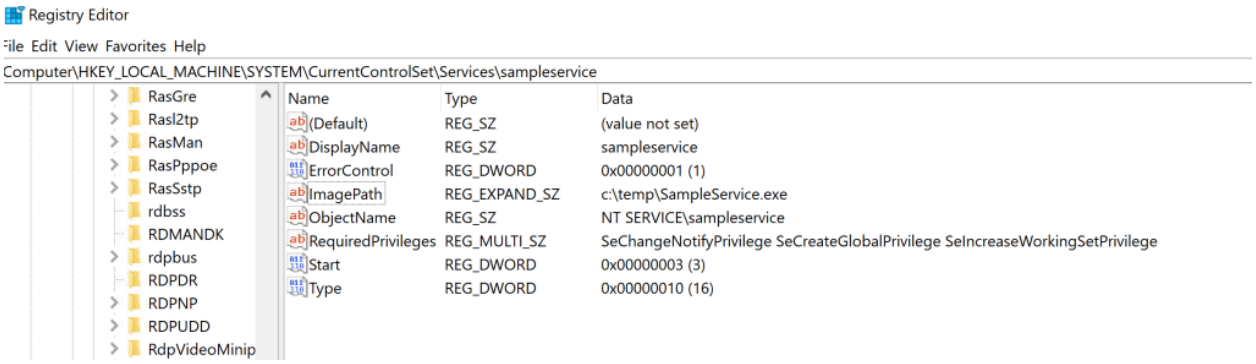
Is it possible to remove this default privilege? There is a good news, yes! We can configure the privileges directly in registry or by using the "sc.exe" command:

```
Administrator: Command Prompt

C:\>sc privs sampleservice SeChangeNotifyPrivilege/SeCreateGlobalPrivilege/SeIncreaseWorkingSetPrivilege
[SC] ChangeServiceConfig2 SUCCESS

C:\>_
```

And these values will be written into registry:



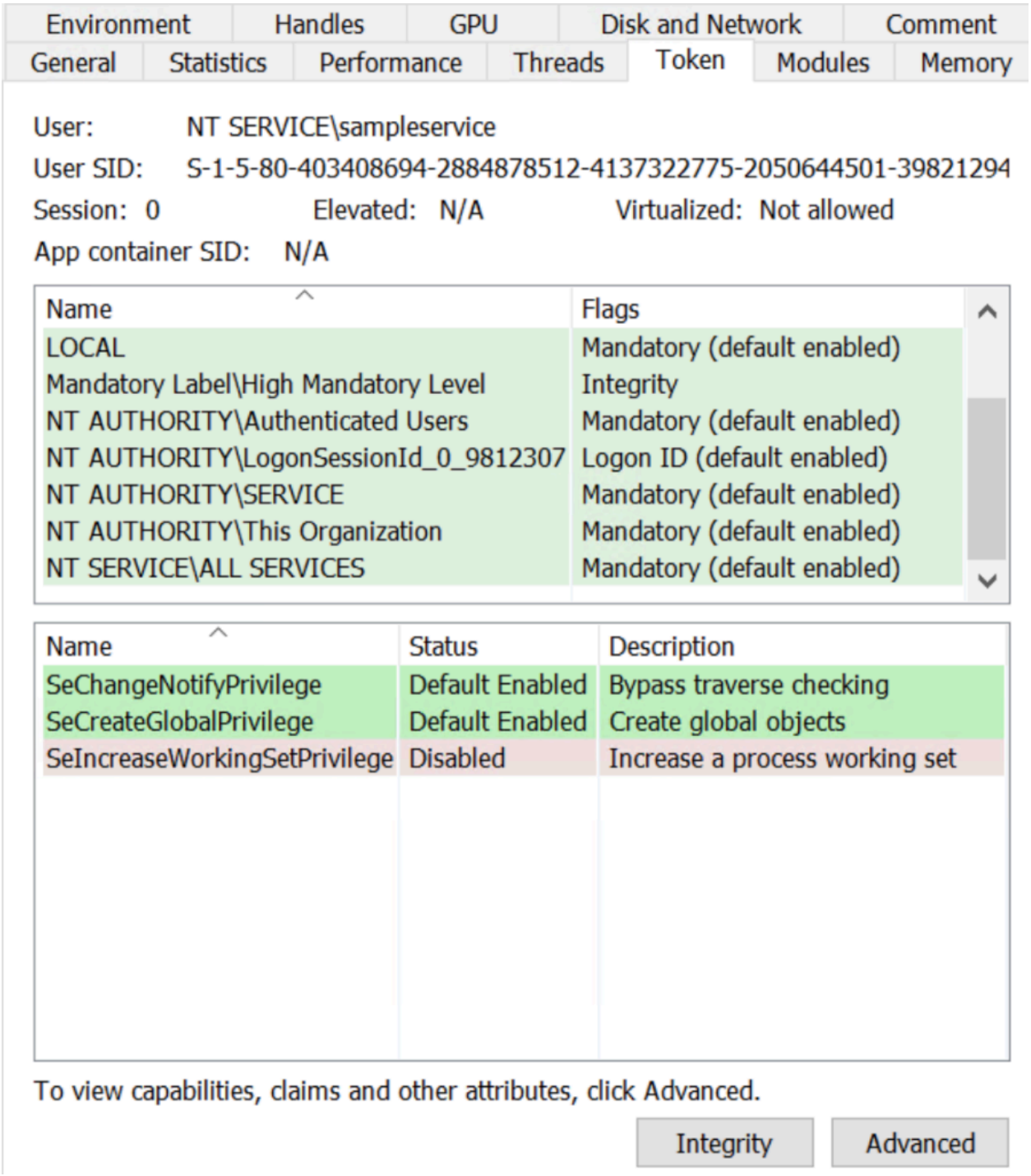
Let’s see if the the privilege is removed:

```
C:\Windows\system32>whoami & whoami /priv
whoami & whoami /priv
nt service\sampleservice

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeCreateGlobalPrivilege       Create global objects          Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled

C:\Windows\system32>_
```



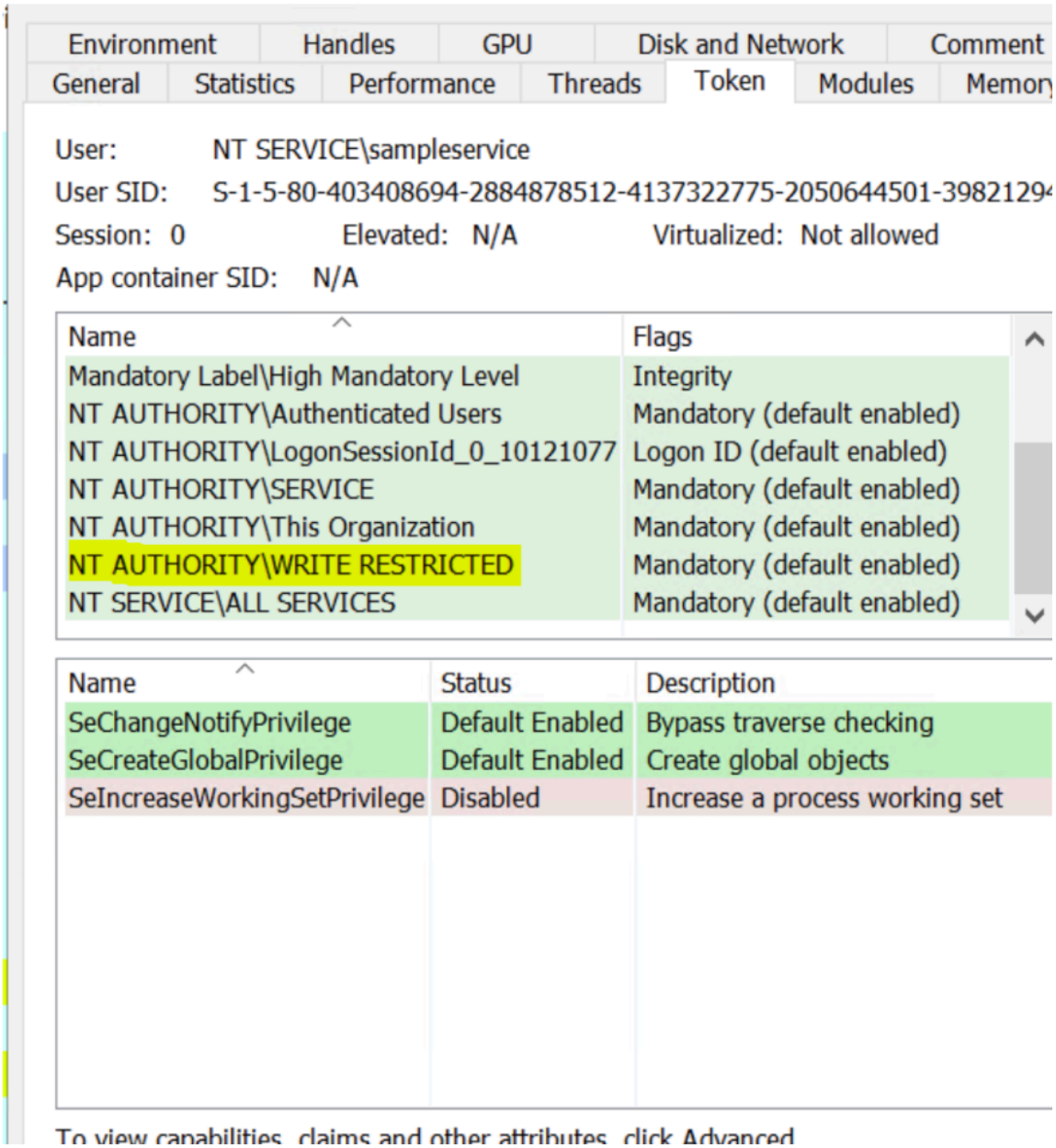
Oh yes! The dangerous privilege has been removed and it's no more possible to get him back, for example using @itm4n 's trick described in this great post.

Write Restricted Token

If you add this extra group to your service tokens, you can further limit the permissions of your service account.

```
C:\>sc sidtype sampleservice restricted
[SC] ChangeServiceConfig2 SUCCESS

C:\>_
```



This means that by *default* you can only read or execute resources unless you explicitly grant write access:

```
C:\>cacls test
cacls test
C:\test CREATOR OWNER:(OI)(CI)(IO)F
        NT AUTHORITY\SYSTEM:(OI)(CI)F
        BUILTIN\Administrators:(OI)(CI)F
        BUILTIN\Users:(OI)(IO)R
        BUILTIN\Users:(CI)(special access:)
                           READ_CONTROL
                           SYNCHRONIZE
                           FILE_GENERIC_READ
                           FILE_GENERIC_EXECUTE
                           FILE_READ_DATA
                           FILE_WRITE_DATA
                           FILE_APPEND_DATA
                           FILE_READ_EA
                           FILE_EXECUTE
                           FILE_READ_ATTRIBUTES



C:\>echo test > test\test.txt
echo test > test\test.txt
Access is denied.
```

```
C:\>cacls test1
cacls test1
C:\test1 CREATOR OWNER:(OI)(CI)(IO)F
         NT AUTHORITY\SYSTEM:(OI)(CI)F
         BUILTIN\Administrators:(OI)(CI)F
         BUILTIN\Users:(OI)(IO)R
         BUILTIN\Users:(CI)(special access:)
                           READ_CONTROL
                           SYNCHRONIZE
                           FILE_GENERIC_READ
                           FILE_GENERIC_EXECUTE
                           FILE_READ_DATA
                           FILE_WRITE_DATA
                           FILE_APPEND_DATA
                           FILE_READ_EA
                           FILE_EXECUTE
                           FILE_READ_ATTRIBUTES

         NT SERVICE\sampleservice:F


C:\>echo test > test1\test.txt
echo test > test1\test.txt

C:\>type test1\test.txt
type test1\test.txt
test
```

A great research about write restricted tokens can be found in Forshaw's post

Empirically Assessing Windows Service Hardening

https://www.tiraniddo.dev/2020/01/empirically-assessing-windows-service.html

In the past few years there's been numerous exploits for service to system privilege escalation. Primarily they revolve around the fact that system services typically have impersonation privilege. What this means is given access to a suitable token handle of an administrator (say through the Rotten Potato attack) you can impersonate and elevate from a lower-privileged service account to SYSTEM. The problem for discovers of these attacks is that Microsoft do not consider them something which needs to be fixed with a security bulletin, as having *SeImpersonatePrivilege* is basically a massive security hole. However MS go and fix them silently making it unclear if they care or not.

Of course, none of this is really new, Cesar Cerrudo detailed these sorts of service attacks in Token Kidnapping and Token Kidnapping's Revenge. The novel element recently is how to get hold of the access token, for example via negotiating local NTLM authentication. Microsoft seem to have been fighting this fire for almost 10 years and still have not gotten it right. In shades of UAC, a significant security push to make services more isolated and secure has been basically abandoned because (presumably) MS realized it was an indefensible boundary.

That's not to say there hasn't been interesting service account to SYSTEM bugs which Microsoft have fixed. The most recent example is CVE-2019-1322 which was independently discovered by multiple parties (DonkeysTeam, Ilias Dimopoulos and Edward Torkington/Phillip Langlois of NCC). To understand the bug you probably should read up one of the write-ups (NCC one here) but the gist is, the Update Orchestrator Service has a service security descriptor which allowed "NT AUTHORITY\SERVICE" full access. It so happens that all system services, including lower-privileged ones have this group and so you could reconfigure the service (which was running as SYSTEM) to point to any other binary giving a direct service to SYSTEM privilege escalation.

That begs the question, why was CVE-2019-1322 special enough to be fixed and not issues related to impersonation? Perhaps it's because this issue didn't rely on impersonate privileges being present? It is possible to configure services to not

have impersonate privilege, so presumably if you could go from a non-impersonate service to an impersonate service that would count as a boundary? Again probably not, for example this bug which abuses the scheduled task service to regain impersonate privilege wouldn't likely be fixed by Microsoft.

That lack of clarity is why I tweeted to Nate Warfield and ultimately to Matt Miller asking for some advice with respect to the MSRC Security Servicing Guidelines. The result is, even if the service doesn't have impersonate privilege it wouldn't be a defended boundary if all you get is the same user with additional privileges as you can't block yourself from compromising yourself. This is the UAC argument over again, but IMO there's a crucial difference, Windows Service Hardening (WSH) was supposed to fix this problem for us in Vista. Unsurprisingly Cesar Cerrudo also did a presentation about this at the inaugural (maybe?) Infiltrate in 2011.

The question I had was, is WSH still as broken as it was in 2011? Has anything changed which made WSH finally live up to its goal of making a service compromise not equal to a full system compromise? To determine that I thought I'd run an experiment on Windows 10 1909. I'm only interested in the features which WSH touches which led me to the following hypothesis:

"Under Windows Service Hardening one service without impersonate privilege can't write to the resources of another service which does have the privilege, even if the same user, preventing full system compromise."

The hypothesis makes the assumption that if you can write to another service's resources then it's possible to compromise that other service. If that other service has *SeImpersonatePrivilege* then that inevitably leads to full system compromise. Of course that's not necessarily the case, the resource being written to might be uninteresting, however as a proxy this is sufficient as the goal of WSH is to prevent one service modifying the data of another even though they are the same underlying user.

WSH Details

Before going into more depth on the experiment, let's quickly go through the various features of WSH and how they're expressed. If you know all this you can skip to the description of the experiment and the results.

Limited Service Accounts and Reduced Privilege

This feature is by far the oldest attempt to harden services, the introduction of the LOCAL SERVICE (LS) and NETWORK SERVICE (NS) accounts. Prior to the accounts introduction there was only two ways of configuring the user for a system service on Windows, either the fully privileged SYSTEM account or creating a local/domain user which has the "Log on as a Service" right. The two accounts where introduced in XP SP2 (I believe) after worms such as Blaster basically got SYSTEM privilege through remotely attacking exposed services. The two service accounts are not administrator accounts which means they shouldn't be able to directly compromise the system. The accounts are very similar on Windows 10 1909, they are both assigned the following groups*:

BUILTIN\Users

CONSOLE LOGON

Everyone

LOCAL

NT AUTHORITY\Authenticated Users

NT AUTHORITY\LogonSessionId_X_Y

NT AUTHORITY\SERVICE

NT AUTHORITY\This Organization

* Technically this isn't 100% accurate, on my machine the LS account has some extra capability groups, but we'll ignore those for this blog post.

No Administrator group in sight. Each service token gets a unique Logon Session ID SID which will be important later. The service accounts also have a limited set of privileges, as shown below:

SeAssignPrimaryTokenPrivilege

SeAuditPrivilege

SeChangeNotifyPrivilege

SeCreateGlobalPrivilege

SeImpersonatePrivilege

SeIncreaseQuotaPrivilege

SeIncreaseWorkingSetPrivilege

SeShutdownPrivilege

SeSystemTimePrivilege†

SeTimeZonePrivilege

SeUndockPrivilege

† NETWORK SERVICE doesn't have *SeSystemTimePrivilege*.

The two privileges I've highlighted, *SeAssignPrimaryTokenPrivilege* and *SeImpersonatePrivilege* give these accounts effectively full system access when combined with a suitable privileged token. Part of WSH is also giving control over what privileges the service account actually requires. The default is to allow all privileges, however when configuring a service you can specify a list of privileges to restrict the service to. For example the *CDPSvc* service is configured to only require *SeImpersonatePrivilege*. Quite why they bother to put this restriction on the service I don't know ¯_(ツ)_/¯.

What's the difference between LS and NS? The primary difference is LS has no

network credentials, so accessing network resources as that user would only succeed as an anonymous login. NS on the other hand is created with the credentials of the computer account and so can interact with the network for resources allowed by that authentication. This only really matters to domain joined machines, standalone machines would not share the computer account with anyone else.

Per-Service SID

The first big addition in WSH was the Per-Service SID. This SID is automatically added to the group list of default groups shown previously by the SCM when creating the service's primary token. The service SID is also added with the SE_GROUP_OWNER flag set and is not mandatory, which means it can be set as the token's default owner when creating new resources and it can disabled. The basic idea is a service can ACL its resources to this SID to prevent other services from accessing them. The use of a service SID is optional, but the majority of default services are configured to use it. An example SID for *CDPSvc* is as follows:

S-1-5-80-3433512109-503559027-1389316256-1766580070-2256751264

The SID is derived by generating a SHA1 hash of the service name and adding that as the SID's RIDs (with an extra 80 at the start to signify it's a service SID). The use of a hash should make it extremely unlikely two services would generate the same SID.

Of course it's up to the service to actually ACL their resources appropriately. To aid in that the token's default DACL is also configured to the following (for *CDPSvc*):

- Type : Allowed
- Name : NT AUTHORITY\SYSTEM
- Access: Full Access

- Type : Allowed

- Name : OWNER RIGHTS

- Access: ReadControl

- Type : Allowed

- Name : NT SERVICE\CDPSvc

- Access: Full Access

The three entries grant SYSTEM and the service SID full access to any resources with this DACL. It then limits the owner of the resource through OWNER RIGHTS to only READ_CONTROL access. This directly prevents one service account accessing the resources of another for write access. Unfortunately the default DACL is only applied when there's no other access control specified, either explicitly at creation time or due to inheritance.

One other thing to point out is that Windows still has shared services through the use of SVCHOST. If multiple services are registered in a specific SVCHOST instance then the SCM will create the token with all service SIDs in the group list and default DACL even if a service isn't currently loaded in the host. That has become less of an issue since Windows 1703, as long as you have greater that 3.5GB of RAM services will run in separate SVCHOST instances and all services will be totally separate.

Write-Restricted Token

The second big addition to WSH was the concept of Write-Restricted (WR) tokens. Restricted token's have existed since Windows 2000 and are created using the *NtFilterToken* system call. The basic concept is the token can have a list of additional groups which are consulted when ever an access check is performed.

First the access check is run on the default group list, if access would be granted the access check is run again on the restricted SIDs. If the second check is successful then the access check passes, if not access is denied.

Restricted tokens are used for sandboxing (such as in Chrome) but are difficult to setup correctly as it blocks all access equally including reading critical files on disk. WR tokens solve the access problem by only blocking write access but leaving read and execute access alone.

In order for a service configured as WR to write to a resource the associated security descriptor must contain the required access for one of the following restricted SIDs.

Everyone

NT AUTHORITY\LogonSessionId_X_Y

NT AUTHORITY\WRITE RESTRICTED

NT SERVICE\SERVICE_NAME

The WRITE RESTRICTED SID is a special group SID which resources can apply if they expect a service to write to the resource. This SID is also added to the token's groups by the SCM so that it can be used to pass both checks. By combining service SIDs and WR the amount of resources a service can modify should be significantly reduced.

And the Rest

There's a few things which are technically part of service hardening which won't really consider for the experiment:

The main one is additional rules in the firewall to block network services or requests being made from a service. This is arguably more to prevent remote compromise than it is to prevent cross-service attacks.

Another is Session 0 Isolation and System Integrity Level. Session 0 Isolation was introduced to prevent Shatter Attacks, by preventing any windows being created by a service on the same desktop as a normal user. System Integrity Level through UIPI then prevents attacks even if the service did create a window on a normal user desktop as it'd be at a much higher IL (even than Administrators). The System IL does admittedly also have a security access check function but it's not that important for cross-service attacks.

Experiment Procedure

On to the experiment itself. Based on the hypothesis I presented earlier the goal is to determine if you can write to resources of one service from another service even though they're the same user. To make this testable I decided on the following procedure:

Step 1: Build an access token for a service which doesn't exist on the system.
Step 2: Enumerate all resources of a specific type which are owned by the token owner and perform an access check using the token.
Step 3: Collate the results based on the type of resource and whether write access was granted.

The reason for choosing to build a token for a non-existent service is it ensures we should only see the resources that could be shared by other services as the same user, not any resources which are actually designed to be accessible by being created by a service. These steps need to be repeated for different access tokens, we'll use the following five:

- LOCAL SERVICE
- LOCAL SERVICE, Write Restricted

- NETWORK SERVICE
- NETWORK SERVICE, Write Restricted
- Control

We'll test both normal service SID and WR versions of the access token to see if it makes much of a difference. One thing to determine is what to use as a control. Ideally the control would be another service account with WSH disabled. However I couldn't find a way to disable WSH entirely to do this test, so instead we need some other control. If our hypothesis holds and WSH is effective we'd expect no resources to be writable, therefore we need to pick a control account where we know this is not true. The easiest is just to use the current logged on user account, it should be able to access almost all its own resources.

What resources do we want to inspect? The obvious type is Process/Thread resources. Getting write access to either of these in another service is probably a trivial to get full system compromise through impersonate. We'd want to get a bigger picture however, it'd be useful to include Files, Registry keys and Named Kernel Objects. These resources might not directly lead to compromise but it does give us a general idea of the maximum impact.

It's worth noting that the hypothesis made a point to specify writing to the resources of a service which has impersonate privilege from one which does not. However this experimental process will only base the analysis on whether the resource is owned by the service user. This is intentional, it'd be too complex to attribute the resource to a specific service in all cases. However an assumption is made that more services running as a specific user have impersonate privilege than do not, therefore in all probability any resource you can write to is probably owned by one of them. We could verify that assumption if we liked, but I'll probably not.

Finally, a good experiment should be something which can be repeatable and verifiable. To that end I'll provide all the code necessary to perform the steps,

written in PowerShell and using my *NtObjectManager* module. If you want to re-run the experiment you should be able to do so and produce a very similar set of results.

Experiment Procedure Detail

On to specific PowerShell steps to perform the experiment. First off you'll need my NtObjectManager module, specifically at least version 1.1.25 as I've added a few extra commands to simplify the process. You will also need to run all the commands as the SYSTEM user, some command will need it (such as getting access tokens) others benefit for the elevated privileges. From an admin command prompt you can create a SYSTEM PowerShell console using the following command:

```
Start-Win32ChildProcess -RequiredPrivilege
SeTcbPrivilege,SeBackupPrivilege,SeRestorePrivilege,SeDebugPrivilege powershell
```

This command will find a SYSTEM process to create the new process from which also has, at a minimum, the specified list of privileges. Due to the way the process is created it'll also have full access to the current desktop so you can spawn GUI applications running at system if you need them.

The experiment will be run on a VM of Windows 1909 Enterprise updated to December 2019 from a split-token admin user account. This just ensures the minimum amount of configuration changes and additional software is present. Of course there's going to be variability on the number of services running at any one time, there's not a lot which can be done about that. However it's expected that the result should be same even if the individual resources available are not. If you were concerned you could rerun the experiment on multiple different installs of Windows at different times of day and aggregate the results.

Creating the Access Tokens

We need to create 5 access tokens for the test. Ideally we'd like to create the four service tokens using the exact method used by the SCM. We could register our unknown service and start the service to steal its token. There is also an

undocumented *RGetServiceProcessToken* SCM RPC method in newer versions of Windows 10. However I think creating a service risks some resources being populated with that service's identity which might not be what we really want. Instead we can use LogonUserExExW which is what the SCM uses, with the LOGON32_LOGON_SERVICE type to create LS and NS tokens. This will work as long as we have *SeTcbPrivilege*. We'll then just add the appropriate groups, convert to WR, and remove privileges as necessary. We can get to the *LogonUserExExW* API using *Get-NtToken.* I've wrapped up everything into a function *Get-ServiceToken*, you can see the full function in the final script. Using this function we can create all the tokens we need using the following commands:

```
$tokens = @()
$tokens += Get-ServiceToken LocalService FakeService
$tokens += Get-ServiceToken LocalService FakeService -WriteRestricted
$tokens += Get-ServiceToken NetworkService FakeService
$tokens += Get-ServiceToken NetworkService FakeService -WriteRestricted
```

For the control token we'll get the unmodified session access token for the current desktop. Even though we're running as SYSTEM as we're running on the same desktop we can just use the following command:

```
$tokens += Get-NtToken -Session -Duplicate
```

Random note. When calling *LogonUserExExW* and requesting a service SID as an additional group the call will fail with access denied. However this only happens if the service SID is the first NT Authority SID in the additional groups list. Putting any other NT Authority SID, including our new logon session SID before the service SID makes it work. Looking at the code in LSASRV (possibly the function *LsapCheckVirtualAccountRestriction*) it looks like the use of a service SID should be restricted to the first process (based on its PID) that used a service SID which would be the SCM. However if another NT Authority SID is placed first the checking loop sets a boolean flag which prevents the loop checking any more SIDs and so the service SID is ignored. I've no idea if this is a bug or not, however as

you need TCB privilege to set the additional groups I don't think it's a security issue.

Resource Checking and Result Collation

With the 5 tokens in hand we can progress to assessing accessible resources. The original purpose of my Sandbox Analysis tools was finding accessible resources from a sandbox process, however the same code is capable of finding resources accessible from any access token, including service tokens.

First as way of example lets run checks for process and threads:

```
$ps = Get-AccessibleProcess -Tokens $tokens `
    -CheckMode ProcessOnly -AllowEmptyAccess
$ts = Get-AccessibleProcess -Tokens $tokens `
    -CheckMode ThreadOnly -AllowEmptyAccess
```

We can pass a list of tokens to the checking command, this improves performance as we only do the enumeration of resources for every token group then do the access check. Each generated access result has a *TokenId* property which indicates the unique ID of the token which was used for the check, this allows us to extract the correct results later. We also specify the *AllowEmptyAccess* option, which will generate a result even if the access check fails and the token has no access to the resource. This will be useful to allow us to assess what resources are owned by the token's owner SID but we were not granted access.

Let's do the rest of the resources:

```
$os = Get-AccessibleObject \ -Recurse `
    -Tokens $tokens -AllowEmptyAccess
```

```
$fs = Get-AccessibleFile -Win32Path "$env:SystemDrive\" `
    -FormatWin32Path -Recurse -Tokens $tokens -AllowEmptyAccess
$ks = Get-AccessibleKey \Registry -FormatWin32Path -Recurse `
    -Tokens $tokens -AllowEmptyAccess
```

We'll only get the accessible files on the system drive in this case as that'll be the only drive in the VM. Note that *Get-AccessibleObject* doesn't check ALPC ports, it's not possible to open an ALPC port by name and read its security descriptor. We'll ignore ALPC ports for this experiment, as it's probably worthy of a topic all on its own.

We now have all the results we need in five variables along with the tokens. If you want to run it yourself the final script is on Github here. It'll take a fair amount of time to run but once it's complete you'll find 5 CSV files in the current directory containing the results for each token.

Experiment Results

We now need to do our basic analysis of the results. Let's start with calculating the percentage of writable resources for each token type relative to the total number of resources. From my single experiment run I got the following table:

Token	Writable	Writable (WR)	Total
Control	99.83%	N/A	13171
Network Service	65.00%	0.00%	300
Local Service	62.89%	0.70%	574

As we expected the control token had almost 100% of the owned resources writable by the user. However for the two service accounts both had over 60% of

their owned resources writable when using an unrestricted token. That level is almost completely eliminated when using a WR token, there were no writable resources for NS and only 4 resources writable from LS, which was less than 1%. Those 4 resources were just Events, from a service perspective not very exciting though there were ACL'ed to everyone which is unusual.

Just based on these numbers alone it would seem that WSH really is a failure when used unrestricted but is probably fine when used in WR mode. It'd be interesting to dig into what types are writable in the unrestricted mode to get a better understanding of where WSH is failing. This is what I've summarized in the following table:

Type	LS Writable%	LS Writable	NS Writable%	NS Writable
Directory	0.28%	1	0.51%	1
Event	1.66%	6	0.51%	1
File	74.24%	268	48.72%	95
Key	22.44%	81	49.23%	96
Mutant	0.28%	1	0.51%	1
Process	0.28%	1	0.00%	0
Section	0.55%	2	0.00%	0
SymbolicLink	0.28%	1	0.51%	1
Thread	0.00%	0	0.00%	0

The clear winners, if there is such a thing is Files and Registry Keys taking up over 95% of the resources which are writable. Based on what we know about how WSH works this is understandable. The likelihood is any keys/files are getting their security through inheritance from the parent container. This will typically result in at least the owner field being the service account granted WRITE_DAC access, or the inherited DACL will contain an OWNER CREATOR SID which results an explicit

access for the service account.

What is perhaps more interesting is the results for Processes and Threads, neither NS or LS have any writable threads and only LS has a single writable process. This primary reason for the lack of writable threads and processes is due to the default DACL which is used for new processes when an explicit DACL isn't specified. The DACL has a OWNER RIGHTS SID granted only READ_CONTROL access, the result is that even if the owner of the resource is the service account it isn't possible to write to it. The only way to get full access as per the default DACL is by having the specific service SID in your group list.

Why does LS have one writable process? This I think is probably a "bug" in the Audio Service which creates the AUDIODG process. If we look at the security descriptor of the AUDIODG process we see the following:

<Owner>
- Name : NT AUTHORITY\LOCAL SERVICE

<DACL>
- Type : Allowed
- Name : NT SERVICE\Audiosrv
- Access: Full Access

- Type : Allowed
- Name : NT AUTHORITY\Authenticated Users
- Access: QueryLimitedInformation

The owner is LS which will grant WRITE_DAC access to the resource if nothing else is in the DACL to stop it. However the default DACL's OWNER RIGHTS SID is missing from the DACL, which means this was probably set explicitly by the Audio Service to grant *Authenticated Users* query access. This results in the access not being correctly restricted from other service accounts. Of course AUDIODG has *SeImpersonatePrivilege* so if you find yourself inside a LS unrestricted process with no impersonate privilege you can open AUDIODG (if running) for WRITE_DAC, change the DACL to grant full access and get back impersonate privileges.

If you look at the results one other odd thing you'll notice is that while there are readable threads there are no readable processes, what's going on? If we look at a normal LS service process' security descriptor we see the following:

<Owner>
 - Name : NT AUTHORITY\LogonSessionId_0_202349

<DACL>
 - Type : Allowed
 - Name : NT AUTHORITY\LogonSessionId_0_202349
 - Access: Full Access

 - Type : Allowed
 - Name : BUILTIN\Administrators
 - Access: QueryInformation|QueryLimitedInformation

We should be able to see the reason, the owner is not LS, but instead the logon session SID which is unique per-service. This blocks other LS processes from having any access rights by default. Then the DACL only grants full access to the logon session SID, even administrators are apparently not the be trusted (though they can typically just bypass this using *SeDebugPrivilege*). This security descriptor is almost certainly set explicitly by the SCM when creating the process.

Is there anything else interesting in writable resources outside of the files and keys? The one interesting result shared between NS and LS is a single writable Object Directory. We can take a look at the results to find out what directories these are, to see if they share any common purpose. The directory paths are *\Sessions\0\DosDevices\00000000-000003e4* for NS and *\Sessions\0\DosDevices\00000000-000003e5* for LS. These are the service account's DOS Device directory, the default location to start looking up drive mappings. As the accounts can write to their respective directory this gives another angle of attack, you can compromise any service process running as the same used by dropping a mapping for the C: drive and waiting the process to load a DLL. Leaving that angle open seems sloppy, but it's not like there are no alternative routes to compromise another service.

I think that's the limit of my interest in analysis. I've put my results up on Google Drive here if you want to play around yourself.

Conclusions

Even though I've not run the experiment on multiple machines, at different times with different software I think I can conclude that WSH does not provide any meaningful security boundary when used in its default unrestricted mode. Based on the original hypothesis we can clearly write to resources not created by a service and therefore could likely fully compromise the system. The implementation does do a good job of securing process and thread resources which provide trivial elevation routes but that can be easily compromised if there's appropriate processes running (including some COM services). I can fully support this not being something MS would want to defend through issuing bulletins.

However when used in WR mode WSH is much more comprehensive. I'd argue that as long as a service doesn't have impersonate privilege then it's effectively sandboxed if running in with a WR token. MS already support sandbox escapes as a defended boundary so I'm not sure why WR sandboxes shouldn't also be included as part of that. For example if the trick using the Task Scheduler worked from a WR service I'd see that as circumventing a security boundary, however I don't work in MSRC so I have no influence on what is or is not fixed.

Of course in an ideal world you wouldn't use shared accounts at all. Versions of Windows since 7 have support for Virtual Service Accounts where the service user is the service SID rather than a standard service account and the SCM even limits the service's IL to High rather than System. Of course by default these accounts still have impersonate privilege, however you could also remove that.

Understanding Windows Service Hardening

Last Updated on Wed, 06 Jan 2021 | Windows 7 Resources

Windows Service Hardening (WSH) is a feature of Windows Vista and later versions that is designed to protect critical network services running on a system If a service is compromised, WSH reduces the potential damage that can occur by reducing the attack surface that could be potentially exploited by some forms of malicious code . Because network services (both those built into the operating system and those installed by third-party applications) are by their nature exposed to the network (which itself is usually connected to the Internet), they provide a vector by which attackers can try to compromise a system. WSH implements the following protection improvements over previous versions of Windows:

■ Configuring services to run whenever possible within the lower-privileged LocalService or NetworkService context instead of the LocalSystem context favored by many services in previous versions of Windows .

■ Implementing a new type of per-service security identifier (service SID) that extends the Windows access control model to services and the system resources they access. When a service is started by the Service Control Manager (SCM), the SID is added to the secondary SIDs list of the process token if the service opted for doing this .

■ Applying a write-restricted access token to the process for each service so that any attempt to access a system resource that does not have an explicit allow access control entry (ACE) for the service SID will fail.

■ Tightening control over the generic SvcHost . exe grouping and distribution of services .

■ Reducing the number of privileges assigned to services to only those needed by the service

Understanding Service SIDs

Service SIDs are of the form S-1-5-80-{SHA1 hash of short service name} and complement the existing set of user, group, machine, and special SIDs used by previous versions of Windows. Service SIDs are secondary SIDs that are added to the SIDs list of the service process token when the SCM starts the service. The

primary SID for a service is the built-in identity (LocalService, NetworkService, or LocalSystem) under which the service runs.

To have a service SID added to its token, the service must first opt in to doing so . Opt in is normally done by the operating system or application when the service is started. Administrators can manually opt in user-mode services by using the sc sidtype command, which can configure the service SID as either RESTRICTED, UNRESTRICTED, or NONE. For example, sc sidtype service_name restricted will add the service SID for the service to its service process token and also make it a write-restricted token. This means, for example, that any registry key used by the service must be explicitly assigned permissions to allow the service to access it . On the other hand, sc sidtype service_name unrestricted adds the SID of the service so that access check operations requesting that SID on the service token will succeed. Finally, sc sidtype service_name none does not include any SID in the token . For more information, type sc sidtype ? at a command prompt.

NOTE To query the SID type of a service, you can use the sc qsidtype command.

Some services in Windows Vista and later versions ship out of box as UNRESTRICTED, and most services will fail to start if changed to RESTRICTED . Third-party applications, such as antivirus software, can be designed to opt in to having service SIDs and can be designed to run either RESTRICTED or UNRESTRICTED . If the local administrator changes an existing service SID type from NONE to UNRESTRICTED, she gets the service having SID type with probably zero regression or issues with this service . (A SID type of UNRESTRICTED is sufficient for network traffic filtering.)

NOTE The service SIDs of all the configured services per process are always present in the process. Only the running services have their SIDs enabled; the SIDS of non-running services are there, but in a disabled state. However, the filtering platform considers all SIDs to be activated, regardless of whether the service is in a disabled state.

Windows Firewall and WSH

You can use service SIDs to restrict ways that services can interact with system objects, the file system, the registry, and events. For example, by changing the

permissions of the firewall driver object using the Windows Firewall service SID, this driver will accept communication only from the Windows Firewall service .

WSH also protects services by using rules similar to those used by Windows Firewall. These rules are called service restriction rules, and they are built into Windows and can specify things such as which ports the service should listen on or which ports the service should send data over. An example of a built-in WSH rule might be "The DNS client service should send data only over port UDP/53 and should never listen on any port." These rules add additional protection to network services because network objects, such as ports, do not support ACLs . ISVs can extend this protection to third-party services they develop by using the public Component Object Model (COM) APIs for WSH found at http://msdn.microsoft.com/en-us/library /aa365489.aspx. However, WSH rules don't actually allow traffic (assuming Windows Firewall is turned on); instead, they define the restricted traffic that can be allowed to/from a service, regardless of the administrator-created firewall rules . WSH rules are thus a sandbox for the service.

WSH rules are also merged into the filtering process performed when Windows Firewall with Advanced Security decides whether to pass or drop a packet. In other words, when making decisions about traffic destined to or originating from services, Windows Firewall rules and WSH rules work closely together to decide whether to allow or drop traffic . For more information on how service restriction rules merge with Windows Firewall rules, see the section titled "Understanding Windows Firewall Policy Storage and Rule Merge Logic" later in this chapter

NOTE An assumption behind WSH is that the services being protected are running under either the NetworkService or LocalService accounts. Services running under the LocalSystem account are omnipotent. In other words, they can turn off Windows Firewall with Advanced Security or ignore its rules; and therefore, they are not protected.

Troubleshoot specific Services thus

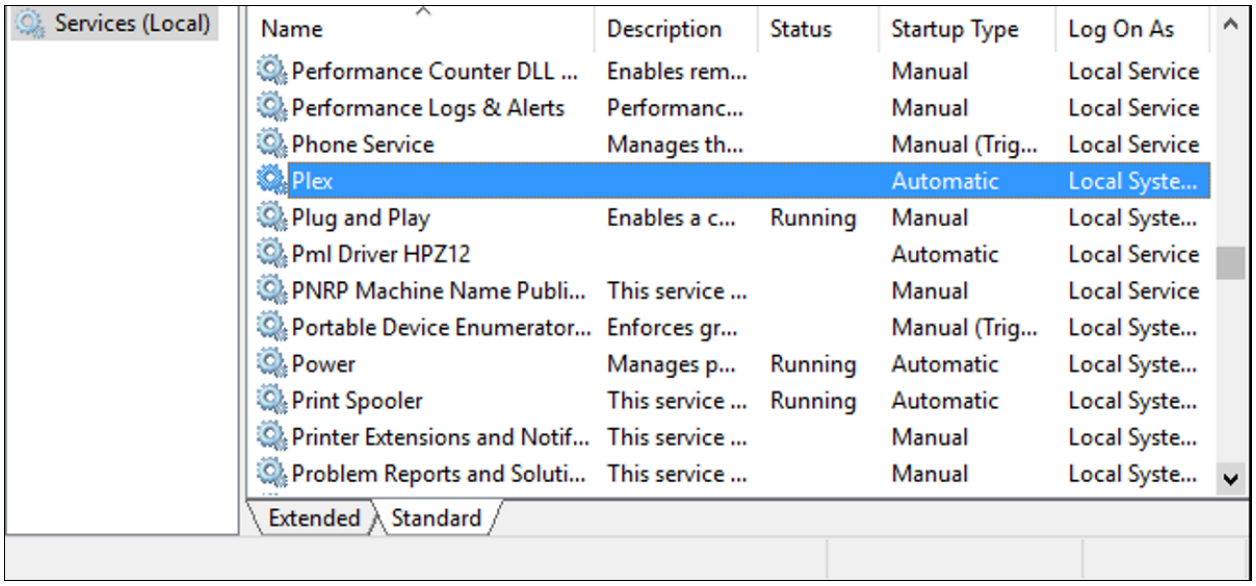
If you are facing problems in starting some specific Services, check if any of these posts can help you:

https://www.thewindowsclub.com/windows-services-will-start

- Windows Time, Windows Firewall, Windows Event Log, services fail to start
- Windows could not start the Windows Update service on Local Computer
- Windows Time Service not working
- Windows Firewall service does not start
- Windows Event Log Service not starting
- Windows Security Center service can't be started
- Windows could not start the WLAN AutoConfig service
- Windows Search service stops
- Windows Defender Service Couldn't Be Started
- User Profile Service failed the logon
- Group Policy Client Service failed to start
- Problem uploading to the Windows Error Reporting service
- Background Intelligent Transfer Service giving problems
- Failed to connect to a Windows service
- Cryptographic Service Provider reported an error
- Windows Wireless Service is not running on this computer.

How to Run Any Program as a Background Service in Windows

https://www.howtogeek.com/50786/using-srvstart-to-run-any-application-as-a-windows-service/



If you're like most Windows users, you have lots of great little utilities that run when you start Windows. While this works great for most apps, there are some that would be nice to start even before a user logs in to the PC. To do this, you'll need to run the app as a Windows service.

Windows services are a special class of programs that are configured to launch and run in the background, usually without any sort of user interface and without needing a user to log in to the PC. Many gamers and power users know them as those things you used to disable to help speed up your system, though that's really not necessary any more.

The primary advantage of running an app as a service is that you can have a program start before a user to log in. That can be particularly important with apps that provide important services you want to be available when you're away from your computer.

RELATED: ***Understanding and Managing Windows Services***

A perfect example of this is Plex, a media server app that can stream local content to just about any device you own. Sure, you could let it sit in the system tray like a

normal program, but what if the computer restarts due to a power outage or scheduled updates? Until you log back in on the PC, Plex wouldn't be available. That's irritating if you have to run to another room to start Plex back up while your popcorn gets cold, and super irritating if you're out of town and trying to stream your media over the Internet. Setting up Plex as a Service would solve that problem.

Before getting started, you should be aware of a couple of important caveats to running an app as a service:

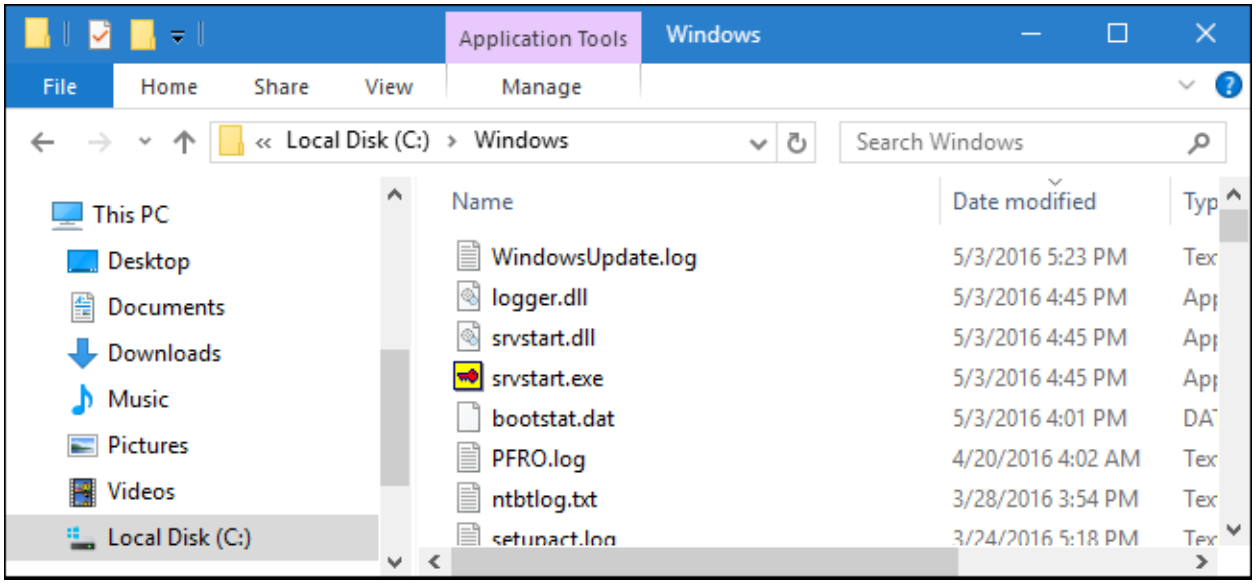
- The app will not put an icon in the system tray. If you need the interface available regularly for an app, it may not be best suited to run as a service.
- When you need to make configuration changes or updates, you'll need to stop the service, run the program as a regular app, do what you need to do, stop the program, and then start the service again.
- If the program is already set up to run when Windows starts, you'll need to disable that so that you don't end up with two instances running. Most programs have an option in the interface for toggling this setting. Others may add themselves to your Startup folder, so you can remove them there.

Ready to roll? Let's talk about how to set it up.

Step One: Install SrvStart

To run an app as a service, you're going to need a small, third-party utility. There are several out there, but our favorite is SrvStart. It was originally designed for Windows NT, and will work with just about any version of Windows from Windows XP on up.

To get started, head over to the SrvStart download page and grab the utility. The download contains just four files (two DLL and two EXE files). There's no installer; instead, copy these to your computer's C:\Windows folder these to your main Windows folder to "install" SrvStart.



We're also going to assume that you've already installed and set up whatever program you're going to turn into a service, but if you haven't, now would be a good time to do that too.

Step Two: Create a Configuration File for the New Service

Next, you'll want to create a configuration file that SrvStart will read to create the service. There's a lot you can do with SrvStart, and you can read the full details on all the configuration options on the documentation page. For this example, we are only going to use two commands: startup, which specifies the program to launch, and shutdown_method, which tells SrvStart how to close the program when the respective service is stopped.

Fire up Notepad and create your configuration file using the format below. Here, we're using Plex, but you can create a file for any program you want to run as a service. The startup command simply specifies the path where the executable file resides. For the shutdown_method command, we're using

the winmessage parameter, which causes SrvStart to send a Windows close message to any windows opened by the service.

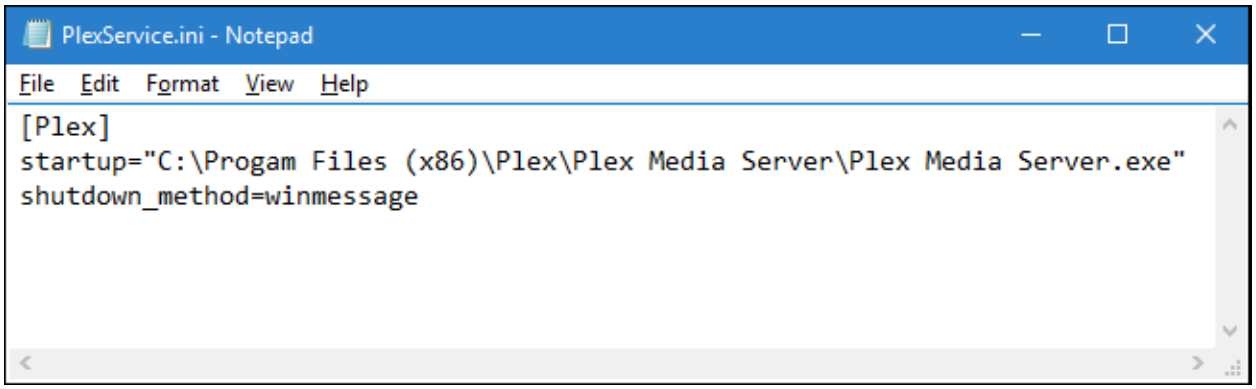
[Plex]

startup="C:\Program Files (x86)\Plex\Plex Media Server\Plex Media Server.exe"

shutdown_method=winmessage

ADVERTISEMENT

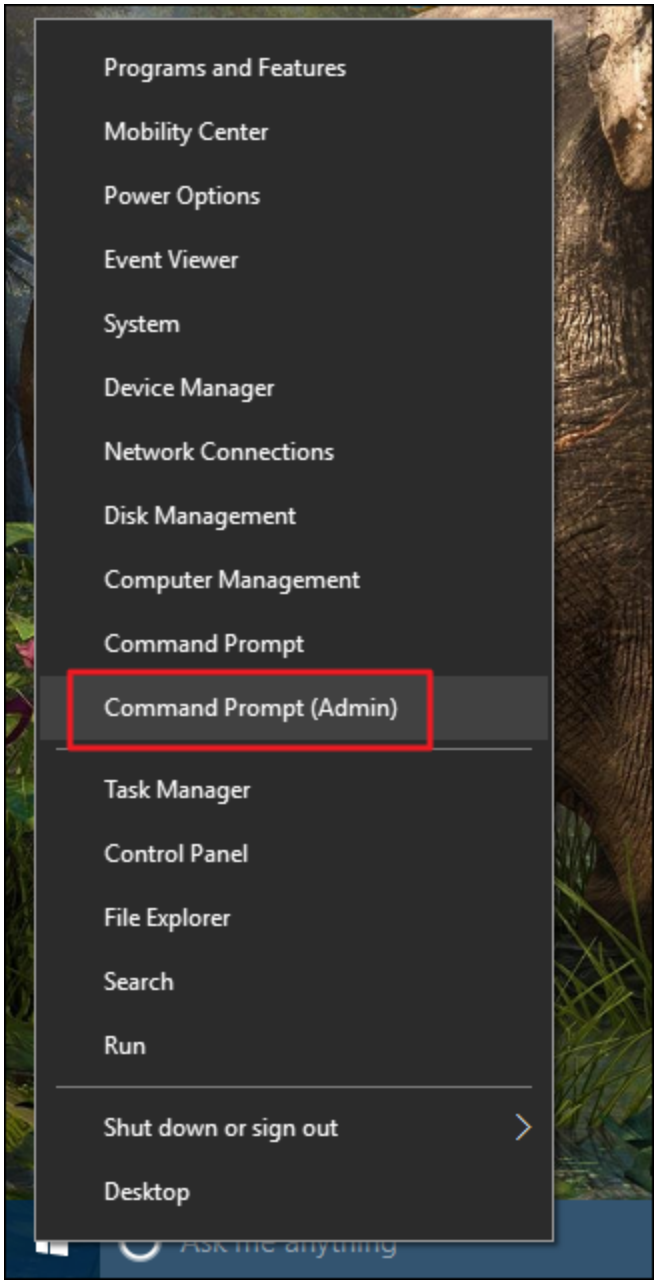
Obviously, adjust the path and name according to the program you're launching.



Save the new configuration file wherever you like, and replace the .txt extension with a .ini extension. Make note of the file name, since we'll need it in the next step. For ease of typing at the Command Prompt, we suggest saving this file temporarily right on your C: drive.

Step Three: Use the Command Prompt to Create the New Service

Your next step is using the Windows Service Controller (SC) command to create the new service based on the criteria in your configuration file. Open Command Prompt by right-clicking the Start menu (or pressing Windows+X), choosing "Command Prompt (Admin)", and then clicking Yes to allow it to run with administrative privileges.



At the Command Prompt, use the following syntax to create the new service:

SC CREATE <servicename> Displayname= "<servicename>" binpath= "srvstart.exe <servicename> -c <path to srvstart config file>" start= <starttype>

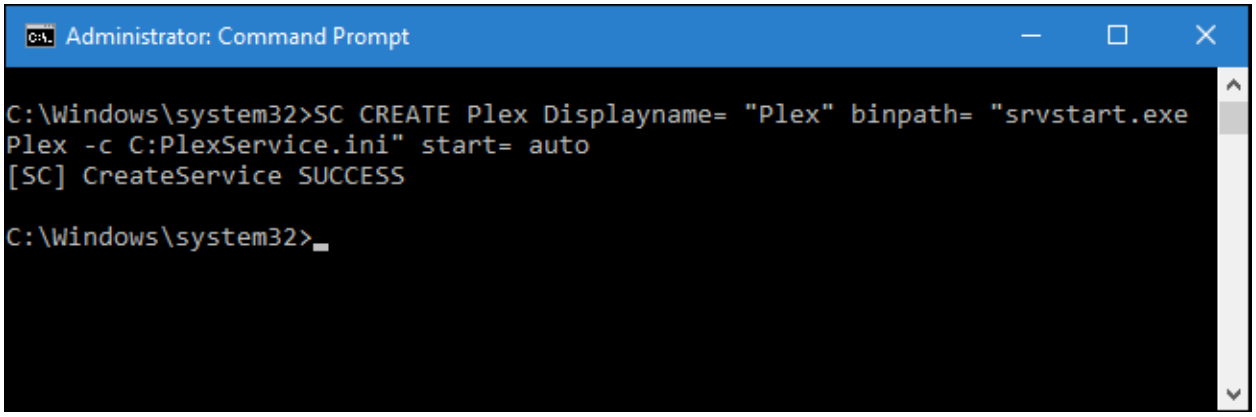
There are a couple of things to note in that command. First, each equal sign (=) has a space after it. That's required. Also, the <servicename> value is entirely up to you. And, finally, for the <starttype> value, you'll want to use auto so that the service starts automatically with Windows.

So in our Plex example, the command would look like this:

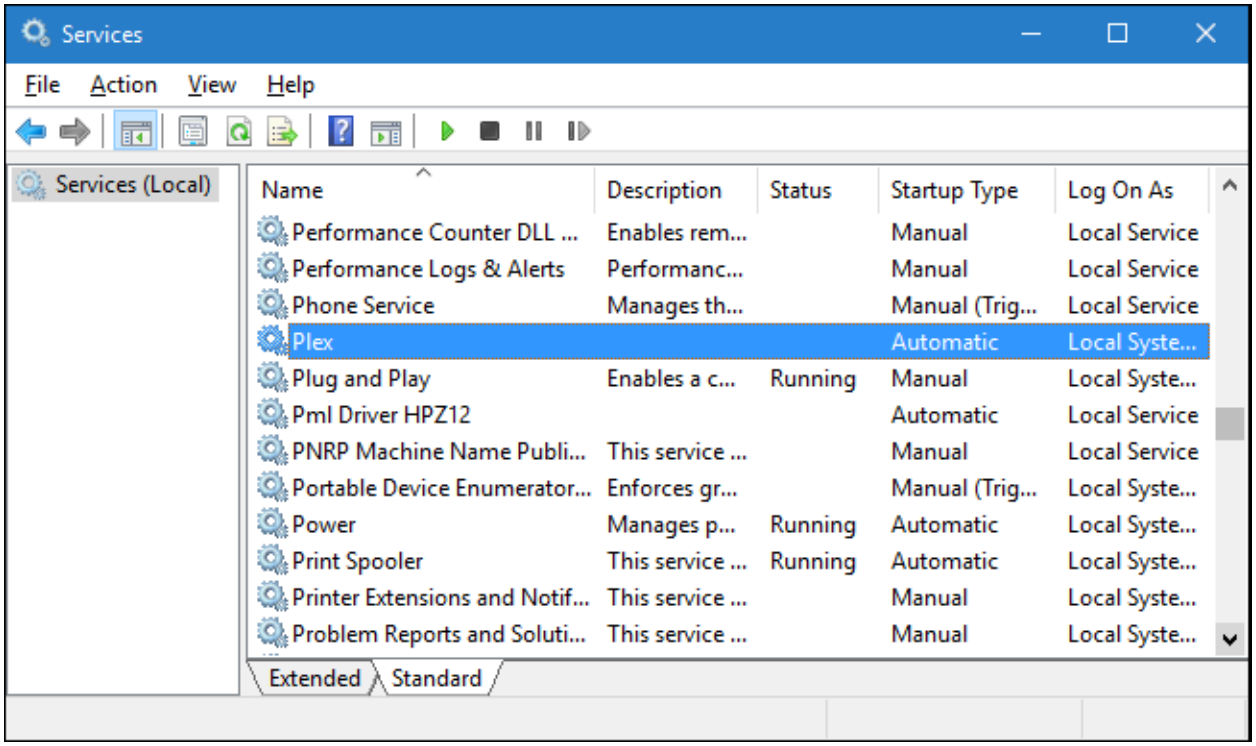
SC CREATE Plex Displayname= "Plex" binpath= "srvstart.exe Plex -c C:PlexService.ini" start= auto

Yes, you read that right: I used C:PlexService.ini instead of C:\PlexService.ini . The command requires you to remove the slash.

When you run the command, you should receive a SUCCESS message if everything goes well.



From this point on, your new service will run whenever Windows starts. If you open the Windows Services interface (just click Start and type “Services”), you can find and configure the new service just like you would any other.



And that's all there is to it. If you have apps that start with Windows and you'd rather they start without needing a user to log in, it's easy enough to turn any app into a service. We've only just touched on the basic method for creating and running a new service, but there's a lot more you can do with SrvStart to fine tune how a service runs. Be sure to check out the documentation if you'd like to learn more.