

Netsh- Ultimate Guide with Examples

Netsh is a great command-line utility that is used to manage, configure, and troubleshoot local or remote network settings. The power of Netsh, Network Shell, comes from the different extensions, better known as contexts, it provides.

Having all the options in one tool is great, but they are also a common problem. You probably don't use it daily, so which switches (options) do you need to use? And did you know you can use the tool to easily troubleshoot network issues?

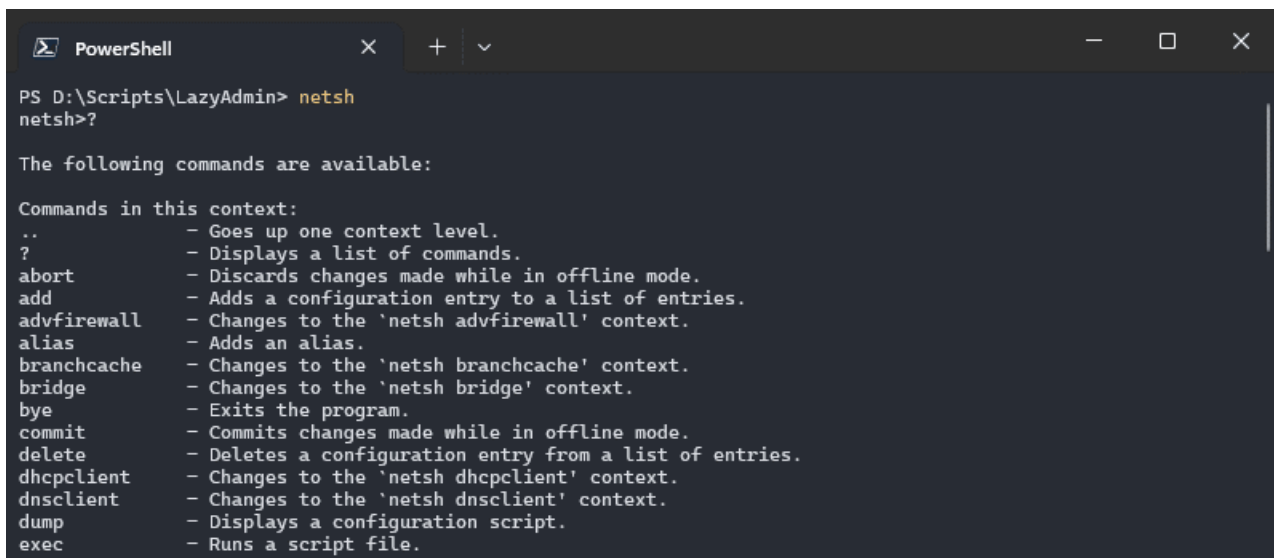
In this article, we are going to take a look at Netsh with the help of commonly used examples. What are the best practices? And how do you troubleshoot network issues with the tool?

The Basics

Before we deep dive into Netsh and look into all the options, let's first start with the basics. Netsh uses a context-based approach, similar to how you navigate through folders in a file system. A context is a group of commands relevant to a specific network configuration area, like `winsock`, `int ip`, `wlan`, or `advfirewall`.

To get started, you can open Windows Terminal or PowerShell and type `netsh`. This will open netsh in interactive mode, allowing you to navigate through the different contexts and options. To view all options, simply type `?` and to exit the shell, type `bye` or `quit`.

netsh
?



```
PS D:\Scripts\LazyAdmin> netsh
netsh>

The following commands are available:

Commands in this context:
..          - Goes up one context level.
?           - Displays a list of commands.
abort       - Discards changes made while in offline mode.
add         - Adds a configuration entry to a list of entries.
advfirewall - Changes to the 'netsh advfirewall' context.
alias       - Adds an alias.
branchcache - Changes to the 'netsh branchcache' context.
bridge      - Changes to the 'netsh bridge' context.
bye         - Exits the program.
commit      - Commits changes made while in offline mode.
delete      - Deletes a configuration entry from a list of entries.
dhcpclient  - Changes to the 'netsh dhcpclient' context.
dnsclient   - Changes to the 'netsh dnsclient' context.
dump        - Displays a configuration script.
exec        - Runs a script file.
```

Commonly I don't use the interactive mode, but find it easier to just type the complete command in one line. But we can also open one of the contexts directly from the command line. To do this there are two options, which may look the same, but have an important difference:

```
netsh wlan
```

```
# Or to open the WLAN context in interactive mode
```

```
netsh -c wlan
```

The first option will just output all the possible commands for the netsh wlan context. But when we use the **-c** switch instead, then we will open netsh interactively in the wlan context.

Netsh Contexts

As mentioned, each context groups commands for a specific part of the network. And because it's an interactive tool, we can easily browse through the different contexts and see which commands are available.

I am not going to fully explain all contexts in this article, we will focus on the most commonly used ones. But below are all contexts that are available with a short description:

Contexts	Description
winsock	Manages Windows Sockets settings and catalog reset for network troubleshooting
int ip	Handles TCP/IP configs, IP stack reset, and static IP assignment
wlan	Manages wireless networks, profiles, and Wi-Fi connections
advfirewall	Configures advanced firewall rules and security profiles
interface	Controls network adapters and IP configurations
trace	Captures network diagnostics and trace logs for troubleshooting
http	Manages HTTP settings, IP listeners, and server configurations
dhcp	Controls DHCP server settings, scopes, and IP distribution
bridge	Manages network bridge connections between segments
routing	Controls routing protocols and static route configurations
ras	Configures remote access services for VPN and dial-up
lan	Manages Local Area Network resources and settings
wins	Controls Windows Internet Name Service for NetBIOS resolution
nap	Manages Network Access Protection health requirements
ikev2	Controls IKEv2 VPN connection settings
ipsec	Configures IPsec policies for secure network communications
rpc	Manages Remote Procedure Call settings
httpstunnel	Controls HTTP tunneling through restrictive networks
wins proxy	Enables cross-subnet WINS server communication
nap client	Manages NAP client health compliance settings

netsh commands

Netsh Examples

As you can see there are a lot of contexts and each context has its own commands (and sometimes even sub-commands). The best way to learn and understand the netsh utility is by examples in my opinion. Below you will find some of the most commonly used netsh commands.

Winsock Reset

The command `netsh winsock reset` is quite often recommended on the internet when you are searching for network problems. But what does this command actually do?

The Windows Sockets (Winsock) API is responsible for network communication between applications and the operating system, Windows in this case. The Winsock catalog can get corrupted which can lead to connectivity issues.

We can reset the WinSock catalog back to the default settings with the following command:

```
netsh winsock reset
```

IP Reset

When it comes to solving networking issues, the second most recommended command to run is `netsh int ip reset`. This command will reset your TCP/IP stack back to the default settings.

When we are talking about the TCP/IP stack, then we mean everything related to your IP Address configuration. This also means that DHCP and DNS configuration will be reset to the default settings, so if you had a static IP Address configured, it will be removed and set to DHCP again.

```
netsh int ip reset
```

For the command to work, it's important to restart your computer afterward. The command is commonly used in combination with the Winsock reset command.

Managing Wireless Networks

One of the common use cases for netsh is to view or manage wireless networks. I wrote an article on how you can view, export, and import your wireless network passwords a [couple of years ago](#). And these are exactly the use cases where you can see how useful this utility is.

The WLAN context allows you to manage your Wireless network connection. We can not only view, import, or export wireless settings, but we can also configure network priorities, or connect to a specific network.

For example to view all stored wireless networks on the device:

```
netsh wlan show profiles
```

We can even zoom in on the wireless network and view all profile details by specifying the profile name:

```
netsh wlan show profiles name="Local UniFi Express"
```

```
PowerShell
PS D:\Scripts\LazyAdmin> netsh wlan show profiles name="Local UniFi Express"

Profile Local UniFi Express on interface Wi-Fi:
=====

Applied: All User Profile

Profile information
=====
Version           : 1
Type              : Wireless LAN
Name              : Local UniFi Express
Control options   :
    Connection mode : Connect automatically
    Network broadcast : Connect only if this network is broadcasting
    AutoSwitch      : Do not switch to other networks
    MAC Randomization : Disabled

Connectivity settings
=====
Number of SSIDs    : 1
SSID name          : "Local UniFi Express"
Network type       : Infrastructure
```

The WLAN context comes with a lot more powerful tools, make sure that you [read my detailed](#) article about it to learn more.

Advanced Firewall Management

The advfirewall context allows you to manage pretty much all aspects of the Windows firewall. We can view the current configuration, enable or disable firewall rules, add custom rules, and configure profiles.

To view the current firewall configuration, we can use the following options:

show all profiles

```
netsh advfirewall show allprofiles
```

View all firewall rules

```
netsh advfirewall firewall show rule name=all profile=domain
```

The first option will show all firewall profiles and if they are enabled or not. The second command displays all firewall rules. We can fine-tune the command to show only the rules from a specific profile or even show a specific rule based on the name of it.

When it comes to managing the firewall, we can have a lot of options. I will show you some of the most convenient ones to know:

To troubleshoot network issues, you can temporarily turn off the firewall. This way you can quickly see if a firewall might be causing the connection issue. We can disable all profiles or a specific profile:

Disable all profiles

```
netsh advfirewall set allprofiles state off
```

Disable a specific profile

```
netsh advfirewall set domainprofile state off
```

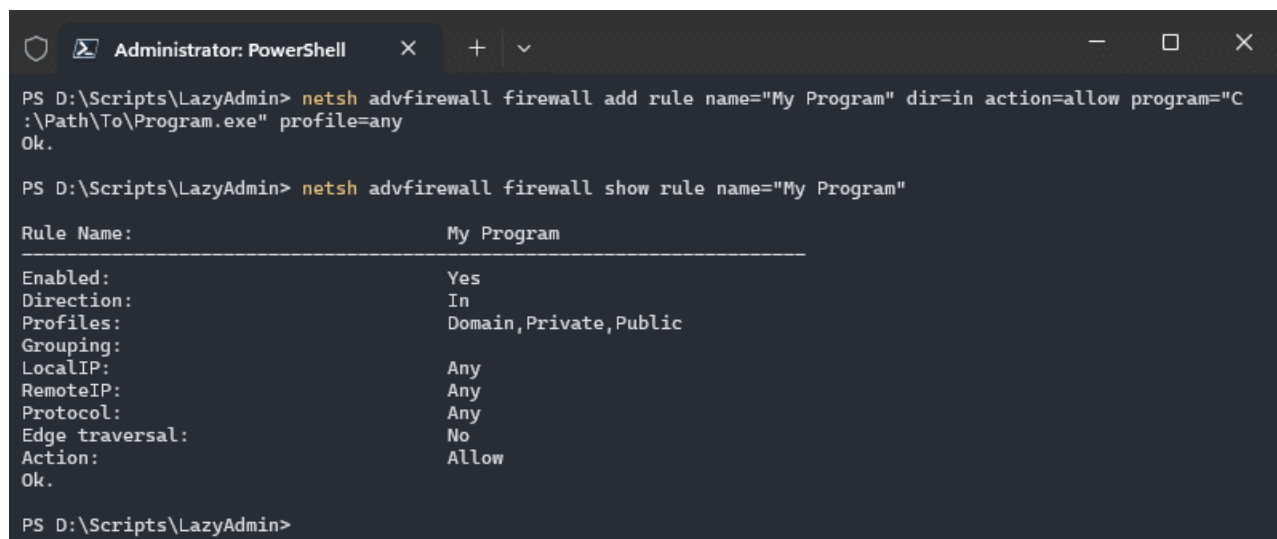
We can also reset the firewall back to the default settings. But before you do that, I recommend that you first make an export of the current settings. Both can be done in the netsh firewall context:

```
# Export the current settings
netsh advfirewall export "C:\temp\Firewall-Config-Export.wfw"
# Reset the firewall to default settings
netsh advfirewall reset
```

Another common use case for the netsh firewall commands is to add or remove firewall rules. We can create the rules based on protocol, port number, or a specific program. Below you will find some examples.

The rules are enabled by default, but you can also create the rule and leave it disabled by adding the parameter `enable=no`. And if you want to block a program or port, simply replace the `action=allow` with `action=block` of course.

```
# Allow specific port (80 and 443)
netsh advfirewall firewall add rule name="Web Server" dir=in action=allow protocol=TCP
localport=80,443
# Allow specific program
netsh advfirewall firewall add rule name="My Program" dir=in action=allow
program="C:\Path\To\Program.exe" profile=any
# Allow specific protocol (ICMP ping)
netsh advfirewall firewall add rule name="Allow ICMP" dir=in action=allow
protocol=icmpv4
```



```
Administrator: PowerShell
PS D:\Scripts\LazyAdmin> netsh advfirewall firewall add rule name="My Program" dir=in action=allow program="C
:\Path\To\Program.exe" profile=any
Ok.
PS D:\Scripts\LazyAdmin> netsh advfirewall firewall show rule name="My Program"

Rule Name:                               My Program
-----
Enabled:                                     Yes
Direction:                               In
Profiles:                               Domain,Private,Public
Grouping:
LocalIP:                                   Any
RemoteIP:                                 Any
Protocol:                                 Any
Edge traversal:                             No
Action:                                   Allow
Ok.
PS D:\Scripts\LazyAdmin>
```

Interface Management

We can use the `netsh interface` context to view and configure network interfaces on the device. A good place to start is to view the current interfaces and their connection status. This will tell you which interfaces (Wi-Fi, Ethernet or other adapters) are currently being used:

netsh interface show interface

To view current configuration of an interface, for example, your wireless network adapter, we need to open the **ipv4** context first. Then we can show the config of a specific interface:

netsh interface ipv4 show config "Wi-Fi"

```
PowerShell
PS D:\Scripts\LazyAdmin> netsh interface ipv4 show config "Wi-Fi"

Configuration for interface "Wi-Fi"
    DHCP enabled:                Yes
    IP Address:                  192.168.51.136
    Subnet Prefix:               192.168.51.0/24 (mask 255.255.255.0)
    Default Gateway:            192.168.51.1
    Gateway Metric:              0
    InterfaceMetric:            30
    Statically Configured DNS Servers: 1.1.1.1
                                   8.8.8.8
    Register with which suffix:  Primary only
    WINS servers configured through DHCP: None

PS D:\Scripts\LazyAdmin> |
```

Another commonly used option is to view the route table. This allows you to troubleshoot connection issues to other subnets.

netsh interface ipv4 show route

```
PowerShell
netsh interface ipv4>show route

Publish Type Met Prefix Idx Gateway/Interface Name
----
No Manual 0 0.0.0.0/0 2 192.168.51.1
No System 256 127.0.0.0/8 1 Loopback Pseudo-Interface 1
No System 256 127.0.0.1/32 1 Loopback Pseudo-Interface 1
No System 256 127.255.255.255/32 1 Loopback Pseudo-Interface 1
No System 256 192.168.51.0/24 2 Wi-Fi
No System 256 192.168.51.136/32 2 Wi-Fi
No System 256 192.168.51.255/32 2 Wi-Fi
No System 256 192.168.76.0/24 20 VMware Network Adapter VMnet1
No System 256 192.168.76.1/32 20 VMware Network Adapter VMnet1
No System 256 192.168.76.255/32 20 VMware Network Adapter VMnet1
No System 256 192.168.209.0/24 16 VMware Network Adapter VMnet8
No System 256 192.168.209.1/32 16 VMware Network Adapter VMnet8
No System 256 192.168.209.255/32 16 VMware Network Adapter VMnet8
No System 256 224.0.0.0/4 1 Loopback Pseudo-Interface 1
No System 256 224.0.0.0/4 19 OpenVPN Data Channel Offload for NordVPN
```

Besides view the configuration, we can of course also configure settings in the **netsh interface** context.

For example, to assign an static IP address and configure a DNS server to a network adapter (interface), we can use the following commands:

Set a static IP Address

```
netsh interface ip set address name="Ethernet" static 192.168.1.10 255.255.255.0 192.168.1.1
```

Set the DNS server

```
netsh interface ip set dns name="Ethernet" static 8.8.8.8
```

To set the adapter back to DHCP mode, we need to change the source to DHCP:

```
# Set the IP Address to DHCP
```

```
netsh interface ipv4 set address name="Ethernet" source=dhcp
```

```
# Set DNS to DHCP
```

```
netsh interface ipv4 set dns name="Ethernet" source=dhcp
```

Another command to use is to disable IPv6 on a machine. We will need to do this for each interface with the following command:

```
netsh interface ipv6 set interface "Ethernet" disabled=yes
```

WinHTTP Proxy

Proxy settings are commonly used in corporate environments, but can also be used with VPN providers. The netsh winhttp context allows you to view and configure proxies settings.

To check if there is any proxy service configure, we can use the following command:

```
netsh winhttp show proxy
```

Adding a proxy service is done with the set proxy command. When adding a proxy server, you often have domains that should not go through the proxy server. These can be added in the bypass-list. In the example below, all local addresses will bypass the proxy:

```
netsh winhttp set proxy proxy-server="proxy.company.com:8080" bypass-list="*.local;  
<local>"
```

To remove all proxy servers from the device, we can simply reset the proxy configuration:

```
netsh winhttp reset proxy
```

Troubleshooting with Netsh Trace

What most don't know is that the netsh utility is also a powerful diagnostic tool for troubleshooting network issues. The netsh trace context offers a list with predefined scenarios to monitor, capture and analyze specific network components.

Each trace scenario is targeted to a specific network operator or traffic pattern, like Wi-Fi connection, DNS operations or TCP/IP flows. When you run a scenario, then a detail log with all the network traffic, packet details, errors, etc will be created.

The log files can be opened in tools like Microsoft Message Analyzer or Wireshark for further analyzes.

To view all available trace scenarios, we can use the following command:

```
netsh trace show scenarios
```

As you will see, there are quite a lot of options available. Some of common scenarios are:

- **NetConnection** – Captures general network connectivity issues.
- **NetEvent** – Logs network events at a high level.

- **IPSec** – Monitors IPsec connections for troubleshooting VPNs or secure communication.
- **WebIO** – Tracks HTTP/HTTPS traffic, useful for web application troubleshooting.
- **WFP** – Captures events related to Windows Filtering Platform, relevant to firewall and packet filtering issues.
- **Wireless** – Monitors Wi-Fi activity, helpful for diagnosing wireless connection issues.

Running a scenario is done in a couple of steps.

First you will need to start the trace. This will tell netsh to start capturing all the details as defined in the scenario. You will need to specify the location for the trace file (.etl) and of course the scenario that you want to run:

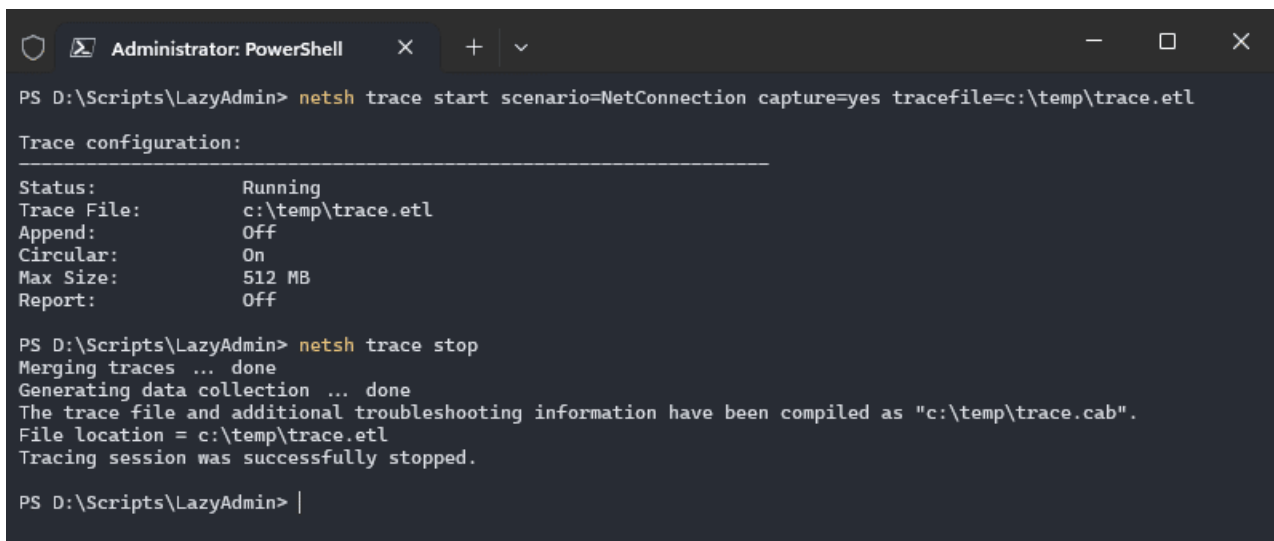
```
netsh trace start scenario=NetConnection capture=yes tracefile=c:\temp\trace.etl
```

With the trace running, you will need to repeat the steps that cause the network issues. The trace will run in the background and capture all the details.

When the problems that you want to capture has occurred, you will need to stop the trace.

```
netsh trace stop
```

It takes a couple of minutes to collect all the data and create the trace file. But when done, you can analyze the ETL file in a network analyzer like Wireshark.



```

Administrator: PowerShell
PS D:\Scripts\LazyAdmin> netsh trace start scenario=NetConnection capture=yes tracefile=c:\temp\trace.etl

Trace configuration:
-----
Status:           Running
Trace File:       c:\temp\trace.etl
Append:           Off
Circular:         On
Max Size:         512 MB
Report:           Off

PS D:\Scripts\LazyAdmin> netsh trace stop
Merging traces ... done
Generating data collection ... done
The trace file and additional troubleshooting information have been compiled as "c:\temp\trace.cab".
File location = c:\temp\trace.etl
Tracing session was successfully stopped.

PS D:\Scripts\LazyAdmin> |

```

Troubleshoot Wireless Network

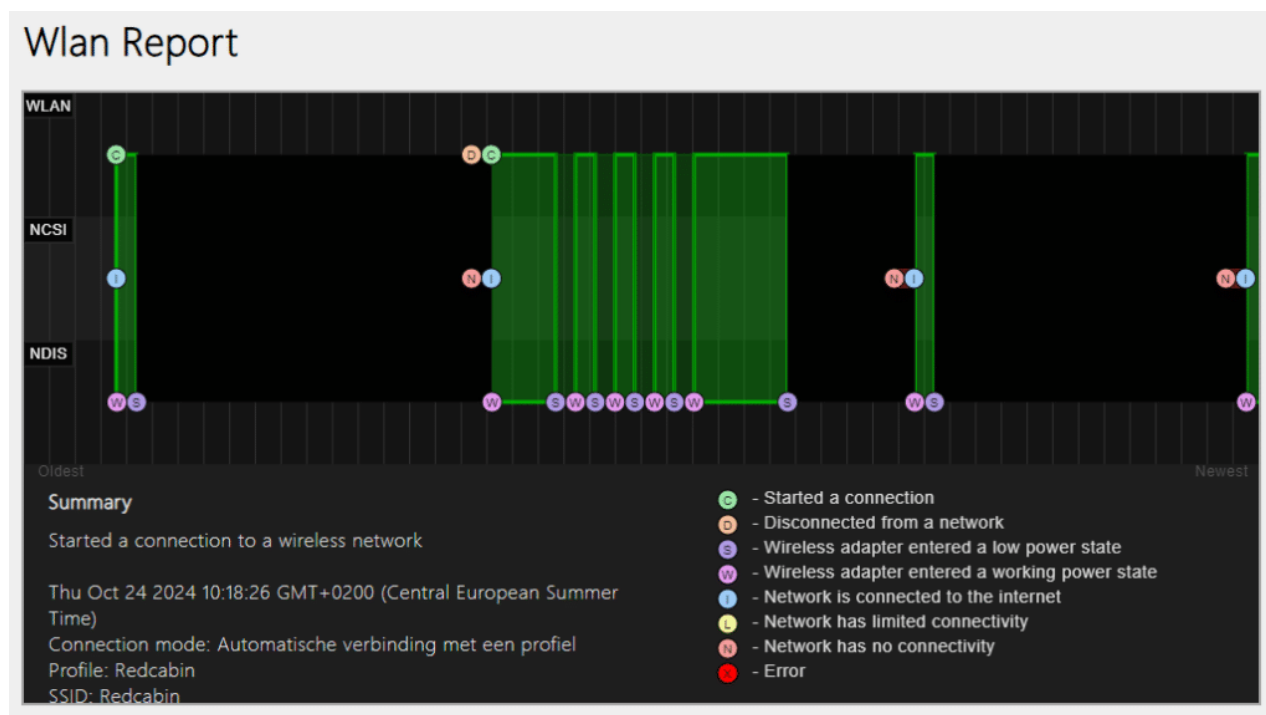
When we are talking about network troubleshooting, then there is one unique feature in netsh that is worth mentioning. To troubleshoot wireless network connections, we can create a WLAN HTML report.

This report will show in a graph all the wireless network connections, when the connection is started, disconnected, length of the connection, powerstate and more. The report can be really helpful when troubleshooting unstable wireless connection on a client device.

To create the report, you can run the command below. By default the report will be generate over the last 3 days, but you can change this with the duration parameter

```
netsh wlan show wlanreport duration="7"
```

When you open the report, you will first see a connection graph. This shows when a connection was made, when the adapter entered a different state or any connection errors.



If you scroll a bit down through the report, you will find a lot more useful information regarding the wireless network connections. It will first show all the network adapters information and some output of specific network related commands.

But below that, you will see a summary of all the connections, and an overview of the disconnect reasons and session duration. And at the end of the report, you will find a detailed report per wireless session with all the connection details.

Session Success/Failures

Status	Count
Successes	0
Failures	3
Warnings	15

Disconnect Reasons

Reason	Count
De verbinding met het netwerk wordt door het stuurprogramma verbroken.	8
De verbinding met het netwerk is verbroken omdat de gebruiker een nieuwe verbinding wil maken.	6
De verbinding met het netwerk wordt verbroken vanwege een wijzigingsaanvraag voor de bewerkingsstatus op deze interface	2
De verbinding met het netwerk wordt verbroken vanwege een aanvraag tot verbinding tijdelijk verbreken.	1
De verbinding met het stuurprogramma is verbroken tijdens het koppelen.	1

Wrapping Up

When you need to view or change network related settings, then netsh is a great tool to use. If you get a bit familiar with basics commands, it's often quicker to use netsh then clicking through all the interfaces on Windows these days.

Before changing settings, always more sure that you have a backup of the current settings. You can do that simply with the `dump` option in most contexts. This way you can always revert back to the previous settings.

I hope you found this article useful, if you have any questions, just drop a comment below.