

# Log-Connections.ps1 User Guide

---

By Ken Hartman

web: [www.KennethGHartman.com](http://www.KennethGHartman.com)

email: [kgh@kennethghartman.com](mailto:kgh@kennethghartman.com)

## General Description

The Log-Connections.ps1 file is a PowerShell Script that Logs active TCP connections and includes the process ID (PID) and process name for each connection on a Microsoft Windows computer. The log file name is a parameter that is passed to the script at run time. A log entry is created every time that the list of processes with open connections or listening ports changes. If the ports or remote addresses are not yet established, they are shown as an asterisk (\*).

The Log-Connections script is based on the netstat command, "`netstat -nao`," which can be run at the Windows command prompt to display a snapshot of all connections and listening ports. The `-o` switch tells `netstat` to display the owning process ID that is associated with each process. A limitation of the `netstat` command is that it cannot report the associated process name, just the PID. To achieve this, the Log-Connections PowerShell script calls the `Get-NetworkStatistics` function. This function was written by Shay Levy (<http://PowerShay.com>) and is available at <http://poshcode.org/2701>.

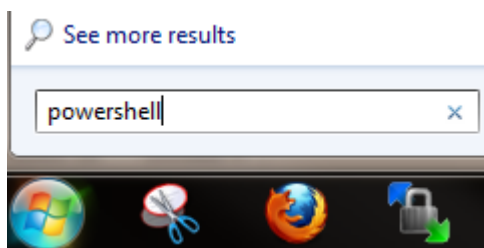
The Log-Connections script calls the `Get-NetworkStatistics` function repeatedly in an infinite loop, comparing the current snapshot with the previous. If there is a change, the current snapshot is time stamped, logged to the file, and optionally passed through to the PowerShell pipeline.

Passing the connections snapshot object through to the PowerShell pipeline allows the data to be manipulated or displayed in real-time by other PowerShell cmdlets. This will be illustrated in the examples that follow.

## Quick Intro to PowerShell

Windows PowerShell is a command-line shell and a scripting language that is built upon the .NET Framework. PowerShell has been around since 2006, but is included in the base OS with Windows 7 and Windows Server 2008 R2.

PowerShell can be invoked by typing "powershell" in the search box above the Windows Start Button,



or by typing "powershell" at the Windows command prompt.

A great website to learn more about PowerShell is at <http://www.powershellpro.com/powershell-tutorial-introduction/>.

## PowerShell Security Features

PowerShell has an script execution policy that by default will prevent scripts from being executed unintentionally. To learn more about execution policies, type the following command in PowerShell:

```
Get-Help about_Execution_Policies
```

To be able to run the following examples, it is suggested that you change the execution policy to "RemoteSigned" so that local scripts can run unsigned and remote scripts can only run if signed by a trusted entity. To do this run the following PowerShell command:

```
Set-ExecutionPolicy RemoteSigned -scope CurrentUser
```

To revert back to the default, just type:

```
Set-ExecutionPolicy Restricted
```

Another feature of PowerShell is that scripts cannot be executed in the current directory by typing just the script name. Instead they must be prefaced with a dot and a backslash, ( ". \" ). This is illustrated in the examples below.

### EXAMPLE 1

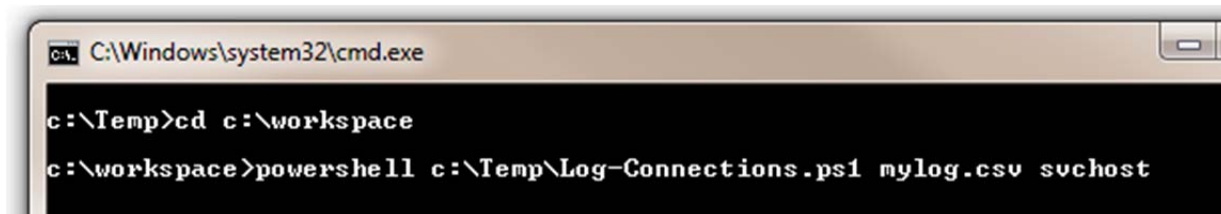
```
powershell .\Log-Connections.ps1 c:\workspace\mylog.csv
```



Example 1 shows the Log-Connections.ps1 script being invoked from the Windows command prompt. The ps1 script and the complete path to the log file are passed in as arguments the command "powershell." Also note the use the file extension of "CSV." This is convenient because on many systems when a CSV file is double clicked it will launch Microsoft Excel. Any other extension, including "TXT" is also acceptable.

### EXAMPLE 2

```
powershell c:\Temp\Log-Connections.ps1 mylog.csv svchost
```



```
C:\Windows\system32\cmd.exe

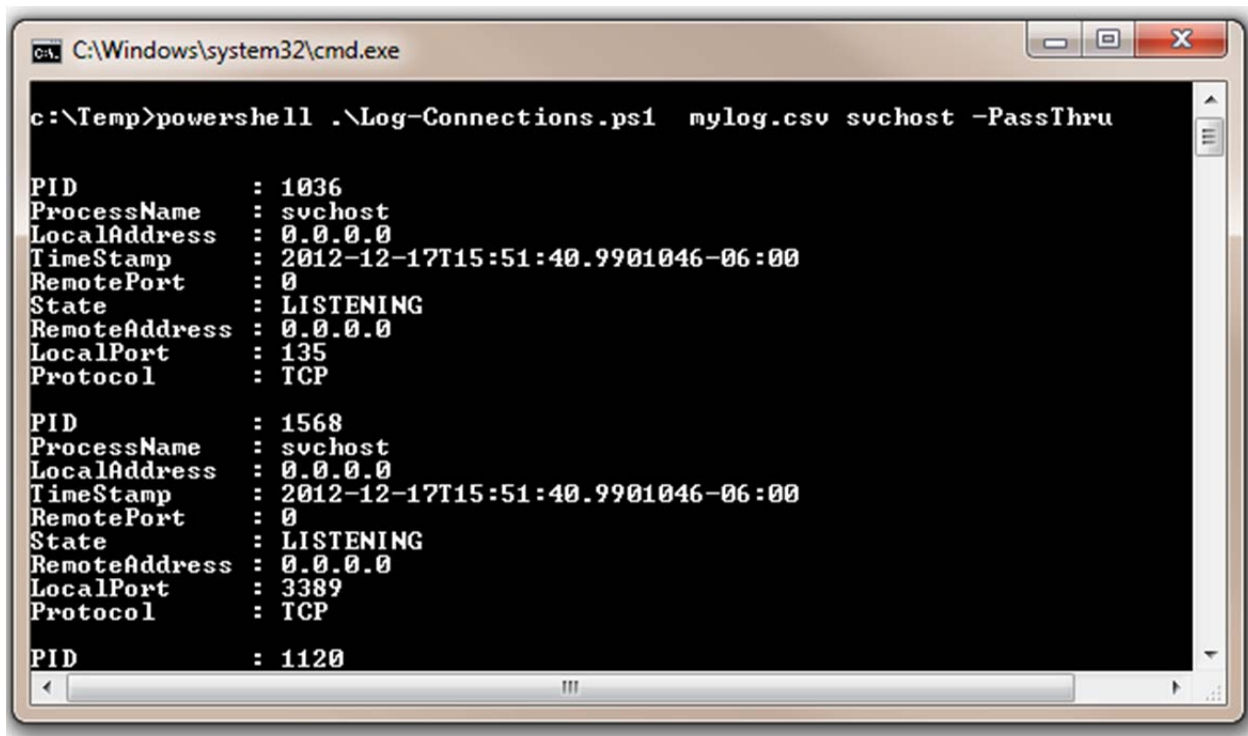
c:\Temp>cd c:\workspace
c:\workspace>powershell c:\Temp\Log-Connections.ps1 mylog.csv svchost
```

Example 2 is very similar to the first one except a process name has been passed in as an argument and just the file name (without the path) has been provided, so the log file will be saved in the current directory, c:\workspace. The full path to the script is provided because it is in c:\temp.

Note: if the log file exists already, the new observations will be appended to the bottom

### EXAMPLE 3

```
powershell .\Log-Connections.ps1 mylog.csv svchost -PassThru
```



```
C:\Windows\system32\cmd.exe

c:\Temp>powershell .\Log-Connections.ps1 mylog.csv svchost -PassThru

PID           : 1036
ProcessName    : svchost
LocalAddress   : 0.0.0.0
TimeStamp      : 2012-12-17T15:51:40.9901046-06:00
RemotePort     : 0
State          : LISTENING
RemoteAddress  : 0.0.0.0
LocalPort      : 135
Protocol       : TCP

PID           : 1568
ProcessName    : svchost
LocalAddress   : 0.0.0.0
TimeStamp      : 2012-12-17T15:51:40.9901046-06:00
RemotePort     : 0
State          : LISTENING
RemoteAddress  : 0.0.0.0
LocalPort      : 3389
Protocol       : TCP

PID           : 1120
```

Using the “-PassThru” switch will cause the script display the results to the screen in a raw format in addition to logging them in the log file.

### EXAMPLE 4

```
.\Log-Connections.ps1 -ProcName svchost -Filepath mylog.csv
```

```
C:\Windows\system32\cmd.exe - powershell

c:\Temp>powershell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Temp> .\Log-Connections.ps1 -ProcName svchost -Filepath mylog.csv
```

In Example 4 the user typed “powershell” at the command prompt to invoke PowerShell. Then the name of the script and its arguments were typed at the PowerShell prompt.

This example also demonstrated the use of the named parameters convention. Passing in the parameter value (e.g. “mylog.csv”) after the parameter name (“-Filepath”) allows the parameters to be passed in out of order.

**EXAMPLE 5**

`.\Log-Connections.ps1`

```
C:\Windows\system32\cmd.exe - powershell

c:\Temp>powershell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Temp> .\Log-Connections.ps1

cmdlet Log-Connections.ps1 at command pipeline position 1
Supply values for the following parameters:
FilePath:
```

Example 6 shows that PowerShell will gracefully request any missing parameters that are mandatory. In this case it is the FilePath value.

**EXAMPLE 6**

`.\Log-Connections.ps1 mylog.csv svchost -PassThru | Format-Table`

```
C:\Windows\system32\cmd.exe - powershell

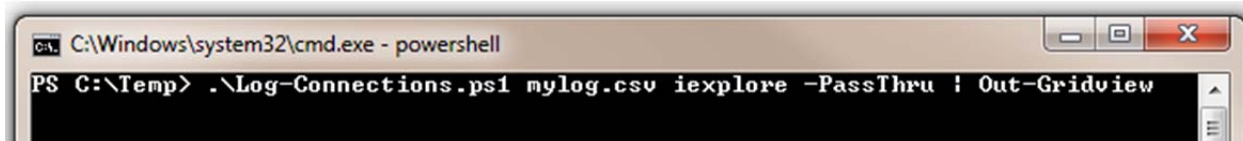
PS C:\Temp> .\Log-Connections.ps1 mylog.csv svchost -PassThru | Format-Table
```

PID	ProcessName	LocalAddress	TimeStamp	RemotePort	State	RemoteAddress	LocalPort	Protocol
1036	svchost	0.0.0.0	2012-...	0	LISTE...	0.0.0.0	135	TCP
1568	svchost	0.0.0.0	2012-...	0	LISTE...	0.0.0.0	3389	TCP
1120	svchost	0.0.0.0	2012-...	0	LISTE...	0.0.0.0	49153	TCP
1232	svchost	0.0.0.0	2012-...	0	LISTE...	0.0.0.0	49154	TCP
364	svchost	0.0.0.0	2012-...	0	LISTE...	0.0.0.0	50326	TCP
1400	svchost	0.0.0.0	2012-...	*	*	*	123	UDP
1232	svchost	0.0.0.0	2012-...	*	*	*	500	UDP
1232	svchost	0.0.0.0	2012-...	*	*	*	4500	UDP
1568	svchost	0.0.0.0	2012-...	*	*	*	5355	UDP
3728	svchost	127.0...	2012-...	*	*	*	1900	UDP
1568	svchost	127.0...	2012-...	*	*	*	49455	UDP

Example 6 illustrates piping the output of the Log-Connections script to the Format-Table cmdlet. The Format-Table cmdlet produces a nice table of the results that will grow in real time.

## EXAMPLE 7

```
.\Log-Connections.ps1 mylog.csv iexplore -PassThru | Out-GridView
```



Using the Out-GridView cmdlet, as shown in Example 7, will produce a grid of the results. The grid grows in real time and can be filtered and sorted. The columns can also be re-arranged.

TimeStamp	PID	ProcessName	LocalAddress	LocalPort	RemoteAddress	RemotePort	State	Protocol
2012-12-17T16:52:10.1680190-06:00								
2012-12-17T16:52:18.0648086-06:00	6980	iexplore	192.168.40.177	55513	173.194.64.106	80	ESTABLISHED	TCP
2012-12-17T16:52:18.0648086-06:00	6980	iexplore	127.0.0.1	56771	*	*		UDP
2012-12-17T16:52:46.6536672-06:00	6980	iexplore	192.168.40.177	55513	173.194.64.106	80	ESTABLISHED	TCP
2012-12-17T16:52:46.6536672-06:00	6980	iexplore	192.168.40.177	55514	173.194.64.106	80	ESTABLISHED	TCP
2012-12-17T16:52:46.6536672-06:00	6980	iexplore	127.0.0.1	56771	*	*		UDP
2012-12-17T16:52:49.6509669-06:00	6980	iexplore	192.168.40.177	55513	173.194.64.106	80	ESTABLISHED	TCP
2012-12-17T16:52:49.6509669-06:00	6980	iexplore	192.168.40.177	55514	173.194.64.106	80	ESTABLISHED	TCP
2012-12-17T16:52:49.6509669-06:00	6980	iexplore	192.168.40.177	55515	74.125.225.34	80	ESTABLISHED	TCP
2012-12-17T16:52:49.6509669-06:00	6980	iexplore	192.168.40.177	55516	173.194.64.94	80	ESTABLISHED	TCP
2012-12-17T16:52:49.6509669-06:00	6980	iexplore	127.0.0.1	56771	*	*		UDP
2012-12-17T16:52:57.3757393-06:00	6980	iexplore	192.168.40.177	55513	173.194.64.106	80	ESTABLISHED	TCP
2012-12-17T16:52:57.3757393-06:00	6980	iexplore	192.168.40.177	55514	173.194.64.106	80	ESTABLISHED	TCP
2012-12-17T16:52:57.3757393-06:00	6980	iexplore	192.168.40.177	55515	74.125.225.34	80	ESTABLISHED	TCP
2012-12-17T16:52:57.3757393-06:00	6980	iexplore	192.168.40.177	55516	173.194.64.94	80	ESTABLISHED	TCP

The grid can also be filtered using the “Add Criteria” button as shown below:

TimeStamp	PID	ProcessName	LocalAddress	LocalPort	RemoteAddress	RemotePort	State	Protocol
2012-12-17T16:52:49.6509669-06:00	6980	iexplore	192.168.40.177	55515	74.125.225.34	80	ESTABLISHED	TCP
2012-12-17T16:52:57.3757393-06:00	6980	iexplore	192.168.40.177	55515	74.125.225.34	80	ESTABLISHED	TCP