

MICROSOFT WINDOWS – CONFIGURE THE MAX LIMIT FOR CONCURRENT TCP CONNECTIONS

To keep the **TCP/IP** stack from taking all resources on the computer, there are different parameters that control how many connections it can handle.

If running applications that are constantly opening and closing connections (**P2P**), or are providing a service which many tries to connect to at the same time (**Web-server like IIS**), then one can improve the performance of these applications by changing the restriction limits.

TcpNumConnections

There is a parameter that limits the maximum number of connections that **TCP** may have open simultaneously.

```
[HKEY_LOCAL_MACHINE \System \CurrentControlSet \Services \Tcpip \Parameters]  
TcpNumConnections = 0x00ffffff (Default = 16,777,214)
```

Note: a 16 Million connection limit sounds very promising, but there are other parameters, which keeps us from ever reaching this limit.

When a client makes a **connect()** call to make a connection to a server, then the client **invisible/implicit** bind the socket to a local dynamic (anonymous, ephemeral, short-lived) port number. The default range for dynamic ports in **Windows** is **1024 to 5000**, thus giving **3977** outbound concurrent connections for each IP Address.

It is possible to change the upper limit with this DWORD registry key:

```
[HKEY_LOCAL_MACHINE \System \CurrentControlSet \Services \Tcpip \Parameters]  
MaxUserPort = 5000 (Default = 5000, Max = 65534)
```

Note: it is possible to reserve port numbers so they aren't used as dynamic ports in case one have a certain application that needs them. This is done by using the [ReservedPorts \(Q812873\)](#) setting.

Note: **Vista** changes the default range from **1024-5000** to **49152-65535**, which can be controlled with the dynamicport setting using **netsh**. More Info [MS KB929851](#).

More Info: [The Cable Guy - Ephemeral, Reserved, and Blocked Port Behavior](#)

More Info: [MS KB Q196271](#)

More Info: [MS KB Q319502](#)

More Info: [MS KB Q319504](#)

More Info: [MS KB Q328476](#)

More Info: [MS KB Q836429](#)

MaxFreeTcbs

For each connection a **TCP Control Block (TCB - Data structure using 0.5 KB pagepool and 0.5 KB non-pagepool)** is maintained. The **TCBs** are pre-allocated and stored in a table, to avoid spending time on allocating/deallocating the **TCBs** every time connections are created/closed.

The **TCB Table** enables *reuse/caching* of **TCBs** and improves memory management, but the static size limits how many connections **TCP** can support simultaneously (**Active + TIME_WAIT**).

Configure the size of the **TCB Table** with this **DWORD** registry key:

```
[HKEY_LOCAL_MACHINE \System \CurrentControlSet \Services \Tcpip \Parameters]  
MaxFreeTcbs = 2000 (Default = RAM dependent, but usual Pro = 1000, Srv=2000)
```

MaxHashTableSize

To make lookups in the **TCB** table faster a hash table has been made, which is optimized for finding a certain active connection. If the **hash table** is too small compared to the total amount of active connections, then extra **CPU** time is required to find a connection.

Configure the size of the hash table with this **DWORD** registry key (*Is allocated from pagepool memory*):

```
[HKEY_LOCAL_MACHINE \System \CurrentControlSet \services \Tcpip \Parameters]  
MaxHashTableSize = 512 (Default = 512, Range = 64-65536)
```

Note: Microsoft recommends for a multiprocessor environment, that the value should not be higher than the maximum amount of concurrent connections (MaxFreeTcbs), also if multiprocessor then it might be interesting to look at the registry-key **NumTcbTablePartitions** (*Recommended value CPU-count multiplied by 4*).

More Info: [MS KB Q151418](#)

More Info: [MS KB Q224585](#)

TcpTimedWaitDelay

If having allocated a **1000 TCBs** then it doesn't mean that one will be able to have a **1000 active connections**. Especially if the application is quickly opening and closing connections, because after a connection is "**closed**" it enters the state **TIME_WAIT**, and will continue to occupy the port number for 4 minutes (**2*Maximum Segment Live, MSL**) before it is actually removed.

This behavior is specified in [RFC 793](#), and prevents attempts to reconnect to the same party, before the old socket is recognized as closed at both sides.

It is possible to change how long a socket should be in **TIME_WAIT** state before it can be re-used freely:

```
[HKEY_LOCAL_MACHINE \System \CurrentControlSet \services \Tcpip \Parameters]  
TcpTimedWaitDelay = 120 (Default = 240 secs, Range = 30-300)
```

More Info: [MS KB Q137984](#)

More Info: [MS KB Q149532](#)

More Info: [MS KB Q832954](#)

MaxFreeTWTcb

Note: with Win2k the reuse of sockets have been changed, so when reaching the limit of more than 1000 connections in **TIME-WAIT** state, then it starts to mark sockets that have been in **TIME_WAIT** state for more than 60 secs as free.

It is possible to configure this limit:

```
[HKEY_LOCAL_MACHINE \System \CurrentControlSet \services \Tcpip \Parameters]  
MaxFreeTWTcbs = 1000 (Default = 1000 sockets)
```

Note: with **Win2k3 SP1** the reuse of sockets have been changed, so when it has to *re-use sockets* in **TIME_WAIT** state, then it checks whether the other party is different from the old socket.

Eliminating the need to fiddle with (***TcpTimedWaitDelay***) and (***MaxFreeTWTcbs***) any more.

KeepAliveTime

If using an application protocol that doesn't implement timeout checking, but relies on the **TCP/IP** timeout checking without specifying how often it should be done, then it is possible to get connections that "**never**" closes, if the remote host disconnects without closing the connection properly.

The **TCP/IP** timeout checking is by default done every **2 hour**, by sending a keep alive packet. It is possible to change how often **TCP/IP** should check the connections (***Affects all TCPIP connections***):

```
[HKEY_LOCAL_MACHINE \System \CurrentControlSet \services \Tcpip \Parameters]  
KeepAliveTime = 1800000 (Default = 7,200,000 milisecs)
```

More Info: [MS KB Q140325](#)

When data is *sent/received* the data is copied back and forth to **non-paged pool memory** for buffering. If there are many connections *receiving/sending* data, then it is possible to exhaust the non-paged pool memory.

The max size of the ***non-paged pool buffer*** allocated for each connection is controlled by **MaxBufferedReceiveBytes** or **TCP/IP Receive Window** depending on which is smallest.

More Info: [MS KB Q296265](#)

Note: if using the **Professional/Home** edition of **Windows** then it is very likely that it is crippled (By **Microsoft**) ***not to handle many concurrent TCP connections***.

Ex. **Microsoft** have officially stated that the backlog limit is **5 (200 when Server)**, so the **Professional** edition is not able to **accept()** more than **5 new connections concurrently**.

More Info: MS KB Q127144

Note: even if having optimized Windows to handle many concurrent connections, then connections might still be refused when reaching a certain limit, in case a **NAT-Router/Firewall** is placed in front of it, which is unable to handle so many concurrent connections.

EnableConnectionRateLimiting

Note: if having activated [SYN-Attack-Protection](#) (Enabled by default in **Win2k3 SP1**) or installed **WinXP SP2**, a limit is introduced on how many connection attempts (**half-open**) one can make simultaneously (**XP SP2 & Vista = 10; Vista SP2 = no limit**).

This will keep worms like blaster and sasser from spreading too fast, but it will also limit other applications that creates many new connections simultaneously (Like **P2P**).

EventID 4226: TCP/IP has reached the security limit imposed on the number of concurrent TCP connect attempts

More Info: www.LvlLord.de

Windows Vista SP2 removes the limit again, but it can be enabled with the following **DWORD** registry setting:

```
[HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \Tcpip \Parameters]
EnableConnectionRateLimiting = 1
```

More Info: MS KB 969710