

# IxChariot TCP HOWTO

## TCP Window Scaling

### 1. What is a TCP window?

TCP uses a 'sliding-window' mechanism to send data from one peer to another. Peer A is allowed to send one 'window' worth of data without having confirmation from Peer B that the data has been received correctly. Once Peer A starts receiving confirmations that data has been received at Peer B, Peer A 'slides' the window forward and sends more data. If the window is smaller than the network capacity, then the network will be underutilized. Most modern equipment can easily sustain data rates up to 1 Gbps in LAN environments with no more than 1-2ms of delay. When the bandwidth or delay is increased significantly, the default settings in most operating systems will prevent full utilization of the network.

### 2. What is the bandwidth-delay product, or BDP?

To determine the capacity of the network, you can use this simple calculation:

$$BDP = \text{bandwidth (in bytes)} \times \text{roundtrip delay (in seconds)}$$

For example, the BDP of a network with a 15 ms round-trip time (ping time) and an expected bandwidth of 1 Gbps is 1,875,000 bytes. To optimally utilize this network, the TCP window must be at least that large and it is recommended to start at twice the BDP. Remember that the round-trip time will increase as the pipe approaches full capacity so the BDP will increase as well.

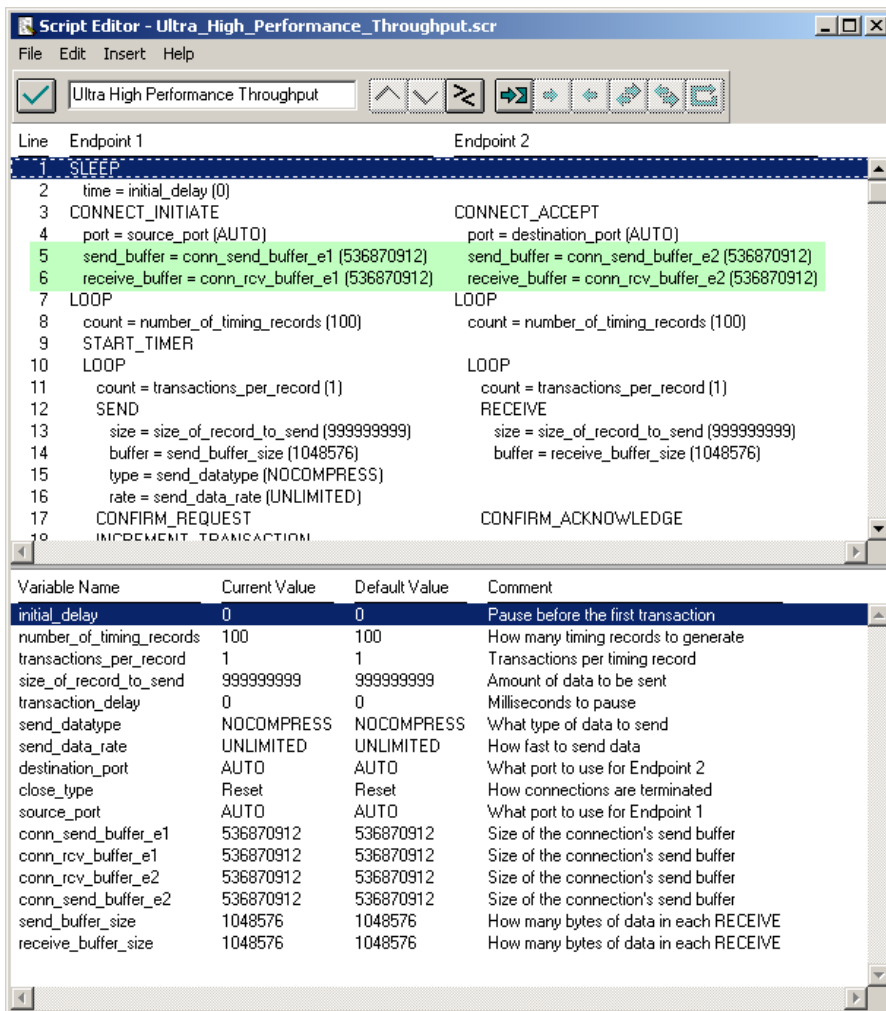
### 3. What is TCP window scaling?

TCP window scaling is an option specified in RFC 1323 that allows TCP connections to use window sizes larger than 64 Kilobytes. This is particularly important for high-bandwidth and/or high-delay networks. This URL has a good explanation of window scaling and how it applies to Windows-based TCP performance:

<http://rdweb.cns.vt.edu/public/notes/win2k-tcpip.htm>

### 4. How do I enable TCP window scaling in a Chariot script?

TCP window scaling in Chariot can set the socket buffer as high as the system maximum. See questions 4 and 5 for system defaults and maximums. If you set the socket buffers higher than the system max, then the system will allocate the system maximum to each pair.



5. How do I enable TCP window scaling on a Windows XP, 2000, or 2003 Server endpoint PC?

There are three registry keys that affect the TCP window scaling capability in a Windows-based machine:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Tcp1323Opts  
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\GlobalMaxTcpWindowSize  
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{xxx}\TcpWindowSize

*Tcp1323Opts* allows the user to enable features specified by RFC 1323 to support TCP window scaling, TCP timestamps and protection against wrapped sequence numbers (PAWS). Set this DWORD to 3 to enable all RFC 1323 features.

*GlobalMaxTcpWindowSize* sets the maximum window size that can be requested by an application. By default, this value does not exist and the system maximum is 64KB. Add this key under Parameters and set it to the maximum value of 1GB, 0x3fffffff.

*TcpWindowSize* controls the default per-socket window size on a system-wide or per-interface basis. If you add this key under Parameters, then every socket will default to this value. If it is added under a particular interface, then all connections on that interface will default to this value. If this value is not specified here, or in the script (per #4 above), then the system default of 17KB will be used.

6. How do I enable TCP window scaling on a Linux-based endpoint PC?

To set the system-wide maximum window size on a Linux PC, use the *wmem\_max* (TX) and *rmem\_max* (RX) values in the */proc* filesystem. For example, to set the maximum to 1MB enter these commands:

```
echo 1048576 > /proc/sys/net/core/wmem_max
echo 1048576 > /proc/sys/net/core/rmem_max
```

To set the system wide default values, change the tcp\_wmem and tcp\_rmem values. These values are in triplets that indicate the "minimum default maximum" values for each socket created.

```
echo "4096 524288 1048576" > /proc/sys/net/ipv4/tcp_wmem
echo "4096 524288 1048576" > /proc/sys/net/ipv4/tcp_rmem
```

## Controlling TCP Packet Sizes

### 7. How do I control packet size in TCP tests?

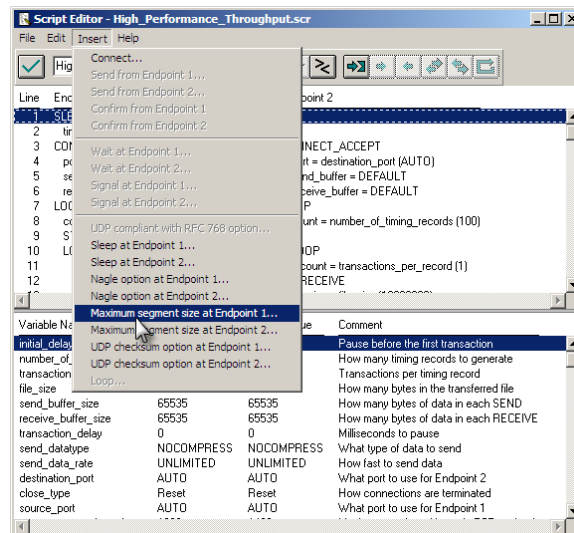
There are three ways to control packet size in TCP tests. First, on systems that support it, the MSS option (#9) is the best way to do this. The next best option is to force the MTU size as described in #10 and #11. Finally, the packet size can be controlled by setting the send\_buffer\_size to the desired payload size and disabling Nagle's algorithm for that pair. Nagle's algorithm is enabled by default on all systems to prevent small packets from flooding the network; by disabling Nagle you can force small packets to be sent. You may see erratic or low-performing behavior on some systems from using this method.

### 8. What is the TCP MSS?

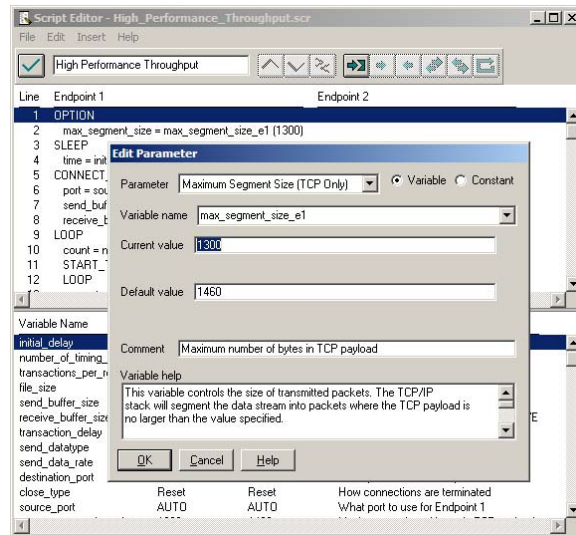
MSS stands for Maximum Segment Size. The default MSS is based on the MTU size of the network interface, i.e.  $MSS = MTU - 40$  (20 bytes IP header + 20 bytes TCP header). Some systems, like Linux, allow this to be adjusted at the socket level. On these systems, you can insert a 'Maximum Segment Size' option at the top of a Chariot script to control the size of frames

### 9. How do I control the MSS from a Chariot script?

The MSS can be controlled by the endpoint on a per-connection basis when running on Linux or IBM AIX platforms. To enable this option, first select Insert->Maximum segment size option at Endpoint X...



Next, set the MSS to your desired value:



#### 10. How do I change the packet size/MSS/MTU on a Windows system?

Microsoft Windows does not allow per-connection modification of the MSS; it can only be modified by changing the actual MTU size of the interface. This can be done by modifying the following registry key: HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{xxxx-xxx-xxx-xxx}\MTU. The best way to find the correct interface key is to look at the IP address field in each one.

#### 11. How do I change the MTU on Linux system?

On a Linux system, the MTU is configured using the 'ifconfig' command. It is best to disable the interface, set the MTU and then re-enable the interface.

```
i fconfig ethX down
i fconfig ethX mtu 1300
i fconfig ethX up
```