



## Managing Caching Performance and Differentiated Services

---

This chapter explains how to configure TCP stack parameters for increased performance and throughput and how to configure Type of Service (ToS) and differentiated services code point (DSCP) settings to mark content for different classes of service. It contains the following sections:

- [Configuring TCP Settings for Increased Cache Performance, page 10-1](#)
- [Configuring IP Differentiated Services, page 10-9](#)
- [Enabling the MTU Discovery Utility, page 10-10](#)

### Configuring TCP Settings for Increased Cache Performance

Although Content Engines are typically used to increase the throughput of HTTP streams over TCP end to end, adjusting TCP parameters for better performance is often overlooked when configuring content caching. For data transactions and queries between client and servers, the size of windows and buffers is important, and fine-tuning the TCP stack parameters therefore becomes the key to maximizing cache performance. The relevant TCP parameters to maximize cache performance and throughput include the ability to tune timeout periods, client and server receive and send buffer sizes, and TCP window scaling behavior.



#### Note

Because of the complexities involved in TCP parameters, care is advised in tuning these parameters. In nearly all environments, the default TCP settings are adequate. Fine tuning of TCP settings is for network administrators with adequate experience and full understanding of TCP operation details.

To configure TCP settings for the Content Engine, follow these steps:

- Step 1** Choose **Devices > Devices**. The Devices window appears, listing all the device types configured in the ACNS network.
- Step 2** Click the **Edit** icon next to the Content Engine for which you want to configure TCP settings. The Device Home for Content Engine window appears.
- Step 3** In the Contents pane, choose **General Settings > Network > TCP**. The TCP Settings for Content Engine window appears.

To configure TCP parameters, you need to configure three sets of settings in this window:

- TCP General Settings
- TCP Client Settings
- TCP Server Settings

See [Figure 10-1](#) and [Figure 10-2](#). [Table 10-1](#) describes the fields in this window and provides the corresponding CLI global configuration commands.

**Figure 10-1 TCP Settings Window—General Settings**

The screenshot displays the Cisco Application and Content Networking System web interface. The main content area is titled "TCP Settings for Content Engine, CONTENTENGINE". It shows the current applied settings from the Content Engine, CONTENTENGINE. The settings are organized into three sections:

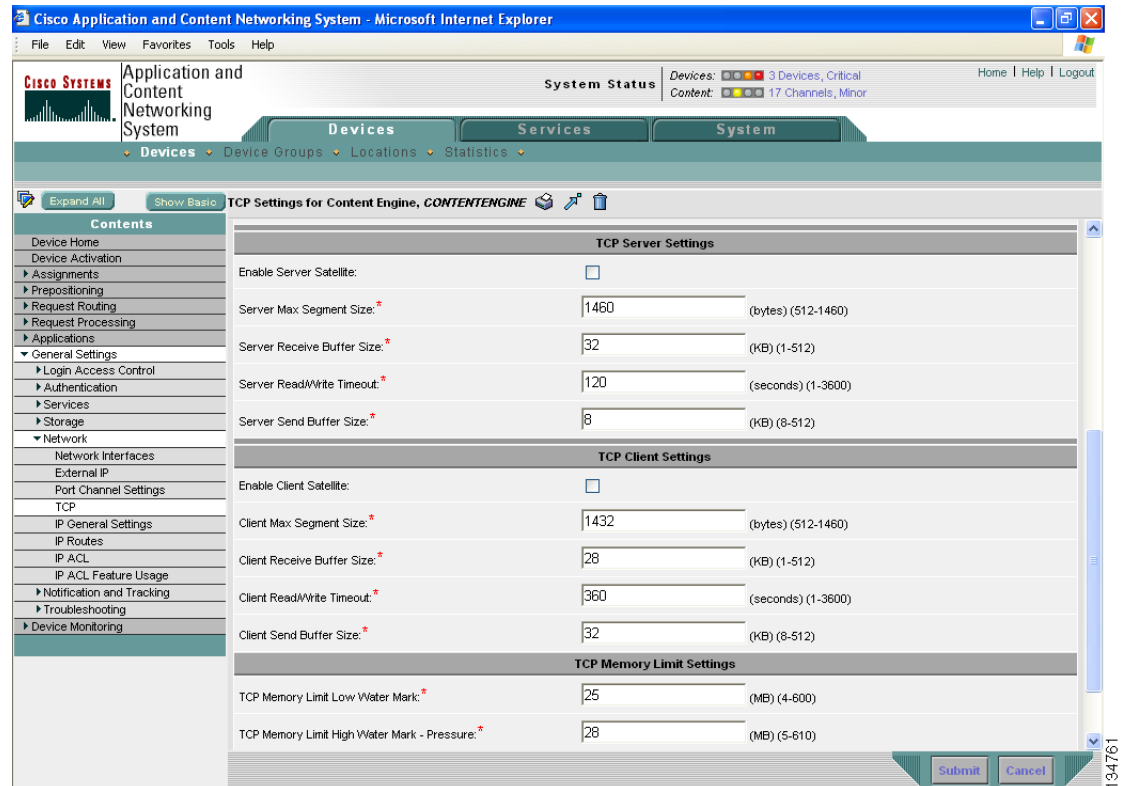
TCP General Settings	
Enable Explicit Congestion Notification:	<input checked="" type="checkbox"/>
Enable Type Of Service:	<input checked="" type="checkbox"/>
Congestion Window Size:	2 (segments) (1-10)
Retransmit Time Multiplier:	1 (1-3)
Initial Slow Start Threshold:	2 (2-10)
Keepalive Probe Count:	4 (1-10)
Keepalive Probe Interval:	75 (seconds) (1-120)
Keepalive Timeout:	90 (seconds) (1-120)

TCP Server Settings	
Enable Server Satellite:	<input type="checkbox"/>
Server Max Segment Size:	1460 (bytes) (512-1460)
Server Receive Buffer Size:	32 (KB) (1-512)

The interface also includes a "Contents" sidebar on the left with various navigation options like "Device Home", "Assignments", "Request Routing", "Applications", "General Settings", "Login Access Control", "Authentication", "Services", "Storage", "Network", "Network Interfaces", "External IP", "Port Channel Settings", "TCP", "IP General Settings", "IP Routes", "IP ACL", "IP ACL Feature Usage", "Notification and Tracking", "Troubleshooting", and "Device Monitoring". At the bottom right, there are "Submit" and "Cancel" buttons.

Figure 10-2 TCP Settings Window—Server and Client Settings



**Step 4** Under the TCP General Settings section, follow these steps:

- a. To enable reduction of delay and packet loss in data transmissions, check the **Enable Explicit Congestion Notification** check box. It provides TCP support for RFC 2581. (See the “[About Explicit Congestion Notification](#)” section on page 10-7.)
- b. To enable the TCP Type of Service, check the **Enable Type Of Service** check box. It is disabled by default.
- c. In the Congestion Window Size field, specify the initial congestion window value in segments. The default is 2 segments. (See the “[About Congestion Windows](#)” section on page 10-7.)
- d. In the ReTransmit Time Multiplier field, specify the factor used to modify the length of the retransmit timer by 1 to 3 times the base value determined by the TCP algorithm. The default is 1, which leaves the times unchanged. (See the “[About the Retransmit Time Multiplier](#)” section on page 10-7.)



**Note** Modify this factor with caution. It can improve throughput when TCP is used over slow reliable connections but should never be changed in an unreliable packet delivery environment.

- e. In the Initial Slow Start Threshold field, specify the threshold for slow start in segments. The default is 2 segments. (See the “[About TCP Slow Start](#)” section on page 10-8.)
- f. In the Keepalive Probe Count field, specify the number of times the Content Engine can retry a connection before the connection is declared unsuccessful. The default is 4 attempts.

- g. In the Keepalive Probe Interval field, specify the length of time that the Content Engine keeps an idle connection open. The default is 300 seconds.
- h. In the Keepalive Timeout field, specify the length of time that the Content Engine keeps a connection open before disconnecting. The default is 90 seconds.

**Step 5** Under the TCP Client Settings section, follow these steps:

- a. To set client TCP compliance to the RFC 1323 standard, check the **Enable Client Satellite** check box. (See the [“About TCP-Over-Satellite Extensions”](#) section on page 10-8.)




---

**Note** If you enable a client satellite, the server satellite is automatically enabled, and if you enable the server satellite, the client satellite is automatically enabled.

---

- b. In the Client Max Segment Size field, specify the maximum packet size sent to clients. The default is 1432 bytes.
- c. In the Client Receive Buffer Size field, specify the TCP receiving buffer size (in kilobytes) for incoming TCP packets. The default is 32 KB.
- d. In the Client Read/Write Timeout field, specify the period after which the Content Engine times out trying to read or write to the network. The default is 120 seconds.
- e. In the Client Send Buffer Size field, specify the TCP sending buffer size (in kilobytes) for outgoing TCP packets. The default is 32 KB.

**Step 6** Under the TCP Server Settings section, follow these steps:

- a. To set server TCP server compliance to the RFC 1323 standard, check the **Enable Server Satellite** check box. (See the [“About TCP-Over-Satellite Extensions”](#) section on page 10-8.)




---

**Note** If you enable a server satellite, the client satellite is automatically enabled, and the reverse.

---

- b. In the Server Max Segment Size field, specify the maximum packet size sent to the server. The default is 1460 bytes.
- c. In the Server Receive Buffer Size field, specify the TCP receiving buffer size (in kilobytes) for incoming TCP packets. The default is 32 KB.
- d. In the Server Read/Write Timeout field, specify the period after which the Content Engine times out trying to read or write to the network. The default is 120 seconds.
- e. In the Server Send Buffer Size field, specify the TCP sending buffers size (in kilobytes) for outgoing TCP packets. The default is 32 KB.

**Step 7** To save the settings, click **Submit**. A “Click Submit to Save” message appears in red next to the Current Settings line when there are pending changes to be saved after you have applied default or device group settings. You can also revert to the previously configured settings by clicking **Reset**. The **Reset** button is visible only when you have applied default or group settings to change the current device settings but have not yet submitted the changes.

If you try to leave this window without saving the modified settings, a warning dialog box prompts you to submit the changes. This dialog box only appears if you are using the Internet Explorer browser.

**Table 10-1 TCP Settings**

GUI Parameter	Function	CLI Command
<b>TCP General Settings</b>		
Enable Explicit Congestion Notification	Enables reduction of delay and packet loss.	<b>tcp ecn enable</b>
Enable Type Of Service	Enables Type of Service.	<b>tcp type-of-service enable</b>
Congestion Window Size	Congestion window size in segments.	<b>tcp cwnd-base</b> <i>segments</i>
ReTransmit Time Multiplier	Factor used to modify the length of the retransmit timer.	<b>tcp increase-xmit-timer-value</b> <i>value</i>
Initial Slow Start Threshold	Threshold for slow start in segments.	<b>tcp init-ss-threshold</b> <i>value</i>
Keepalive Probe Count	Number of times the Content Engine can retry a connection before it is considered unsuccessful.	<b>tcp keepalive-probe-cnt</b> <i>count</i>
Keepalive Probe Interval	Length of time that the Content Engine keeps an idle connection open.	<b>tcp keepalive-probe-interval</b> <i>seconds</i>
Keepalive Timeout	Length of time that the Content Engine keeps a connection open before disconnecting.	<b>tcp keepalive-timeout</b> <i>seconds</i>
<b>TCP Client Settings</b>		
Enable Client Satellite	Sets client TCP compliance to the RFC 1323 standard.	<b>tcp client-satellite</b>
Client Max Segment Size	Maximum packet size sent to clients.	<b>tcp client-mss</b> <i>max_seg_size</i>
Client Receive Buffer Size	TCP receiving buffer size in kilobytes (1-512) for incoming TCP packets.	<b>tcp client-receive-buffer</b> <i>kbytes</i>
Client Read/Write Timeout	Period after which the Content Engine times out when attempting to read or write to the network.	<b>tcp client-rw-timeout</b> <i>seconds</i>
Client Send Buffer Size	TCP sending buffers size in kilobytes (1-512) for outgoing TCP packets.	<b>tcp client-send-buffer</b> <i>kbytes</i>
<b>TCP Server Settings</b>		
Enable Server Satellite	Sets TCP server compliance to the RFC 1323 standard.	<b>tcp server-satellite</b>
Server Max Segment Size	Maximum packet size sent to the server.	<b>tcp server-mss</b> <i>max_seg_size</i>
Server Receive Buffer Size	TCP receiving buffer size in kilobytes (1-512) for incoming TCP packets.	<b>tcp server-receive-buffer</b> <i>kbytes</i>

Table 10-1 TCP Settings (continued)

GUI Parameter	Function	CLI Command
Server Read/Write Timeout	Period after which the Content Engine times out when attempting to read or write to the network.	<b>tcp server-rw-timeout</b> <i>seconds</i>
Server Send Buffer	TCP sending buffer size in kilobytes (1-512) for outgoing TCP packets.	<b>tcp server-send-buffer</b> <i>kbytes</i>

## Configuring TCP Memory Limits

The TCP memory limit settings allow you to control the amount of memory that can be used by the TCP subsystem send and receive buffers.



### Caution

Do not modify the default values unless you know what you are doing. The default values are device dependent and have been chosen after extensive testing. They should not be changed under normal conditions. Increasing these values can result in the TCP subsystem using more memory, which might render the system unresponsive. Decreasing these values can result in increased response times and lower performance.

To configure the TCP memory limit settings from the Content Distribution Manager GUI, choose **Devices > Devices** (or **Device Groups**) > **General Settings** > **Network** > **TCP**. The TCP Settings window appears. (See [Figure 10-2](#).)

[Table 10-2](#) lists the TCP memory limit settings in the Content Distribution Manager GUI and the corresponding CLI commands.

Table 10-2 TCP Memory Limit Settings

Content Distribution Manager GUI Parameter	Function	CLI Command
TCP Limit Low Water Mark	The lower limit (in MBytes) of memory pressure mode, below which TCP enters into normal memory allocation mode. The range is 4–600.	<b>tcp memory-limit low-water-mark</b> <i>megabytes</i>
TCP Memory Limit High Water Mark–Pressure	The upper limit (in megabytes) of normal memory allocation mode, beyond which TCP enters into memory pressure mode. The range is 5–610.	<b>high-water-mark-pressure</b> <i>megabytes</i>
TCP Memory Limit High Water Mark–Absolute	The absolute limit (in MBytes) on TCP memory usage. The range is 6–620.	<b>high-water-mark-absolute</b> <i>megabytes</i>

Table 10-3 describes the default values for each command parameter, which are based on the total amount of memory for the device.

**Table 10-3** Default TCP Memory Limit Settings

Total System Memory	Low	Pressure	Absolute
1 GB, 2 GB, or 4 GB	360 MB	380 MB	400 MB
512 MB	180 MB	190 MB	200 MB
256 MB	25 MB	28 MB	30 MB

The following conditions must be satisfied whenever these default values are changed:

- The low water mark must be a number that is less than the high water mark pressure setting.
- The high water mark pressure must be a number that is less than the high water mark absolute setting:

```
low-water-mark < high-water-mark-pressure < high-water-mark-absolute
```

## About Explicit Congestion Notification

The TCP Explicit Congestion Notification (ECN) feature allows an intermediate router to notify the end hosts of impending network congestion. It also provides enhanced support for TCP sessions associated with applications that are sensitive to delay or packet loss, including Telnet, web browsing, and transfer of audio and video data. The major issue with ECN is the need to change the operation of both the routers and the TCP software stacks to accommodate the operation of ECN.

## About Congestion Windows

The congestion window (*cwnd*) is a TCP state variable that limits the amount of data that a TCP sender can transmit onto the network before receiving an acknowledgment (ACK) from the receiving side of the TCP transmission. The TCP *cwnd* variable is implemented by the TCP congestion avoidance algorithm. The goal of the congestion avoidance algorithm is to continually modify the sending rate so that the sender automatically senses any increase or decrease in available network capacity during the entire data flow. When congestion occurs (manifested as packet loss), the sending rate is first lowered and then gradually increased as the sender continues to probe the network for additional capacity.

## About the Retransmit Time Multiplier

The TCP sender uses a timer to measure the time that has elapsed between sending a data segment and receiving the corresponding ACK from the receiving side of the TCP transmission. When this retransmit timer expires, the sender (according to the RFC standards for TCP congestion control) must reduce its sending rate. However, because the sender is not reducing its sending rate in response to network congestion, the sender is not able to make any valid assumptions about the current state of the network. Therefore, in order to avoid congesting the network with an inappropriately large burst of data, the sender implements the slow start algorithm, which reduces the sending rate to one segment per transmission. (See the next section, “[About TCP Slow Start](#).”)

You can modify the sender's retransmit timer by using the Retransmit Time Multiplier field in the Content Distribution Manager GUI or the **tcp increase-xmit-timer-value** global configuration command in the CLI. The retransmit time multiplier modifies the length of the retransmit timer by one to three times the base value, as determined by the TCP algorithm that is being used for congestion control.

When making adjustments to the retransmit timer, be aware that they affect performance and efficiency. If the retransmit timer is triggered too early, the sender pushes duplicate data onto the network unnecessarily; if the timer is triggered too slowly, the sender remains idle for too long, unnecessarily slowing data flow.

## About TCP Slow Start

Slow start is one of four congestion control algorithms used by TCP. The slow start algorithm controls the amount of data being inserted into the network at the beginning of a TCP session, when the capacity of the network is not known.

For example, if a TCP session began by inserting a large amount of data into the network, much of the initial burst of data would probably be lost. Instead, TCP initially transmits a modest amount of data that has a high probability of successful transmission. TCP then probes the network by sending increasing amounts of data as long as the network does not show signs of congestion.

The slow start algorithm begins by sending packets at a rate that is determined by the congestion window, or *cwnd* variable. (See “[About Congestion Windows](#).”) The algorithm continues to increase the sending rate until it reaches the limit set by the slow start threshold (*ssthresh*) variable. (Initially, the value of the *ssthresh* variable is adjusted to the receiver's maximum window size [RMSS]. However, when congestion occurs, the *ssthresh* variable is set to half the current value of the *cwnd* variable, marking the point of the onset of network congestion for future reference.)

The starting value of the *cwnd* variable is set to that of the sender maximum segment size (SMSS), which is the size of the largest segment that the sender can transmit. The sender sends a single data segment, and because the congestion window is equal to the size of one segment, the congestion window is now full. The sender then waits for the corresponding ACK from the receiving side of the transmission. When the ACK is received, the sender increases its congestion window size by increasing the value of the *cwnd* variable by the value of one SMSS. Now the sender can transmit two segments before the congestion window is again full and the sender is once more required to wait for the corresponding ACKs for these segments. The slow start algorithm continues to increase the value of the *cwnd* variable and therefore increase the size of the congestion window by one SMSS for every ACK received. If the value of the *cwnd* variable increases beyond the value of the *ssthresh* variable, then the TCP flow control algorithm changes from the slow start algorithm to the congestion avoidance algorithm.

## About TCP-Over-Satellite Extensions

The Content Engine has the ability to turn on TCP-over-satellite extensions (as documented in RFC 1323) to maximize performance and end-to-end throughput over satellite-type connections.

The large number of satellites available to network infrastructures has increased the amount of bandwidth available in the air. Taking advantage of these connections through satellite-type connections has created new challenges in the use of TCP transactions and acknowledgments:

- Latency—Round trip times to satellites orbiting 24,000 miles above the earth are 550 milliseconds for a single satellite hop. Buffer size must be set to prevent low-throughput connections.
- Bit errors—Packet loss can occur in a land-based device-to-satellite connection in addition to the losses caused by regular network congestion.

- Asymmetric bandwidth—Return bandwidth from satellites can be narrower than receiving bandwidth, thereby affecting performance.

Use the fields provided under the TCP Server Settings and TCP Client Settings sections to set the TCP connection so that it complies with RFC 1323.

## Configuring IP Differentiated Services

In the ACNS network, you can configure Content Engines, Content Routers, and Content Distribution Managers for Type of Service (ToS) or differentiated services code point (DSCP) either through the CLI or through the Content Distribution Manager GUI.

The differentiated services architecture is based on a simple model where traffic entering a network is classified and possibly conditioned at the boundaries of the network. The class of traffic is then identified with a differentiated services (DS) code point or bit marking in the IP header. Within the core of the network, packets are forwarded according to the per-hop behavior associated with the DS code point.

To set the global ToS or DSCP values for the IP header from the CLI, use the **ip dscp** global configuration command:

```
ip dscp {client {cache-hit {match-server | set-dscp dscp-packets | set-tos tos-packets} | cache-miss {match-server | set-dscp dscp-packets | set-tos tos-packets}} | server {match-client | set-dscp dscp-packets | set-tos tos-packets}}
```

To configure TOS and DSCP values using the Content Distribution Manager GUI, follow these steps:

- 
- Step 1** From the Content Distribution Manager GUI, choose **Devices > Devices** (or **Devices > Device Groups**).
  - Step 2** Click the **Edit** icon next to the name of the device or device group that you want to configure.
  - Step 3** In the Contents pane, choose **General Settings > Network > IP General Settings**. The IP General Settings window appears.
  - Step 4** Under the IP General Settings heading, you can choose to enable the MTU discovery utility by checking the **Enable Path MTU Discovery** check box. (For more information, see the next section, “[Enabling the MTU Discovery Utility](#).”)
  - Step 5** To configure DSCP or TOS responses to the client for a cache hit, check the **Set IP DSCP Client Cache-Hit** check box to enable the options in the GUI, and choose one of the following methods for marking the IP packets:
    - **Match Server**
    - **DSCP**
    - **TOS**
  - Step 6** To configure DSCP or TOS responses to the client for a cache miss, check the **Set IP DSCP Client Cache-Miss** check box, and choose a method for marking the IP packets.
  - Step 7** To configure DSCP or TOS for outgoing requests to the server, check the **Set IP DSCP Server** check box, and choose a method for marking the IP packets.
  - Step 8** To save the settings, click **Submit**.
-

## Enabling the MTU Discovery Utility

Cisco ACNS software supports the IP Path Maximum Transmission Unit (MTU) Discovery mechanism, as defined in RFC 1191. When enabled, the Path MTU Discovery utility discovers the largest IP packet size allowable between the various links along the forwarding path and automatically sets the correct value for the packet size. By using the largest MTU that the links will bear, the sending device can minimize the number of packets it must send.

IP Path MTU Discovery is useful when a link in a network goes down, forcing the use of another, different MTU-sized link. IP Path MTU Discovery is also useful when a connection is first being established and the sender has no information at all about the intervening links.

**Note**

---

IP Path MTU Discovery is a process initiated by the sending device. If a server does not support IP Path MTU Discovery, the receiving device will have no mechanism available to avoid fragmenting datagrams generated by the server.

---

To enable this autodiscovery utility for the Content Engine or Content Router from the CLI, use the **ip path-mtu-discovery enable** global configuration command. To enable it using the Content Distribution Manager GUI, check the **Enable Path MTU Discovery** check box in the IP General Settings window for the device (**Devices > Devices > General Settings > Network > IP General Settings**). By default, this feature is disabled. With the feature disabled, the sending device uses a packet size that is the lesser of 576 bytes and the next hop MTU. Existing connections are not affected when this feature is turned on or off.