QoS: Header Compression Configuration Guide, Cisco IOS Release 12.4

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Header Compression

Header compression is a mechanism that compresses the header in a packet before the packet is transmitted. For Cisco IOS XE Software, Cisco provides RTP header compression (used for RTP packets).

This module contains a high-level overview of header compression. Before configuring header compression, you should understand the information in this module.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About Header Compression

Header Compression Defined

Header compression is a mechanism that compresses the header in a data packet before the packet is transmitted. Header compression reduces network overhead and speeds up the transmission of packets. Header compression also reduces the amount of bandwidth consumed when the packets are transmitted.

RTP Functionality and Header Compression

Real-Time Transport Protocol (RTP) provides end-to-end network transport functions for applications that support audio, video, or simulation data over unicast or multicast services.

RTP provides support for real-time conferencing of groups of any size within the Internet. This support includes source identification support for gateways such as audio and video bridges, and support for multicast-to-unicast translators. RTP provides QoS feedback from receivers to the multicast group and support for the synchronization of different media streams.

RTP includes a data portion and a header portion. The data portion of RTP is a thin protocol that provides support for the real-time properties of applications, such as continuous media, including timing reconstruction, loss detection, and content identification. The header portion of RTP is considerably larger than the data portion. The header portion consists of the IP segment, the User Datagram Protocol (UDP) segment, and the RTP segment. Given the size of the IP/UDP/RTP segment combinations, it is inefficient to send the IP/UDP/RTP header without compressing it.

To avoid the unnecessary consumption of available bandwidth, RTP header compression is used on a link-by-link basis.

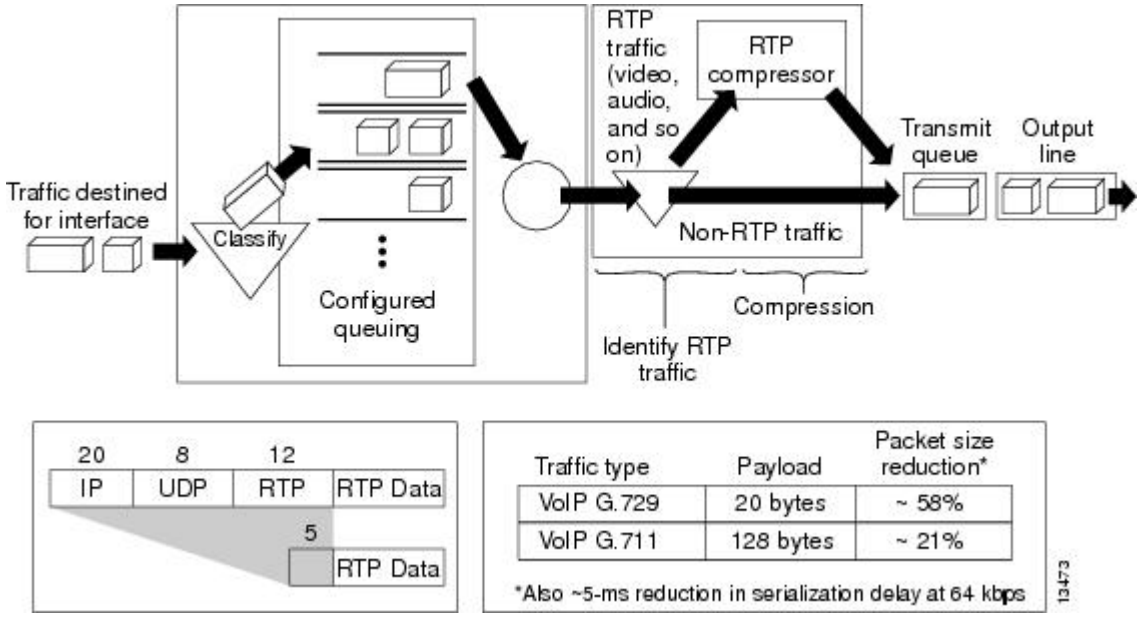
- How RTP Header Compression Works, page 2
- Why Use RTP Header Compression, page 3

How RTP Header Compression Works

RTP header compression compresses the RTP header (that is, the combined IP, UDP, and RTP segments) in an RTP packet. The figure below illustrates this process and shows how RTP header compression treats incoming packets.

In this example, packets arrive at an interface and the packets are classified. After the packets are classified, they are queued for transmission according to the configured queuing mechanism.

Figure 1 RTP Header Compression



For most audio applications, the RTP packet typically has a 20- to 128-byte payload.

RTP header compression identifies the RTP traffic and then compresses the IP header portion of the RTP packet. The IP header portion consists of an IP segment, a UDP segment, and an RTP segment. In the figure above, the minimal 20 bytes of the IP segment, combined with the 8 bytes of the UDP segment, and

the 12 bytes of the RTP segment, create a 40-byte IP/UDP/RTP header. In the figure above, the RTP header portion is compressed from 40 bytes to approximately 5 bytes.

Note RTP header compression is supported on serial interfaces using Frame Relay, HDLC, or PPP encapsulation. It is also supported over ISDN interfaces.

Why Use RTP Header Compression

RTP header compression accrues major gains in terms of packet compression because although several fields in the header change in every packet, the difference from packet to packet is often constant, and therefore the second-order difference is zero. The decompressor can reconstruct the original header without any loss of information.

RTP header compression also reduces overhead for multimedia RTP traffic. The reduction in overhead for multimedia RTP traffic results in a corresponding reduction in delay; RTP header compression is especially beneficial when the RTP payload size is small, for example, for compressed audio payloads of 20 to 50 bytes.

Use RTP header compression on any WAN interface where you are concerned about bandwidth and where there is a high portion of RTP traffic. RTP header compression can be used for media-on-demand and interactive services such as Internet telephony. RTP header compression provides support for real-time conferencing of groups of any size within the Internet. This support includes source identification support for gateways such as audio and video bridges, and support for multicast-to-unicast translators. RTP header compression can benefit both telephony voice and multicast backbone (MBONE) applications running over slow links.

Note Using RTP header compression on any high-speed interfaces--that is, anything over T1 speed--is not recommended. Any bandwidth savings achieved with RTP header compression may be offset by an increase in CPU utilization on the router.

Additional References

The following sections provide references related to header compression.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Quality of Service Solutions Command Reference*
MQC	"Applying QoS Features Using the MQC" module
RTP header compression	"Configuring RTP Header Compression" module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco IOS XE Software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1144	*Compressing TCP/IP Headers for Low-Speed Serial Links*
RFC 2507	*IP Header Compression*
RFC 2508	*Compressing IP/UDP/RTP Headers for Low-Speed Serial Links*
RFC 3544	*IP Header Compression over PPP*
RFC 3550	*A Transport Protocol for Real-Time Applications*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Glossary

compression --The running of a data set through an algorithm that reduces the space required to store the data set or the bandwidth required to transmit the data set.

decompression --The act of reconstructing a compressed header.

HDLC --High-Level Data Link Control. A bit-oriented synchronous data link layer protocol developed by International Organization for Standardization (ISO). Derived from Synchronous Data Link Control (SDLC), HDLC specifies a data encapsulation method on synchronous serial links using frame characters and checksums.

header --A chain of subheaders.

incorrect decompression --The circumstance in which a compressed and then decompressed header is different from the uncompressed header. This variance is usually due to a mismatched context between the compressor and decompressor or bit errors during transmission of the compressed header.

ISDN --Integrated Services Digital Network. A communication protocol offered by telephone companies that permits telephone networks to carry data, voice, and other source traffic.

MQC --Modular Quality of Service Command-Line Interface. The MQC allows you to create traffic classes and policy maps and then attach the policy maps to interfaces. The policy maps apply QoS features to your network.

PPP --Point-to-Point Protocol. A protocol that provides router-to-router and host-to-network connections over synchronous and asynchronous circuits.

regular header --A normal, uncompressed header. A regular header does not carry a context identifier (CID) or generation association.

RTP --Real-Time Transport Protocol. A protocol that is designed to provide end-to-end network transport functions for applications that transmit real-time data, such as audio, video, or simulation data, over unicast or multicast network services. RTP provides such services as payload type identification, sequence numbering, timestamping, and delivery monitoring to real-time applications.

subheader --An IPv6 base header, an IPv6 extension header, an IPv4 header, a UDP header, an RTP header, or a TCP header.

UDP --User Datagram Protocol. A connectionless transport layer protocol in the TCP/IP protocol stack. UDP is a simple protocol that exchanges datagrams without acknowledgments or guaranteed delivery, requiring that error processing and retransmission be handled by other protocols. UDP is defined in RFC 768.

Configuring RTP Header Compression

Header compression is a mechanism that compresses the IP header in a packet before the packet is transmitted. Header compression reduces network overhead and speeds up the transmission of either Real-Time Transport Protocol (RTP) or Transmission Control Protocol (TCP) packets.

Cisco provides two types of header compression: RTP header compression and TCP header compression. This module describes the concepts and tasks related to configuring RTP header compression.

Note RTP header compression is configured on a per-interface (or subinterface) basis. If you want to configure RTP header compression on a per-class basis, see the "Configuring Class-Based RTP and TCP Header Compression" module.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Configuring RTP Header Compression

- Before configuring RTP header compression, read the information in the "Header Compression" module.
- You must configure RTP header compression on both ends of the network.

Information About Configuring RTP Header Compression

Configurable RTP Header-Compression Settings

With RTP header compression, you can configure the maximum size of the compressed IP header, the maximum time between transmitting full-header packets, and the maximum number of compressed packets between full headers. These settings are configured using the following three commands:

- **ip header-compression max-header**
- **ip header-compression max-time**
- **ip header-compression max-period**

The **ip header-compression max-header** command allows you to define the maximum size of the IP header of a packet to be compressed. Any packet with an IP header that exceeds the maximum size is sent uncompressed.

The **ip header-compression max-time**command allows you tospecify the maximum time between transmitting full-header packets, and the **ip header-compression max-period** command allows you to specify the maximum number of compressed packets between full headers. With the **ip header-compression max-time** and **ip header-compression max-period** commands, the full-header packet is transmitted at the specified time period or when the maximum number of packets is reached, respectively. The counters for both the time period and the number of packets sent are reset after the full-header packet is sent.

For more information about these commands, see the Cisco IOS Quality of Service Solutions Command Reference.

RTP Header-Compression Keywords

When you configure RTP header compression, you can specify the circumstances under which the RTP packets are compressed and the format that is used when the packets are compressed. These circumstances and formats are defined by the following keywords:

- **passive**
- **iphc-format**
- **ietf-format**

These keywords (described below) are available with many of the quality of service (QoS) commands used to configure RTP header compression, such as the **ip rtp header-compression** command. For more information about the **ip rtp header-compression** command, these keywords, and the other QoS commands, see the Cisco IOS Quality of Service Solutions Command Reference.

The passive Keyword

By default, the **ip rtp header-compression** command compresses outgoing RTP traffic. If you specify the **passive** keyword, outgoing RTP traffic is compressed only if *incoming* RTP traffic on the *same* interface is compressed. If you do not specify the **passive** keyword, *all* outgoing RTP traffic is compressed.

The **passive** keyword is ignored on PPP interfaces.

The iphc-format Keyword

The **iphc-format**keyword indicates that the IP Header Compression (IPHC) format of header compression will be used. For PPP and HDLC interfaces, when the **iphc-format** keyword is specified, TCP header compression is also enabled. Since both RTP and TCP header compression are enabled, both UDP and TCP packets are compressed.

The **iphc-format** keyword includes checking whether the destination port number is even and is in the ranges of 16,385 to 32,767 (for Cisco audio) or 49,152 to 65,535 (for Cisco video). Valid RTP packets that meet the criteria (that is, the port number is even and is within the specified range) are compressed using the compressed RTP packet format. Otherwise, packets are compressed using the less-efficient compressed non-TCP packet format.

The **iphc-format**keyword is not available for interfaces that use Frame Relay encapsulation.

Note The header compression format (in this case, IPHC) must be the same at *both* ends of the network. That is, if you specify the **iphc-format** keyword on the local router, you must also specify the **iphc-format** keyword on the remote router.

The ietf-format Keyword

The **ietf-format** keyword indicates that the Internet Engineering Task Force (IETF) format of header compression will be used. For HDLC interfaces, the **ietf-format** keyword compresses only UDP packets. For PPP interfaces, when the **ietf-format**keyword is specified, TCP header compression is also enabled. Since both RTP header compression and TCP header compression are enabled, both UDP packets and TCP packets are compressed.

With the **ietf-format** keyword, any even destination port number higher than 1024 can be used. Valid RTP packets that meet the criteria (that is, the port number is even and is higher than 1024) are compressed using the compressed RTP packet format. Otherwise, packets are compressed using the less-efficient compressed non-TCP packet format.

The **ietf-format** keyword is not available for interfaces that use Frame Relay encapsulation.

Note The header compression format (in this case, IETF) must be the same at *both* ends of the network. That is, if you specify the **ietf-format**keyword on the local router, you must also specify the **ietf-format** keyword on the remote router.

Enhanced RTP Header Compression

The Cisco IOS Release 12.3(11)T introduced a feature that enhances the functionality of RTP header compression. This feature is called Enhanced CRTP for Links with High Delay, Packet Loss, and Reordering (ECRTP).

The ECRTP feature is also known as Enhanced RTP Header Compression. It includes modifications and enhancements to RTP header compression to achieve robust operation over unreliable point-to-point links. This is accomplished by repeating updates and sending absolute (uncompressed) values in addition to delta values for selected context parameters.

During compression of an RTP stream, a session context is defined. For each context, the session state is established and shared between the compressor and the decompressor. The context state consists of the full IP/UDP/RTP headers, a few first-order differential values, a link sequence number, a generation number, and a delta encoding table. Once the context state is established, compressed packets may be sent.

RTP header compression was designed for reliable point-to-point links with short delays. It does not perform well over links with a high rate of packet loss, packet reordering, and long delays. Packet loss results in context corruption, and because of long delay, packets are discarded before the context is repaired. To correct the behavior of RTP header compression over such links, several enhancements have been made to the RTP header compression functionality. The enhancements reduce context corruption by changing the way that the compressor updates the context at the decompressor; updates are repeated and include additions to full and differential context parameters.

With these enhancements, RTP header compression performs well over links with packet loss, packet reordering, and long delays.

RTP Header Compression over Satellite Links

The Cisco IOS Release 12.3(2)T introduced a feature called RTP Header Compression over Satellite Links. The RTP Header Compression over Satellite Links feature allows you to use RTP header compression over an asymmetric link (such as a satellite link), where the uplink and downlink connections are on separate interfaces. This feature provides improved system performance by reducing network overhead and speeding up transmission of RTP packets.

Periodic Refreshes of a Compressed Packet Stream

RTP header compression is a mechanism that compresses the IP header in a packet before the packet is transmitted. RTP header compression requires a context status feedback mechanism to recover when the compressed packet stream experiences packet channel loss. If the round-trip time of the packet between the uplink and the downlink is lengthy or if a feedback path does not exist, the chance of loss propagation is greatly increased when a packet is dropped from the link. For instance, if a feedback path does not exist, a compressed packet stream may never recover. This situation presents a need for a configurable option that allows periodic refreshes of the compressed packet stream using full-header packets.

The periodic-refresh Keyword

When you configure header compression, you can configure periodic refreshes of the compressed packet stream using the **periodic-refresh** keyword. The **periodic-refresh** keyword is available with the following commands:

- **ip rtp header-compression**
- **frame-relay ip rtp header-compression**
- **frame-relay map ip rtp header-compression**

For more information about these commands, see the Cisco IOS Quality of Service Solutions Command Reference.

Optional Disabling of Context-Status Messages

During header compression, a session context is defined. For each context, the session state is established and shared between the compressor and the decompressor. The context state consists of the full IP/UDP/RTP headers, a few first-order differential values, a link sequence number, a generation number, and a delta encoding table. This information is included in the context-status messages.

You can disable the sending of context-status messages in instances either when the time it takes for the packet to traverse the uplink and the downlink portions of the data path is greater than the refresh period (in which case, the sending of the context-status message would not be useful) or when a feedback path does not exist.

Disabling the context-status messages can be accomplished by using the **ip header-compression disable-feedback** command. For more information about this command, see the Cisco IOS Quality of Service Solutions Command Reference.

How to Configure RTP Header Compression

Enabling RTP Header Compression on an Interface

To enable RTP header compression on an interface, perform the following steps.

Note To enable RTP header compression on an interface that uses Frame Relay encapsulation, skip these steps and complete the steps in the Enabling RTP Header Compression on an Interface That Uses Frame Relay Encapsulation, page 13 instead.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **encapsulation** *encapsulation-type*
5. **ip address** *ip-address mask* [**secondary**]
6. **ip rtp header-compression** [**passive** | **iphc-format** | **ietf-format**] [**periodic-refresh**]
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**encapsulation** *encapsulation-type* **Example:** `Router(config-if)# encapsulation ppp`	Sets the encapsulation method used by the interface. • Enter the encapsulation method.
Step 5	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 209.165.200.225 255.255.255.224`	Sets a primary or secondary IP address for an interface. • Enter the IP address and mask for the associated IP subnet.
Step 6	**ip rtp header-compression** [**passive** | **iphc-format** | **ietf-format**] [**periodic-refresh**] **Example:** `Router(config-if)# ip rtp header-compression`	Enables RTP header compression.
Step 7	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Enabling RTP Header Compression on an Interface That Uses Frame Relay Encapsulation

To enable RTP header compression on an interface that uses Frame Relay encapsulation, perform the following steps.

Note The encapsulation type is specified by using either the **cisco** or **ietf** keyword of the **frame-relayinterface-dlci** command. The **cisco** keyword specifies Cisco proprietary encapsulations, and the **ietf** keyword specifies IETF encapsulations. However, note the following points about these keywords:

- Frame Relay interfaces do not support IETF encapsulations when RTP header compression is enabled. Therefore, the **ietf**keyword is not available for Frame Relay interfaces and is not listed in the command syntax shown below.
- The **cisco** keyword is available for use on point-to-point subinterfaces *only* .

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **encapsulation frame-relay**
5. **ip address** *ip-address mask* [**secondary**]
6. **frame-relay interface-dlci** *dlci* [**cisco**]
7. **frame-relay ip rtp header-compression** [**active** | **passive**][**periodic-refresh**]
8.
9. **frame-relay map ip** *ip-address dlci* [**broadcast**] **rtpheader-compression** [**active** | **passive**] [**periodic-refresh**] [**connections***number*]
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**encapsulation frame-relay** **Example:** `Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation.
Step 5	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 209.165.200.225 255.255.255.224`	Sets a primary or secondary IP address for an interface. • Enter the IP address and mask for the associated IP subnet.
Step 6	**frame-relay interface-dlci** *dlci* [**cisco**] **Example:** `Router(config-if)# frame-relay interface-dlci 20`	Assigns a data-link connection identifier (DLCI) to a specified Frame Relay interface on the router.
Step 7	**frame-relay ip rtp header-compression** [**active** | **passive**] [**periodic-refresh**] **Example:** `Router(config-if)# frame-relay ip rtp header-compression`	Enables RTP header compression for all Frame Relay maps on a physical interface.
Step 8		

	Command or Action	Purpose
Step 9	**frame-relay map ip** *ip-address dlci* [**broadcast**] **rtpheader-compression** [**active** | **passive**] [**periodic-refresh**] [**connections***number*] **Example:** `Router(config-if)# frame-relay map ip 10.108.175.220 180 rtp header-compression periodic-refresh`	Assigns to an IP map header-compression characteristics that differ from the compression characteristics of the interface with which the IP map is associated. • Enter the IP address, DLCI number, and any optional keywords and arguments.
Step 10	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Enabling Enhanced RTP Header Compression

The Enhanced RTP Header Compression feature (also known as ECRTP) includes modifications and enhancements to RTP header compression to achieve robust operation over unreliable point-to-point links. Enhanced RTP header compression is intended for use on networks subject to high rates of packet loss, packet reordering, and long delays. For more information about Enhanced RTP header compression, see the Enhanced RTP Header Compression, page 9.

To enable enhanced RTP header compression, perform the following steps.

- Configure a serial link using HDLC encapsulation or configure an interface using PPP encapsulation.
- Ensure that RTP header compression is enabled on the interface. See the Enabling RTP Header Compression on an Interface, page 11.

Note Enhanced RTP header compression is not supported on interfaces that use Frame Relay encapsulation.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **encapsulation** *encapsulation-type*
5. **ip address** *ip-address mask* [**secondary**]
6. **ip rtp header-compression** [**passive** | **iphc-format** | **ietf-format**] [**periodic-refresh**]
7. **ip header-compression recoverable-loss** {**dynamic** | *packet-drops*}
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**encapsulation** *encapsulation-type* **Example:** `Router(config-if)# encapsulation ppp`	Sets the encapsulation method used on the interface. • Enter the encapsulation method.
Step 5	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 209.165.200.225 255.255.255.224`	Sets a primary or secondary IP address for an interface. • Enter the IP address and mask for the associated IP subnet.
Step 6	**ip rtp header-compression** [**passive** | **iphc-format** | **ietf-format**] [**periodic-refresh**] **Example:** `Router(config-if)# ip rtp header-compression ietf-format`	Enables RTP header compression.

	Command or Action	Purpose
Step 7	**ip header-compression recoverable-loss** {**dynamic** | *packet-drops*} **Example:** `Router(config-if)# ip header-compression recoverable-loss dynamic`	Enables ECRTP on an interface. **Note** Enter the **dynamic** keyword to enable dynamic packet loss recovery, or enter the *packet-drops*argument to specify the maximum number of consecutive packet drops that are acceptable.
Step 8	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Enabling RTP Header Compression over a Satellite Link

To enable RTP header compression over a satellite link, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **ip address** *ip-address mask* [**secondary**]
5. **ip rtp header-compression** [**passive** | **iphc-format** | **ietf-format**] [**periodic-refresh**]
6. **ip header-compression disable-feedback**
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 209.165.200.225 255.255.255.224`	Sets a primary or secondary IP address for an interface. • Enter the IP address and mask for the associated IP subnet.
Step 5	**ip rtp header-compression** [**passive** | **iphc-format** | **ietf-format**] [**periodic-refresh**] **Example:** `Router(config-if)# ip rtp header-compression ietf-format periodic-refresh`	Enables RTP header compression. **Note** For RTP header compression over a satellite link, use the **periodic-refresh** keyword.
Step 6	**ip header-compression disable-feedback** **Example:** `Router(config-if)# ip header-compression disable-feedback`	(Optional) Disables the context status feedback messages from the interface or link.
Step 7	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Specifying the Header-Compression Settings

With RTP header compression, you can configure the maximum size of the compressed IP header, the time period for an automatic resend of full-header packets, and the number of packets transmitted before a new full-header packet is sent.

To specify these header-compression settings, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **ip header-compression max-header** *max-header-size*
5.
6. **ip header-compression max-time** *length-of-time*
7.
8. **ip header-compression max-period** *number-of-packets*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**ip header-compression max-header** *max-header-size* **Example:** `Router(config-if)# ip header-compression max-header 100`	Specifies the maximum size of the compressed IP header. • Enter the maximum size of the compressed IP header, in bytes.
Step 5		

	Command or Action	Purpose
Step 6	**ip header-compression max-time** *length-of-time* **Example:** `Router(config-if)# ip header-compression max-time 30`	Specifies the maximum amount of time to wait before the compressed IP header is refreshed. • Enter the amount of time, in seconds.
Step 7		
Step 8	**ip header-compression max-period** *number-of-packets* **Example:** `Router(config-if)# ip header-compression max-period 160`	Specifies the maximum number of compressed packets between full headers. • Enter the maximum number of compressed packets between full headers.
Step 9	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Changing the Number of Header-Compression Connections

For PPP and HDLC interfaces, the default is 16 compression connections. For interfaces that use Frame Relay encapsulation, the default is 256 compression connections.

To change the default number of header-compression connections, perform the following steps.

• Implications of Changing the Number of Header-Compression Connections, page 20

Implications of Changing the Number of Header-Compression Connections

Each header-compression connection sets up a compression cache entry, so you are in effect specifying the maximum number of cache entries and the size of the cache. Too few cache entries for the specified interface can lead to degraded performance, and too many cache entries can lead to wasted memory. Choose the number of header-compression connections according to the network requirements.

Note **Header-Compression Connections on HDLC and Frame Relay Interfaces**

For HDLC interfaces and Frame Relay interfaces (that is, interfaces that use Frame Relay encapsulation), the number of header-compression connections on *both sides* of the network must match. That is, the number configured for use on the local router must match the number configured for use on the remote router.

Header-Compression Connections on PPP Interfaces

For PPP interfaces, if the header-compression connection numbers on both sides of the network do not match, the number used is "autonegotiated." That is, any mismatch in the number of header-compression connections between the local router and the remote router will be automatically negotiated to the lower of the two numbers. For example, if the local router is configured to use 128 header-compression connections, and the remote router is configured to use 64 header-compression connections, the negotiated number will be 64.

Note This autonegotiation function applies to PPP interfaces *only* . For HDLC interfaces and inte
Frame Relay encapsulation, no autonegotiation occurs.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **ip rtp compression-connections** *number*
5.
6. **frame-relay ip rtp compression-connections** *number*
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**ip rtp compression-connections** *number* **Example:** `Router(config-if)# ip rtp compression-connections 150`	Specifies the total number of RTP header-compression connections that can exist on an interface. • Enter the number of compression connections. **Note** This command can be used for PPP interfaces, HDLC interfaces, or interfaces that use Frame Relay encapsulation.
Step 5		
Step 6	**frame-relay ip rtp compression-connections** *number* **Example:** `Router(config-if)# frame-relay ip rtp compression-connections 150`	Specifies the maximum number of RTP header-compression connections that can exist on a Frame Relay interface (that is, an interface using Frame Relay encapsulation). • Enter the number of compression connections. **Note** This command can be used for interfaces that use Frame Relay encapsulation only.
Step 7	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Displaying Header-Compression Statistics

You can display header-compression statistics, such as the number of packets sent, received, and compressed, by using either the **show ip rtp header-compression** command or the **show frame-relay ip rtp header-compression** command.

To display header-compression statistics, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **show ip rtp header-compression** [*interface-type interface-number*] [**detail**]
3.
4. **show frame-relay ip rtp header-compression** [**interface** *type number*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip rtp header-compression** [*interface-type interface-number*] [**detail**] **Example:** `Router# show ip rtp header-compression` **Example:**	Displays RTP header-compression statistics for one or all interfaces.
Step 3		
Step 4	**show frame-relay ip rtp header-compression** [**interface** *type number*] **Example:** `Router# show frame-relay ip rtp header-compression`	Displays Frame Relay RTP header-compression statistics for one or all interfaces.
Step 5	**end** **Example:** `Router# end`	(Optional) Exits privileged EXEC mode.

Configuration Examples for RTP Header Compression

Example Enabling RTP Header Compression on an Interface

In the following example, RTP header compression is enabled on serial interface 0.

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# encapsulation ppp

Router(config-if)# ip address 209.165.200.225 255.255.255.224

Router(config-if)# ip rtp header-compression

Router(config-if)# end
```

Example Enabling RTP Header Compression on an Interface That Uses Frame Relay Encapsulation

In the following example, RTP header compression is enabled on serial interface 0. Frame Relay encapsulation has been enabled on this interface by using the **encapsulationframe-relay** command.

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# encapsulation frame-relay

Router(config-if)# ip address 209.165.200.225 255.255.255.224

Router(config-if)# frame-relay interface-dlci 20

Router(config-if)# frame-relay ip rtp header-compression

Router(config-if)# end
```

Example Enabling Enhanced RTP Header Compression

In the following example, ECRTP is enabled on serial interface 0. PPP encapsulation is enabled on the interface (a prerequisite for configuring ECRTP on a serial interface). Also, dynamic loss recovery has been specified by using the **dynamic** keyword of the **ip header-compression recoverable-loss** command.

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# encapsulation ppp

Router(config-if)# ip address 209.165.200.225 255.255.255.224

Router(config-if)# ip rtp header-compression ietf-format

Router(config-if)# ip header-compression recoverable-loss dynamic

Router(config-if)# end
```

Example Enabling RTP Header Compression over a Satellite Link

In the following example, RTP header compression is enabled on the serial interface 0. In this example, serial interface 0 is a satellite link in the network topology. The **periodic-refresh** keyword has been specified, which means that the compressed IP header will be refreshed periodically. Also, the context-status messages have been turned off (disabled).

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# ip address 209.165.200.225 255.255.255.224

Router(config-if)# ip rtp header-compression ietf-format periodic-refresh

Router(config-if)# ip header-compression disable-feedback

Router(config-if)# end
```

Example Specifying the Header-Compression Settings

In the following example, the maximum size of the compressed IP header (100 bytes) has been specified by using the **ip header-compression max-header** command.

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# ip header-compression max-header 100

Router(config-if)# end
```

Example Changing the Number of Header-Compression Connections

In the following example, the number of header-compression connections has been changed to 150 by using the **ip rtp compression-connections**command.

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# ip rtp compression-connections 150

Router(config-if)# end
```

Example Displaying Header-Compression Statistic

You can use the **show ip rtp header-compression**command to display header-compression statistics such as the number of packets received, sent, and compressed. The following is sample output from the **show ip rtp header-compression** command. In this example, ECRTP has been enabled on serial interface 0.

```
Router# show ip rtp header-compression serial0
RTP/UDP/IP header compression statistics:
  Interface Serial0 (compression on, IETF, ECRTP)
    Rcvd:    1473 total, 1452 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    1234 total, 1216 compressed, 0 status msgs, 379 not predicted
             41995 bytes saved, 24755 bytes sent
             2.69 efficiency improvement factor
    Connect: 16 rx slots, 16 tx slots,
             6 misses, 0 collisions, 0 negative cache hits, 13 free contexts
             99% hit ratio, five minute miss rate 0 misses/sec, 0 max
```

Additional References

The following sections provide references related to configuring RTP header compression.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Quality of Service Solutions Command Reference*
Frame Relay	"Frame Relay Queueing and Fragmentation at the Interface" module
Header compression overview	"Header Compression" module
TCP header compression	"Configuring TCP Header Compression" module
Class-based RTP and TCP header compression	"Configuring Class-Based RTP and TCP Header Compression" module
IPHC profiles and header compression	"Configuring Header Compression Using IPHC Profiles" module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 2507	*IP Header Compression*

RFC	Title
RFC 2508	*Compressing IP/UDP/RTP Headers for Low-Speed Serial Links*
RFC 3544	*IP Header Compression over PPP*
RFC 3545	*Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Glossary

compression --The running of a data set through an algorithm that reduces the space required to store the data set or the bandwidth required to transmit the data set.

context --The state that the compressor uses to compress a header and that the decompressor uses to decompress a header. The context is the uncompressed version of the last header sent and includes other information used to compress and decompress the packet.

context-state packet --A special packet sent from the decompressor to the compressor to communicate a list of (TCP or NON_TCP/RTP) context identifiers (CIDs) for which synchronization has been lost. This packet is sent only over a single link, so it requires no IP header.

DLCI --data-link connection identifier. A value that specifies a permanent virtual circuit (PVC) or switched virtual circuit (SVC) in a Frame Relay network. In the basic Frame Relay specification, DLCIs are locally significant (connected devices might use different values to specify the same connection). In the Local Management Interface (LMI) extended specification, DLCIs are globally significant (DLCIs specify individual end devices).

ECRTP --Enhanced Compressed Real-Time Transport Protocol. A compression protocol that is designed for unreliable point-to-point links with long delays.

encapsulation --A method of wrapping data in a particular protocol header. For example, Ethernet data is wrapped in a specific Ethernet header before network transit. Also, when dissimilar networks are bridged, the entire frame from one network is simply placed in the header used by the data link layer protocol of the other network.

full header (header refresh) --An uncompressed header that updates or refreshes the context for a packet stream. It carries a CID that will be used to identify the context. Full headers for non-TCP packet streams also carry the generation of the context that they update or refresh.

HDLC --High-Level Data Link Control. A bit-oriented synchronous data link layer protocol developed by the International Organization for Standardization (ISO). Derived from Synchronous Data Link Control (SDLC), HDLC specifies a data encapsulation method on synchronous serial links using frame characters and checksums.

header --A chain of subheaders.

IETF --Internet Engineering Task Force. A task force that consists of over 80 working groups responsible for developing Internet standards.

IPHC --IP Header Compression. A protocol capable of compressing both TCP and UDP headers.

ISDN --Integrated Services Digital Network. A communication protocol offered by telephone companies that permits telephone networks to carry data, voice, and other source traffic.

lossy serial links --Links in a network that are prone to lose packets.

packet stream --The sequence of packets whose headers are similar and share context. For example, headers in an RTP packet stream have the same source and final destination address and the same port numbers in the RTP header.

PPP --Point-to-Point Protocol. A protocol that provides router-to-router and host-to-network connections over synchronous and asynchronous circuits.

regular header --A normal, uncompressed header. A regular header does not carry a context identifier (CID) or generation association.

RTP --Real-Time Transport Protocol. A protocol that is designed to provide end-to-end network transport functions for applications that transmit real-time data, such as audio, video, or simulation data, over unicast or multicast network services. RTP provides such services as payload type identification, sequence numbering, timestamping, and delivery monitoring to real-time applications.

subheader --An IPv6 base header, an IPv6 extension header, an IPv4 header, a UDP header, an RTP header, or a TCP header.

Feature Information for Configuring RTP Header Compression

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1 Feature Information for Configuring RTP Header Compression

Feature Name	Releases	Feature Information
RTP Header Compression over Satellite Links	12.3(2)T	The RTP Header Compression over Satellite Links feature allows customers to use RTP header compression over an asymmetric link (such as a satellite link), where the uplink and downlink connections are on separate interfaces.
Enhanced CRTP for Links with High Delay, Packet Loss and Reordering	12.3(11)T	The Enhanced Compressed Real-Time Transport Protocol (ECRTP) for Links with High Delay, Packet Loss, and Reordering feature includes modifications and enhancements to CRTP to achieve robust operation over unreliable point-to-point links. This is accomplished by repeating updates and sending absolute (uncompressed) values in addition to delta values for selected context parameters.
RTP Header Compression RTP Header Compression over Satellite Links	15.0(1)S	The RTP Header Compression and RTP Header Compression over Satellite Links features were integrated into the Cisco IOS Release 15.0(1)S release.

Configuring TCP Header Compression

Header compression is a mechanism that compresses the IP header in a packet before the packet is transmitted. Header compression reduces network overhead and speeds up the transmission of either Real-Time Transport Protocol (RTP) or TCP packets.

Cisco provides two types of header compression: RTP header compression and TCP header compression. This module describes the concepts and tasks related to configuring TCP header compression.

Note TCP header compression is configured on a per-interface (or subinterface) basis. If you want to configure TCP header compression on a per-class basis, see the "Configuring Class-Based RTP and TCP Header Compression" module.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Configuring TCP Header Compression

- Before configuring TCP header compression, read the information in the "Header Compression" module.
- You must configure TCP header compression on both ends of the network.

Information About Configuring TCP Header Compression

TCP Header-Compression Keywords

When you configure TCP header compression, you can specify the circumstances under which the TCP packets are compressed and the format that is used when the packets are compressed. These circumstances and formats are defined by the following keywords:

- **passive**
- **iphc-format**
- **ietf-format**

These keywords (described below) are available with many of the quality of service (QoS) commands used to configure TCP header compression, such as the **ip tcp header-compression** command. For more information about the **ip tcp header-compression** command, these keywords, and the other QoS commands, see the Cisco IOS Quality of Service Solutions Command Reference.

The passive Keyword

By default, the **ip tcp header-compression** command compresses outgoing TCP traffic. If you specify the **passive** keyword, outgoing TCP traffic is compressed only if *incoming* TCP traffic on the *same* interface is compressed. If you do not specify the **passive** keyword, *all* outgoing TCP traffic is compressed.

The **passive** keyword is ignored for PPP interfaces.

The iphc-format Keyword

The **iphc-format**keyword indicates that the IP Header Compression (IPHC) format of header compression will be used. For PPP and HDLC interfaces, when the **iphc-format** keyword is specified, RTP header compression is also enabled. Since both TCP and RTP header compression are enabled, both TCP and UDP packets are compressed.

The **iphc-format**keyword is not available for interfaces that use Frame Relay encapsulation.

Note The header compression format (in this case, IPHC) must be the same at *both* ends of the network. That is, if you specify the **iphc-format** keyword on the local router, you must also specify the **iphc-format** keyword on the remote router.

The ietf-format Keyword

The **ietf-format** keyword indicates that the Internet Engineering Task Force (IETF) format of header compression will be used. For HDLC interfaces, the **ietf-format** keyword compresses only TCP packets. For PPP interfaces, when the **ietf-format**keyword is specified, RTP header compression is also enabled. Since both TCP header compression and RTP header compression are enabled, both TCP packets and UDP packets are compressed.

The **ietf-format** keyword is not available for interfaces that use Frame Relay encapsulation.

Note The header compression format (in this case, IETF) must be the same at *both* ends of the network. That is, if you specify the **ietf-format**keyword on the local router, you must also specify the **ietf-format** keyword on the remote router.

Maximum Compressed IP Header Size and TCP Header Compression

With TCP header compression, you can configure the maximum size of the compressed IP header by using the **ip header-compression max-header** command.

The **ip header-compression max-header** command allows you to define the maximum size of the IP header of a packet to be compressed. Any packet with an IP header that exceeds the maximum size is sent uncompressed. For more information about the **ip header-compression max-header** command, see the Cisco IOS Quality of Service Solutions Command Reference.

How to Configure TCP Header Compression

Enabling TCP Header Compression on an Interface

Note To enable TCP header compression on an interface that uses Frame Relay encapsulation, skip these steps and complete the steps in the Enabling TCP Header Compression on an Interface That Uses Frame Relay Encapsulation, page 35 instead.

To enable TCP header compression on an interface, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **encapsulation** *encapsulation-type*
5. **ip address** *ip-address mask* [**secondary**]
6. **ip tcp header-compression** [**passive** | **iphc-format** | **ietf-format**]
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**encapsulation** *encapsulation-type* **Example:** `Router(config-if)# encapsulation ppp`	Sets the encapsulation method used by the interface. • Enter the encapsulation method.
Step 5	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 209.165.200.225 255.255.255.224`	Sets a primary or secondary IP address for an interface. • Enter the IP address and mask for the associated IP subnet.
Step 6	**ip tcp header-compression** [**passive** | **iphc-format** | **ietf-format**] **Example:** `Router(config-if)# ip tcp header-compression ietf-format`	Enables TCP header compression.
Step 7	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Enabling TCP Header Compression on an Interface That Uses Frame Relay Encapsulation

To enable TCP header compression on an interface that uses Frame Relay encapsulation, perform the following steps.

Note The encapsulation type is specified by using either the **cisco** or **ietf** keyword of the **frame-relayinterface-dlci** command. The **cisco** keyword specifies Cisco proprietary encapsulations, and the **ietf** keyword specifies IETF encapsulations. However, note the following points about these keywords:

- Frame Relay interfaces do not support IETF encapsulations when TCP header compression is enabled. Therefore, the **ietf**keyword is not available for Frame Relay interfaces and is not listed in the command syntax shown below.
- The **cisco** keyword is available for use on point-to-point subinterfaces *only* .

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **encapsulation frame-relay**
5. **ip address** *ip-address mask* [**secondary**]
6. **frame-relay interface-dlci** *dlci* [**cisco**]
7. **frame-relay ip tcp header-compression** [**passive**]
8.
9. **frame-relay map ip** *ip-address dlci* [**broadcast**] **tcpheader-compression** [**active** | **passive**] [**connections***number*]
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**encapsulation frame-relay** **Example:** `Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation.
Step 5	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 209.165.200.225 255.255.255.224`	Sets a primary or secondary IP address for an interface. • Enter the IP address and mask for the associated IP subnet.
Step 6	**frame-relay interface-dlci** *dlci* [**cisco**] **Example:** `Router(config-if)# frame-relay interface-dlci 20`	Assigns a data-link connection identifier (DLCI) to a specified Frame Relay interface on the router or access server. • Enter the DLCI number.
Step 7	**frame-relay ip tcp header-compression** [**passive**] **Example:** `Router(config-if)# frame-relay ip tcp header-compression`	Enables TCP header compression for all Frame Relay maps on a physical interface.
Step 8		

	Command or Action	Purpose
Step 9	**frame-relay map ip** *ip-address dlci* [**broadcast**] **tcpheader-compression** [**active** | **passive**] [**connections***number*] **Example:** `Router(config-if)# frame-relay map ip 10.108.175.200 190 tcp header-compression active`	Assigns to an IP map header-compression characteristics that differ from the compression characteristics of the interface with which the IP map is associated. • Enter the IP address, DLCI number, and any optional keywords and arguments.
Step 10	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Enabling Special-VJ Format TCP Header Compression

To enable the special Van Jacobson (VJ) format of TCP header compression so that context IDs are included in compressed packets, perform the following steps.

Enable TCP header compression using the **ip tcp header-compression** command before configuring the special-VJ format.

Note This task is unnecessary if IPHC was configured on an interface using the **iphc-profile**command.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **encapsulation ppp**
5. **ip address** *ip-address mask* [**secondary**]
6. ip tcp header-compression
7. ip header-compression special-vj
8. **ip tcp compression-connections** *number*
9. **exit**
10. **iphc-profile** *profile-name* **van-jacobson**
11. **special-vj**
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial 0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**encapsulation ppp** **Example:** `Router(config-if)# encapsulation ppp`	**(Optional) Sets the encapsulation method used by the interface.**
Step 5	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 209.165.200.225 255.255.255.224`	Sets a primary or secondary IP address for an interface. • Enter the IP address and mask for the associated IP subnet.
Step 6	ip tcp header-compression **Example:** `Router(config-if)# ip header-compression`	`Enables TCP header compression.`
Step 7	ip header-compression special-vj **Example:** `Router(config-if)# ip header-compression special-vj`	**Enables the special VJ format of TCP header compression.**

	Command or Action	Purpose
Step 8	**ip tcp compression-connections** *number* **Example:** `Router(config-if)# ip tcp compression-connections 16`	`Specifies the total number of TCP header compression connections that can exist on an interface.`
Step 9	**exit** **Example:** `Router(config-if)# exit`	Exits the current configuration mode.
Step 10	**iphc-profile** *profile-name* **van-jacobson** **Example:** `Router(config)# iphc-profile profile1 van-jacobson`	Creates an IP Header Compression (IPHC) profile and enters IPHC profile configuration mode.
Step 11	**special-vj** **Example:** `Router(config-iphcp)# special-vj`	**Enables the special VJ format of TCP header compression so that context IDS are included in compressed packets.**
Step 12	**end** **Example:** `Router(config-if)# end`	(Optional) Exits the current configuration mode.

Changing the Maximum Size of the Compressed IP Header

By default, the maximum size of the compressed IP header is 168 bytes. When you configure TCP header compression, you can change this size to suit the needs of your network.

To change the maximum size of the compressed IP header, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **ip header-compression max-header** *max-header-size*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**ip header-compression max-header** *max-header-size* **Example:** `Router(config-if)# ip header-compression max-header 100`	Specifies the maximum size of the compressed IP header. • Enter the maximum size of the compressed IP header, in bytes.
Step 5	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Changing the Number of Header-Compression Connections

For PPP and HDLC interfaces, the default is 16 compression connections. For interfaces that use Frame Relay encapsulation, the default is 256 compression connections.

To change the default number of header-compression connections, perform the following steps.

• Implications of Changing the Number of Header-Compression Connections, page 40

Implications of Changing the Number of Header-Compression Connections

Each header-compression connection sets up a compression cache entry, so you are in effect specifying the maximum number of cache entries and the size of the cache. Too few cache entries for the specified interface can lead to degraded performance, and too many cache entries can lead to wasted memory. Choose the number of compression connections according to the network requirements.

Note **Header-Compression Connections on HDLC and Frame Relay Interfaces**

For HDLC interfaces and Frame Relay interfaces (that is, interfaces that use Frame Relay encapsulation), the number of header-compression connections on *both sides* of the network must match. That is, the number configured for use on the local router must match the number configured for use on the remote router.

Header-Compression Connections on PPP Interfaces

For PPP interfaces, if the header-compression connection numbers on both sides of the network do not match, the number used is "autonegotiated." That is, any mismatch in the number of header-compression connections between the local router and the remote router will be automatically negotiated to the lower of the two numbers. For example, if the local router is configured to use 128 header-compression connections, and the remote router is configured to use 64 header-compression connections, the negotiated number will be 64.

Note This autonegotiation function applies to PPP interfaces *only* . For HDLC interfaces and inte
Frame Relay encapsulation, no autonegotiation occurs.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **ip tcp compression-connections** *number*
5.
6. **frame-relay ip tcp compression-connections** *number*
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**ip tcp compression-connections** *number* **Example:** `Router(config-if)# ip tcp compression-connections 150`	Specifies the total number of TCP header compression connections that can exist on an interface. • Enter the number of compression connections. **Note** This command can be used for PPP interfaces, HDLC interfaces, or interfaces that use Frame Relay encapsulation.
Step 5		
Step 6	**frame-relay ip tcp compression-connections** *number* **Example:** `Router(config-if)# frame-relay ip tcp compression-connections 150`	Specifies the maximum number of TCP header compression connections that can exist on an interface that use Frame Relay encapsulation. • Enter the number of compression connections. **Note** This command can be used for interfaces that use Frame Relay encapsulation *only* .
Step 7	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Displaying Header-Compression Statistics

You can display header-compression statistics, such as the number of packets sent, received, and compressed, by using either the **show ip tcp header-compression** command or the **show frame-relay ip tcp header-compression** command.

To display header-compression statistics, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **show ip tcp header-compression** [*interface-type interface-number*] [**detail**]
3.
4. **show frame-relay ip tcp header-compression** [**interface** *type number*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip tcp header-compression** [*interface-type interface-number*] [**detail**] **Example:** `Router# show ip tcp header-compression` **Example:**	Displays TCP/IP header compression statistics.
Step 3		
Step 4	**show frame-relay ip tcp header-compression** [**interface** *type number*] **Example:** `Router# show frame-relay ip tcp header-compression` **Example:**	Displays Frame Relay TCP/IP header compression statistics for one or all interfaces.
Step 5	**end** **Example:** `Router# end`	(Optional) Exits privileged EXEC mode.

Configuration Examples for TCP Header Compression

Example Enabling TCP Header Compression on an Interface

In the following example, TCP header compression is enabled on serial interface 0.

```
Router> enable
Router# configure terminal
Router(config)# interface serial0
Router(config-if)# encapsulation ppp
Router(config-if)# ip address 209.165.200.225 255.255.255.224
Router(config-if)# ip tcp header-compression ietf-format
Router(config-if)# end
```

Example Enabling TCP Header Compression on an Interface That Uses Frame Relay Encapsulation

In the following example, TCP header compression is enabled on serial interface 0. Frame Relay encapsulation has been enabled on this interface by using the **encapsulationframe-relay** command.

```
Router> enable
Router# configure terminal
Router(config)# interface serial0
Router(config-if)# encapsulation frame-relay
Router(config-if)# ip address 209.165.200.225 255.255.255.224
Router(config-if)# frame-relay interface-dlci 20
Router(config-if)# frame-relay ip tcp header-compression
Router(config-if)# end
```

Example Enabling Special-VJ Format TCP Header Compression

In the following example, TCP header compression is enabled on serial interface 0. The special VJ format has been enabled on this interface by using the **ip header-compression special-vj**, **ip tcp compression-connections**, and the **special-vj**commands:

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# ip address 209.165.200.225 255.255.255.224

Router(config-if)# ip tcp header-compression

Router(config-if)# ip header-compression special-vj

Router(config-if)# ip tcp compression-connections 16

Router(config-if))# exit

Router(config)# iphc-profile profile-name van-jacobson

Router(config-iphcp)# special-vj
Router(config-if)# end
```

Example Changing the Maximum Size of the Compressed IP Header

In the following example, the maximum size of the compressed IP header (100 bytes) has been specified by using the **ip header-compression max-header** command:

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# ip header-compression max-header 100

Router(config-if)# end
```

Example Changing the Number of Header-Compression Connections

In the following example, the number of header-compression connections has been changed to 150 by using the **ip tcp compression-connections**command:

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# ip tcp compression-connections 150

Router(config-if)# end
```

Example Displaying Header-Compression Statistics

You can use the **show ip tcp header-compression**command to display header-compression statistics such as the number of packets received, sent, and compressed. The following is sample output from the **show ip tcp header-compression** command:

```
Router# show ip tcp header-compression serial0
TCP/IP header compression statistics:
  Interface Serial0 (compression on, IETF)
    Rcvd:    53797 total, 53796 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    53797 total, 53796 compressed, 0 status msgs, 0 not predicted
             1721848 bytes saved, 430032 bytes sent
             5.00 efficiency improvement factor
    Connect: 16 rx slots, 16 tx slots,
             1 misses, 0 collisions, 0 negative cache hits, 15 free contexts
             99% hit ratio, five minute miss rate 0 misses/sec, 0 max
```

Additional References

The following sections provide references related to configuring TCP header compression.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Quality of Service Solutions Command Reference*
Frame Relay	"Frame Relay Queueing and Fragmentation at the Interface" module
Header compression overview	"Header Compression" module

Related Topic	Document Title
RTP header compression	"Configuring RTP Header Compression" module
Class-based RTP and TCP header compression	"Configuring Class-Based RTP and TCP Header Compression" module
IPHC profiles and header compression	"Configuring Header Compression Using IPHC Profiles" module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1144	*Compressing TCP/IP Headers for Low-Speed Serial Links*
RFC 2507	*IP Header Compression*
RFC 3544	*IP Header Compression over PPP*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Glossary

compression --The running of a data set through an algorithm that reduces the space required to store the data set or the bandwidth required to transmit the data set.

DLCI --data-link connection identifier. A value that specifies a permanent virtual circuit (PVC) or switched virtual circuit (SVC) in a Frame Relay network. In the basic Frame Relay specification, DLCIs are locally significant (connected devices might use different values to specify the same connection). In the Local Management Interface (LMI) extended specification, DLCIs are globally significant (DLCIs uniquely specify individual end devices).

encapsulation --A method of wrapping data in a particular protocol header. For example, Ethernet data is wrapped in a specific Ethernet header before network transit. Also, when dissimilar networks are bridged, the entire frame from one network is simply placed in the header used by the data link layer protocol of the other network.

full header (header refresh) --An uncompressed header that updates or refreshes the context for a packet stream. It carries a context identifier (CID) that will be used to identify the context. Full headers for non-TCP packet streams also carry the generation of the context that they update or refresh.

HDLC --High-Level Data Link Control. A bit-oriented synchronous data link layer protocol developed by the International Organization for Standardization (ISO). Derived from Synchronous Data Link Control (SDLC), HDLC specifies a data encapsulation method on synchronous serial links using frame characters and checksums.

header --A chain of subheaders.

IETF --Internet Engineering Task Force. A task force that consists of over 80 working groups responsible for developing Internet standards.

IPHC --IP Header Compression. A protocol capable of compressing both TCP and UDP headers.

PPP --Point-to-Point Protocol. A protocol that provides router-to-router and host-to-network connections over synchronous and asynchronous circuits.

regular header --A normal, uncompressed header. A regular header does not carry a context identifier (CID) or generation association.

subheader --An IPv6 base header, an IPv6 extension header, an IPv4 header, a UDP header, an RTP header, or a TCP header.

TCP --Transmission Control Protocol. A connection-oriented transport layer protocol that provides reliable full-duplex data transmission. TCP is part of the TCP/IP protocol stack.

UDP --User Datagram Protocol. A connectionless transport layer protocol in the TCP/IP protocol stack. UDP is a simple protocol that exchanges datagrams without acknowledgments or guaranteed delivery, requiring that error processing and retransmission be handled by other protocols. UDP is defined in RFC 768.

Feature Information for Configuring TCP Header Compression

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 2 Feature Information for Configuring TCP Header Compression

Feature Name	Releases	Feature Information
This table is intentionally left blank because no features were introduced or modified in Cisco IOS Release 12.2(1) or a later release. This table will be updated when feature information is added to this module.	--	--

Configuring Class-Based RTP and TCP Header Compression

Header compression is a mechanism that compresses the IP header in a packet before the packet is transmitted. Header compression reduces network overhead and speeds up the transmission of either Real-Time Transport Protocol (RTP) packets or Transmission Control Protocol (TCP) packets.

Cisco provides two types of header compression: RTP header compression and TCP header compression.

RTP and TCP header compression are typically configured on a per-interface (or subinterface) basis. Class-based RTP and TCP header compression allows you to configure either type of header compression on a per-class basis. This module describes the concepts and tasks related to configuring class-based RTP and TCP header compression.

Note If you want to configure RTP or TCP header compression on a per-interface (or subinterface) basis, see the "Configuring RTP Header Compression" module or the "Configuring TCP Header Compression" module, respectively.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Class-Based RTP and TCP Header Compression

Before configuring class-based RTP and TCP header compression, read the information in the "Header Compression" module.

Restrictions for Class-Based RTP and TCP Header Compression

Class-based RTP and TCP header compression can be enabled on PPP interfaces, High-Level Data Link Control (HDLC) interfaces, and interfaces that use Frame Relay encapsulation. However, note the following points about the header-compression formats supported on these interfaces:

- For PPP and HDLC interfaces, the only supported format for header compression is the IPHC (IP Header Compression) format.
- For interfaces that use Frame Relay encapsulation, the IPHC format is not available. The only supported format for header compression is the Cisco proprietary format.

Information About Class-Based RTP and TCP Header Compression

Class-Based Header Compression and the MQC

Class-based RTP and TCP header compression allows you to configure *either* RTP *or* TCP header compression for a specific class within a policy map (sometimes referred to as a traffic policy). You configure the class and the policy map by using the Modular Quality of Service (QoS) Command-Line Interface (CLI) (MQC). The MQC is a CLI that allows you to create classes within policy maps (traffic policies) and then attach the policy maps to interfaces (or subinterfaces). The policy maps are used to configure and apply specific QoS features (such as RTP or TCP header compression) to your network. For more information about the MQC, see the "Applying QoS Features Using the MQC" module.

Benefits of Class-Based Header Compression

Class-based header compression allows you to compress (and then decompress) a subset of the packets on your network. Class-based header compression acts as a filter; it allows you to specify at a much finer level the packets that you want to compress. For example, instead of compressing all RTP (or TCP) packets that traverse your network, you can configure RTP header compression to compress only those packets that meet certain criteria (for example, protocol type "ip" in a class called "voice)."

Header Compression on Local and Remote Routers

In a typical network topology, header compression is configured at both a local router and a remote router. If you configure class-based RTP header compression (or class-based TCP header compression) on the local router, you must also configure RTP header compression (or TCP header compression) on the remote router.

However, when you configure either RTP or TCP header compression on the remote router, you can choose one of the following:

- You can configure *class-based* RTP or TCP header compression on the remote router (by using the instructions in this module)

or

- You can configure RTP or TCP header compression *directly on the interface* of the remote router (by using the instructions in the "Configuring RTP Header Compression" module or the "Configuring TCP Header Compression" module, respectively).

Note If you configure RTP or TCP header compression directly on the interface of the remote router, you must specify the **iphc-format** keyword for PPP and HDLC interfaces. For Frame Relay interfaces, the **iphc-format** keyword is not supported; only the Cisco proprietary format (that is, the **cisco** keyword) is supported. For more information about the **iphc-format**keyword, see either the "Configuring RTP Header Compression" module or the "Configuring TCP Header Compression" module.

About Header-Compression Connections

Number of Connections Calculated on the Basis of Bandwidth

In class-based RTP and TCP header compression, the number of header-compression connections is calculated on the basis of the amount of available bandwidth.

Note the following points about how bandwidth is used:

- The setting of the **bandwidth** command determines the amount of bandwidth available on the interface.
- The number of header-compression connections is calculated by dividing the available bandwidth by 4 (that is, 4 kilobits per connection).

Header-Compression Connections on HDLC and Frame Relay Interfaces

For HDLC interfaces and Frame Relay interfaces (that is, interfaces that use Frame Relay encapsulation), the number of header-compression connections on *both sides* of the network must match. That is, the number calculated (from the bandwidth setting) for use on the local router must match the number configured (or calculated from the bandwidth setting) for use on the remote router.

Header-Compression Connections on PPP Interfaces

For PPP interfaces, if the header-compression connection numbers on both sides of the network do not match, the number used is "autonegotiated." That is, any mismatch in the number of header-compression connections between the local router and the remote router will be automatically negotiated to the lower of

the two numbers. For example, if the local router is configured to use 128 header-compression connections, and the remote router is configured to use 64 header-compression connections, the negotiated number will be 64.

Note This autonegotiation function applies to PPP interfaces *only* . For HDLC interfaces and interfaces that use Frame Relay encapsulation, no autonegotiation occurs.

How to Configure Class-Based RTP and TCP Header Compression

Enabling RTP or TCP Header Compression for a Class in a Policy Map

With class-based header compression, you can configure either RTP or TCP header compression for a specific class inside a policy map. To specify the class, to create a policy map, and to configure either RTP or TCP header compression for the class inside the policy map, perform the following steps.

Note In the following task, the **match protocol**command is shown in step Enabling RTP or TCP Header Compression for a Class in a Policy Map, page 54 The **match protocol**command matches traffic on the basis on the protocol type and is only an example of a **match** command you can use. You may want to use a different **match** command to specify another criterion. The **match** commands vary by Cisco IOS release. See the command documentation for the Cisco IOS release that you are using for a complete list of **match** commands.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **class-map** [**match-all**| **match-any**] *class-map-name*
4. **match protocol** *protocol-name*
5. **exit**
6. **policy-map** *policy-map-name*
7. **class** {*class-name*| **class-default**}
8. **compression header ip** {**rtp** | **tcp**}
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**class-map** [**match-all**| **match-any**] *class-map-name* **Example:** `Router(config)# class-map class1`	Creates a class map to be used for matching packets to a specified class and enters class-map configuration mode. • Enter the class map name.
Step 4	**match protocol** *protocol-name* **Example:** `Router(config-cmap)# match protocol ip`	(Optional) Matches traffic on the basis of the specified protocol. • Enter the protocol name. **Note** The **match protocol** command matches traffic on the basis of the protocol type. The **match protocol**command is just an example of one of the **match** commands that can be used. The **match** commands vary by Cisco IOS release. See the command documentation for the Cisco IOS release that you are using for a complete list of **match** commands.
Step 5	**exit** **Example:** `Router(config-cmap)# exit`	(Optional) Exits class-map configuration mode.
Step 6	**policy-map** *policy-map-name* **Example:** `Router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters policy-map configuration mode. • Enter the policy map name.

	Command or Action	Purpose
Step 7	**class** {*class-name*| **class-default**} **Example:** `Router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters policy-map class configuration mode. • Enter the class name or the **class-default** keyword.
Step 8	**compression header ip** {**rtp** | **tcp**} **Example:** `Router(config-pmap-c)# compression header ip rtp`	Configures either RTP or TCP header compression for a specific class. • Enter either the **rtp** keyword (for RTP header compression) or the **tcp** keyword (for TCP header compression).
Step 9	**end** **Example:** `Router(config-pmap-c)# end`	(Optional) Exits policy-map class configuration mode.

Attaching the Policy Map to an Interface

After a policy map is created, the next step is to attach the policy map to an interface (or subinterface). To attach the policy map to an interface or subinterface, perform the following steps.

Note You configure class-based RTP and TCP header compression in policy maps. Then you attach those policy maps to an interface by using the **service-policy** command. The **service-policy**command gives you the option of specifying either an input service policy (for input interfaces) or an output service policy (for output interfaces). For class-based RTP and TCP header compression, you can specify output service policies *only* .

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **service-policy output** *policy-map-name*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**service-policy output** *policy-map-name* **Example:** `Router(config-if)# service-policy output policy1`	Specifies the name of the policy map to be attached to the interface in the output direction. • Enter the policy map name. **Note** Policy maps can be attached in the input or output direction of an interface. For class-based RTP and TCP header compression, always use the **output** keyword.
Step 5	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode.

Verifying the Class-Based RTP and TCP Header Compression Configuration

This task allows you to verify that you created the intended configuration and that the feature is functioning correctly. To verify the configuration, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **show policy-map interface** *type number* **output**
3.
4. **show policy-map** *policy-map* **class** *class-name*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show policy-map interface** *type number* **output** **Example:** `Router# show policy-map interface serial0 output`	Displays the packet statistics of all classes that are configured for all service policies on the specified interface. • Enter the interface type and the interface number.
Step 3		
Step 4	**show policy-map** *policy-map* **class** *class-name* **Example:** `Router# show policy-map policy1 class class1`	Displays the configuration for the specified class of the specified policy map. • Enter the policy map name and the class name.
Step 5	**end** **Example:** `Router# end`	(Optional) Exits privileged EXEC mode.

Configuration Examples for Class-Based RTP and TCP Header Compression

Example Enabling RTP or TCP Header Compression for a Class in a Policy Map

In the following example, a class map called class1 and a policy map called policy1 have been configured. Policy1 contains the class called class1, within which RTP header compression has been enabled by using the **compression header ip rtp** command.

```
Router> enable

Router# configure terminal

Router(config)# class-map class1

Router(config-cmap)# match protocol ip

Router(config-cmap)# exit

Router(config)# policy-map policy1

Router(config-pmap)# class class1

Router(config-pmap-c)# compression header ip rtp

Router(config-pmap-c)# end
```

Example Attaching the Policy Map to an Interface

In the following example, the policy map called policy1 has been attached to serial interface 0.

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# service-policy output policy1

Router(config-if)# end
```

Example Verifying the Class-Based RTP and TCP Header Compression Configuration

This section provides sample output from a typical **showpolicy-mapinterface**command.

Note Depending upon the interface in use and the QoS feature enabled (such as Class-Based Weighted Fair Queuing [CBWFQ]), the output you see may vary from that shown below.

The following sample displays the statistics for serial interface 0. In this sample configuration, three classes, called gold, silver, and voice, have been configured. Traffic is classified and grouped into classes on the basis of the IP precedence value and RTP port protocol number.

```
class-map match-all gold
 match ip precedence 2
class-map match-all silver
 match ip precedence 1
class-map match-all voice
 match ip precedence 5
 match ip rtp 16384 1000
```

This sample configuration also contains a policy map called mypolicy, configured as shown below. QoS features such as RTP header compression and CBWFQ are enabled for specific classes within the policy map.

```
policy-map mypolicy
  class voice
    priority 128              ! A priority queue and bandwidth amount are specified.
    compress header ip rtp    ! RTP header compression is enabled for class voice.
  class gold
    bandwidth 100             ! CBWFQ is enabled for class gold.
  class silver
    bandwidth 80              ! CBWFQ is enabled for class silver.
    random-detect             ! WRED is enabled for class silver.
```

Given the classes and policy map configured as shown above, the following content is displayed for serial interface 0:

```
Router# show policy-map interface
 serial0 output
 Serial0
  Service-policy output: mypolicy

    Class-map: voice (match-all)
      880 packets, 58080 bytes
      30 second offered rate 1000 bps, drop rate 0 bps
      Match: ip precedence 5
      Match: ip rtp 16384 1000
      Queueing
        Strict Priority
        Output Queue: Conversation 136
        Bandwidth 128 (kbps) Burst 3200 (Bytes)
        (pkts matched/bytes matched) 880/26510
        (total drops/bytes drops) 0/0
      compress:
          header ip rtp
          UDP/RTP (compression on, IPHC, RTP)
            Sent:    880 total, 877 compressed,
                     31570 bytes saved, 24750 bytes sent
                     2.27 efficiency improvement factor
                     99% hit ratio, five minute miss rate 0 misses/sec, 0 max
                     rate 0 bps

    Class-map: gold (match-all)
      100 packets, 53000 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: ip precedence 2
      Queueing
        Output Queue: Conversation 137
        Bandwidth 100 (kbps) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 100/53000
        (depth/total drops/no-buffer drops) 0/0/0
```

```
  Class-map: silver (match-all)
    878 packets, 1255540 bytes
    30 second offered rate 56000 bps, drop rate 0 bps
    Match: ip precedence 1
    Queueing
      Output Queue: Conversation 138
      Bandwidth 64 (kbps)
      (pkts matched/bytes matched) 878/1255540
      (depth/total drops/no-buffer drops) 0/0/0
       exponential weight: 9
       mean queue depth: 0

class    Transmitted      Random drop      Tail drop    Minimum Maximum  Mark
         pkts/bytes       pkts/bytes       pkts/bytes    thresh  thresh prob
    0       0/0              0/0              0/0           20      40  1/10
    1     878/1255540        0/0              0/0           22      40  1/10
    2       0/0              0/0              0/0           24      40  1/10
    3       0/0              0/0              0/0           26      40  1/10
    4       0/0              0/0              0/0           28      40  1/10
    5       0/0              0/0              0/0           30      40  1/10
    6       0/0              0/0              0/0           32      40  1/10
    7       0/0              0/0              0/0           34      40  1/10
 rsvp       0/0              0/0              0/0           36      40  1/10

  Class-map: class-default (match-any)
    3 packets, 84 bytes
    30 second offered rate 0 bps, drop rate 0 bps
    Match: any
```

Additional References

The following sections provide references related to configuring class-based RTP and TCP header compression.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Quality of Service Solutions Command Reference*
MQC	"Applying QoS Features Using the MQC" module
Header compression overview	"Header Compression" module
RTP header compression	"Configuring RTP Header Compression" module
TCP header compression	"Configuring TCP Header Compression" module
IPHC profiles and header compression	"Configuring Header Compression Using IPHC Profiles" module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1144	*Compressing TCP/IP Headers for Low-Speed Serial Links*
RFC 2507	*IP Header Compression*
RFC 2508	*Compressing IP/UDP/RTP Headers for Low-Speed Serial Links*
RFC 3544	*IP Header Compression over PPP*
RFC 3545	*Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering*
RFC 3550	*A Transport Protocol for Real-Time Applications*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Glossary

bandwidth --The rated throughput capacity of a given network medium.

compression --The running of a data set through an algorithm that reduces the space required to store the data set or the bandwidth required to transmit the data set.

full header (header refresh) --An uncompressed header that updates or refreshes the context for a packet stream. It carries a context identifier (CID) that will be used to identify the context. Full headers for non-TCP packet streams also carry the generation of the context that they update or refresh.

HDLC --High-Level Data Link Control. A bit-oriented synchronous data link layer protocol developed by the International Organization for Standardization (ISO). Derived from Synchronous Data Link Control (SDLC), HDLC specifies a data encapsulation method on synchronous serial links using frame characters and checksums.

header --A chain of subheaders.

MQC --Modular Quality of Service Command-Line Interface. The MQC is a CLI that allows you to create traffic classes and policy maps and then attach the policy maps to interfaces. The policy maps apply QoS features to your network.

PPP --Point-to-Point Protocol. A protocol that provides router-to-router and host-to-network connections over synchronous and asynchronous circuits.

regular header --A normal, uncompressed header. A regular header does not carry a context identifier (CID) or generation association.

RTP --Real-Time Transport Protocol. A protocol that is designed to provide end-to-end network transport functions for applications that transmit real-time data, such as audio, video, or simulation data, over unicast or multicast network services. RTP provides such services as payload type identification, sequence numbering, timestamping, and delivery monitoring to real-time applications.

subheader --An IPv6 base header, an IPv6 extension header, an IPv4 header, a UDP header, an RTP header, or a TCP header.

TCP --Transmission Control Protocol. A connection-oriented transport layer protocol that provides reliable full-duplex data transmission. TCP is part of the TCP/IP protocol stack.

Feature Information for Class-Based RTP and TCP Header Compression

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 3 Feature Information for Class-Based RTP and TCP Header Compression

Feature Name	Releases	Feature Information
Class-Based RTP and TCP Header Compression	12.2(13)T	This feature allows you to configure Real-Time Transport Protocol (RTP) or Transmission Control Protocol (TCP) IP header compression on a per-class basis, when a class is configured within a policy map. Policy maps are created using the Modular Quality of Service (QoS) Command-Line Interface (CLI) (MQC).

Configuring Header Compression Using IPHC Profiles

Header compression is a mechanism that compresses the IP header in a packet before the packet is transmitted. Header compression reduces network overhead and speeds up the transmission of either Real-Time Transport Protocol (RTP) or Transmission Control Protocol (TCP) packets.

One method of configuring header compression on your network is to use an IP header compression (IPHC) profile. An IPHC profile is a kind of template within which you can configure the type of header compression that you want to use, set all of the optional features and parameters for header compression, and then apply the profile to an interface, subinterface, or Frame Relay permanent virtual circuit (PVC).

This module describes the concepts and tasks for configuring header compression using IPHC profiles.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Using IPHC Profiles

Before using IPHC profiles to configure header compression, read the information in the "Header Compression" module.

Restrictions for Using IPHC Profiles

IPHC profiles are not supported on L2TP networks.

Information About Using IPHC Profiles

Benefits of Using IPHC Profiles

An IPHC profile provides a flexible means of enabling header compression and the options associated with header compression. For example, header compression (and the header compression options) can be enabled *once* in an IPHC profile, and then the IPHC profile can be applied to one or more of the following:

- An interface
- A subinterface
- A Frame Relay PVC

IPHC Profile Types

You use the **iphc-profile** command to create the IPHC profile. When you create an IPHC profile, you must specify the IPHC profile type. The IPHC profile choices are Internet Engineering Task Force (IETF) or van-jacobson. You specify the IPHC profile type with the **ietf** keyword or the **van-jacobson** keyword of the **iphc-profile** command.

The ietf profile type conforms with and supports the standards established with RFC 2507, RFC 2508, RFC 3544, and RFC 3545 and is typically associated with non-TCP header compression (for example, RTP header compression). The van-jacobson profile type conforms with and supports the standards established with RFC 1144 and is typically associated with TCP header compression.

Considerations When Specifying the IPHC Profile Type

When specifying the IPHC profile type, consider whether you are compressing TCP traffic or non-TCP (that is, RTP) traffic. Also consider the header compression format capabilities of the remote network link to which you will be sending traffic.

The IPHC profile type that you specify directly affects the header compression format used on the remote network links to which the IPHC profile is applied. *Only* TCP traffic is compressed on remote network links using a van-jacobson IPHC profile, whereas *both* TCP and non-TCP (for example, RTP) traffic is compressed on remote network links using an ietf IPHC profile.

Note The header compression format in use on the router that you are configuring and the header compression format in use on the remote network link must match.

Configurable Header Compression Features and Settings

The specific header compression features and settings that you can configure (that is, enable or modify) are determined by the IPHC profile type that you select (either van-jacobson or ietf) when you create the IPHC profile. There is one set of features and options for the van-jacobson IPHC profile type and another set for the ietf IPHC profile type. Both sets are listed below.

Features and Settings for van-jacobson IPHC Profile Type Header Compression

If you specify van-jacobson as the IPHC profile type, you can enable TCP header compression and set the number of TCP contexts. The table below lists the van-jacobson IPHC profile type header compression features and settings that are available and the command used to enable that feature or setting.

Table 4 van-jacobson IPHC Profile Type Header Compression Features and Settings

Feature or Setting	Command
TCP header compression	**tcp**
Number of contexts available for TCP header compression	**tcp contexts**

Features and Settings for ietf IPHC Profile Type Header Compression

If you specify ietf as the IPHC profile type, you can enable non-TCP header compression (that is, RTP header compression), along with a number of additional features and settings. The table below lists the ietf IPHC profile type header compression features and settings that are available and the command used to enable that feature or setting.

Table 5 ietf IPHC Profile Type Header Compression Features and Settings

Feature or Setting	Command
Non-TCP header compression	**non-tcp**
Number of contexts available for non-TCP header compression	**non-tcp contexts**
RTP header compression	**rtp**
Enhanced Compressed Real-Time Transport Protocol (ECRTP) on an interface	**recoverable-loss**
Context refresh (full-header refresh) options, such as the amount of time to wait before a full-header is refreshed	**refresh max-time refresh max-period refresh rtp**
Context-status feedback messages from the interface or link	**feedback**
Maximum size of the compressed IP header	**maximum header**
TCP header compression	**tcp**

Feature or Setting	Command
Number of contexts available for TCP header compression	**tcp contexts**

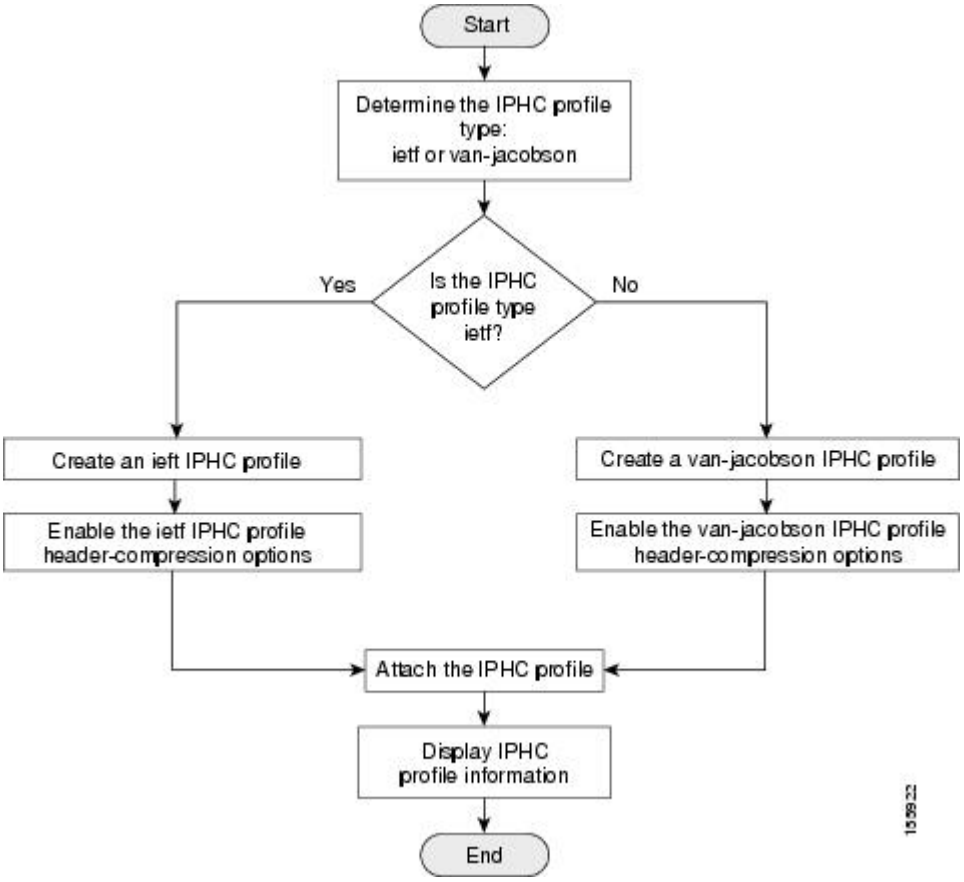
Tasks for Using IPHC Profiles

The tasks for configuring header compression using an IPHC profile are described below.

1. Create the IPHC profile and specify the IPHC profile type (ietf or van-jacobson) that you want to use.
2. Enable or set the header compression features available for the IPHC profile type that you specified when you created the IPHC profile. The header compression features vary by IPHC profile type.
3. Attach the IPHC profile to an interface, subinterface, or Frame Relay PVC.
4. Display information about the IPHC profiles that you have created.

The figure below illustrates the high-level processes for configuring header compression using IPHC profiles.

Figure 2 Flowchart for Configuring Header Compression Using IPHC Profiles



How to Configure Header Compression Using IPHC Profiles

Creating an IPHC Profile

The first task is to create an IPHC profile. When you create an IPHC profile, you can create either an ietf IPHC profile or a van-jacobson IPHC profile, by using the corresponding keyword of the **iphc-profile** command.

To create either an ietf IPHC profile or a van-jacobson IPHC profile, complete the following steps.

Before completing the steps listed below, determine the type of IPHC profile that you want to create: ietf or van-jacobson. The IPHC profile type that you create directly affects the header compression options available for you.

For more information about IPHC profile types and considerations for selecting one or the other, see the IPHC Profile Types, page 66.

Note The IPHC profile name must be unique and cannot be longer than 32 characters. IPHC profile names exceeding this maximum are truncated to 32 characters.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **iphc-profile** *profile-name* {**ietf** | **van-jacobson**}
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**iphc-profile** *profile-name* {**ietf** | **van-jacobson**} **Example:** `Router(config)# iphc-profile profile2 ietf`	Creates an IPHC profile and enters IPHC-profile configuration mode. • Enter the IPHC profile name and the IPHC profile type keyword.
Step 4	**end** **Example:** `Router(config-iphcp)# end`	(Optional) Exits IPHC-profile configuration mode.

• What to Do Next, page 70

What to Do Next

So far you have created either an ietf IPHC profile or a van-jacobson IPHC profile.

The next step is to enable or set any additional header compression features or options available for the type of IPHC profile that you created.

Choose one of the following:

- To enable or set any of the header compression features available for a van-jacobson IPHC profile, complete the steps in the Enabling the Options for van-jacobson IPHC Profile Type Header Compression, page 70 section below.
- To enable or set any of the header compression features available for an ietf IPHC profile, complete the steps in the Enabling the Options for ietf IPHC Profile Type Header Compression, page 72.

Enabling the Options for van-jacobson IPHC Profile Type Header Compression

If you created a van-jacobson IPHC profile, you can enable TCP header compression and set the number of TCP contexts.

Note If you created an ietf IPHC profile, the header compression options available to you are documented in the Enabling the Options for ietf IPHC Profile Type Header Compression, page 72.

To enable TCP header compression set the number of TCP contexts, complete the following steps.

The IPHC profile must exist.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **iphc-profile** *profile-name*
4. **tcp**
5. **tcp contexts** {**absolute** *number-of-contexts* | **kbps-per-context** *kbps*}
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**iphc-profile** *profile-name* **Example:** `Router(config)# iphc-profile profile2`	Specifies the IPHC profile and enters IPHC-profile configuration mode. • Enter the IPHC profile name.
Step 4	**tcp** **Example:** `Router(config-iphcp)# tcp`	(Optional) Enables TCP header compression.
Step 5	**tcp contexts** {**absolute** *number-of-contexts* | **kbps-per-context** *kbps*} **Example:** `Router(config-iphcp)# tcp contexts absolute 25`	(Optional) Sets the number of TCP contexts. • Enter either the **absolute** keyword and the fixed number or the **kbps-per-context** keyword and the number of kbps to allow for each context.

	Command or Action	Purpose
Step 6	**end** **Example:** `Router(config-iphcp)# end`	(Optional) Exits IPHC-profile configuration mode.

- What to Do Next, page 72

What to Do Next

The next step is to attach the IPHC profile to an interface, a subinterface, or a Frame Relay PVC. For the instructions to follow, see the Attaching the IPHC Profile, page 75.

Enabling the Options for ietf IPHC Profile Type Header Compression

Note If you created a van-jacobson IPHC profile, complete the tasks in the Enabling the Options for van-jacobson IPHC Profile Type Header Compression, page 70.

If you created an ietf IPHC profile, you can enable or set a variety of header compression options. These options include enabling non-TCP header compression, enabling RTP header compression, and enabling ECRTP. For a list of the additional header compression features or settings available with an ietf IPHC profile, see the Enabling the Options for ietf IPHC Profile Type Header Compression, page 72.

The IPHC profile must exist.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **iphc-profile** *profile-name*
4. **non-tcp**
5. **non-tcp contexts** {**absolute** *number-of-contexts* | **kbps-per-context** *kbps* }
6. **rtp**
7. **recoverable-loss** {**dynamic** | *packet-drops*}
8. **refresh max-period** {*number-of-packets* | **infinite**}
9. **refresh max-time** {*length-of-time* | **infinite**}
10. **refresh rtp**
11. **feedback**
12. **maximum header** *max-header-size*
13. **tcp**
14. Router(config-iphcp)# tcp contexts absolute 75
15. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**iphc-profile** *profile-name* **Example:** `Router(config)# iphc-profile profile3`	Specifies the IPHC profile and enters IPHC-profile configuration mode. • Enter the IPHC profile name.
Step 4	**non-tcp** **Example:** `Router(config-iphcp)# non-tcp`	(Optional) Enables non-TCP header compression.
Step 5	**non-tcp contexts** {**absolute** *number-of-contexts* | **kbps-per-context** *kbps* } **Example:** `Router(config-iphcp)# non-tcp contexts absolute 75`	(Optional) Sets the number of contexts available for non-TCP header compression. • Enter either the **absolute** keyword and the fixed number or the **kbps-per-context** keyword and the number of kbps to allow for each context.
Step 6	**rtp** **Example:** `Router(config-iphcp)# rtp`	(Optional) Enables RTP header compression. **Note** This command automatically enables non-TCP header compression.
Step 7	**recoverable-loss** {**dynamic** | *packet-drops*} **Example:** `Router(config-iphcp)# recoverable-loss 5`	(Optional) Enables ECRTP.

	Command or Action	Purpose
Step 8	**refresh max-period** {*number-of-packets* | **infinite**} **Example:** `Router(config-iphcp)# refresh max-period 700`	(Optional) Sets the number of packets sent between full-header refresh occurrences. • Enter the number of packets sent between full-header refresh occurrences, or enter the **infinite** keyword to indicate no limitation on the number of packets sent between full-header refresh occurrences. **Note** Non-TCP header compression must be enabled first.
Step 9	**refresh max-time** {*length-of-time* | **infinite**} **Example:** `Router(config-iphcp)# refresh max-time infinite`	(Optional) Sets the amount of time to wait before a full-header refresh occurrence. • Enter the length of time, in seconds, to wait before a full-header refresh occurrence, or enter the **infinite** keyword to indicate no limitation on the time between full-header refreshes. **Note** Non-TCP header compression must be enabled first.
Step 10	**refresh rtp** **Example:** `Router(config-iphcp)# refresh rtp`	(Optional) Enables a context refresh for RTP header compression. **Note** RTP header compression must be enabled first.
Step 11	**feedback** **Example:** `Router(config-iphcp)# feedback`	(Optional) Disables the context-status feedback messages from the interface or link. **Note** TCP or non-TCP header compression must be enabled first.
Step 12	**maximum header** *max-header-size* **Example:** `Router(config-iphcp)# maximum header 75`	(Optional) Specifies the maximum size of the compressed IP header. • Enter the maximum size of the compressed IP header, in bytes. **Note** TCP or non-TCP header compression must be enabled first.
Step 13	**tcp** **Example:** `Router(config-iphcp)# tcp`	(Optional) Enables TCP header compression.
Step 14	Router(config-iphcp)# tcp contexts absolute 75	(Optional) Sets the number of contexts available for TCP header compression. • Enter either the **absolute** keyword and the fixed number or the **kbps-per-context** keyword and the number of kbps to allow for each context.

	Command or Action	Purpose
Step 15	**end** **Example:** `Router(config-iphcp)# end`	(Optional) Exits IPHC-profile configuration mode.

Attaching the IPHC Profile

You can attach the IPHC profile (either an ietf IPHC profile or a van-jacobson IPHC profile) to an interface, a subinterface, or a Frame Relay PVC.

Choose one of the following:

- Attaching an IPHC Profile to an Interface, page 75
- Attaching an IPHC Profile to a Frame Relay PVC, page 76

Attaching an IPHC Profile to an Interface

To attach an IPHC profile to an interface or subinterface, complete the following steps.

- The IPHC profile must exist.
- IP must be enabled on the interface or subinterface.
- The type of encapsulation in use on the interface or subinterface must support header compression. Two types of encapsulation that typically support header compression are PPP and HDLC encapsulation.
- Header compression must not already be enabled.
- The interface or subinterface must have sufficient memory.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **iphc-profile** *profile-name*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface fastethernet0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 4	**iphc-profile** *profile-name* **Example:** `Router(config-if)# iphc-profile profile1`	Attaches the IPHC profile to the interface. • Enter the IPHC profile to be attached to the interface specified in Attaching an IPHC Profile to an Interface, page 75.
Step 5	**end** **Example:** `Router(config-if)# end`	(Optional) Exits IPHC-profile configuration mode.

Attaching an IPHC Profile to a Frame Relay PVC

To attach an IPHC profile to a Frame Relay PVC, complete the following steps.

- The IPHC profile must exist.
- On a network that is using Frame Relay encapsulation, IPHC profiles are supported only in the Frame Relay map-class infrastructure.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **map-class frame-relay** *map-class-name*
4. **frame-relay iphc-profile** *profile-name*
5. **exit**
6. **interface** *type number* [*name-tag*]
7. **encapsulation frame-relay**
8. **ip address** *ip-address mask*
9. **frame-relay interface-dlci** *dlci*
10. **class** *name*
11. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**map-class frame-relay** *map-class-name* **Example:** `Router(config)# map-class frame-relay mapclass1`	Creates a map class and enters static map class configuration mode. • Enter the Frame Relay map class name.
Step 4	**frame-relay iphc-profile** *profile-name* **Example:** `Router(config-map-class)# frame-relay iphc-profile profile2`	Attaches the IPHC profile to the Frame Relay map class. • Enter the IPHC profile to be attached to the Frame Relay map class created in Attaching an IPHC Profile to a Frame Relay PVC, page 76.
Step 5	**exit** **Example:** `Router(config-map-class)# exit`	Exits static map class configuration mode.
Step 6	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial2/0`	Configures an interface type and enters interface configuration mode. • Enter the interface type and the interface number.
Step 7	**encapsulation frame-relay** **Example:** `Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation on the interface.

	Command or Action	Purpose
Step 8	**ip address** *ip-address mask* **Example:** `Router(config-if)# ip address 209.165.200.225 255.255.255.224`	Sets a primary IP address for an interface. • Enter the IP address and mask for the associated IP subnet.
Step 9	**frame-relay interface-dlci** *dlci* **Example:** `Router(config-if)# frame-relay interface-dlci 100`	Assigns a data-link connection identifier (DLCI) to a specified Frame Relay interface on the router or access server and enters Frame Relay DLCI configuration mode. • Enter the DLCI number to be used on the specified interface.
Step 10	**class** *name* **Example:** `Router(config-fr-dlci)# class mapclass1`	Associates a map class with a specified DLCI. • Enter the name of the map class to associate with the specified DLCI.
Step 11	**end** **Example:** `Router(config-fr-dlci)# end`	(Optional) Exits Frame Relay DLCI configuration mode.

Displaying the IPHC Profile Statistics

In this task, you can display statistical information about the IPHC profiles that you have created and configured. Displaying the IPHC profile statistics allows you to confirm that the IPHC profile is configured as you intended.

Information reported includes the IPHC profile name and profile type, the type of header compression enabled, whether any optional header compression features (such as the number of contexts) are enabled, and the name of the interface to which the IPHC profile is attached (if applicable).

To display the IPHC profile statistics, complete the following steps.

The IPHC profile must exist.

SUMMARY STEPS

1. **enable**
2. **show iphc-profile** [*profile-name*]
3. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show iphc-profile** [*profile-name*] **Example:** `Router# show iphc-profile profile1`	Displays configuration information for one or more IPHC profiles. • (Optional) Enter the name of the IPHC profile you want to display. If you do not specify an IPHC profile name, all IPHC profiles are displayed.
Step 3	**end** **Example:** `Router# end`	(Optional) Exits privileged EXEC mode.

Configuration Examples for Using IPHC Profiles

Example Creating an IPHC Profile

In the following example, a van-jacobson IPHC profile called profile1 has been created.

```
Router> enable
Router# configure terminal
Router(config)# iphc-profile profile1 van-jacobson
Router(config-iphcp)# end
```

In the following example, an ietf IPHC profile called profile2 has been created.

```
Router> enable

Router# configure terminal

Router(config)# iphc-profile profile2 ietf

Router(config-iphcp)# end
```

Example Enabling TCP Header Compression

In the following example, TCP header compression has been enabled in a van-jacobson IPHC profile called profile1. Additionally, the number of TCP contexts has been set to 25.

```
Router> enable

Router# configure terminal

Router(config)# iphc-profile profile1

Router(config-iphcp)# tcp

Router(config-iphcp)# tcp contexts absolute 25

Router(config-iphcp)# end
```

Example Enabling Non-TCP Header Compression

In the following example, RTP header compression has been enabled in an ietf IPHC profile called profile2. Additionally, ECRTP has been enabled with the **recoverable-loss** command, and the size of the compressed IP header has been set to 75 bytes.

```
Router> enable

Router# configure terminal

Router(config)# iphc-profile profile2

Router(config-iphcp)# rtp

Router(config-iphcp)# recoverable-loss 5

Router(config-iphcp)# maximum header 75

Router(config-iphcp)# end
```

Example Attaching the IPHC Profile

In the following example, an IPHC profile called profile1 is attached to serial interface 0.

```
Router> enable

Router# configure terminal

Router(config)# interface serial0

Router(config-if)# iphc-profile profile1

Router(config-if)# end
```

In the following example, an IPHC profile called profile2 is attached to a Frame Relay map class called mapclass1.

```
Router> enable

Router# configure terminal

Router(config)# map-class frame-relay mapclass1

Router(config-map-class)# frame-relay iphc-profile profile2

Router(config-map-class)# exit

Router(config)# interface serial2/0

Router(config-if)# encapsulation frame-relay

Router(config-if)# ip address 209.165.200.225 255.255.255.224

Router(config-if)# frame-relay interface-dlci 100

Router(config-fr-dlci)# class mapclass1

Router(config-fr-dlci)# end
```

Example Reporting IPHC Profile Statistics

The following is sample output from the **show iphc-profile** command. In this output, information about two IPHC profiles, profile21 and 20, is displayed.

```
Router# show iphc-profile
IPHC Profile "profile21"
Type: VJ
  Compressing: TCP
  Contexts   : TCP fixed at 150
  Controlled interfaces: (1)
    Se3/1
IPHC Profile "profile20"
```

```
Type: IETF
  Compressing: TCP NON-TCP (RTP)
  Contexts   : TCP 1 for each 0 kbits NON-TCP 1 for each 0 kbits
  Refresh    : NON-TCP and RTP every 5 seconds or 256 packets
  Controlled interfaces: (1)
    Se3/0
```

Additional References

The following sections provide references related to IPHC profiles.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Quality of Service Solutions Command Reference*
QoS functionality overview	"Quality of Service Overview" module
Header compression overview	"Header Compression" module
RTP header compression	"Configuring RTP Header Compression" module
TCP header compression	"Configuring TCP Header Compression" module
Class-based RTP and TCP header compression	"Configuring Class-Based RTP and TCP Header Compression" module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1144	*Compressing TCP/IP Headers for Low-Speed Serial Links*
RFC 2507	*IP Header Compression*
RFC 2508	*Compressing IP/UDP/RTP Headers for Low-Speed Serial Links*
RFC 3544	*IP Header Compression over PPP*
RFC 3545	*Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Configuring Header Compression Using IPHC Profiles

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 6 Feature Information for Configuring Header Compression Using IPHC Profiles

Feature Name	Releases	Feature Information
IPHC Profiles	12.4(9)T	The IPHC Profiles feature allows you to configure header compression in a kind of template ("profile") and to apply the profile to interfaces, subinterfaces, or Frame Relay PVCs. The following commands were introduced by this feature: **feedback**, **iphc-profile**, **maximum header**, **non-tcp**, **non-tcp contexts**, **recoverable-loss**, **refresh max-period**, **refresh max-time**, **refresh rtp**, **rtp**, **show iphc-profile**, **tcp**, **tcp contexts**.