

# Microsoft Point-to-Point Compression (MPPC)

---

## Feature Summary

Microsoft Point-to-Point Compression (MPPC) is a scheme used to compress Point-to-Point Protocol (PPP) packets between Cisco and Microsoft client devices. The MPPC algorithm is designed to optimize bandwidth utilization in order to support multiple simultaneous connections. The MPPC algorithm uses a Lempel-Ziv (LZ) based algorithm with a continuous history buffer, called a dictionary.

The Compression Control Protocol (CCP) configuration option for MPPC is 18.

Exactly one MPPC datagram is encapsulated in the PPP information field. The PPP protocol field indicates the hexadecimal type of 00FD for all compressed datagrams. The maximum length of the MPPC datagram transmitted over PPP is the same as the MTU of the PPP interface; however this length cannot be greater than 8192 bytes since the history buffer is limited to 8192 bytes. If compressing the data results in data expansion, the original data is sent as an uncompressed MPPC packet.

The history buffers between compressor and decompressor are synchronized by maintaining 12 bit coherency count. If the decompressor detects that the coherency count is out of sequence, then the error recovery process begins and is described through the following steps:

- 1 Reset Request (RR) packet sent from the decompressor.
- 2 The compressor then flushes the history buffer and sets the flushed bit in the next packet it sends.
- 3 Upon receiving the flushed bit set packet, the decompressor flushes the history buffer.

Synchronization is achieved without CCP using the Reset Acknowledge (RA) packet that can be additionally time-consuming.

Compression negotiation between a router and a Windows 95 client occurs through the following steps:

- 1 Windows 95 sends a request for both STAC (option 17) and MPPC (option 18) compression.
- 2 The router sends a negative acknowledgement (NAK) requesting only MPPC.
- 3 Windows 95 resends the request for MPPC.
- 4 The router sends an acknowledgment (ACK) confirming MPPC compression negotiation.

## Benefits

MPPC provides the following benefits:

- Supports point-to-point connections for Microsoft clients running Windows NT and Windows 95.
- Optimizes bandwidth utilization.
- Operates under Multilink PPP (MLP) and Virtual Private Dial-Up Networks (VPDN). CCP runs under the bundled MLP interfaces. Compression takes place before the packets are handed off to the MLP layer in the transmitter side, and decompression occurs after the MLP layer distributes received packets to each interface in the bundled MLP group; therefore, MPPC is transparent to the MLP layer.

---

**Note** Windows NT only supports MPPC compression. Windows 95 supports both MPPC and STAC compression methods.

---

## List of Terms

**acknowledgment (ACK)**—A notification sent from one network device to another to acknowledge that some event (receipt of a message, for example) has occurred. Usually abbreviated as ACK.

**Compression Control Protocol (CCP)**—A protocol that provides a defined method of negotiating data compression over the Point-to-Point Protocol (PPP).

**Lempel-Ziv algorithm (LZ)**—A compression algorithm, based on a dynamically encoded dictionary, that replaces a continuous stream of characters with codes. The symbols represented by the codes are stored in memory in a dictionary-style list. LZ-style compression is used in a number of compression algorithms, such as Stacker (STAC) compression. In contrast, statistical compression methods use a fixed and, oftentimes, a non-adaptive encoding method, which is best used where data is consistent and predictable.

**Link Control Protocol (LCP)**—A protocol that establishes, configures, and tests data link connections used by PPP.

**Multilink PPP (MLP)**—A protocol that allows packets to be fragmented, whereby the fragments can be simultaneously sent over multiple point-to-point links to the same remote address. MLP provides bandwidth on demand and reduces transmission latency across WAN links.

**negative acknowledgment (NAK)**—A response sent from a receiving device to a sending device indicating that the information received contained errors. Typically abbreviated as NAK.

**Point-to-Point Protocol (PPP)**—A protocol that encapsulates network layer protocol information over point-to-point links. The RFC for PPP is RFC 1661.

**Stacker compression (STAC)**—A data compression algorithm developed by STAC Electronics that is based on the LZ compression algorithm. Although effective, the STAC compression algorithm uses more CPU resources to perform compression. Cisco IOS software uses an optimized version of STAC compression.

**Virtual Private Dial-Up Network (VPDN)**—A networking service that allows separate and autonomous protocol domains to share common access infrastructure including modems, access servers, and ISDN routers. VPDN uses the Layer 2 Forwarding protocol (L2F) which permits the tunneling of link level frames.

## Restrictions

The following restrictions apply to the MPPC feature:

- MPPC is only supported with PPP encapsulation.
- Compression can be processor intensive because it requires a reserved block of memory to maintain the history buffer. Do not enable modem or hardware compression because it may cause performance degradation, compression failure or data expansion.
- Both ends of the point-to-point link must be using the same compression method (STAC, Predictor or MPPC, for example).

## Platforms

This feature is supported on these platforms:

- Cisco 1000 series
- Cisco 1600 series
- Cisco 2500 series
- Cisco 3600 series
- Cisco 4000 series (Cisco 4000, 4000-M, 4500, 4500-M, 4700, 4700-M)
- Cisco 5200
- Cisco 5300
- Cisco 7200 series
- Cisco 7500 series

## Prerequisites

PPP encapsulation must be enabled. For information on how to configure PPP encapsulation refer to the Cisco IOS Release 11.3 *Dial Solutions Configuration Guide*.

## Supported MIBs and RFCs

This feature supports the following RFC:

- RFC 2118

No MIBs are supported by this feature.

## Configuration Task

Once the router is configured for PPP encapsulation, you can configure the interface to perform MPPC as described in the following section.

## Configure the Interface to Perform MPPC

There is only one command required to configure MPPC. The existing **compress** command now supports the **mppc** keyword, which prepares the interface to initiate CCP and negotiates MPPC with the Microsoft client. To configure MPPC on the interface, perform the following task in interface configuration mode:

Task	Command
Enable MPPC on the interface.	<b>compress [mppc [ignore-pfc]]</b>

The **ignore-pfc** keyword instructs the router to ignore the protocol field compression flag negotiated by LCP. For example, the uncompressed standard protocol field value for IP is 0x0021 and 0x21 when compression is enabled. When the **ignore-pfc** option is enabled, the router will continue to use the uncompressed value (0x0021). Using the **ignore-pfc** option is helpful for some asynchronous driver devices which use an uncompressed protocol field (0x0021), even though the pfc is negotiated between peers. Figure 1 displays protocol rejections when the **debug ppp negotiation** command is enabled. These errors can be remedied by setting the **ignore-pfc** option.

**Figure 1 Sample Debug PPP Negotiation Showing Protocol Reject**

```

PPP Async2: protocol reject received for protocol = 0x2145
PPP Async2: protocol reject received for protocol = 0x2145
PPP Async2: protocol reject received for protocol = 0x2145
    
```

## Configuration Examples

This section provides the following examples:

- MPPC BRI Interface Configuration Example
- MPPC Asynchronous Interface Configuration Example
- MPPC Serial and Virtual Template Interface Configuration Example

### MPPC BRI Interface Configuration Example

The following example configures BRI interface 0 to perform MPPC:

```

interface BRI0
 ip unnumbered ethernet0
 encapsulation ppp
 isdn spid1 5551234
 dialer map ip 172.21.71.74 5551234
 dialer-group 1
 compress mppc
    
```

## MPPC Asynchronous Interface Configuration Example

The following example configures asynchronous interface 1 to implement MPPC and ignore the protocol field compression flag negotiated by LCP:

```
interface async1
  ip unnumbered ethernet0
  encapsulation ppp
  async default routing
  async dynamic routing
  async mode interactive
  peer default ip address 172.21.71.74
  compress mppc ignore-pfc
```

## MPPC Serial and Virtual Template Interface Configuration Example

The following example creates a virtual access interface (virtual-template interface 1) and serial interface 0, which is configured for X.25 encapsulation. MPPC values are configured on the virtual-template interface and will ignore the negotiated protocol field compression flag.

```
interface ethernet0
  ip address 172.20.30.102 255.255.255.0
  !
interface virtual-template1
  ip unnumbered ethernet0
  peer default ip address pool vtemp1
  compress mppc ignore-pfc
  !
interface serial0
  no ipaddress
  no ip mroute-cache
  encapsulation x25
  x25 win 7
  x25 winout 7
  x25 ips 512
  x25 ops 512
  clock rate 50000
  !
ip local pool vtemp1 172.20.30.103 172.20.30.104
ip route 0.0.0.0 0.0.0.0 172.20.30.1
  !
translate x25 31320000000000 virtual-template 1
```

## Command Reference

This section documents the modified **compress** command. The **compress** command is revised to add the **mppc** keyword that supports compression for Microsoft clients, such as Windows 95 and Windows NT. Only the information that applies to the **mppc** keyword is provided. Refer to the Cisco IOS Release 11.3 *Dial Solutions Command Reference* for other keywords available with the **compress** command.

### compress mppc

To configure software compression for PPP encapsulation, use the **compress** interface configuration command. To disable compression on the interface, use the **no** form of this command.

**compress** [**mppc** [**ignore-pfc**]]  
**no compress** [**mppc** [**ignore-pfc**]]

#### Syntax Description

<b>mppc</b>	Specifies that the MPPC compression algorithm will be used.
<b>ignore-pfc</b>	(Optional) Specifies that the protocol field compression flag negotiated through LCP will be ignored.

#### Default

PPP compression is disabled.

#### Command Mode

Interface configuration

#### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3 T.

The **compress mppc** command is used for point-to-point connections between Cisco and Microsoft clients.

PPP encapsulation supports predictor, STAC LZS-DCP, and MPPC compression algorithms.

Compression is a means of optimizing traffic by running a data set through specific algorithms that either reduce space or bandwidth required to transmit the data set. Some algorithms reduce packet header size while others condense data within packets.

MPPC implements an LZ based compression algorithm that uses a compression dictionary to compress PPP packets connecting to Microsoft clients.

Compression is performed in software and may significantly affect system performance. We recommend that you disable compression if CPU load exceeds 65 percent. To display the CPU load, use the **show process cpu EXEC** command.

Both end-point devices must be configured to use the same compression method (predictor, Stacker or MPPC).

If the majority of your traffic is already compressed files, we recommend that you not use compression. If the files are already compressed, the additional processing time spent in attempting unsuccessfully to compress them again will slow system performance.

### Example

The following example configures asynchronous interface 0 to perform MPPC compression and ignore the protocol field compression flag:

```
interface async0
  encapsulation ppp
  compress mppc ignore-pfc
```

### Related Commands

**show compress**  
**show process cpu**

## Debug Commands

The **debug ppp** command is revised to add the **mppc** keyword that supports compression for Microsoft clients such, as Windows 95 and Windows NT. Only the information that applies to **debug ppp mppc** is provided. Refer to the Cisco IOS Release 11.3 *Debug Command Reference* for complete **debug ppp** information.

## debug ppp

Use the **debug ppp** EXEC command with the **mppc** keyword to display information specific to the exchange of PPP connections using MPPC. To disable debugging output, use the **no** form of this command.

**debug ppp mppc**  
**[no] debug ppp mppc**

### Syntax Description

**mppc** Causes the **debug ppp** command to display decompressed MPPC header compression information for incoming packets.

### Usage Guidelines

Use this command to debug PPP connections on point-to-point links using MPPC.

This command is useful for obtaining incorrect packet sequence number information where MPPC compression is enabled.

The **debug ppp** command currently exists; only the **mppc** keyword is added to support MPPC.

Refer to the Cisco IOS Release 11.3 *Debug Command Reference* for other keywords available with the **debug ppp** command.



**Caution** The **debug ppp mppc** command is CPU intensive and should be used with caution. This command should be disabled immediately after debugging.

### Sample Display

Figure 1 shows a typical PPP packet exchange between the router and Microsoft client where the MPPC header sequence numbers increment correctly. Sample Debug PPP MPPC Output

```
Router# debug ppp mppc

MPPC Decomp. :mppc_hdr = 0x2027,
               hdr->coherency_count = 39, coherency_count = 39
MPPC Decomp. :mppc_hdr = 0x2028,
               hdr->coherency_count = 40, coherency_count = 40
MPPC Decomp. :mppc_hdr = 0x2029,
               hdr->coherency_count = 41, coherency_count = 41
```

Table 1 describes the fields for the debug ppp mppc output.1

**Table 1** Debug PPP MPPC Fields

Field	Description
MPPC Decomp.	Debugging decompression information
mppc_hdr = 0x2027	Bit setting and coherency count
hdr->coherency_count = 39	Received sequence number
coherency_count = 39	Expected sequence number

Figure 2 shows the output from **debug ppp negotiation** and **debug ppp error** commands which can be used to troubleshoot initial ppp negotiation and setup errors. This example shows a virtual interface (virtual interface 1) during normal PPP operation and CCP negotiation. Refer to the Cisco IOS Release 11.3 *Debug Command Reference* for details and field descriptions about these debug commands.

**Figure 2 Sample Debug PPP Negotiation and Debug PPP Errors Output**

```

Router# debug ppp nego error
Vt1 PPP: Unsupported or un-negotiated protocol. Link arp
VPDN: Chap authentication succeeded for p5200
Vt1 PPP: Phase is DOWN, Setup
Vt1 VPDN: Virtual interface created for dinesh@cisco.com
Vt1 VPDN: Set to Async interface
Vt1 PPP: Phase is DOWN, Setup
Vt1 VPDN: Clone from Vtemplate 1 filterPPP=0 blocking
Vt1 CCP: Re-Syncing history using legacy method
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
Vt1 PPP: Treating connection as a dedicated line
Vt1 PPP: Phase is ESTABLISHING, Active Open
Vt1 LCP: O CONFREQ [Closed] id 1 len 25
Vt1 LCP:   ACCM 0x000A0000 (0x0206000A0000)
Vt1 LCP:   AuthProto CHAP (0x0305C22305)
Vt1 LCP:   MagicNumber 0x000FB69F (0x0506000FB69F)
Vt1 LCP:   PFC (0x0702)
Vt1 LCP:   ACFC (0x0802)
Vt1 VPDN: Bind interface direction=2
Vt1 PPP: Treating connection as a dedicated line
Vt1 LCP: I FORCED CONFREQ len 21
Vt1 LCP:   ACCM 0x000A0000 (0x0206000A0000)
Vt1 LCP:   AuthProto CHAP (0x0305C22305)
Vt1 LCP:   MagicNumber 0x12A5E4B5 (0x050612A5E4B5)
Vt1 LCP:   PFC (0x0702)
Vt1 LCP:   ACFC (0x0802)
Vt1 VPDN: PPP LCP accepted sent & rcv CONFACK
Vt1 PPP: Phase is AUTHENTICATING, by this end
Vt1 CHAP: O CHALLENGE id 1 len 27 from "l_4000"
Vt1 CHAP: I RESPONSE id 20 len 37 from "dinesh@cisco.com"
Vt1 CHAP: O SUCCESS id 20 len 4
Vt1 PPP: Phase is UP
Vt1 IPCP: O CONFREQ [Closed] id 1 len 10
Vt1 IPCP:   Address 15.2.2.3 (0x03060F020203)
Vt1 CCP: O CONFREQ [Not negotiated] id 1 len 10
Vt1 CCP:   MS-PPC supported bits 0x00000001 (0x120600000001)
Vt1 IPCP: I CONFREQ [REQsent] id 1 len 34
Vt1 IPCP:   Address 0.0.0.0 (0x030600000000)
Vt1 IPCP:   PrimaryDNS 0.0.0.0 (0x810600000000)
Vt1 IPCP:   PrimaryWINS 0.0.0.0 (0x820600000000)
Vt1 IPCP:   SecondaryDNS 0.0.0.0 (0x830600000000)
Vt1 IPCP:   SecondaryWINS 0.0.0.0 (0x840600000000)
Vt1 IPCP: Using the default pool
Vt1 IPCP: Pool returned 11.2.2.5
Vt1 IPCP: O CONFREQ [REQsent] id 1 len 16
Vt1 IPCP:   PrimaryWINS 0.0.0.0 (0x820600000000)
Vt1 IPCP:   SecondaryWINS 0.0.0.0 (0x840600000000)
Vt1 CCP: I CONFREQ [REQsent] id 1 len 15
Vt1 CCP:   MS-PPC supported bits 0x00000001 (0x120600000001)
Vt1 CCP:   Stacker history 1 check mode EXTENDED (0x1105000104)
Vt1 CCP: Already accepted another CCP option, rejecting this STACKER
Vt1 CCP: O CONFREQ [REQsent] id 1 len 9
Vt1 CCP:   Stacker history 1 check mode EXTENDED (0x1105000104)
Vt1 IPCP: I CONFACK [REQsent] id 1 len 10

```

## Debug Commands

---

```
Vi1 IPCP:   Address 15.2.2.3 (0x03060F020203)
Vi1 CCP: I CONFACK [REQsent] id 1 len 10
Vi1 CCP:   MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: I CONFREQ [ACKrcvd] id 2 len 10
Vi1 CCP:   MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: O CONFACK [ACKrcvd] id 2 len 10
Vi1 CCP:   MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: State is Open
Vi1 IPCP: I CONFREQ [ACKrcvd] id 2 len 22
Vi1 IPCP:   Address 0.0.0.0 (0x030600000000)
Vi1 IPCP:   PrimaryDNS 0.0.0.0 (0x810600000000)
Vi1 IPCP:   SecondaryDNS 0.0.0.0 (0x830600000000)
Vi1 IPCP: O CONFNAK [ACKrcvd] id 2 len 22
Vi1 IPCP:   Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP:   PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP:   SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: I CONFREQ [ACKrcvd] id 3 len 22
Vi1 IPCP:   Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP:   PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP:   SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: O CONFACK [ACKrcvd] id 3 len 22
Vi1 IPCP:   Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP:   PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP:   SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: State is Open
Vi1 IPCP: Install route to 11.2.2.5
```