Configuring MPLS RSVP TE

This chapter describes how to configure Multiprotocol Label Switching (MPLS) Resource Reservation Protocol (RSVP) Traffic Engineering (TE) on Cisco NX-OS devices.

This chapter includes the following sections:

Finding Feature Information

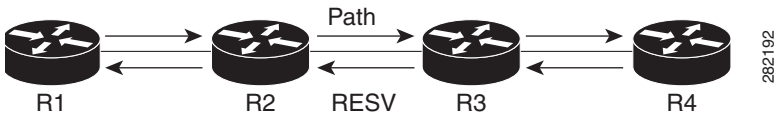
Your software release might not support all the features documented in this module. For the latest caveats and feature information, see the Bug Search Tool at https://tools.cisco.com/bugsearch/ and the release notes for your software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "New and Changed Information" chapter or the Feature History table below.

Information About MPLS RSVP TE

RSVP is a signaling protocol that reserves resources, such as for IP unicast and multicast flows, and requests quality-of-service (QoS) parameters for applications. The protocol was extended in MPLS RSVP TE to enable RSVP to set up label switched paths (LSPs) that can be used for TE in MPLS networks.

Figure 13-1 shows how RSVP sets up an LSP from router R1 through router R4 that can be used for TE in an MPLS environment.

Figure 13-1 Example of RSVP Used to Set Up an MPLS LSP



The LSP setup is driven by the TE application on the headend router R1 and is identified as a session that specifies the tailend router for the LSP (R4), a tunnel identifier, and an extended tunnel identifier, which is typically the local address of R1.

The headend RSVP component signals a PATH message destined toward R4. The PATH message can include policy link-admission control information, which identifies the sender that is setting up the path, and a flow specification that defines the resources desired on the path.

Each hop along the path examines the PATH message, verifies the policy control information, saves the path state that is associated with the session, and sets aside the requested resources specified by the sender. When the tailend router is reached, a hop-by-hop reservation (RESV) message is initiated by R4 toward R1, along the reverse direction taken by the PATH message.

At each node including the tailend, the session-state is updated, the earmarked resources are reserved for the session, and an MPLS label is allocated for use by the prior hop. When the RESV reaches the headend router, the LSP setup for the session is complete.

The reservation state in each router is considered as a soft state, which means that periodic PATH and RESV messages must be sent at each hop to maintain the state. If there is a failure to establish or maintain a session at any hop, RSVP provides messages to propagate the error along the path, and to tear down the existing reservation.

Overview

RSVP interacts with TE to support the MPLS TE functionality.

The TE process contains the following functionalities:

- End-point control, which is associated with establishing and managing TE tunnels at the headend and tailend.
- Link-management, which manages link resources to do resource-aware routing of TE LSPs and to program MPLS labels.
- Fast Reroute (FRR), which manages the LSPs that need protection and to assign backup tunnel information to these LSPs.

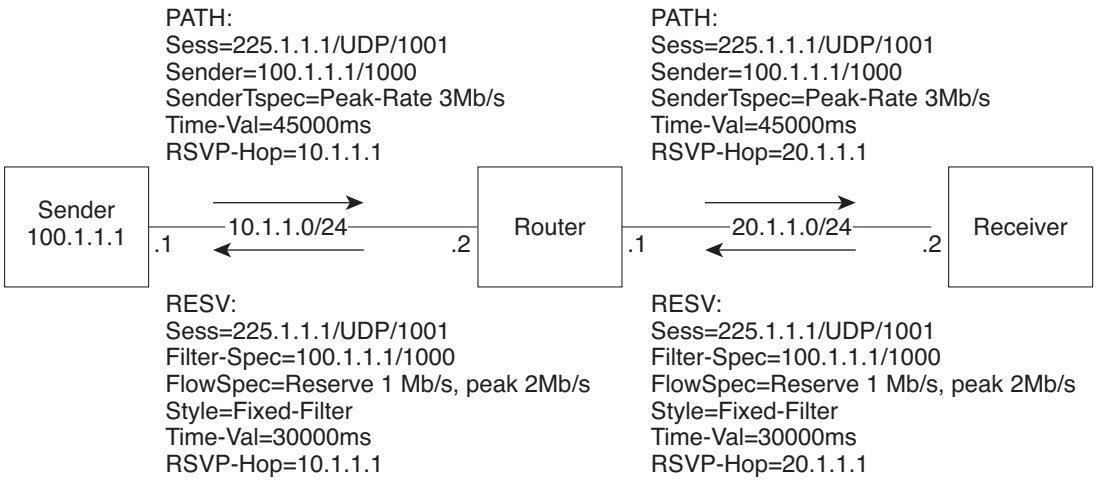
Note The interactions between TE and RSVP assume the existence of the end-point control, link-management, and FRR functionality within TE.

RSVP Core Functionality

The RSVP core functionality specifies RSVP messages and the objects required to set up resource reservations for IP unicast and multicast flows.

The primary RSVP messages are PATH and RESV messages. Senders send a PATH message from the source to the receiver to specify the reservation requirements of a data flow. Receivers send a RESV message to the sender to reserve resources for the flow. The primary object is the Session object, which identifies the data flow via the destination address, IP protocol ID, and destination port. Other key objects include the Sender-Template and Sender-Tspec, which the switch uses to qualify the sender and traffic specification in the PATH message, and the Filter-Spec, FlowSpec, and Style, which the switch uses in the RESV message to further classify the flow, specify its resource requirements, and designate the reservation style. An example that shows a resource reservation for the {225.1.1.1, UDP, 1001} data flow through the PATH and RESV messages and their objects is shown in Figure 13-2. The figure also shows the Time-Val and RSVP-Hop objects that track the PATH refresh and previous hop in the PATH message and the RESV refresh and next hop in the RESV message.

Figure 13-2 RSVP Reservation via PATH and RESV Messages



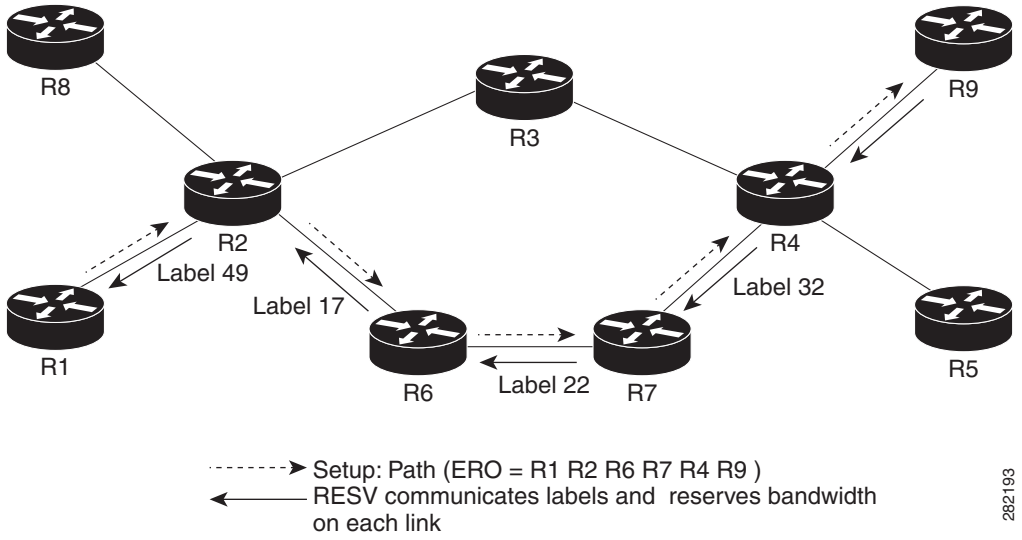
In addition to the PATH and RESV messages, RSVP also specifies support messages that include PATH-ERROR, RESV-ERROR, PATH-TEAR, RESV-TEAR, and RESV-CONFIRM messages that are used to handle error situations, to tear down existing reservations, and to confirm the setup of an existing reservation. Other message objects include an Integrity object that provides integrity protection to RSVP messages, a Policy-Data object that identifies the sender and receiver credentials, a Scope object that carries an explicit list of senders to which a RESV is to be sent, an Error-Spec object that provides error information, a Resv-Confirm object that identifies the receiver interested in the confirmation message, and an Adspec object that carries flow advertisement information.

RSVP TE (RFC 3209, 5151)

RSVP TE builds on the RSVP core protocol, defines new objects, and modifies existing objects used in the PATH and RESV objects for LSP establishment. The base *Session* construct for RSVP TE is based on the triple {Tunnel Remote Address, Tunnel ID, Extended Tunnel ID}. The *Sender Template* object contains the {IPv4 tunnel sender address, LSP-ID}. The PATH message was extended to contain a *Label-Request* object (LRO) that results in a label being assigned during the RESV, a *Session-Attribute* object that is used to provide additional requirements for a session, and an *Explicit-Route* object (ERO) that specifies the data path traversed by the PATH message, which could be independent of IP-routing. The RESV message was modified to include a *Label* object that contains the MPLS label and a *Record-Route* object (RRO) to record the path taken followed by the RESV message. The *Flowspec* object was also modified to set up a reservation on LSPs.

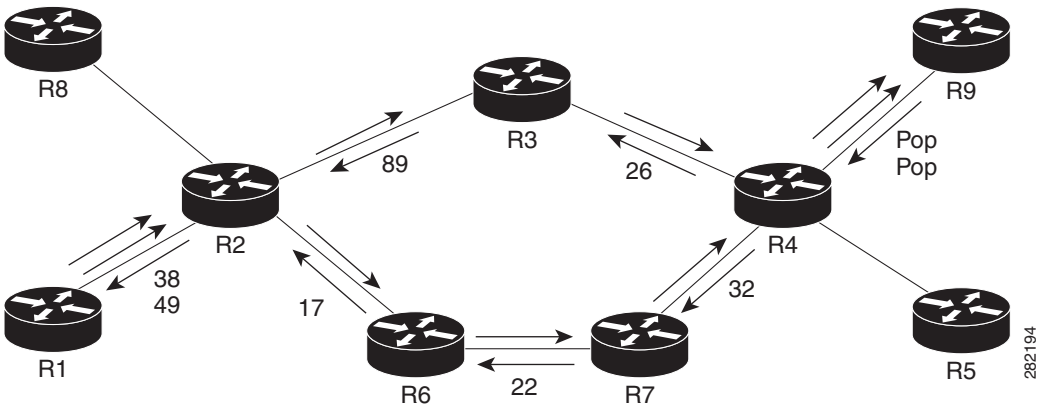
An example of an LSP path setup that uses procedures described in RSVP TE is shown in Figure 13-3. The figure shows a PATH message that is sent from router R1 to Router R9 with an *ERO* object of R1-R2-R6-R7-R4-R9. The PATH message contains a *LRO* object in the message. The RESV message shows labels that are assigned in the reverse direction of the PATH message, which is done via the *Label* object. The LSP setup is driven by the TE process on the router.

Figure 13-3 MPLS LSP Setup Using RSVP TE



For RSVP TE path reoptimization, the switch must reroute an LSP to a new data path before destroying the existing LSP, which is known as a make-before-break. Path reoptimization, as shown in Figure 13-4, is achieved when the switch sends two PATH messages with the same session ID but with different sender templates. When the receiver receives this PATH message, it recognizes that make-before-break is in progress and sends a RESV to reserve resources using the shared-explicit reservation *Style* object, which allows for sharing the resource requirements of the two paths before the original path is torn down.

Figure 13-4 TE Path Reoptimization Using RSVP



RSVP TE Explicit Routing (Strict, Loose)

RSVP TE explicit routes are particular paths in the network topology that you can specify as abstract nodes, which could be a sequence of IP prefixes or a sequence of autonomous systems, in the ERO. The explicit path could be administratively specified, or automatically computed using an algorithm such as constrained shortest path first (CSPF).

The explicit path specified in the ERO may be a strict path or a loose path.

A strict path means that a network node and its preceding node in the ERO must be adjacent and directly connected.

A loose hop means that a network node specified in the ERO must be in the path but is not required to be directly connected to its preceding node. If a loose hop is encountered during ERO processing, the node that processes the loose hop can update the ERO with one or more nodes along the path from itself to the next node in the ERO. The advantage of a loose path is that the entire path does not need to be specified or known when creating the ERO. The disadvantage of a loose path is that it can result in forwarding loops during transients in the underlying routing protocol.

RSVP Hello

The RSVP hello functionality was introduced in RFC 3209 to monitor communication failures with a neighbor. A hello message can contain a *Hello-Request* object or a *Hello-Ack* object. The hello message is periodic and unidirectional, so that each neighbor can issue a Hello-Request with a period that is independent of its neighbor, and a Hello-Ack must also be sent in response to the Hello-Request. The Hello-Request includes a source-instance that is echoed by the neighbor in its Hello-Ack. The Hello-Request also includes a destination-instance that echoes the destination-instance value used by the neighbor in its Hello-Ack. Instance values are fixed during a session and must be changed when communication breaks down, which enables a router to identify a communication failure with its neighbor, apart from a hello timeout.

RSVP Fast Reroute

When a router's link or neighboring node fails, the router often detects this failure by receiving an interface-down notification. When a router notices that an interface has gone down, it switches LSPs going out that interface onto their respective backup tunnels (if any).

RSVP establishes backup LSP-based tunnels for the local repair of TE LSPs. RSVP uses the facility backup method in which a PLR creates one or more bypass tunnels that can be used to protect multiple LSPs.

The Fast-Reroute object is used in the PATH message and contains a flag that identifies the backup method to be used as facility-backup. The Fast-Reroute object specifies setup and hold priorities, which are included in a set of attribute filters and bandwidth requirements to be used in the selection of the backup path.

The Session-Attribute object signals in the PATH message that local protection is desired for an LSP, requires label recording when doing a record route, explicitly specifies a desire for node or bandwidth protection on an LSP, and specifies the use of a shared-explicit style of a reservation by the egress node.

The *RRO* object reports in the RESV message the availability or use of local protection on an LSP, and whether bandwidth and node protection are available for that LSP.

The signaling of the FRR requirements is initiated at the TE tunnel headend. PLRs along the path act on the FRR requirements based on the backup tunnel availability at the PLR, and signal the backup tunnel selection information to the headend. When an FRR event is triggered, the PLR sends PATH messages through the backup tunnel to the merge point (MP) where the backup tunnel rejoins the original LSP. The MP also sends RESV messages to the PLR using the RSVP-Hop object that is included by the PLR in its PATH message. This process prevents the original LSP from being torn down by the MP. Also, the PLR signals the tunnel headend with a PATH-ERROR message to indicate the failure along the LSP and that FRR is in active use for that LSP. This information is used by the headend to signal a new LSP for the TE tunnel, and to tear down the existing failed path after the new LSP is set up through make-before-break techniques.

You can detect FRR link failures by monitoring the interface states that are configured for the links or by using Bidirectional Forwarding Detection (BFD) to detect the health of the neighbor.

Refresh Reduction

RSVP requires that the path and reservation state that are set up during LSP signaling must be refreshed by periodically sending PATH refresh and RESV refresh messages. Refresh messages are used to synchronize the state between RSVP neighbors and to recover from lost RSVP messages. Periodic refresh signaling, however, can result in scaling issues and reliability and latency issues depending on the refresh period.

A refresh-reduction-capable-bit in the RSVP message common header is set by nodes to indicate support for refresh reduction capability. A Message-ID extension defines three new objects, a Message-ID object, a Message-ID-Ack object, and a Message-ID-Nack object. The Message-ID object contains a sender selected value, which when combined with the sender's IP address identifies the particular RSVP message and the state that it represents.

Reliable Messages

To support a reliable RSVP message exchange, a sender includes the Message-ID object in the RSVP message and sets an Ack-Desired flag in the Message-ID object to indicate that it wants an explicit acknowledgement of the message from the neighbor. The sender retransmits unacknowledged messages at a rate faster than the standard refresh period, until a retry limit is reached. The Message-ID-Ack object is used by a receiver to acknowledge receipt of the messages. The object can be sent in an explicit Ack message to the receiver, or it can be combined with another RSVP message if the message is ready to be sent to the sender.

To reduce the volume of the PATH and RESV refresh messages, RSVP-Refresh-Red defines a summary refresh (Srefresh) message that can be sent between RSVP neighbors. The Srefresh message contains a summary refresh extension that carries a list of Message-ID objects that correspond to PATH and RESV messages that originally established the PATH and RESV state. An RSVP node that receives an Srefresh message matches each listed Message-ID object with the installed PATH and RESV state and updates the state as if a normal RSVP refresh message has been received. If the matching state is not found, the Srefresh sender is notified through an Ack message or a combined Ack message, that contains a Message-ID-Nack object.

Message Authentication

RSVP enables particular users to obtain preferential access to network resources under the control of an admission control function that protects RSVP message integrity hop-by-hop by transmitting an authenticating digest of the message prepared using a shared secret authentication key, a sequence number, and a keyed-hash algorithm in the Integrity object of the RSVP message. This process allows the message receiver to identify playbacks and to thwart replay attacks. The scheme may also use an RSVP Integrity Challenge and response messages with a Challenge object to initialize the sequence number used between the sender and receiver The keyed-hash algorithms that are supported in this project are HMAC-MD5 and HMAC-SHA1.

Note For authentication to work properly, two connected switches must be configured with authentication. You must also synchronize the keychain (key-id/key-strings) configuration between the switches participating in the authenticated exchange.

The **authentication** [**neighbor address** *IP-address*] **key-chain** *key-chain-name* command is used to authenticate the neighbor.

Note You must configure keychain parameters before the **authentication** commands can take effect.

```
switch# configure terminal
switch(config)# key chain key1
switch(config-keychain)# key 4660
switch(config-keychain-key)# key-string qwertyui
===============Now configuring Auth for RSVP===========
switch(config)# ip rsvp
switch(config-ip-rsvp)# authentication key key1
switch(config-ip-rsvp)#

switch(config)# ip rsvp
switch(config-ip-rsvp)# authentication key-chain key1

switch(config)# feature mpls traffic-engineering
switch(config)# interface eth 2/1
switch(config-if)# ip rsvp authentication lifetime 20:30:30


[no] authentication [neighbor address <IP-address>] key-chain <key-chain-name>
[no] authentication [neighbor address <IP-address>] type {md5 | sha-1}
[no] authentication [neighbor address <IP-address>] lifetime <hh:mm:ss>
[no] authentication [neighbor address <IP-address>] window-size  <value>
[no] authentication [neighbor address <IP-address>] challenge

[no] ip rsvp authentication key-chain <key-chain-name>
[no] ip rsvp authentication type {md5 | sha-1}
[no] ip rsvp authentication lifetime <hh:mm:ss>
[no] ip rsvp authentication window-size  <value>
```

These sets of commands are completely independent and depend on the type of authentication required. For example, if you want to globally configure authentication for RSVP, you would use the following set of commands.

Note The optional **neighbor address** keyword and argument is not present.

```
switch(config)# ip rsvp
switch(config-ip-rsvp)# authentication key-chain key-chain-name
switch(config-ip-rsvp)# authentication type {md5 | sha-1}
switch(config-ip-rsvp)# authentication lifetime <hh:mm:ss>
switch(config-ip-rsvp)# authentication window-size  <value>
switch(config-ip-rsvp)# authentication challenge
```

```
[no] authentication key-chain <key-chain-name>
[no] authentication type {md5 | sha-1}
[no] authentication lifetime <hh:mm:ss>
[no] authentication window-size  <value>
[no] authentication challenge
```

If a per-interface authentication is needed, that is, if all RSVP neighbors on the other side of the interface are to be authenticated, then an interface set of commands is used:

```
[no] ip rsvp authentication key-chain <key-chain-name>
[no] ip rsvp authentication type {md5 | sha-1}
[no] ip rsvp authentication lifetime <hh:mm:ss>
[no] ip rsvp authentication window-size  <value>
```

If a user has only some specific neighbors that require to be authenticated then this is a set to be used.

Note The **neighbor** keyword and argument is no longer optional:

```
switch(config)# ip rsvp
switch(config-ip-rsvp)# authentication neighbor address <IP-address> key-chain
<key-chain-name>
switch(config-ip-rsvp)# authentication neighbor address <IP-address> type {md5 | sha-1}
switch(config-ip-rsvp)# authentication neighbor address <IP-address> lifetime <hh:mm:ss>
switch(config-ip-rsvp)# authentication neighbor address <IP-address> window-size  <value>
switch(config-ip-rsvp)# authentication neighbor address <IP-address> challenge
```

```
[no] authentication neighbor address <IP-address> key-chain <key-chain-name>
[no] authentication neighbor address <IP-address> type {md5 | sha-1}
[no] authentication neighbor address <IP-address> lifetime <hh:mm:ss>
[no] authentication neighbor address <IP-address> window-size  <value>
[no] authentication neighbor address <IP-address> challenge
```

To verify that this command has been configured correctly, use the **show ip rsvp authentication** command.

RSVP Bundle Messages

RSVP bundle messages consist of a bundle header followed by a variable number of standard RSVP messages. The bundle message aggregates multiple RSVP messages within a single PDU, though they can only be sent to RSVP neighbors that support bundling. The maximum size of an RSVP message is one IP datagram.

```
switch(config-ip-rsvp)# signalling refresh reduction bundle-max-size
```

Graceful Restart

The RSVP graceful restart (GR) is based on using RSVP hellos and adds new objects; for example, a Restart-Capability object to the Hello message and a Recovery-Label object to the sender template that forms part of the PATH message. The RSVP graceful restart procedure also adds a new message; for example, the RECOVERY PATH message has been added to help the restart.

An example that depicts the RSVP GR functionality is shown in Figure 13-5. The figure shows a TE tunnel between R1 and R3 and RSVP hellos being exchanged between routers R1 and R2, and R2 and R3 respectively. The Hello message contains a Restart-Capability object that defines a restart time and a recovery time for the router that originated the message. If router R2 restarts, the neighboring routers R1 and R3 detect the failure when four Hello messages with the Hello-Ack object are not received from R2. These routers start a restart timer based on the restart time specified by R2 in its Restart-Capability object. If R2 fails to restart during this restart-period, R1 and R3 tearing down the existing TE LSP between R1 and R4.

Figure 13-5 RSVP Graceful Restart Example

If R2 restarts during the restart period, it sends a Hello message to R1 and R3, which check the source-instance in the Hello against the value used by R2 before the failure.

If the value is unchanged, then R1 and R3 associate the R2 failure with a control channel failure and send summary refresh messages to R2 to refresh the state. Otherwise, R1 and R3 assume that R2 has restarted and check the recovery time sent by R2 in its HELLO *Restart-Capability* object.

If the recovery time is set to 0, R1 and R3 assume that R2 was not able to preserve its forwarding state, otherwise the value specified by R2 is considered as the recovery period for R2. During the recovery period, R1 sends PATH messages to R2 with Recovery-Label objects that contain labels previously sent by R2 to R1. This process enables R2 to recover the labels that it has given to R1 and locate the corresponding outgoing label and interface for the LSP (for example from TE). R2 then sends a corresponding PATH message to R3 with a Suggested-Label object. On receipt of the PATH message, R3 sends a RESV to R2.

If the downstream router R3 passes a label that does not match the label in the Suggested-Label object, R2 must reconfigure itself to use this label or generate a RESV error. R2 should not send data upstream using the label in the Suggested-Label object until the downstream router passes a label to R2 in its RESV messages.

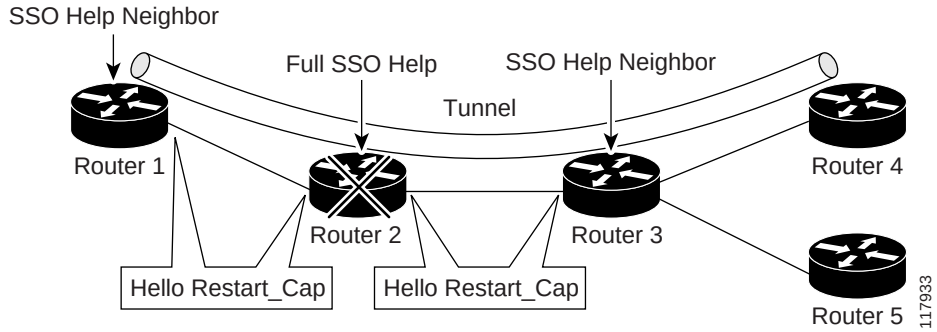
Overview of MPLS TE and RSVP Graceful Restart

RSVP graceful restart allows TE RSVP-enabled nodes to recover gracefully after a node failure in the network so that the RSVP state after the failure is restored as quickly as possible. The node failure can be completely transparent to other nodes in the network.

RSVP graceful restart preserves the label values and forwarding information and works with third-party or Cisco routers seamlessly.

RSVP graceful restart depends on RSVP hello messages to detect that a neighbor went down. Hello messages include Hello Request or Hello Acknowledgment (ACK) objects between two neighbors. As shown in Figure 13-6, the RSVP graceful restart extension to these messages adds an object called Hello Restart_Cap, which tells neighbors that a node may be able to reconnect if a failure occurs.

Figure 13-6 How RSVP Graceful Restart Works



The Hello Restart_Cap object has two values: the restart time, which is the sender's time to restart the RSVP_TE component and exchange hello messages after a failure; and the recovery time, which is the desired time that the sender wants the receiver to synchronize with the RSVP and MPLS databases.

In Figure 13-6, RSVP graceful restart help neighbor support is enabled on routers 1 and 3 so that they can help a neighbor recover after a failure, but they cannot do self recovery. Router 2 has full SSO help support enabled, which means that it can do self recovery after a failure or help its neighbor to recover. Router 2 has two RPs, one that is active and one that is standby (backup). A TE LSP is signaled from router 1 to router 4.

Router 2 does checkpointing; it copies state information from the active RP to the standby RP, which ensures that the standby RP has the latest information. If an active RP fails, the standby RP can take over.

Routers 2 and 3 exchange periodic graceful restart hello messages every 10,000 milliseconds (ms) (10 seconds), and so do routers 2 and 1 and routers 3 and 4. Assume that router 2 advertises its restart time = 60,000 ms (60 seconds) and its recovery time = 60,000 ms (60 seconds) as shown in the following example:

The debug from the sender side is as follows:

```
2011 Mar 15 11:47:46.200375 rsvp: [3016] HELLO-MSG: RSVP-HELLO: Sending msg_type [1=Req,
2=Ack] 1 from 10.1.1.1 to 10.1.1.2
```

The receiver debugs are as follows:

```
2011 Mar 15 11:48:26.012252 rsvp: [2967] HELLO-MSG: Received HELLO msg with object length
24 on i/f 10.21.1.2
2011 Mar 15 11:48:26.012344 rsvp: [2967] HELLO-MSG:  RSVP-HELLO: Received HELLO REQUEST
message from 10.1.1.2
2011 Mar 15 11:48:26.012371 rsvp: [2967] HELLO-MSG:  Received message with dst_address
(10.1.1.3) matching router ID
2011 Mar 15 11:48:26.012393 rsvp: [2967] HELLO-MSG:  RSVP-HELLO:
rsvp_hello_process_incoming_gr_message: restart_time 30000 recovery_time 120000
2011 Mar 15 11:48:26.012432 rsvp: [2967] HELLO-MSG:  Rcvd:Nbr 10.1.1.2  old_src_inst
845400891 new_src_inst 845400891, hc_event 0 hi_nbr_hello_state 1 hello_dst_inst
215306973, hi_my_src_inst 215306973
```

```
23:33:36: Outgoing Hello:
23:33:36: version:1 flags:0000 cksum:883C ttl:255 reserved:0 length:32
23:33:36: HELLO type HELLO REQUEST length 12:
23:33:36: Src_Instance: 0x6EDA8BD7, Dst_Instance: 0x00000000
23:33:36: RESTART_CAP type 1 length 12:
23:33:36: Restart_Time: 0x0000EA60, Recovery_Time: 0x0000EA60
```

Router 3 records this information into its database. Also, both neighbors maintain the neighbor status as UP. However, router 3's control plane fails at some point (for example, a primary RP failure). As a result, RSVP and TE lose their signaling information and states although data packets continue to be forwarded by the line cards.

When router 3 declares communication with router 2 lost, router 3 starts the restart time to wait for the duration advertised in router 2's restart time previously recorded (60 seconds). Routers 1 and 2 suppress all RSVP messages to router 3 except hellos. Router 3 keeps sending the RSVP PATH and RESV refresh messages to routers 4 and 5 so that they do not expire the state for the LSP; however, routers 1 and 3 suppress these messages for router 2.

When routers 1 and 3 receive the hello message from router 2, routers 1 and 3 check the recovery time value in the message. If the recovery time is 0, router 3 knows that router 2 was not able to preserve its forwarding information, and routers 1 and 3 delete the RSVP states that they had with router 2.

If the recovery time is greater than 0, router 1 sends router 2 PATH messages for each LSP that it had previously sent through router 2. If these messages were previously refreshed in summary messages, they are sent individually during the recovery time. Each of these PATH messages includes a Recovery_Label object that contains the label value received from router 2 before the failure.

When router 3 receives a PATH message from router 2, router 3 sends a RESV message upstream. However, router 3 suppresses the RESV message until it receives a PATH message. When router 2 receives the RESV message, it installs the RSVP state and reprograms the forwarding entry for the LSP.

Benefits of MPLS TE and RSVP Graceful Restart

State information recovery—RSVP graceful restart allows a node to do self recovery or to help its neighbor recover state information when there is an RP failure or the device has undergone an SSO.

Session information recovery—RSVP graceful restart allows session information recovery with minimal disruption to the network.

Increased availability of network services—A node can do a graceful restart to help itself or a neighbor recover its state by keeping the label bindings and state information, which provides a faster recovery of the failed node and does not affect currently forwarded traffic.

Configuring MPLS RSVP TE Graceful Restart

To alter the interval at which RSVP GR hellos are sent out by the local RSVP router, use the global **signalling hello graceful-restart refresh interval** command that applies to all neighbors. Graceful restart is on by default in Cisco NX-OS software and the default for the hello interval is 10 seconds.

```
switch(config-ip-rsvp)# signalling hello graceful-restart
switch(config-ip-rsvp)# signalling hello graceful-restart refresh interval
switch(config-ip-rsvp)# signalling hello graceful-restart refresh misses
switch(config-ip-rsvp)# signalling hello graceful-restart send restart-time
switch(config-ip-rsvp)# signalling hello graceful-restart send recovery-time
switch(config-ip-rsvp)# signalling hello reroute

switch(config-if)# ip rsvp signalling hello reroute
switch(config-if)# ip rsvp signalling hello reroute state-timeout refresh interval time
```

```
switch(config-if)# ip rsvp signalling hello reroute state-timeout refresh misses

switch(config-ip-rsvp)# signalling initial-retransmit-delay
switch(config-ip-rsvp)# signalling refresh interval
switch(config-ip-rsvp)# signalling refresh misses
switch(config-ip-rsvp)# signalling refresh reduction
switch(config-ip-rsvp)# signalling refresh reduction ack-delay
switch(config-ip-rsvp)# signalling refresh reduction bundle-max-size
switch(config-ip-rsvp)# signalling patherr state-removal
switch(config-ip-rsvp)# signalling rate-limit
```

To verify that this command has been configured correctly, use the **show ip rsvp graceful-restart** command.

RSVP Nonstop-Routing

RSVP nonstop routing (NSR) provides stateful high availability (HA) functionality to NX-OS. Two forms of service-level HA supported by RSVP NSR are as follows:

- Restartability—When an application crashes or hangs, it can be restarted by the system manager on the same supervisor.
- Switchover—If the kernel on the active supervisor fails, the active role can be switched to the standby supervisor.

Hello State Timer

The Hello State Timer (HST) provides for teardown of unprotected LSPs on non-FRR interfaces or LSPs with no backup on FRR interfaces by using RSVP hellos to detect a neighbor failure. The HST requires that the switch maintain hello communication with neighbors through which there are no protected LSPs running. If the hello communication with a neighbor is lost, HST initiates a PATH ERROR on the router that is upstream of the failed router and a PATH TEARDOWN on the neighbor that is downstream of the failed router (where there are instances running on both ends). The primary advantage of the HST is that it can detect an LSP failure and tear down the associated LSPs, and then free up requisite bandwidth more quickly than IGP.

Licensing Requirements for MPLS RSVP TE

Product	License Requirement
Cisco NX-OS	MPLS RSVP TE requires an MPLS license. For a complete explanation of the Cisco NX-OS licensing scheme and how to obtain and apply licenses, see the *Cisco NX-OS Licensing Guide.*

Prerequisites for MPLS RSVP TE

MPLS RSVP TE has the following prerequisites:

- Your network must support Multiprotocol Label Switching (MPLS).

Guidelines and Limitations for MPLS RSVP TE

MPLS RSVP TE has the following guidelines and limitations:

- The MPLS TE feature must be enabled.

Default Settings for MPLS RSVP TE

Table 13-1 lists the default settings for MPLS RSVP TE.

Table 13-1 Default Settings for MPLS RSVP TE

Parameters	Default
MPLS RSVP TE feature	Enabled

Configuring MPLS RSVP TE

This section includes the following topics:

Configuring RSVP Message Authentication

You can configure message authentication for MPLS RSVP TE.

Prerequisites

You must have the MPLS TE feature enabled (see the "Configuring MPLS TE" section on page 11-4).

Ensure that you are in the correct VDC (or use the **switchto vdc** command).

SUMMARY STEPS

1. **configure terminal**
2. **key chain** *key-chain-name*
3. **key** *key-identifier-number*
4. **key-string**
5. **exit**
6. **exit**
7. **ip rsvp**
8. **authentication** [**neighbor address** *IP-address*] **key-chain** *key-chain-name*
9. **authentication** [**neighbor address** *IP-address*] **type** {**md5** | **sha-1**}
10. **authentication** [**neighbor address** *IP-address*] **lifetime** *hh:mm:ss*

11. **authentication** [**neighbor address** *IP-address*] **window-size** *value*
12. **authentication** [**neighbor address** *IP-address*] **challenge**
13. **exit**
14. **exit**

DETAILED STEPS

	Command	Purpose
Step 1	**configure terminal** **Example:** switch# configure terminal switch(config)#	Enters global configuration mode.
Step 2	**key chain** *key-chain-name* **Example:** switch(config)# key chain ***key1*** switch(config-keychain)#	Enters the keychain management configuration mode and assigns a name for the key chain to be configured. The maximum size for the key-chain name is 63 alphanumeric characters.
Step 3	**key** *key-identifier-number* **Example:** switch(config-keychain)# key 4660 switch(config-keychain-key)#	Enters a keychain configuration ID and enters the keychain string configuration mode. The key-chain identifier number has a range from 0 to 65535.
Step 4	**key-string** **Example:** switch(config-keychain-key)# key-string qwertyui	Enters a keychain string for the keychain configuration ID.
Step 5	**exit** **Example:** switch(config-keychain-key)# exit switch(config-keychain)#	Exits the keychain string assignment mode and returns to keychain configuration ID mode.
Step 6	**exit** **Example:** switch(config-keychain)# exit switch(config)#	Exits the keychain configuration ID mode and returns to global configuration mode.
Step 7	**ip rsvp** **Example:** switch(config)# ip rsvp switch(config-ip-rsvp)#	Enters the RSVP configuration mode.
Step 8	**authentication** [**neighbor address** *IP-address*] **key-chain** *key-chain-name* **Example:** switch(config-ip-rsvp)# authentication neighbor address 10.0.0.2 key-chain key1	Activates RSVP cryptographic authentication for a neighbor or globally. The key-chain information is provided in a separate command. Use the **no** form of the command to disable global authentication.

	Command	Purpose
Step 9	**authentication** [**neighbor address** *IP-address*] **type** {**md5** | **sha-1**} **Example:** `switch(config-ip-rsvp)# authentication neighbor address 10.0.0.2 type sha-1`	Specifies the algorithm used to generate cryptographic signatures messages for a neighbor or globally. By default, the authentication type is md5. To revert to the default md5 authentication configuration, use the **no** form of the command.
Step 10	**authentication** [**neighbor address** *IP-address*] **lifetime** *hh:mm:ss* **Example:** `switch(config-ip-rsvp)# authentication neighbor address 10.0.0.2 lifetime 10:30:30`	Controls how long RSVP maintains security associations with a neighbor or globally. The default lifetime is 30 minutes. To revert to the default lifetime, use the **no** form of the command.
Step 11	**authentication** [**neighbor address** *IP-address*] **window-size** *value* **Example:** `switch(config-ip-rsvp)# authentication neighbor address 10.0.0.2 window-size 2`	Specifies the tolerance for out-of-sequence messages for a neighbor or globally. The default value is 1, which means all out-of-sequence messages are dropped. Use the **no** form of the command to return to the default configuration.
Step 12	**authentication** [**neighbor address** *IP-address*] **challenge** **Example:** `switch(config-ip-rsvp)# authentication neighbor address 10.0.0.2 challenge`	Makes RSVP use a challenge-response handshake with a neighbor.

Step 13	**exit**
	
	Example:
	`switch(config-ip-rsvp)# exit`
	`switch(config)#`
		Exits RSVP configuration mode and returns to global configuration mode.

Step 14	**exit**
	
	Example:
	`switch(config)# exit`
	`switch#`
		Exits global configuration mode.

Configuring Hello for MPLS RSVP TE

You can configure hellos for MPLS RSVP TE.

Note MPLS TE supports a single IGP process or instance. Do not configure MPLS TE in more than one IGP process or instance.

Prerequisites

You must have the MPLS TE feature enabled (see the "Configuring MPLS TE" section on page 11-4).

Ensure that you are in the correct VDC (or use the **switchto vdc** command).

SUMMARY STEPS

1. **configure terminal**
2. **ip rsvp**
3. **signalling hello graceful-restart**
4. **signalling hello graceful-restart refresh interval** *time*
5. **signalling hello graceful-restart refresh misses** *refresh-misses*
6. **signalling hello graceful-restart send restart-time** *time*
7. **signalling hello graceful-restart send recovery-time** *time*
8. **signalling hello reroute**

DETAILED STEPS

	Command	Purpose

Step 1	**configure terminal**
	
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**ip rsvp**
	
	Example:
	`switch(config)# ip rsvp`
	`switch(config-ip-rsvp)#`
		Enters RSVP configuration mode.

Step 3	**signalling hello graceful-restart** **Example:** `switch(config-ip-rsvp)# signalling hello graceful-restart`	Globally enables signaling of node-based hellos for graceful restart functionality. The command is by default on at the global level. Use the **no** form of the command to disable sending of hellos for graceful restart.
Step 4	**signalling hello graceful-restart refresh interval** *time* **Example:** `switch(config-ip-rsvp)# signalling hello graceful-restart refresh interval 15`	Configures the interval at which RSVP graceful-restart hello messages are sent to each neighbor. The default value is 10 seconds. Use the **no** form of the command to return to the default behavior.
Step 5	**signalling hello graceful-restart refresh misses** *refresh-misses* **Example:** `switch(config-ip-rsvp)# signalling hello graceful-restart refresh misses 6`	Configures the number of consecutive missed hello message before a neighbor is declared down or unreachable. The default value is 4. Use the **no** form of the command to return to the default behavior.
Step 6	**signalling hello graceful-restart send restart-time** *time* **Example:** `switch(config-ip-rsvp)# signalling hello graceful-restart send restart-time 20`	Configures the restart time that is advertised in the Restart-Capability object in hello messages. The default restart time is 30 seconds. Use the **no** form of the command to return to the default behavior.

	Command	Purpose
Step 7	**signalling hello graceful-restart send recovery-time** *time* **Example:** `switch(config-ip-rsvp)# signalling hello graceful-restart send recovery-time 150`	Configures the recovery time that is advertised in the Restart-Capability object in hello messages. The default recovery time is 120 seconds. Use the **no** form of the command to return to the default behavior.
Step 8	**signalling hello reroute** **Example:** `switch(config-ip-rsvp)# signalling hello reroute`	Globally enables the use of HST hellos. Sending of HST hellos is interface based, and configuration of reroute hello signaling is required at the global level and per-interface to enable sending of HST hellos on an interface. The command is by default off. Use the **no** form of the command to disable sending of reroute hellos.

Other Configurations for MPLS RSVP TE

You can use other configuration commands for MPLS RSVP TE.

Prerequisites

You must have the MPLS TE feature enabled (see the "Configuring MPLS TE" section on page 11-4).

Ensure that you are in the correct VDC (or use the **switchto vdc** command).

SUMMARY STEPS

1. **configure terminal**
2. **ip rsvp**
3. **signalling initial-retransmit-delay** *time*
4. **signalling refresh interval** *time*
5. **signalling refresh misses** *refresh-missed*
6. **signalling refresh reduction**
7. **signalling refresh reduction ack-delay** *time*
8. **signalling refresh reduction bundle-max-size** *value*
9. **signalling patherr state-removal**
10. **signalling rate-limit** [**burst** *value*] [**period** *time*]

DETAILED STEPS

	Command	Purpose
Step 1	**configure terminal** **Example:** switch# configure terminal switch(config)#	Enters global configuration mode.
Step 2	**ip rsvp** **Example:** switch(config)# ip rsvp switch(config-ip-rsvp)#	Enters RSVP configuration mode.
Step 3	**signalling initial-retransmit-delay** *time* **Example:** switch(config-ip-rsvp)# signalling initial-retransmit-delay 3	Configures the minimum amount of time that RSVP waits for an acknowledgement before retransmitting the same message. The default value is 1 second. Use the **no** form of the command to return to the default value.
Step 4	**signalling refresh interval** *time* **Example:** switch(config-ip-rsvp)# signalling refresh interval 40	Configures the frequency at which the RSVP state is refreshed. The default value is 30 seconds. Use the **no** form of the command to return to the default behavior.
Step 5	**signalling refresh misses** *refresh-missed* **Example:** switch(config-ip-rsvp)# signalling refresh misses 5	Specifies the number of refresh messages that can be missed before RSVP deems a state to be expired. The default value is 4. Use the **no** form of the command to return to the default behavior.
Step 6	**signalling refresh reduction** **Example:** switch(config-ip-rsvp)# signalling refresh reduction	Configures the RSVP refresh reduction. Use the **no** form of the command to disable RSVP refresh reduction. By default, refresh reduction is enabled.
Step 7	**signalling refresh reduction ack-delay** *time* **Example:** switch(config-ip-rsvp)# signalling refresh reduction ack-delay 300	Configures the maximum amount of time RSVP holds on to an acknowledgement before sending it. The default value is 250 ms (0.25sec). Use the **no** form of the command to return to the default value.
Step 8	**signalling refresh reduction bundle-max-size** *value* **Example:** switch(config-ip-rsvp)# signalling refresh reduction bundle-max-size 5000	Configures the bundle maximum send message size in the range from 0 to 65000 bytes, with the default being 4096 bytes. Set the value of 0 to disable sending of bundle messages.

	Command	Purpose
Step 9	**signalling patherr state-removal** **Example:** `switch(config-ip-rsvp)# signalling patherr state-removal`	Deletes the PATH state automatically when forwarding a PATH-ERROR message, which eliminates the need to send a subsequent PATH-TEAR message. Use the **no** form of the command to disable the path state deletion on a PATH ERROR.
Step 10	**signalling rate-limit** [**burst** *value*] [**period** *time*] **Example:** `switch(config-ip-rsvp)# signalling rate-limit burst 10 period 30`	Sets the rate limit for the number of messages that are sent to a neighboring router. The default burst is 8 messages in an interval of 20 ms. Use the **no** form of the command to return to the default behavior.

Verifying the MPLS RSVP TE Configuration

To display the MPLS RSVP TE configuration, perform one of the following tasks:

Command	Purpose
show ip rsvp	Displays global RSVP information.
show ip rsvp authentication	Displays the database for security associations that RSVP has established with neighbors. Use the **interface** or **neighbor** optional keyword to display the authentication information for the specified interface or neighbor. Use the **detail** optional keyword to display detailed authentication information.
show ip rsvp counters	Displays RSVP packet counters. Use the **interface** optional keyword to display interface RSVP packet counters. Use the **teardown** optional keyword to display RSVP teardown counters and the **all** optional keyword to display all counters.
show ip rsvp fast-reroute	Displays RSVP FRR information. This command is different than the Cisco IOS version, and is used to display RSVP centric FRR information. Use the **detail** optional keyword to display detailed fast-reroute information.
show ip rsvp graceful-restart	Displays restart information for RSVP.
show ip rsvp interface	Displays information about all interfaces with RSVP enabled. Use the **interface name** optional keyword to display information for the specified interface. Use the **detail** optional keyword to display detailed interface information and the **backup-tunnel** optional keyword to display backup-tunnel information known to RSVP.

Command	Purpose
show ip rsvp neighbor	Displays information about all RSVP neighbors. To display information about a neighbor, use the **neighbor** optional keyword. Use the **detail** optional keyword to display detailed neighbor information.
show ip rsvp reservation	Displays all reservations RSVP knows about on a router. Use the destination IP address, sender IP address, destination port, and/or source port information to filter the reservation information. Use the **detail** optional keyword to view detailed reservation display.
show ip rsvp sender	Displays all path-states RSVP knows about on a router. To display information about a particular sender or destination, use the sender IP address, destination IP address, destination port, and/or source port with the command. Use the **detail** optional keyword to view detailed sender information.
show ip rsvp session	Displays all sessions that RSVP knows about on a router. To display information about a particular destination, use the **destination IP-address** optional keyword with the command.
show ip rsvp hello instance	Displays RSVP hello instance information. Use the **detail** optional keyword to view detailed hello instance information.
show ip rsvp hello client lsp	Displays RSVP hello client LSP database. Use the **detail** optional keyword to view details of the LSP.
show ip rsvp hello client neighbor	Displays RSVP hello neighbor information. Use the **detail** optional keyword to view detailed hello neighbor information.
show ip rsvp signalling rate-limit	Displays RSVP globally configured signalling rate-limit information.
show ip rsvp signalling refresh reduction	Displays RSVP globally configured refresh reduction information.
show ip rsvp signalling refresh misses	Displays RSVP globally configured refresh misses information.
show ip rsvp signalling refresh interval	Displays RSVP globally configured refresh interval information.
show ip rsvp internal	Displays internal counters, event-history buffers, memory statistics or persistent store information. Use additional filters with these suboptions to filter the display the filtered information.

For detailed information about the fields in the output from these commands, see the *Cisco NX-OS MPLS Command Reference*.

Verification Examples for MPLS RSVP TE

This section includes the following topics:

Example: Verifying the RSVP

The following example shows how to verify the global RSVP parameters:

```
switch# show ip rsvp
RSVP Process
        Supervisor State: Active
        Start Type: configuration [stateless]
        High Availability: Enabled [ok]
        Graceful Restart: disabled
        Hello State Timeout: (null:no-enum-table)
        Router id: 1.1.1.20
        Patherr State Removal: Disabled
        Local Epoch: 0xab446c

Registered RSVP Clients
        MPLS TE [Service-Access-Point 288, ID 1, Batch-Time 50 msec]
                [Listener: Flags 0x7, Events 0x7ffff, ClientHintLen 24]

Message Bundling
        Enabled [Transmit-delay 50 msec, Max-Size 4096 bytes]

Refresh Parameters
        Interval 45 sec, Miss-Limit 4

Refresh-Reduction
```

```
        Enabled [Initial-Retransmit-Delay 5000 msec]
        [Rapid-Retransmit Disabled, Ack-Delay 400 msec]

Rate-Limit
        Disabled [Limit 100 messages, Interval 1000 msec]

GR Recovery Timer
        Not running

Authentication
        Disabled
```

Example: Verifying the RSVP Neighbor

The following example shows how to verify the RSVP neighbor:

```
switch# show ip rsvp neighbor
Address         Interface      RouterID        State    Expiry     LastSend
3.0.206.6       Ethernet1/7    1.1.1.6         UP,RR    14 minutes    4 sec
```

Example: Verifying the RSVP Reservation

The following example shows how to verify RSVP reservations:

```
switch# show ip rsvp reservation

Total Reservation States: 3000
To            From          Pro DPort Sport Next Hop      I/F           Fi
1.1.1.13      1.1.1.20      0   20000 19    3.0.206.6     Eth1/7        SE
1.1.1.13      1.1.1.20      0   20001 19    3.0.206.6     Eth1/7        SE
.
.
.
1.1.1.13      1.1.1.20      0   22997 17    3.0.206.6     Eth1/7        SE
1.1.1.13      1.1.1.20      0   22998 17    3.0.206.6     Eth1/7        SE
1.1.1.13      1.1.1.20      0   22999 17    3.0.206.6     Eth1/7        SE
```

Example: Verifying the RSVP Sender

The following example shows how to verify the RSVP senders:

```
switch# show ip rsvp sender
Total Sender States: 3000
To              From            Pro DPort Sport Prev Hop        I/F
1.1.1.13        1.1.1.20        0   20000 19    none            None
1.1.1.13        1.1.1.20        0   20001 19    none            None
.
.
.
1.1.1.13        1.1.1.20        0   22998 17    none            None
1.1.1.13        1.1.1.20        0   22999 17    none            None
```

Example: Verifying the RSVP Sessions

The following example shows how to verify RSVP sessions:

```
switch(config-if-te)# show ip rsvp session
Total Sessions: 4
Type Destination     DPort Proto/ExtTunID  PSBs  RSBs  Reqs  PXSBs RXSBs
LSP4 10.10.10.10     10    10.10.10.15     1     1     0     1     0
LSP4 10.10.10.10     11    10.10.10.15     1     1     0     1     0
LSP4 10.10.10.10     12    10.10.10.15     1     1     0     1     0
LSP4 10.10.10.10     13    10.10.10.15     1     1     0     1     0
```

Example: Verifying the RSVP Signaling Rate Limit

The following example shows how to verify the RSVP signaling rate limit:

```
switch# show ip rsvp signalling rate-limit
Rate-Limiting: Disabled
  Limit: 100
  Interval (msec): 1000
switch# show ip rsvp signalling refresh ?
  interval   Display interval for refresh messages
  misses     Display misses required to trigger state timeout
  reduction  Display refresh reduction parameters
```

Example: Verifying the RSVP Signaling Refresh Interval

The following example shows how to verify the RSVP signaling refresh interval:

```
switch# show ip rsvp signalling refresh interval
Refresh interval (sec): 45
```

Example: Verifying the RSVP Signaling Refresh Misses

The following example shows how to verify the RSVP signaling refresh misses:

```
switch# show ip rsvp signalling refresh misses
Refresh misses: 4
```

Example: Verifying the RSVP Signaling Refresh Reduction

The following example shows how to verify the RSVP signaling refresh reduction:

```
switch# show ip rsvp signalling refresh reduction
Refresh Reduction: Enabled
  ACK delay (msec): 400
  Initial retransmit delay (msec): 5000
  Local epoch: 0xab446c
  Message IDs: in use 6000, total allocated 33005, freed 27005
```

Example: Verifying the RSVP Counters

The following example shows how to verify the RSVP counters:

```
switch# show ip rsvp counters
All Interfaces    Recv       Xmit                         Recv       Xmit
Packet            40641      40847     PacketError        0          0
Path              0          301       Resv               0          0
PathError         0          0         ResvError          0          0
PathTear          0          0         ResvTear           0          0
ResvConf          0          0         RTearConf          0          0
Ack               5          2         SRefresh           38341      38510
Hello             0          0         IntegrityChallenge 0          0
IntegrityResponse 0          0

Bundle            1672       1875
Path              0          9192      Resv               9135       0
PathError         3559       0         ResvError          0          0
PathTear          0          6000      ResvTear           3557       0
Ack               0          0
switch#
```

Example: Verifying All of the RSVP Counters

The following example shows how to verify all of the RSVP counters:

```
switch# show ip rsvp counters all
Teardown Reason                Path       Resv
UNSPECIFIED                    0          0
PATH TIMEOUT                   0          0
RESV TIMEOUT                   0          0
SIGNALED                       4448       4448
MGMT                           0          0
POLICY                         0          0
PROXY                          3069       3069
NO_RESOURCES                   0          0
PREEMPTED                      0          0
MSG_ERROR                      0          0
INTERNAL                       0          0
TRAFFIC_CONTROL                0          0
POLICY_SYNC                    0          0
GR_TIMEOUT                     0          0
LINK_NBOR_DOWN                 0          0
LOCAL-SEND_PERR_PSR            0          0
NETWORK_PERR_PSR               0          0
HST_TIMEOUT                    0          0
PLR BACKUP DELETE              0          0
CLI-CLEAR                      0          0
RESTART-COMMAND                0          0
INTERFACE-DELETE               0          0
Sent:
  Messages successfully authenticated:    0
  Messages internal failure:              0
Received:
  Messages successfully authenticated:    0
  Total receive errors:                   0
  Receive Errors:
    Missing INTEGRITY object:             0
    Incorrect digest:                     0
    Digest type mismatch:                 0
    Duplicate sequence number:            0
```

```
    Out-of-range sequence number:         0
Challenge Handshake:
  Challenges Received:                    0
  Challenges Responded:                   0
  Initiations:                            0
  Timeouts:                               0
  Retransmissions:                        0
  Responses Received:                     0
  Drops during Challenge:                 0
  Incorrect challenge response:           0
  Duplicate challenge response:           0
  Late challenge response:                0
All Interfaces    Recv       Xmit                        Recv       Xmit
Packet            40645      40851     PacketError       0          0
Path              0          301       Resv              0          0
PathError         0          0         ResvError         0          0
PathTear          0          0         ResvTear          0          0
ResvConf          0          0         RTearConf         0          0
Ack               5          2         SRefresh          38345      38514
Hello             0          0         IntegrityChallenge 0         0
IntegrityResponse 0          0

Bundle            1672       1875
Path              0          9192      Resv              9135       0
PathError         3559       0         ResvError         0          0
PathTear          0          6000      ResvTear          3557       0
Ack               0          0
```

Example: Verifying the RSVP Counters for Teardown

The following example shows how to verify the RSVP counters for teardown:

```
switch# show ip rsvp counters teardown
Teardown Reason                Path       Resv
UNSPECIFIED                    0          0
PATH TIMEOUT                   0          0
RESV TIMEOUT                   0          0
SIGNALED                       4448       4448
MGMT                           0          0
POLICY                         0          0
PROXY                          3069       3069
NO_RESOURCES                   0          0
PREEMPTED                      0          0
MSG_ERROR                      0          0
INTERNAL                       0          0
TRAFFIC_CONTROL                0          0
POLICY_SYNC                    0          0
GR_TIMEOUT                     0          0
LINK_NBOR_DOWN                 0          0
LOCAL-SEND_PERR_PSR            0          0
NETWORK_PERR_PSR               0          0
HST_TIMEOUT                    0          0
PLR BACKUP DELETE              0          0
CLI-CLEAR                      0          0
RESTART-COMMAND                0          0
INTERFACE-DELETE               0          0
```

Example: Verifying the RSVP Counters Authentication

The following example shows how to verify the RSVP counters authentication:

```
switch# show ip rsvp counters authentication
Sent:
  Messages successfully authenticated:    0
  Messages internal failure:              0
Received:
  Messages successfully authenticated:    0
  Total receive errors:                   0
  Receive Errors:
    Missing INTEGRITY object:             0
    Incorrect digest:                     0
    Digest type mismatch:                 0
    Duplicate sequence number:            0
    Out-of-range sequence number:         0
Challenge Handshake:
  Challenges Received:                    0
  Challenges Responded:                   0
  Initiations:                            0
  Timeouts:                               0
  Retransmissions:                        0
  Responses Received:                     0
  Drops during Challenge:                 0
  Incorrect challenge response:           0
  Duplicate challenge response:           0
  Late challenge response:                0
```

Example: Verifying the RSVP FRR

The following example shows how to verify the RSVP FRR:

```
switch# show ip rsvp fast-reroute
    A - Active      R - Ready      U - Unassigned
Destination     TunID Source          Backup         Protected-I/f   Hop

State
Fast-Reroute Summary:
  Total Reroutable Paths: 0
    Active: 0, Ready: 0, Unassigned: 0
```

Example: Verifying the RSVP Hello Client LSP

The following example shows how to verify the RSVP hello client LSP:

```
switch# show ip rsvp hello client lsp

Local           Remote          tun_id lsp_id subgrp_orig     subgrp_id FLAGS
```

Example: Verifying the RSVP Hello Graceful-Restart

The following example shows how to verify the RSVP hello graceful-restart:

```
switch(config-if-te)# show ip rsvp hello graceful-restart
Graceful Restart: Enabled (full mode)
  Refresh interval: 10000 msecs
  Refresh misses: 4
```

```
  DSCP: 0xc0
  Advertised restart time: 30000 msecs
  Advertised recovery time: 120000 msecs
  Maximum wait for recovery: 3600000 msecs
```

Example: Verifying the RSVP Hello Instance

The following example shows how to verify the RSVP hello instance:

```
switch# show ip rsvp hello instance

Active Instances:
  - None -

Passive Instances:
  - None -
```

Example: Verifying the RSVP Interface

The following example shows how to verify the RSVP interface:

```
switch(config-if-te)# show ip rsvp interface
Interface          Ifindex    IOD    MPLS     Config   State
Ethernet2/2        0x1a081000 37     enabled  None     Up
Ethernet2/7        0x1a086000 42     enabled  None     Down
loopback0          0x14000000 45     enabled  None     Up
```

Additional References for MPLS RSVP TE

For additional information related to implementing MPLS RSVP TE, see the following sections:

- Related Document, page 13-27
- MIBs, page 13-27

Related Document

Related Topic	Document Title
MPLS TE commands	*Cisco NX-OS Multiprotocol Label Switching Command Reference*

MIBs

MIB	MIBs Link
• CISCO-IETF-FRR-MIB • MPLS TE-STD-MIB	To locate and download Cisco 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

Feature History for MPLS RSVP TE

Table 13-2 lists the release history for this feature.

Table 13-2 Feature History for MPLS RSVP TE

Feature Name	Releases	Feature Information
MPLS RSVP TE	5.2(1)	This feature was introduced.