Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software

Multiprotocol Label Switching (MPLS) is a standards-based solution, driven by the Internet Engineering Task Force (IETF), devised to convert the Internet and IP backbones from best-effort networks into business-class transport media.

Resource Reservation Protocol (RSVP) is a signaling protocol that enables systems to request resource reservations from the network. RSVP processes protocol messages from other systems, processes resource requests from local clients, and generates protocol messages. As a result, resources are reserved for data flows on behalf of local and remote clients. RSVP creates, maintains, and deletes these resource reservations.

RSVP provides a secure method to control quality-of-service (QoS) access to a network.

MPLS Traffic Engineering (MPLS-TE) and MPLS Optical User Network Interface (MPLS O-UNI) use RSVP to signal label switched paths (LSPs).

Feature History for Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router. Support was added for ACL-based prefix filtering.
Release 3.3.0	No modification.
Release 3.4.0	No modification.
Release 3.4.1	Support was added for RSVP authentication.
Release 3.5.0	No modification.
Release 3.6.0	No modification.
Release 3.7.0	No modification.

Contents

Prerequisites for Implementing RSVP for MPLS-TE and MPLS O-UNI

The following are prerequisites are required to implement RSVP for MPLS-TE and MPLS O-UNI:

- You must be in a user group associated with a task group that includes the proper task IDs for MPLS RSVP commands.
- Either a composite mini-image plus an MPLS package, or a full image, must be installed.

Information About Implementing RSVP for MPLS-TE and MPLS O-UNI

To implement MPLS RSVP, you must understand the following concepts, which are described in the sections that follow:

For information on how to implement RSVP authentication, see How to Implement RSVP Authentication, page MPC-71.

Overview of RSVP for MPLS-TE and MPLS O-UNI

RSVP is a network control protocol that enables Internet applications to signal LSPs for MPLS-TE, and LSPs for O-UNI. The RSVP implementation is compliant with the IETF RFC 2205, RFC 3209, and OIF2000.125.7.

When configuring an O-UNI LSP, the RSVP session is bidirectional. The exchange of data between a pair of machines actually constitutes a single RSVP session. The RSVP session is established using an Out-Of-Band (OOB) IP Control Channel (IPCC) with the neighbor. The RSVP messages are transported over an interface other than the data interface.

RSVP supports extensions according to OIF2000.125.7 requirements, including:

- Generalized Label Request
- Generalized UNI Attribute
- UNI Session
- New Error Spec sub-codes

RSVP is automatically enabled on interfaces on which MPLS-TE is configured. For MPLS-TE LSPs with non-zero bandwidth, the RSVP bandwidth has to be configured on the interfaces. There is no need to configure RSVP, if all MPLS-TE LSPs have zero bandwidth. For O-UNI, there is no need for any RSVP configuration.

RSVP Refresh Reduction, defined in RFC2961, includes support for reliable messages and summary refresh messages. Reliable messages are retransmitted rapidly if the message is lost. Because each summary refresh message contains information to refresh multiple states, this greatly reduces the amount of messaging needed to refresh states. For refresh reduction to be used between two routers, it must be enabled on both routers. Refresh Reduction is enabled by default.

Message rate limiting for RSVP allows you to set a maximum threshold on the rate at which RSVP messages are sent on an interface. Message rate limiting is disabled by default.

The process that implements RSVP is restartable. A software upgrade, process placement or process failure of RSVP or any of its collaborators, has been designed to ensure Nonstop Forwarding (NSF) of the data plane.

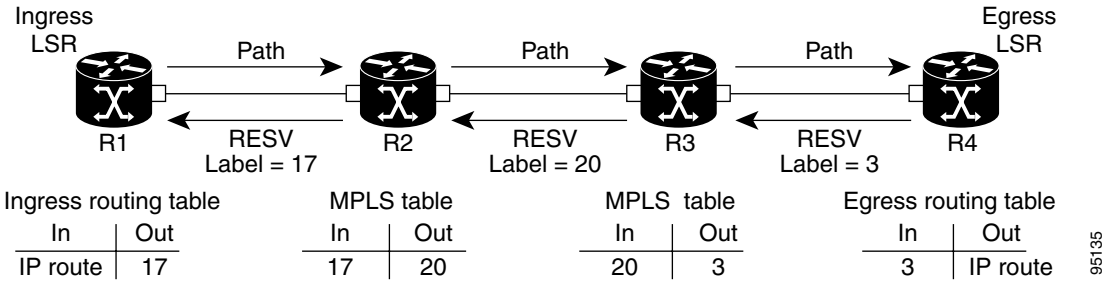
RSVP supports graceful restart, which is compliant with RFC 3473. It follows the procedures that apply when the node reestablishes communication with the neighbor's control plane within a configured restart time.

It is important to note that RSVP is not a routing protocol. RSVP works in conjunction with routing protocols and installs the equivalent of dynamic access lists along the routes that routing protocols calculate. Because of this, implementing RSVP in an existing network does not require migration to a new routing protocol.

LSP Setup

LSP setup is initiated when the LSP head node sends path messages to the tail node (see Figure 7).

Figure 7 RSVP Operation



The Path messages reserve resources along the path to each node, creating Path soft states on each node. When the tail node receives a path message, it sends a reservation (RESV) message with a label back to the previous node. When the reservation message arrives at the previous node, it causes the reserved resources to be locked and forwarding entries are programmed with the MPLS label sent from the tail-end node. A new MPLS label is allocated and sent to the next node upstream.

When the reservation message reaches the head node, the label is programmed and the MPLS data starts to flow along the path.

Figure 7 illustrates an LSP setup for non-O-UNI applications. In the case of an O-UNI application, the RSVP signaling messages are exchanged on a control channel, and the corresponding data channel to be used is acquired from the LMP Manager module based on the control channel. Also the O-UNI LSP's are by default bidirectional while the MPLS-TE LSP's are uni-directional.

High Availability

RSVP has been designed to ensure nonstop forwarding under the following constraints:

- Ability to tolerate the failure of one or more MPLS/O-UNI processes.
- Ability to tolerate the failure of one RP of a 1:1 redundant pair.
- Hitless software upgrade.

The RSVP high availability (HA) design follows the constraints of the underlying architecture where processes can fail without affecting the operation of other processes. A process failure of RSVP or any of its collaborators does not cause any traffic loss or cause established LSPs to go down. When RSVP restarts, it recovers its signaling states from its neighbors. No special configuration or manual intervention are required. You may configure RSVP graceful restart, which offers a standard mechanism to recover RSVP state information from neighbors after a failure.

Graceful Restart

RSVP graceful restart provides a control plane mechanism to ensure high availability, which allows detection and recovery from failure conditions while preserving nonstop forwarding services on the systems running Cisco IOS XR software.

RSVP graceful restart provides a mechanism that minimizes the negative effects on MPLS traffic caused by the following types of faults:

- Disruption of communication channels between two nodes when the communication channels are separate from the data channels. This is called control channel failure.
- The control plane of a node fails but the node preserves its data forwarding states. This is called node failure.

The procedure for RSVP graceful restart is described in the "Fault Handling" section of RFC 3473: *Generalized MPLS Signaling, RSVP-TE Extensions*. One of the main advantages of using RSVP graceful restart is recovery of the control plane while preserving nonstop forwarding and existing labels.

Graceful Restart: Standard and Interface-Based

When you configure RSVP graceful restart, Cisco IOS XR software sends and expects node-id address based Hello messages (that is, Hello Request and Hello Ack messages). The RSVP graceful restart Hello session is not established if the neighbor router does not respond with a node-id based Hello Ack message.

You can also configure graceful restart to respond (send Hello Ack messages) to interface-address based Hello messages sent from a neighbor router in order to establish a graceful restart Hello session on the neighbor router. If the neighbor router does not respond with node-id based Hello Ack message, however, the RSVP graceful restart Hello session is not established.

Cisco IOS XR software provides two commands to configure graceful restart:

- **signalling hello graceful-restart**
- **signalling hello graceful-restart interface-based**

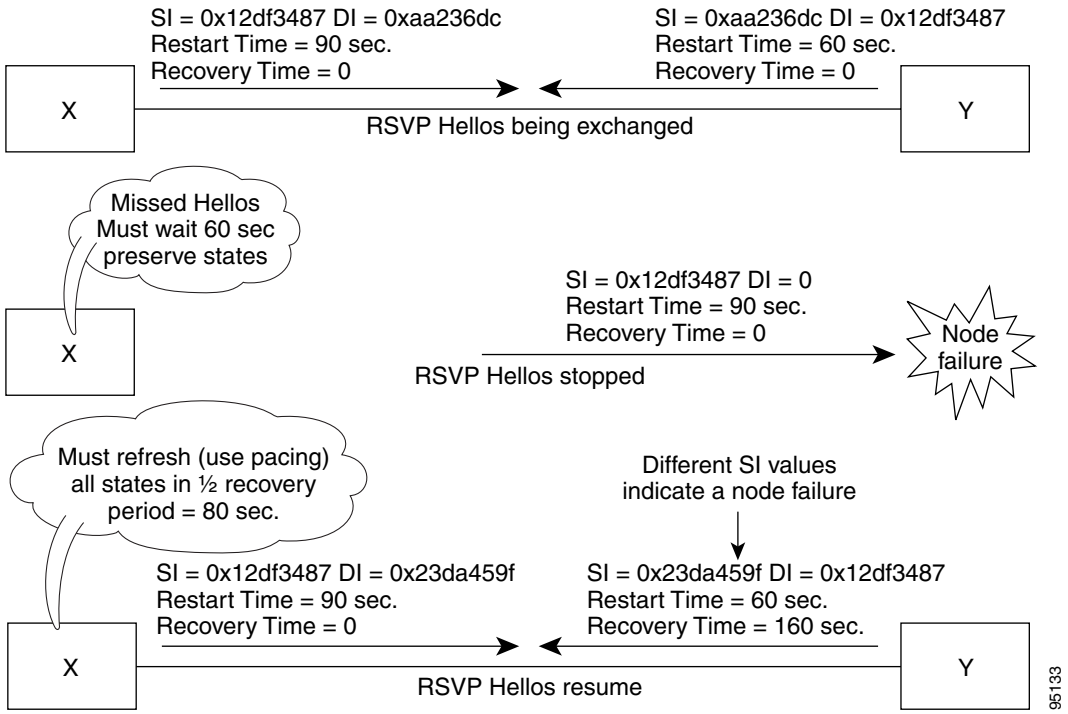
Note By default, graceful restart is disabled. To enable interface-based graceful restart, you must first enable standard graceful restart. You cannot enable interface-based graceful restart independently.

For detailed configuration steps, refer to Enabling Graceful Restart, page MPC-63.

Graceful Restart: Figure

Figure 8 illustrates how RSVP graceful restart handles a node failure condition.

Figure 8 Node Failure with RSVP



RSVP graceful restart requires the use of RSVP hello messages. Hello messages are used between RSVP neighbors. Each neighbor can autonomously issue a hello message containing a hello request object. A receiver that supports the hello extension replies with a hello message containing a hello acknowledgement (ACK) object. This means that a hello message contains either a hello Request or a hello ACK object. These two objects have the same format.

The restart cap object indicates a node's restart capabilities. It is carried in hello messages if the sending node supports state recovery. The restart cap object has the following two fields:

- Restart Time: Time after a loss in Hello messages within which RSVP hello session can be reestablished. It is possible for a user to manually configure the Restart Time.
- Recovery Time: Time that the sender waits for the recipient to re-synchronize states after the re-establishment of hello messages. This value is computed and advertised based on number of states that existed before the fault occurred.

For graceful restart, the hello messages are sent with an IP Time to Live (TTL) of 64. This is because the destination of the hello messages can be multiple hops away. If graceful restart is enabled, hello messages (containing the restart cap object) are send to an RSVP neighbor when RSVP states are shared with that neighbor.

Restart cap objects are sent to an RSVP neighbor when RSVP states are shared with that neighbor. If the neighbor replies with hello messages containing the restart cap object, the neighbor is considered to be graceful restart capable. If the neighbor does not reply with hello messages or replies with hello messages that do not contain the restart cap object, RSVP backs off sending hellos to that neighbor. If graceful restart is disabled, no hello messages (Requests or ACKs) are sent. If a hello Request message is received from an unknown neighbor, no hello ACK is sent back.

ACL-based Prefix Filtering

RSVP provides for the configuration of extended access lists (ACLs) to forward, drop, or perform normal processing on RSVP Router-Alert (RA) packets. Prefix filtering is designed for use at core access routers in order that RA packets (identified by a source/destination address) can be seamlessly forwarded across the core from one access point to another (or, conversely to be dropped at this node). RSVP applies prefix filtering rules only to RA packets because RA packets contain source and destination addresses of the RSVP flow.

Note RA packets forwarded due to prefix filtering must not be sent as RSVP bundle messages, because bundle messages are hop-by-hop and do not contain RA. Forwarding a Bundle message does not work, because the node receiving the messages is expected to apply prefix filtering rules only to RA packets.

For each incoming RSVP RA packet, RSVP inspects the IP header and attempts to match the source/destination IP addresses with a prefix configured in an extended ACL. The results are as follows:

- If an ACL does not exist, the packet is processed like a normal RSVP packet.
- If the ACL match yields an explicit permit (and if the packet is not locally destined), the packet is forwarded. The IP TTL is decremented on all forwarded packets.
- If the ACL match yields an explicit deny, the packet is dropped.

If there is no explicit permit or explicit deny, the ACL infrastructure returns an implicit (default) deny. RSVP may be configured to drop the packet. By default, RSVP processes the packet if the ACL match yields an implicit (default) deny.

Information About Implementing RSVP Authentication

Before implementing RSVP authentication, you must configure a keychain first. The name of the keychain must be the same as the one used in the keychain configuration. For more information about configuring keychains, see *Cisco IOS XR System Security Configuration Guide*.

Note RSVP authentication supports only keyed-hash message authentication code (HMAC) type algorithms.

To implement RSVP authentication on Cisco IOS XR software, you must understand the following concepts:

- RSVP Authentication Functions, page MPC-58
- RSVP Authentication Design, page MPC-58
- Global, Interface, and Neighbor Authentication Modes, page MPC-58
- Security Association, page MPC-59
- Key-source Key-chain, page MPC-60
- Guidelines for Window-Size and Out-of-Sequence Messages, page MPC-61
- Caveats for Out-of-Sequence, page MPC-61

RSVP Authentication Functions

You can carry out the following tasks with RSVP authentication:

- Set up a secure relationship with a neighbor by using secret keys that are known only to you and the neighbor.
- Configure RSVP authentication in global, interface, or neighbor configuration modes.
- Authenticate incoming messages by checking if there is a valid security relationship that is associated based on key identifier, incoming interface, sender address, and destination address.
- Add an integrity object with message digest to the outgoing message.
- Use sequence numbers in an integrity object to detect replay attacks.

RSVP Authentication Design

Network administrators need the ability to establish a security domain to control the set of systems that initiates RSVP requests.

The RSVP authentication feature permits neighbors in an RSVP network to use a secure hash to sign all RSVP signaling messages digitally, thus allowing the receiver of an RSVP message to verify the sender of the message without relying solely on the sender's IP address.

The signature is accomplished on a per-RSVP-hop basis with an RSVP integrity object in the RSVP message as defined in RFC 2747. This method provides protection against forgery or message modification. However, the receiver must know the security key used by the sender to validate the digital signature in the received RSVP message.

Network administrators manually configure a common key for each RSVP neighbor on the shared network.

The following reasons explain how to choose between global, interface, or neighbor configuration modes:

- Global configuration mode is optimal when a router belongs to a single security domain (for example, part of a set of provider core routers). A single common key set is expected to be used to authenticate all RSVP messages.
- Interface, or neighbor configuration mode, is optimal when a router belongs to more than one security domain. For example, a provider router is adjacent to the provider edge (PE), or a PE is adjacent to an edge device. Different keys can be used but not shared.

Global configuration mode configures the defaults for interface and neighbor interface modes. These modes, unless explicitly configured, inherit the parameters from global configuration mode, as follows:

- Window-size is set to 1.
- Lifetime is set to 1800.
- The **key-source key-chain** command is set to none or disabled.

Global, Interface, and Neighbor Authentication Modes

You can configure global defaults for all authentication parameters including key, window size, and lifetime. These defaults are inherited when you configure authentication for each neighbor or interface. However, you can also configure these parameters individually on a neighbor or interface basis, in which case the global values (configured or default) are no longer inherited.

Note RSVP uses the following rules when choosing which authentication parameter to use when that parameter is configured at multiple levels (interface, neighbor, or global). RSVP goes from the most specific to least specific; that is, neighbor, interface, and global.

Global keys simplify the configuration and eliminate the chances of a key mismatch when receiving messages from multiple neighbors and multiple interfaces. However, global keys do not provide the best security.

Interface keys are used to secure specific interfaces between two RSVP neighbors. Because many of the RSVP messages are IP routed, there are many scenarios in which using interface keys are not recommended. If all keys on the interfaces are not the same, there is a risk of a key mismatch for the following reasons:

- When the RSVP graceful restart is enabled, RSVP hello messages are sent with a source IP address of the local router ID and a destination IP address of the neighbor router ID. Because multiple routes can exist between the two neighbors, the RSVP hello message can traverse to different interfaces.
- When the RSVP Fast Reroute (FRR) is active, the RSVP Path and Resv messages can traverse multiple interfaces.
- When Generalized Multiprotocol Label Switching (GMPLS) optical tunnels are configured, RSVP messages are exchanged with router IDs as the source and destination IP addresses. Since multiple control channels can exist between the two neighbors, the RSVP messages can traverse different interfaces.

Neighbor-based keys are particularly useful in a network in which some neighbors support RSVP authentication procedures and others do not. When the neighbor-based keys are configured for a particular neighbor, you are advised to configure all the neighbor's addresses and router IDs for RSVP authentication.

Security Association

A security association (SA) is defined as a collection of information that is required to maintain secure communications with a peer to counter replay attacks, spoofing, and packet corruption.

Table 2 lists the main parameters that define a security association.

Table 2 Security Association Main Parameters

Parameter	Description
src	IP address of the sender.
dst	IP address of the final destination.
interface	Interface of the SA.
direction	Send or receive type of the SA.
Lifetime	Expiration timer value that is used to collect unused security association data.
Sequence Number	Last sequence number that was either sent or accepted (dependent of the direction type).
key-source	Source of keys for the configurable parameter.
keyID	Key number (returned form the key-source) that was last used.

Table 2 Security Association Main Parameters (continued)

Parameter	Description
digest	Algorithm last used (returned from the key-source).
Window Size	Specifies the tolerance for the configurable parameter. The parameter is applicable when the direction parameter is the receive type.
Window	Specifies the last *window size* value sequence number that is received or accepted. The parameter is applicable when the direction parameter is the receive type.

An SA is created dynamically when sending and receiving messages that require authentication. The neighbor, source, and destination addresses are obtained either from the IP header or from an RSVP object, such as a HOP object, and whether the message is incoming or outgoing.

When the SA is created, an expiration timer is created. When the SA authenticates a message, it is marked as recently used. The lifetime timer periodically checks if the SA is being used. If so, the flag is cleared and is cleaned up for the next period unless it is marked again.

Table 3 shows how to locate the source and destination address keys for an SA that is based on the message type.

Table 3 Source and Destination Address Locations for Different Message Types

Message Type	Source Address Location	Destination Address Location
Path	HOP object	SESSION object
PathTear	HOP object	SESSION object
PathError	HOP object	IP header
Resv	HOP object	IP header
ResvTear	HOP object	IP header
ResvError	HOP object	IP header
ResvConfirm	IP header	CONFIRM object
Ack	IP header	IP header
Srefresh	IP header	IP header
Hello	IP header	IP header
Bundle	—	—

Key-source Key-chain

The key-source key-chain is used to specify which keys to use.

You configure a list of keys with specific IDs and have different lifetimes so that keys are changed at predetermined intervals automatically, without any disruption of service. Rollover enhances network security by minimizing the problems that could result if an untrusted source obtained, deduced, or guessed the current key.

RSVP handles rollover by using the following key ID types:

- On TX, use the youngest eligible key ID.
- On RX, use the key ID that is received in an integrity object.

For more information about implementing keychain management on Cisco IOS XR Software, see *Cisco IOS XR System Security Configuration Guide*.

Guidelines for Window-Size and Out-of-Sequence Messages

The following guidelines are required for window-size and out-of-sequence messages:

- The default window-size is set to 1. If a single message is received out-of-sequence, RSVP rejects it and displays a message.
- When RSVP messages are sent in burst mode (for example, tunnel optimization), some messages can become out-of-sequence for a short amount of time.
- The window size can be increased by using the **window-size** command. When the window size is increased, replay attacks can be detected with duplicate sequence numbers.

Caveats for Out-of-Sequence

The following caveats are listed for out-of-sequence:

- When RSVP messages traverse multiple interface types with different maximum transmission unit (MTU) values, some messages can become out-of-sequence if they are fragmented.
- Packets with some IP options may be reordered.
- A change in QoS configurations may lead to a transient reorder of packets.
- QoS policies can cause a reorder of packets in a steady state.

Because all out-of-sequence messages are dropped, the sender must retransmit them. Because RSVP state timeouts are generally long, out-of-sequence messages during a transient state do not lead to a state timeout.

How to Implement RSVP

RSVP requires coordination among several routers, establishing exchange of RSVP messages to set up LSPs. Depending on the client application, RSVP requires some basic configuration, as described in the following sections:

- Configuring Traffic Engineering Tunnel Bandwidth, page MPC-61
- Confirming DiffServ-TE Bandwidth, page MPC-62
- Configuring MPLS O-UNI Bandwidth, page MPC-63
- Enabling Graceful Restart, page MPC-63
- Configuring ACL-based Prefix Filtering, page MPC-65
- Verifying RSVP Configuration, page MPC-68

Configuring Traffic Engineering Tunnel Bandwidth

To configure traffic engineering tunnel bandwidth, you must first set up TE tunnels and configure the reserved bandwidth per interface (there is no need to configure bandwidth for the data channel or the control channel).

Cisco IOS XR software supports two DS-TE modes: Prestandard and IETF.

The configuration steps for each option are described in the following sections in *Implementing MPLS Traffic Engineering on Cisco IOS XR Software*:

- Configuring a Prestandard Diff-Serv TE Tunnel, page MPC-127
- Configuring an IETF Diff-Serv TE Tunnel Using RDM, page MPC-129
- Configuring an IETF Diff-Serv TE Tunnel Using MAM, page MPC-131

Note For prestandard DS-TE you do not need to configure bandwidth for the data channel or the control channel. There is no other specific RSVP configuration required for this application.

Note When no RSVP bandwidth is specified for a particular interface, you can specify zero bandwidth in the LSP setup if it is configured under RSVP interface configuration mode or MPLS-TE configuration mode.

Confirming DiffServ-TE Bandwidth

Perform this task to confirm DiffServ-TE bandwidth.

In RSVP global and subpools, reservable bandwidths are configured per interface to accommodate TE tunnels on the node. Available bandwidth from all configured bandwidth pools is advertised using IGP. RSVP is used to signal the TE tunnel with appropriate bandwidth pool requirements.

SUMMARY STEPS

1. **configure**
2. **rsvp**
3. **interface** *interface-id*
4. **bandwidth** *total-bandwidth max-flow* **sub-pool** *sub-pool-bw*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**rsvp** **Example:** `RP/0/RP0/CPU0:router(config)# rsvp`	Enters RSVP configuration mode.

	Command or Action	Purpose
Step 3	**interface** *interface-id* **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# interface pos 0/2/0/0`	Enters interface configuration mode for the RSVP protocol.
Step 4	**bandwidth** *total-bandwidth max-flow* **sub-pool** *sub-pool-bw* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth 1000 100 sub-pool 150`	Sets the reservable bandwidth, the maximum RSVP bandwidth available for a flow and the sub-pool bandwidth on this interface.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring MPLS O-UNI Bandwidth

For this application you do not need to configure bandwidth for the data channel or the control channel. There is no other specific RSVP configuration needed for this application.

Enabling Graceful Restart

Perform this task to enable graceful restart for implementations using both node-id- and interface-based hellos.

RSVP graceful restart provides a control plane mechanism to ensure high availability, which allows detection and recovery from failure conditions while preserving nonstop forwarding services.

SUMMARY STEPS

1. **configure**
2. **rsvp**
3. **signalling graceful-restart**
4. **signalling graceful-restart interface-based**
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure terminal`	Enters global configuration mode
Step 2	**rsvp** **Example:** `RP/0/RP0/CPU0:router(config)# rsvp`	Enters the RSVP configuration submode.
Step 3	**signalling graceful-restart** **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# signalling graceful-restart`	Enables the graceful restart process on the node.

	Command or Action	Purpose
Step 4	**signalling graceful-restart interface-based** **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# signalling graceful-restart interface-based`	Enables interface-based graceful restart process on the node.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring ACL-based Prefix Filtering

This section includes two procedures associated with RSVP Prefix Filtering:

- Configuring ACLs for Prefix Filtering, page MPC-65
- Configuring RSVP Packet Dropping, page MPC-66

Configuring ACLs for Prefix Filtering

Perform this task to configure an extended access list ACL that identifies the source and destination prefixes used for packet filtering.

Note The extended ACL needs to be configured separately using extended ACL configuration commands.

SUMMARY STEPS

1. **configure**
2. **rsvp**
3. **signalling prefix-filtering access-list**

4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode
Step 2	**rsvp** **Example:** `RP/0/RP0/CPU0:router(config)# rsvp`	Enters the RSVP configuration submode.
Step 3	**signalling prefix-filtering access-list** **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# signalling prefix-filtering access-list banks`	Enter an extended access list name as a string.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring RSVP Packet Dropping

Perform this task to configure RSVP to drop RA packets when the ACL match returns an implicit (default) deny.

Note The default behavior will perform normal RSVP processing on RA packets when the ACL match returns an implicit (default) deny.

SUMMARY STEPS

1. **configure**
2. **rsvp**
3. **signalling prefix-filtering default-deny-action drop**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode
Step 2	**rsvp** **Example:** `RP/0/RP0/CPU0:router(config)# rsvp`	Enters the RSVP configuration submode.

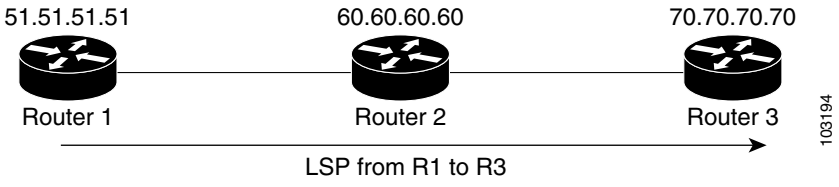
	Command or Action	Purpose
Step 3	**signalling prefix-filtering default-deny-action** **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# signalling prefix-filtering default-deny-action`	Drops RA messages.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Verifying RSVP Configuration

Figure 9 illustrates the topology that forms the basis for this section.

Figure 9 Sample Topology



To verify RSVP configuration, perform the following steps.

SUMMARY STEPS

1. **show rsvp session**
2. **show rsvp counters messages summary**
3. **show rsvp counters events**
4. **show rsvp interface** *type interface-id* [**detail**]

5. **show rsvp graceful-restart**
6. **show rsvp graceful-restart** [**neighbors** *ip-address* | **detail**]
7. **show rsvp interface**
8. **show rsvp neighbor**

DETAILED STEPS

Step 1 show rsvp session

Use this command to verify that all routers on the path of the LSP are configured with at least one Path State Block (PSB) and one Reservation State Block (RSB) per session. For example:

```
RP/0/RP0/CPU0:router# show rsvp session

Type Destination Add DPort  Proto/ExtTunID  PSBs  RSBs  Reqs
---- --------------- ----- --------------- ----- ----- -----
LSP4     172.16.70.70     6     10.51.51.51     1     1     0
```

In the example above, the output represents an LSP from ingress (head) router 10.51.51.51 to egress (tail) router 172.16.70.70. The tunnel ID (a.k.a destination port) is 6.

- If no states can be found for a session that should be up, verify the application (for example, MPLS-TE and O-UNI) to see if everything is in order.
- If a session has one PSB but no RSB, this indicates that either the Path message is not making it to the egress (tail) router or the reservation message is not making it back to the router R1 in question.

Go to the downstream router R2 and display the session information:

- If R2 has no PSB, either the path message is not making it to the router or the path message is being rejected (for example, due to lack of resources).
- If R2 has a PSB but no RSB, go to the next downstream router R3 to investigate.
- If R2 has a PSB and an RSB, this means the reservation is not making it from R2 to R1 or is being rejected.

Step 2 show rsvp counters messages summary

Use this command to verify whether RSVP message are being transmitted and received. For example:

```
RP/0/RP0/CPU0:router# show rsvp counters messages summary

 All RSVP Interfaces    Recv     Xmit                        Recv     Xmit
    Path                   0       25    Resv                   30        0
    PathError              0        0    ResvError               0        1
    PathTear               0       30    ResvTear               12        0
    ResvConfirm            0        0    Ack                    24       37
    Bundle                 0             Hello                   0     5099
    SRefresh            8974     9012    OutOfOrder              0
    Retransmit                     20    Rate Limited                     0
```

Step 3 show rsvp counters events

Use this command to see how many RSVP states have expired. Since RSVP uses a soft-state mechanism, some failures will lead to RSVP states to expire due to lack of refresh from the neighbor. For example:

```
RP/0/RP0/CPU0:router# show rsvp counters events

mgmtEthernet0/0/0/0                        tunnel6
 Expired Path states           0          Expired Path states           0
```

```
 Expired Resv states           0          Expired Resv states           0
 NACKs received                0          NACKs received                0
POS0/3/0/0                              POS0/3/0/1
 Expired Path states           0          Expired Path states           0
 Expired Resv states           0          Expired Resv states           0
 NACKs received                0          NACKs received                0
POS0/3/0/2                              POS0/3/0/3
 Expired Path states           0          Expired Path states           0
 Expired Resv states           0          Expired Resv states           1
 NACKs received                0          NACKs received                1
```

Step 4 **show rsvp interface** *type interface-id* [**detail**]

Use this command to verify that refresh reduction is working on a particular interface. For example:

```
RP/0/RP0/CPU0:router# show rsvp interface pos0/3/0/3 detail

INTERFACE: POS0/3/0/3 (ifh=0x4000D00).
 BW (bits/sec): Max=1000M. MaxFlow=1000M. Allocated=1K (0%). MaxSub=0.
 Signalling: No DSCP marking. No rate limiting.
 States in: 1. Max missed msgs: 4.
 Expiry timer: Running (every 30s). Refresh interval: 45s.
 Normal Refresh timer: Not running. Summary refresh timer: Running.
 Refresh reduction local: Enabled. Summary Refresh: Enabled (4096 bytes max).
 Reliable summary refresh: Disabled.
 Ack hold: 400 ms, Ack max size: 4096 bytes. Retransmit: 900ms.
 Neighbor information:
    Neighbor-IP    Nbor-MsgIds States-out  Refresh-Reduction Expiry(min::sec)
 -------------- -------------- ---------- ------------------ ----------------
    64.64.64.65              1          1            Enabled 14::45
```

Step 5 **show rsvp graceful-restart**

Use this command to verify that graceful restart is enabled locally. For example:

```
RP/0/RP0/CPU0:router# show rsvp graceful-restart

Graceful restart: enabled  Number of global neighbors: 1
Local MPLS router id: 10.51.51.51
Restart time: 60 seconds  Recovery time: 0 seconds
Recovery timer: Not running
Hello interval: 5000 milliseconds  Maximum Hello miss-count: 3
```

Step 6 **show rsvp graceful-restart** [**neighbors** *ip-address* | **detail**]

Use this command to verify that graceful restart is enabled on the neighbor(s). In the following examples, the neighbor 192.168.60.60 is not responding to hello messages:

```
RP/0/RP0/CPU0:router# show rsvp graceful-restart neighbors

Neighbor          App  State Recovery       Reason                Since  LostCnt
--------------- ----- ------ -------- ------------ -------------------- --------
192.168.60.60     MPLS   INIT     DONE          N/A  12/06/2003 19:01:49        0

RP/0/RP0/CPU0:router# show rsvp graceful-restart neighbors detail

Neighbor: 192.168.60.60 Source: 10.51.51.51 (MPLS)
  Hello instance for application MPLS
    Hello State: INIT        (for 3d23h)
    Number of times communications with neighbor lost: 0
    Reason: N/A
  Recovery State: DONE
  Number of Interface neighbors: 1
    address: 10.64.64.65
  Restart time: 0 seconds  Recovery time: 0 seconds
```

```
  Restart timer: Not running
  Recovery timer: Not running
  Hello interval: 5000 milliseconds  Maximum allowed missed Hello messages: 3
```

Step 7 **show rsvp interface**

Use this command to verify available RSVP bandwidth. For example:

```
RP/0/RP0/CPU0:router# show rsvp interface

Interface   MaxBW    MaxFlow  Allocated       MaxSub
----------- -------- -------- --------------- --------
Et0/0/0/0          0        0        0 (  0%)        0
PO0/3/0/0      1000M    1000M        0 (  0%)        0
PO0/3/0/1      1000M    1000M        0 (  0%)        0
PO0/3/0/2      1000M    1000M        0 (  0%)        0
PO0/3/0/3      1000M    1000M       1K (  0%)        0
```

Step 8 **show rsvp neighbor**

Use this command to verify RSVP neighbors. For example:

```
RP/0/RP0/CPU0:router# show rsvp neighbor detail

Global Neighbor: 40.40.40.40
Interface Neighbor: 1.1.1.1
Interface: POS0/0/0/0
Refresh Reduction: "Enabled" or "Disabled".
Remote epoch: 0xXXXXXXXX
Out of order messages: 0
Retransmitted messages: 0
Interface Neighbor: 2.2.2.2
Interface: POS0/1/0/0
Refresh Reduction: "Enabled" or "Disabled".
Remote epoch: 0xXXXXXXXX
Out of order messages: 0
Retransmitted messages: 0
```

How to Implement RSVP Authentication

There are three types of RSVP authentication modes—global, interface, and neighbor. The sections that follow describe how to implement RSVP authentication for each mode:

- Configuring Global Configuration Mode RSVP Authentication, page MPC-72
- Configuring an Interface for RSVP Authentication, page MPC-76
- Configuring RSVP Neighbor Authentication, page MPC-82
- Verifying the Details of the RSVP Authentication, page MPC-88
- Eliminating Security Associations for RSVP Authentication, page MPC-88

Configuring Global Configuration Mode RSVP Authentication

This section includes the following procedures for RSVP authentication in global configuration mode, as follows:

Enabling RSVP Authentication Using the Keychain in Global Configuration Mode

Perform this task to enable RSVP authentication for cryptographic authentication by specifying the keychain in global configuration mode.

Note You must configure a keychain before completing this task (see *Cisco IOS XR System Security Configuration Guide).*

SUMMARY STEPS

1. **configure**
2. **rsvp authentication**
3. **key-source key-chain** *key-chain-name*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode

Step 2	`rsvp authentication`
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-auth)#`
		Enters RSVP authentication configuration mode.

	Command or Action	Purpose
Step 3	**key-source key-chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-auth)# key-source key-chain mpls-keys`	Specifies the source of the key information to authenticate RSVP signaling messages. The *key-chain-name* argument is used to specify the name of the keychain. The maximum number of characters is 32.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Lifetime for RSVP Authentication in Global Configuration Mode

Perform this task to configure a lifetime value for RSVP authentication in global configuration mode.

SUMMARY STEPS

1. **configure**
2. **rsvp authentication**
3. **life-time** *seconds*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-auth)#`
		Enters RSVP authentication configuration mode.

Step 3	**life-time** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-auth)# life-time 2000`	Controls how long Resource Reservation Protocol (RSVP) maintains security associations with other trusted RSVP neighbors. • Use the *seconds* argument to specify the length of time (in seconds) that RSVP maintains idle security associations with other trusted RSVP neighbors. Range is from 30 to 86400. The default value is 1800.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Window Size for RSVP Authentication in Global Configuration Mode

Perform this task to configure the window size for RSVP authentication in global configuration mode.

SUMMARY STEPS

1. **configure**
2. **rsvp authentication**
3. **window-size** {*N*}
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-auth)#`
		Enters RSVP authentication configuration mode.

	Command or Action	Purpose
Step 3	**window-size** {*N*} **Example:** `RP/0/RP0/CPU0:router(config-rsvp-auth)# window-size 33`	Specifies the maximum number of Resource Reservation Protocol (RSVP) authenticated messages that can be received out-of-sequence. • Use the *N* argument to specify the Size of the window to restrict out-of-sequence messages. The range is from 1 to 64. The default value is 1, in which case all out-of-sequence messages are dropped.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring an Interface for RSVP Authentication

This section contains the following procedures for configuring an interface for RSVP authentication:

- Specifying the RSVP Authentication Keychain in Interface Mode, page MPC-76
- Configuring a Lifetime for an Interface for RSVP Authentication, page MPC-78
- Configuring the Window Size for an Interface for RSVP Authentication, page MPC-80

Specifying the RSVP Authentication Keychain in Interface Mode

Perform this task to specify RSVP authentication keychain in interface mode.

You must configure a keychain first (see *Cisco IOS XR System Security Configuration Guide*).

SUMMARY STEPS

1. **configure**
2. **rsvp interface** {*type interface-id*}
3. **authentication**
4. **key-source key-chain** *key-chain-name*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp interface** {*type interface-id*}
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp interface POS 0/2/1/0`
	`RP/0/RP0/CPU0:router(config-rsvp-if)#`
		Enters RSVP interface configuration mode.

Step 3	**authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)#`
		Enters RSVP authentication configuration mode.

	Command or Action	Purpose
Step 4	**key-source key-chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-auth)# key-source key-chain mpls-keys`	Specifies the source of the key information to authenticate RSVP signaling messages. The *key-chain-name* argument is used to specify the name of the keychain. The maximum number of characters is 32.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Lifetime for an Interface for RSVP Authentication

Perform this task to configure a lifetime for the security association for an interface.

SUMMARY STEPS

1. **configure**
2. **rsvp interface** {*type interface-id*}
3. **authentication**
4. **life-time** *seconds*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp interface** {*type interface-id*}
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp interface POS 0/2/1/0`
	`RP/0/RP0/CPU0:router(config-rsvp-if)#`
		Enters RSVP interface configuration mode.

Step 3	**authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)#`
		Enters RSVP authentication configuration mode.

	Command or Action	Purpose
Step 4	**life-time** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-auth)# life-time 2000`	Controls how long Resource Reservation Protocol (RSVP) maintains security associations with other trusted RSVP neighbors. • Use the *seconds* argument to specify the length of time (in seconds) that RSVP maintains idle security associations with other trusted RSVP neighbors. Range is from 30 to 86400. The default value is 1800.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Window Size for an Interface for RSVP Authentication

Perform this task to configure the window size for an interface for RSVP authentication to check the validity of the sequence number received.

SUMMARY STEPS

1. **configure**
2. **rsvp interface** {*type interface-id*}
3. **authentication**
4. **window-size** {*N*}
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp interface** {*type interface-id*}
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp interface POS 0/2/1/0`
	`RP/0/RP0/CPU0:router(config-rsvp-if)#`
		Enters RSVP interface configuration mode.

Step 3	**authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)#`
		Enters RSVP interface authentication configuration mode.

	Command or Action	Purpose
Step 4	**window-size** {*N*} **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-auth)# window-size 33`	Specifies the maximum number of Resource Reservation Protocol (RSVP) authenticated messages that can be received out-of-sequence. • Use the *N* argument to specify the size of the window to restrict out-of-sequence messages. The range is from 1 to 64. The default value is 1, in which case all out-of-sequence messages are dropped.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring RSVP Neighbor Authentication

This section contains the following procedures for RSVP neighbor authentication:

- Specifying the Keychain for RSVP Neighbor Authentication, page MPC-82
- Configuring a Lifetime for RSVP Neighbor Authentication, page MPC-84
- Configuring the Window Size for RSVP Neighbor Authentication, page MPC-86

Specifying the Keychain for RSVP Neighbor Authentication

Perform this task to specify the keychain RSVP neighbor authentication.

You must configure a keychain first (see *Cisco IOS XR System Security Configuration Guide).*

SUMMARY STEPS

1. **configure**
2. **rsvp neighbor** *IP address* **authentication**
3. **key-source key-chain** *key-chain-name*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp neighbor** *IP address* **authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp neighbor 1.1.1.1 authentication`
	`P/0/RP0/CPU0:router(config-rsvp-nbor-auth)#`
		Enters neighbor authentication configuration mode. Use the **rsvp neighbor** command to activate Resource Reservation Protocol (RSVP) cryptographic authentication for a neighbor.
		• Use the *IP address* argument to specify the IP address of the neighbor. A single IP address for a specific neighbor; usually one of the neighbor's physical or logical (loopback) interfaces.
		• Use the **authentication** keyword to configure the RSVP authentication parameters.

	Command or Action	Purpose
Step 3	**key-source key-chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# key-source key-chain mpls-keys`	Specifies the source of the key information to authenticate RSVP signaling messages. The *key-chain-name* argument is used to specify the name of the keychain. The maximum number of characters is 32.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Lifetime for RSVP Neighbor Authentication

Perform this task to configure a lifetime for security association for RSVP neighbor authentication mode.

SUMMARY STEPS

1. **configure**
2. **rsvp neighbor** *IP address* **authentication**
3. **life-time** *seconds*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp neighbor** *IP address* **authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp neighbor 1.1.1.1 authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)#`
		Enters RSVP neighbor authentication configuration mode. Use the **rsvp neighbor** command to specify a neighbor under RSVP.
		• Use the *IP address* argument to specify the IP address of the neighbor. A single IP address for a specific neighbor; usually one of the neighbor's physical or logical (loopback) interfaces.
		• Use the **authentication** keyword to configure the RSVP authentication parameters.

	Command or Action	Purpose
Step 3	**life-time** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# life-time 2000`	Controls how long Resource Reservation Protocol (RSVP) maintains security associations with other trusted RSVP neighbors. • Use the *seconds* argument to specify the length of time (in seconds) that RSVP maintains idle security associations with other trusted RSVP neighbors. Range is from 30 to 86400. The default value is 1800.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Window Size for RSVP Neighbor Authentication

Perform this task to configure the RSVP neighbor authentication window size to check the validity of the sequence number received.

SUMMARY STEPS

1. **configure**
2. **rsvp neighbor** *IP address* **authentication**
3. **window-size** {*N*}
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp neighbor** *IP address* **authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp neighbor 1.1.1.1 authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)#`
		Enters RSVP neighbor authentication configuration mode. Use the **rsvp neighbor** command to specify a neighbor under RSVP.
		• Use the *IP address* argument to specify the IP address of the neighbor. A single IP address for a specific neighbor; usually one of the neighbor's physical or logical (loopback) interfaces.
		• Use the **authentication** keyword to configure the RSVP authentication parameters.

	Command or Action	Purpose
Step 3	**window-size** {*N*} **Example:** `RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# window-size 33`	Specifies the maximum number of Resource Reservation Protocol (RSVP) authenticated messages that can be received out-of-sequence. • Use the *N* argument to specify the Size of the window to restrict out-of-sequence messages. The range is from 1 to 64. The default value is 1, in which case all out-of-sequence messages are dropped.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Verifying the Details of the RSVP Authentication

To display the security associations that RSVP has established with other RSVP neighbors, use the **show rsvp authentication** command.

Eliminating Security Associations for RSVP Authentication

To eliminate RSVP authentication SA's, use the **clear rsvp authentication** command. To eliminate RSVP counters for each SA, use the **clear rsvp counters authentication** command.

Configuration Examples for RSVP

The following section gives sample RSVP configurations for some of the supported RSVP features. More details on the commands can be found in the *Resource Reservation Protocol Infrastructure Commands* guide. Examples are provided for the following features:

Bandwidth Configuration (Prestandard): Example

The following example shows the configuration of bandwidth on an interface using prestandard DS-TE mode. The example configures an interface for a reservable bandwidth of 7500, specifies the maximum bandwidth for one flow to be 1000 and adds a sub-pool bandwidth of 2000:

```
rsvp interface pos 0/3/0/0
 bandwidth 7500 1000 sub-pool 2000
```

Bandwidth Configuration (MAM): Example

The following example shows the configuration of bandwidth on an interface using MAM. The example shows how to limit the total of all RSVP reservations on POS interface 0/3/0/0 to 7500 kbps, and allows each single flow to reserve no more than 1000 kbps:

```
rsvp interface pos 0/3/0/0
 bandwidth mam 7500 1000
```

Bandwidth Configuration (RDM): Example

The following example shows the configuration of bandwidth on an interface using RDM. The example shows how to limit the total of all RSVP reservations on PoS interface 0/3/0/0 to 7500 kbps, and allows each single flow to reserve no more than 1000 kbps:

```
rsvp interface pos 0/3/0/0
 bandwidth rdm 7500 1000
```

Refresh Reduction and Reliable Messaging Configuration: Example

Refresh reduction feature as defined by RFC 2961 is supported and enabled by default. The following examples illustrate the configuration for the refresh reduction feature. Refresh reduction is used with a neighbor only if the neighbor supports it also.

Changing the Refresh Interval and the Number of Refresh Messages

The following example shows how to configure the refresh interval to 30 seconds on POS 0/3/0/0 and how to change the number of refresh messages the node can miss before cleaning up the state from the default value of 4 to 6:

```
rsvp interface pos 0/3/0/0
 signalling refresh interval 30
 signalling refresh missed 6
```

Configuring Retransmit Time Used in Reliable Messaging

The following example shows how to set the retransmit timer to 2 seconds. To prevent unnecessary retransmits, the retransmit time value configured on the interface must be greater than the ACK hold time on its peer.

```
rsvp interface pos 0/4/0/1
 signalling refresh reduction reliable retransmit-time 2000
```

Configuring Acknowledgement Times

The following example shows how to change the acknowledge hold time from the default value of 400 ms, to delay or speed up sending of ACKs, and the maximum acknowledgment message size from default size of 4096 bytes.

```
rsvp interface pos 0/4/0/1
 signalling refresh reduction reliable ack-hold-time 1000
rsvp interface pos 0/4/0/1
 signalling refresh reduction reliable ack-max-size 1000
```

Note Make sure retransmit time on the peers' interface is at least twice the amount of the ACK hold time to prevent unnecessary retransmissions.

Changing the Summary Refresh Message Size

The following example shows how to set the summary refresh message maximum size to 1500 bytes:

```
rsvp interface pos 0/4/0/1
 signalling refresh reduction summary max-size 1500
```

Disabling Refresh Reduction

If the peer node does not support refresh reduction or for any other reason you want to disable refresh reduction on an interface, use the following commands to disable refresh reduction on that interface:

```
rsvp interface pos 0/4/0/1
 signalling refresh reduction disable
```

Configuring Graceful Restart: Example

RSVP graceful restart is configured globally or per interface (as are refresh-related parameters). The following examples show how to enable graceful restart, set the restart time, and change the hello message interval.

Enabling Graceful Restart

RSVP graceful restart is enabled by default. If disabled, enable it with the following command:

```
rsvp signalling graceful-restart
```

Enabling Interface-Based Graceful Restart

Configure the RSVP graceful restart feature on an interface using the following command:

```
signalling hello graceful-restart interface-based
```

Changing the Restart-Time

Configure the restart time that is advertised in hello messages sent to neighbor nodes:

```
rsvp signalling graceful-restart restart-time 200
```

Changing the Hello Interval

Configure the interval at which RSVP graceful restart hello messages are sent per neighbor, and change the number of hellos missed before the neighbor is declared down:

```
rsvp signalling hello graceful-restart refresh interval 4000
rsvp signalling hello graceful-restart refresh misses 4
```

Configuring ACL-based Prefix Filtering: Example

In the following example, when RSVP receives a Router Alert (RA) packet from source address 1.1.1.1 and 1.1.1.1 is not a local address, the packet is forwarded with IP TTL decremented. Packets destined to 2.2.2.2 are dropped. All other RA packets are processed as normal RSVP packets.

```
show run ipv4 access-list
  ipv4 access-list rsvpacl
  10 permit ip host 1.1.1.1 any
  20 deny ip any host 2.2.2.2
  !
show run rsvp
  rsvp
  signalling prefix-filtering access-list rsvpacl
  !
```

Setting DSCP for RSVP Packets: Example

The following configuration can be used to set the Differentiated Services Code Point (DSCP) field in the IP header of RSVP packets:

```
rsvp interface pos0/2/0/1
 signalling dscp 20
```

Configuration Examples for RSVP Authentication

This section provides the following configuration examples:

RSVP Authentication Global Configuration Mode: Example

The following configuration is used to enable authentication of all RSVP messages and to increase the default lifetime of the SAs:

```
rsvp
 authentication
  key-source key-chain default_keys
  life-time 3600
 !
!
```

Note The specified keychain (default_keys) must exist and contain valid keys, or signaling will fail.

RSVP Authentication for an Interface: Example

The following configuration is used to enable authentication of all RSVP messages that are being sent or received on one interface only, and sets the window-size of the SA's:

```
rsvp
 interface GigabitEthernet0/6/0/0
  authentication
   window-size 64
  !
 !
```

Note Because the key-source keychain configuration is not specified, the global authentication mode keychain is used and inherited. The global keychain must exist and contain valid keys or signaling fails.

RSVP Neighbor Authentication: Example

The following configuration is used to enable authentication of all RSVP messages that being sent to and received from only a particular IP address:

```
rsvp
 neighbor 10.0.0.1
  authentication
   key-source key-chain nbr_keys
  !
 !
!
```

RSVP Authentication by Using All the Modes: Example

The following configuration shows how to perform the following functions:

- Authenticates all RSVP messages.
- Authenticates the RSVP messages to or from 10.0.0.1 by setting the keychain for the **key-source key-chain** command to nbr_keys, SA lifetime is set to 3600, and the default window-size is set to 1.
- Authenticates the RSVP messages not to or from 10.0.0.1 by setting the keychain for the **key-source key-chain** command to default_keys, SA lifetime is set to 3600, and the window-size is set 64 when using GigabitEthernet0/6/0/0; otherwise, the default value of 1 is used.

```
rsvp
 interface GigabitEthernet0/6/0/0
  authentication
   window-size 64
  !
 !
 neighbor 10.0.0.1
  authentication
   key-source key-chain nbr_keys
  !
 !
 authentication
  key-source key-chain default_keys
  life-time 3600
 !
!
```

Note If a keychain does not exist or contain valid keys, this is considered a configuration error because signaling fails. However, this can be intended to prevent signaling. For example, when using the above configuration, if the nbr_keys does not contain valid keys, all signaling with 10.0.0.1 fails.

Additional References

The following section provides references related to implementing MPLS RSVP:

Related Documents

Related Topic	Document Title
Cisco IOS XR MPLS RSVP commands	*RSVP Infrastructure Commands on Cisco IOS XR Software* module in the *Cisco IOS XR MPLS Command Reference*
Cisco CRS-1 getting started material	Cisco IOS XR *Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
OIF2000.125.7	*User Network Interface (UNI) 1.0 Signaling Specification*

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs[1]	Title
RFC 2205	*Resource Reservation Protocol Version 1 Functional Specification*
RFC 2747	*RSVP Cryptographic Authentication*
RFC 3209	*RSVP-TE: Extensions to RSVP for LSP Tunnels*
RFC 2961	*RSVP Refresh Overhead Reduction Extensions*
RFC 3473	*Generalized MPLS Signaling, RSVP-TE Extensions*
RFC 4090	*Fast Reroute Extensions to RSVP-TE for LSP Tunnels*

1. Not all supported RFCs are listed.

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport