GRE Configuration Guide

First Published: 2025-09-15

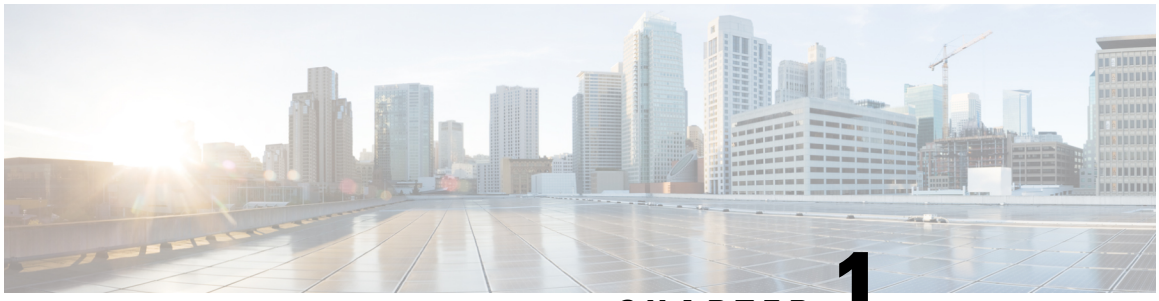
Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Read Me First

Only supported features are documented. To confirm or clarify all the supported features for a platform, go to Cisco Feature Navigator.

Read Me First

CONTENTS



CHAPTER 1

GRE Tunnel IP Source and Destination VRF Membership

Feature history for GRE tunnel IP source and destination VRF membership

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature Name and Description	Supported Platform
Cisco IOS XE 17.18.1	Generic Routing Encapsulation(GRE) Tunnel IP Source and Destination VRF Membership: GRE Tunnel IP Source and Destination VRF Membership feature allows you to configure the source and destination of a tunnel to belong to any VPN VRF table.	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

GRE tunnel IP source and destination VRF memberships

A GRE tunnel IP source and destination VRF membership is a tunneling feature that

- allows the source and destination of a GRE tunnel to be assigned to any Virtual Routing and Forwarding (VRF) table,

- associates the VPN membership of a customer site with specific VRF instances, and
- enforces that the tunnel becomes disabled if there is no route to its destination.

A VRF (Virtual Routing and Forwarding) table is a virtualized routing table that stores and maintains separate IP route information for each VPN, defining the VPN membership of each user site attached to a network access server. Each VRF table contains a unique IP routing table, a corresponding Cisco Express Forwarding (CEF) table, and relevant protocol parameters.

Previously, GRE IP tunnels required the destination to be in the global routing table. This feature allows more flexible tunnel topology by enabling both source and destination to reside within any VRF, supporting complex VPN deployments.

Restrictions for GRE Tunnel IP Source and Destination VRF Membership

This topic provides information to recognize unsupported GRE tunnel configurations tied to VRF membership.

- Both ends of the tunnel must reside within the same VRF.
- The VRF associated with the tunnel vrf command is the same as the VRF associated with the physical interface over which the tunnel sends packets (outer IP packet routing).
- The VRF associated with the tunnel by using the ip vrf forwarding command is the VRF that the packets are to be forwarded in as the packets exit the tunnel (inner IP packet routing).
- The feature does not support the fragmentation of multicast packets passing through a multicast tunnel.
- The feature does not support the ISIS (Intermediate System to intermediate system) protocol.
- Keepalive is not supported on VRF aware GRE tunnels.
- The following restrictions are applicable on the Cisco C9350 Series Smart Switches:

  IPv6 ICMP response packets are not supported over IPv4 GRE tunnels.
- The following restrictions are applicable on the Cisco C9610 Series Smart Switches:
  - Each interface must be configured with a unique combination of tunnel source and destination.
  - Only 16 unique tunnel sources are supported.
  - BFD is not supported on GRE tunnels.
  - GRE tunnels cannot be part of a routing protocol if it is formed over a Layer 3 VLAN Switch Virtual Interface (SVI).

Configure GRE tunnel IP source and destination VRF membership

Configure a GRE tunnel with source and destination interfaces associated with specific VRF instances on a Cisco device.

Use this task to enable GRE tunnels where both source and destination may belong to different VRF routing domains.

Follow these steps to configure GRE Tunnel IP Source and Destination VRF Membership:

Procedure

Step 1 **enable**

Example:

```
Device>enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device#configure terminal
```

Enters global configuration mode.

Step 3 **interface tunnel***number*

Example:

```
Device(config)#interface tunnel 0
```

Enters interface configuration mode for the specified interface.

- *number* is the number associated with the tunnel interface.

Step 4 **ip vrf forwarding***vrf-name*

Example:

```
Device(config-if)#ip vrf forwarding green
```

Associates a virtual private network (VPN) routing and forwarding (VRF) instance with an interface or subinterface.

- *vrf-name* is the name assigned to a VRF.

Step 5 **ip address***ip-address subnet-mask*

Example:

```
Device(config-if)#ip address 10.7.7.7 255.255.255.255
```

Specifies the interface IP address and subnet mask.

- *ip-address* specifies the IP address of the interface.
- *subnet-mask* specifies the subnet mask of the interface.

Step 6 **tunnel source** { *ip-address* | *type number* }

Example:

```
Device(config-if)#tunnel source loop 0
```

Specifies the source of the tunnel interface.

- *ip-address* specifies the IP address to use as the source address for packets in the tunnel.
- *type* specifies the interface type (for example, serial).
- *number* specifies the port, connector, or interface card number. The numbers are assigned at the factory at the time of installation or when added to a system, and can be displayed using the show interfaces command.

Step 7 **tunnel destination** {*hostname* | *ip-address*}

Example:

```
Device(config-if)#tunnel destination 10.5.5.5
```

Defines the tunnel destination.

- *hostname* specifies the name of the host destination.
- *ip-address* specifies the IP address of the host destination.

Step 8 **tunnel vrf***vrf-name*

Example:

```
Device(config-if)#tunnel vrf finance1
```

Associates a VPN routing and forwarding (VRF) instance with a specific tunnel destination.

- *vrf-name* is the name assigned to a VRF.

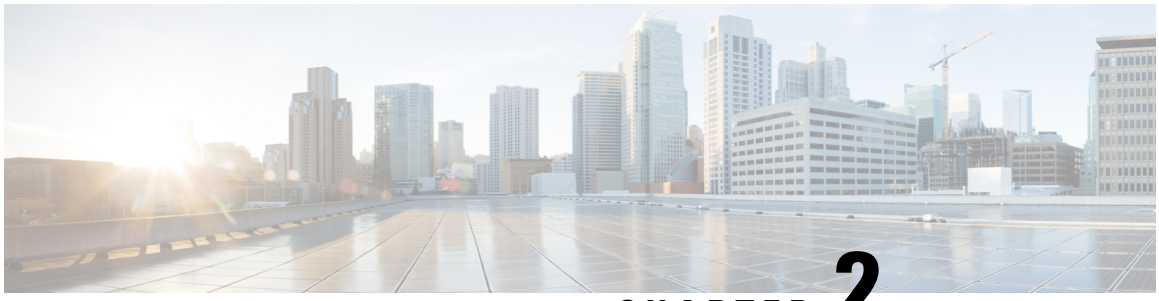
Example: GRE tunnel IP source and destination VRF memberships

This topic explains how GRE tunnel IP source and destination VRF memberships enable routing flexibility by allowing packets to enter a GRE tunnel from one VRF and exit from a different VRF, supporting complex network segmentation and overlay designs.

In this example, packets received on interface e0 using VRF green are forwarded out of the tunnel through interface e1 using VRF blue.

```
ip vrf blue rd 1:1

ip vrf green rd 1:2

interface loop0
ip vrf forwarding blue
ip address 10.7.7.7 255.255.255.255

interface tunnel0
ip vrf forwarding green
ip address 10.3.3.3 255.255.255.0 tunnel source loop 0
```

```
tunnel destination 10.5.5.5 tunnel vrf blue

interface ethernet0
ip vrf forwarding green
ip address 10.1.1.1 255.255.255.0

interface ethernet1
ip vrf forwarding blue
ip address 10.2.2.2 255.255.255.0

ip route vrf blue 10.5.5.5 255.255.255.0 ethernet 1
```



CHAPTER 2

IPv6 over IPv4 GRE Tunnels

Feature history for IPv6 over IPv4 GRE tunnels

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature Name and Description	Supported Platform
Cisco IOS XE 17.18.1	IPv6 over IPv4 GRE Tunnels: GRE tunnels are links between two points, with a separate tunnel for each link. The tunnels are not tied to a specific passenger or transport protocol, but in this case carry IPv6 as the passenger protocol with the GRE as the carrier protocol and IPv4 or IPv6 as the transport protocol.	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

Tunnel modes for IPv6 over IPv4 GRE deployment

The following sections provide information about tunnel models for IPv6 over IPv4 GRE deployment:

GRE IPv4 tunnels for IPv6 traffic

A GRE IPv4 tunnel is a tunneling mechanism that

- encapsulates IPv6 traffic within IPv4 GRE packets,

- enables point-to-point connections between dual-stack devices, and
- provides protocol-independent transport over IPv4 networks.

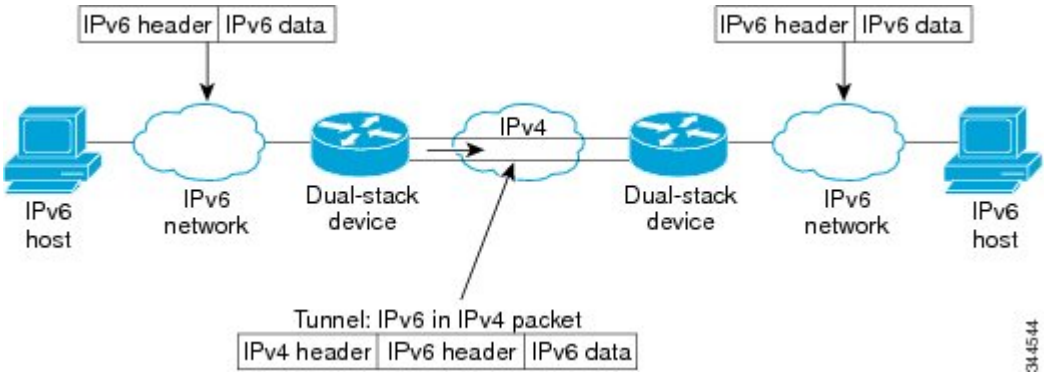
The primary use of GRE tunnels is for stable connections that require regular secure communication between two edge devices or between an edge device and an end system. The edge devices and the end systems must be dual-stack implementations.

IPv6 overlay tunnels

An overlay tunnel is a network tunneling technique that

- encapsulates one protocol (such as IPv6) within another (such as IPv4) for network transport
- enables communication between isolated networks without modifying the intermediate infrastructure, and
- requires tunnel endpoints to support both protocol stacks.

Figure 1: Overlay Tunnels



Note Overlay tunnels reduce the maximum transmission unit (MTU) of an interface by 20 octets (assuming that the basic IPv4 packet header does not contain optional fields). A network that uses overlay tunnels is difficult to troubleshoot. Therefore, overlay tunnels that connect isolated IPv6 networks should not be considered a final IPv6 network architecture. The use of overlay tunnels should be considered as a transition technique toward a network that supports both the IPv4 and IPv6 protocol stacks or just the IPv6 protocol stack.

IPv6 supports GRE type of overlay tunneling. IPv6 over IPv4 GRE Tunnels can carry IPv6, Connectionless Network Service (CLNS), and many other types of packets.

Configure GRE IPv6 tunnels

Establish a GRE tunnel over an IPv6 network to enable transport of IPv4 or IPv6 packets across IPv6 infrastructure.

Use GRE IPv6 tunnels to support routing protocols across IPv6 networks or interconnect remote IPv6 networks over intermediate IPv6 infrastructure.

Before you begin

Ensure the device supports both IPv4 and IPv6 protocol stacks.

- Verify reachability between tunnel source and destination.
- Decide on the tunnel numbers and IPv6 addresses to use.

Procedure

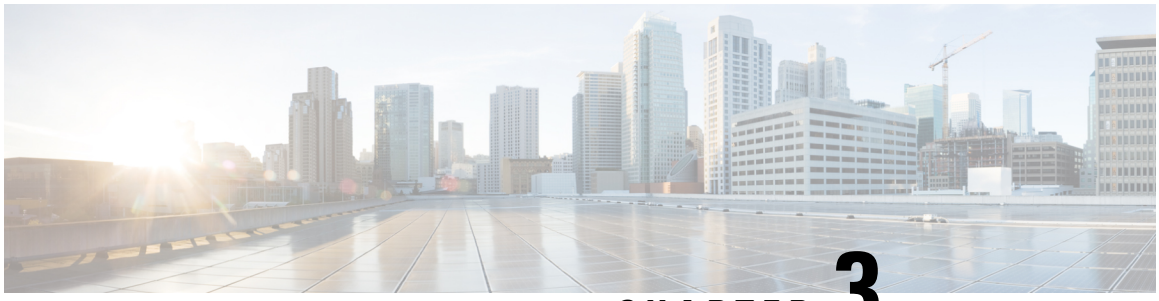
	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Device(config)# interface tunnel 0`	Specifies a tunnel interface and number, and enters interface configuration mode.
Step 4	**ipv6 address** *ipv6-prefix* / *prefix-length* [**eui-64**] **Example:** `Device(config-if)# ipv6 address 3ffe:b00:c18:1::3/127`	Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface.
Step 5	**tunnel source** {*ip-address* | *ipv6-address* | *interface-type interface-number*} **Example:** `Device(config-if)# tunnel source ethernet 0`	Specifies the source IPv4 address or the source interface type and number for the tunnel interface. • If an interface is specified, the interface must be configured with an IPv4 address.
Step 6	**tunnel destination** {*host-name* | *ip-address* | *ipv6-address*} **Example:** `Device(config-if)# tunnel destination 2001:DB8:1111:2222::1/64`	Specifies the destination IPv6 address or hostname for the tunnel interface.

	Command or Action	Purpose
Step 7	**tunnel mode** {**aurp** | **cayman** | **dvmrp** | **eon** | **gre**| **gre multipoint** | **gre ipv6** | **ipip** [**decapsulate-any**] | **iptalk** | **ipv6** | **mpls** | **nos** **Example:** `Device(config-if)# tunnel mode gre ipv6`	Specifies a GRE IPv6 tunnel. **Note** The **tunnel mode gre ipv6** command specifies GRE as the encapsulation protocol for the tunnel.

Example: IPv6 tunnel destination address configuration

To provide an example configuration for setting the tunnel destination address in an IPv6 tunnel.

```
Device> enable
Device# configure terminal
Device(config)# interface Tunnel 0
Device(config-if)# ipv6 address 2001:1:1::1/48
Device(config-if)# tunnel source GigabitEthernet 0/0/0
Device(config-if)# tunnel destination 10.0.0.2
Device(config-if)# tunnel mode gre ipv6
Device(config-if)# exit
!
Device(config)# interface GigabitEthernet0/0/0
Device(config-if)# ip address 10.0.0.1 255.255.255.0
Device(config-if)# exit
!
Device(config)# ipv6 unicast-routing
Device(config)# router isis
Device(config-router)# net 49.0000.0000.000a.00
```



CHAPTER 3

GRE over IPsec

Feature History for GRE over IPsec

This table provides release and related information for the features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature Name and Description	Supported Platform
Cisco IOS XE 17.18.1	GRE over IPsec: The GRE over IPsec feature allows a payload to be GRE encapsulated and transferred securely over an IPsec tunnel. VRF aware GRE over IPsec: VRF support was introduced for GRE over IPsec tunnels.	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

GRE over IPsec

A GRE over IPsec tunnel is a VPN solution that

- encapsulates a variety of traffic types (such as unicast, multicast, broadcast, and MPLS) using Generic Routing Encapsulation (GRE),
- secures the encapsulated packets by encrypting them through Internet Protocol Security (IPsec), and
- enables the flexibility of transporting diverse protocols over a protected IP network
- GRE does not itself provide data confidentiality or authentication; it is used mainly for encapsulating various types of network traffic. IPsec provides confidentiality, integrity, and authentication for the

payload, but it can only secure IP packets. By combining GRE and IPsec, organizations can tunnel multiple protocols securely over the Internet.

VRF aware GRE over IPsec tunnels

A Virtual Routing and Forwarding (VRF) aware GRE over IPsec tunnel is a secure tunneling mechanism that:

- associates each GRE over IPsec tunnel with two VRF domains (Front Door VRF and Inside VRF),
- encapsulates and routes packets using the FVRF for the outer packet and IVRF for the inner packet, and
- supports flexible VPN routing with route separation for service provider and customer networks.

A VRF instance is a per-VPN routing information repository that defines the VPN membership of a customer site attached to the Provider Edge (PE) router. A VRF comprises an IP routing table, a derived Cisco Express Forwarding (CEF) table, a set of interfaces that use the forwarding table, and a set of rules and routing protocol parameters that control the information that is included in the routing table. A separate set of routing and CEF tables is maintained for each VPN customer.

In a VRF aware GRE over IPsec instance, when a packet starts from the service provider network it gets encapsulated based on the security policy. The encapsulated packet is forwarded using the FVRF routing table. When the packet arrives at the PE endpoint the security associations are validated. Then the packet is decapsulated and associated with an IVRF. The packet is forwarded using the IVRF routing table.

One or more IPsec tunnels can terminate on a single interface. The FVRF of all these tunnels is the same and is set to the VRF that is configured on that interface. The IVRF of these tunnels can be different and depends on the VRF that is defined in the Internet Security Association and Key Management Protocol (ISAKMP) profile that is attached to a crypto map entry.

GRE over IPsec configuration options

The following sections explain the procedures that you can perform to configure a GRE over IPsec tunnel interface.

Configure the IKEv2 keyring

Configure the IKEv2 keyring for preshared key authentication between peers.

Use this procedure when the local or remote authentication method is a preshared key in an IKEv2 VPN setup.

On an IKEv2 initiator, the IKEv2 keyring key lookup is performed using the hostname or the address of the peer, in that order. On an IKEv2 responder, the key lookup is performed using the IKEv2 identity or the address of the peer, in that order.

Before you begin

- Access the device in privileged EXEC mode.
- Know the peer hostnames, IP addresses, identities, and preshared keys to be configured.

IKEv2 keyrings are independent of IKEv1 keyrings. The key differences are as follows:

- IKEv2 keyrings support symmetric and asymmetric preshared keys.

- IKEv2 keyrings don't support Rivest, Shamir, and Adleman (RSA) public keys.
- IKEv2 keyrings are specified in the IKEv2 profile and aren't looked up, unlike IKEv1 keys. IKEv1 keys are looked up on receipt of MM1 to negotiate the preshared key authentication method. IKEv2 doesn't negotiate the authentication method.
- IKEv2 keyrings aren't associated with VPN routing and forwarding (VRF) during configuration. The VRF of an IKEv2 keyring is the VRF of the IKEv2 profile that refers to the keyring.
- You can specify a single keyring in an IKEv2 profile, unlike an IKEv1 profile, which can specify multiple keyrings.
- If peers matching different profiles share the same keys, you can specify a single keyring in more than one IKEv2 profile, .
- An IKEv2 keyring is structured as one or more peer subblocks.

Note You can't configure the same identity in more than one peer.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto ikev2 keyring** *keyring-name* **Example:** `Device(config)# crypto ikev2 keyring kyr1`	Defines an IKEv2 keyring. Enters IKEv2 keyring configuration mode.
Step 4	**peer** *name* **Example:** `Device(config-ikev2-keyring)# peer peer1`	Defines the peer or peer group. Enters IKEv2 keyring peer configuration mode.
Step 5	**description** *line-of-description* **Example:** `Device(config-ikev2-keyring-peer)# description this is the first peer`	(Optional) Describes the peer or peer group.
Step 6	**hostname** *name* **Example:**	Specifies the peer using a hostname.

	Command or Action	Purpose
	`Device(config-ikev2-keyring-peer)# hostname host1`	
Step 7	**address** {*ipv4-address* [*mask*] | *ipv6-address prefix*} **Example:** `Device(config-ikev2-keyring-peer)# address 10.0.0.1 255.255.255.0`	Specifies an IPv4 or IPv6 address or range for the peer. **Note** This IP address is the IKE endpoint address and is independent of the identity address.

Step 8	**identity** {**address** {*ipv4-address* | *ipv6-address*} | **fqdn domain** *domain-name* | **email domain** *domain-name* | **key-id** *key-id*}
	Example:
	`Device(config-ikev2-keyring-peer)# identity address 10.0.0.5`
		Identifies the IKEv2 peer through the following identities:
		• E-mail
		• Fully qualified domain name (FQDN)
		Note
		When you use FQDN to identify the peer in the keyring configuration, use the IP address of the peer along with the FQDN
		`crypto ikev2 keyring key1`
		`peer headend-1`
		`address 10.1.1.1 >>>>>>>>>`
		`identity fqdn`
		`NFVIS-headend-1.cisco.com`
		`pre-shared-key Cisco123`
		• IPv4 or IPv6 address
		• Key ID
		Note
		The identity is available for key lookup on the IKEv2 responder only.

Step 9	**pre-shared-key** {**local** | **remote**} [**0** | **6**] *line* **hex** *hexadecimal-string* **Example:** `Device(config-ikev2-keyring-peer)# pre-shared-key local key1`	Specifies the preshared key for the peer.
Step 10	**end** **Example:** `Device(config-ikev2-keyring-peer)# end`	Exits IKEv2 keyring peer configuration mode. Returns to privileged EXEC mode.

Configure an IKEv2 profile (basic)

Create a basic IKEv2 profile with the mandatory configuration commands for establishing secure communications.

IKEv2 profiles define nonnegotiable parameters of the IKE security association (SA) and determine how the device authenticates and communicates with peers.

An IKEv2 profile is a repository of nonnegotiable parameters of the IKE security association (SA) (such as local or remote identities and authentication methods) and services available to authenticated peers that match the profile. An IKEv2 profile must be configured and associated with a crypto map. Use the **set ikev2-profile** *profile-name* command to associate a profile with a crypto map. To disassociate the profile, use the **no** form of the command.

The following rules apply to match statements:

- An IKEv2 profile must contain a match identity or a match certificate statement; otherwise, the profile is considered incomplete and is not used. An IKEv2 profile can have more than one match identity or match certificate statements.
- An IKEv2 profile must have a single match Front Door VPN routing and forwarding (FVRF) statement.
- When a profile is selected, multiple match statements of the same type are logically ORed, and multiple match statements of different types are logically ANDed.
- The match identity and match certificate statements are considered to be the same type of statements and are ORed.
- Configuration of overlapping profiles is considered a misconfiguration. In the case of multiple profile matches, no profile is selected.

Use the **show crypto ikev2 profile** *profile-name* command to display the IKEv2 profile.

Before you begin

Perform this task to configure the mandatory commands for an IKEv2 profile.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables the privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**crypto ikev2 profile** *profile-name* **Example:** `Device(config)# crypto ikev2 profile profile1`	Defines an IKEv2 profile and enters the IKEv2 profile configuration mode.
Step 4	**description** *line-of-description* **Example:** `Device(config-ikev2-profile)# description This is an IKEv2 profile`	(Optional) Describes the profile.

	Command or Action	Purpose
Step 5	**aaa accounting** {**psk** | **cert** | **eap**} *list-name* **Example:** `Device(config-ikev2-profile)# aaa accounting eap list1`	(Optional) Enables authentication, authorization, and accounting (AAA) method lists for IPsec sessions. **Note** If the **psk**, **cert**, or **eap** keyword is not specified, the AAA accounting method list is used irrespective of the peer authentication method.
Step 6	**authentication** {**local** {**rsa-sig** | **pre-share** [**key** {**0** | **6**} *password*}] | **ecdsa-sig** | **eap** [**gtc** | **md5** | **ms-chapv2**] [**username** *username*] [**password** {**0** | **6**} *password*}]} | **remote** {**eap** [**query-identity** | **timeout** *seconds*] | **rsa-sig** | **pre-share** [**key** {**0** | **6**} *password*}] | **ecdsa-sig**}} **Example:** `Device(config-ikev2-profile)# authentication local ecdsa-sig`	Specifies the local or remote authentication method. • **rsa-sig**—Specifies RSA-sig as the authentication method. • **pre-share**—Specifies the preshared key as the authentication method. • **ecdsa-sig**—Specifies ECDSA-sig as the authentication method. • **eap**—Specifies EAP as the remote authentication method. • **query-identity**—Queries the EAP identity from the peer. • **timeout** *seconds*—Specifies the duration, in seconds, to wait for the next IKE_AUTH request after sending the first IKE_AUTH response. **Note** You can specify only one local authentication method but multiple remote authentication methods.
Step 7	**dpd** *interval retry-interval* {**on-demand** | **periodic**} **Example:** `Device(config-ikev2-profile)# dpd 30 6 on-demand`	This step is optional. Configures Dead Peer Detection (DPD) globally for peers matching the profile. By default, the Dead Peer Detection (DPD) is disabled.
Step 8	**dynamic** **Example:** `Device(config-ikev2-profile)# dynamic`	Configures a dynamic IKEv2 profile. **Note** When you configure a dynamic profile, you cannot configure local or remote authentication and identity using the command line interface.

	Command or Action	Purpose
Step 9	**identity local** {**address** {*ipv4-address* | *ipv6-address*} | **dn** | **email** *email-string* | **fqdn** *fqdn-string* | **key-id** *opaque-string*} **Example:** `Device(config-ikev2-profile)# identity local email abc@example.com`	This is an optional step. Specifies the local IKEv2 identity type. **Note** If the local authentication method is a preshared key, the default local identity is the IP address. If the local authentication method is a Rivest, Shamir, and Adleman (RSA) signature, the default local identity is a Distinguished Name.
Step 10	**initial-contact force** **Example:** `Device(config-ikev2-profile)# initial-contact force`	Enforces initial contact processing if the initial contact notification is not received in the IKE_AUTH exchange.
Step 11	**ivrf** *name* **Example:** `Device(config-ikev2-profile)# ivrf vrf1`	This is an optional step. Specifies a user-defined VPN routing and forwarding (VRF) or global VRF if the IKEv2 profile is attached to a crypto map. • If you use the IKEv2 profile for tunnel protection, you must configure the Inside VRF (IVRF) for the tunnel interface on the tunnel interface. **Note** IVRF specifies the VRF for cleartext packets. The default value for IVRF is FVRF.
Step 12	**keyring** {**local** *keyring-name* | **aaa** *list-name* [**name-mangler** *mangler-name* | **password** *password*] } **Example:** `Device(config-ikev2-profile)# keyring aaa keyring1 name-mangler mangler1`	Specifies the local or AAA-based key ring that must be used with the local and remote preshared key authentication method. **Note** You can specify only one key ring. Local AAA is not supported for AAA-based preshared keys. **Note** When using AAA, the default password for a Radius access request is "cisco". You can use the **password** keyword within the **keyring** command to change the password.
Step 13	**lifetime** *seconds* **Example:** `Device(config-ikev2-profile)# lifetime 1000`	Specifies the lifetime, in seconds, for the IKEv2 SA.

	Command or Action	Purpose
Step 14	**match** {**address local** {*ipv4-address* | *ipv6-address* | **interface** *name*} | **certificate** *certificate-map* | **fvrf** {*fvrf-name* | **any**} | **identity remote address** {*ipv4-address* [*mask*] | *ipv6-address prefix*} | {**email** [*domain string*] | **fqdn** [*domain string*]} *string* | **key-id** *opaque-string*} **Example:** `Device(config-ikev2-profile)# match address local interface Ethernet 2/0`	Uses match statements to select an IKEv2 profile for a peer.
Step 15	**pki trustpoint** *trustpoint-label* [**sign** | **verify**] **Example:** `Device(config-ikev2-profile)# pki trustpoint tsp1 sign`	Specifies Public Key Infrastructure (PKI) trustpoints for use with the RSA signature authentication method. **Note** If the **sign** or **verify** keyword is not specified, the trustpoint is used for signing and verification. **Note** In contrast to IKEv1, a trustpoint must be configured in an IKEv2 profile for certificate-based authentication to succeed. There is no fallback for globally configured trustpoints if this command is not present in the configuration. The trustpoint configuration applies to the IKEv2 initiator and responder.
Step 16	**virtual-template** *number* **mode auto** **Example:** `Device(config-ikev2-profile)# virtual-template 1 mode auto`	This is an optional step. Specifies the virtual template for cloning a virtual access interface (VAI). • **mode auto** - Enables the tunnel mode auto selection feature.
Step 17	**shutdown** **Example:** `Device(config-ikev2-profile)# shutdown`	(Optional) Shuts down the IKEv2 profile.
Step 18	**end** **Example:** `Device(config-ikev2-profile)# end`	Exits the IKEv2 profile configuration mode and returns to the privileged EXEC mode.

Attaching an IKEv2 profile to an IPsec profile

Attach an IKEv2 profile to an IPsec profile to enable IKEv2 authentication for secure IPsec communications between devices.

To attach an IKEv2 profile to an IPsec profile, perform the following procedure.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto ipsec transform-set** *transform-set-name* **Example:** `Device(config)# crypto ipsec transform-set tfs`	Defines a transform set. Enters crypto transform configuration mode.
Step 4	**mode tunnel** **Example:** `Device(cfg-crypto-tran)# mode tunnel`	(Optional) Changes the mode associated with the transform set.
Step 5	**crypto IPsec profile** *profile-name* **Example:** `Device(cfg-crypto-tran)# crypto IPsec profile PROF`	Defines the IPsec parameters used for IPsec encryption between two IPsec devices. Enters IPsec profile configuration mode.
Step 6	**set transform-set** *transform-set-name* **Example:** `Device(ipsec-profile)# set transform-set tfs esp-gcm`	Specifies the transform sets used with the crypto map entry.
Step 7	**set ikev2-profile** *profile-name* **Example:** `Device(ipsec-profile)# set ikev2-profile ikev2_prof`	Attaches an IKEv2 profile to an IPSec profile.
Step 8	**exit** **Example:** `Device(ipsec-profile)# exit`	Exits IPsec profile configuration mode. Enters global configuration mode.

Configure a GRE over IPsec tunnel interface

Create a secure GRE tunnel encapsulated within IPsec to protect traffic between two endpoints.

Use this task when you need to transport GRE-encapsulated packets across a secure, encrypted IPsec tunnel.

Before you begin

- Ensure you have created and configured an IPsec profile on the device.
- Gather the tunnel source and destination IP addresses.

To create a GRE over IPsec tunnel and configure a tunnel source and tunnel destination under the tunnel interface, perform the following procedure:

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode. Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *tunnel number*

Example:

```
Device(config)# interface tunnel 100
```

Specifies the interface on which the tunnel will be configured. Enters interface configuration mode.

Step 4 **ip address** *address mask*

Example:

```
Device(config-if)# ip address 128.1.1.1 255.255.255.0
```

Specifies the IP address and mask.

Step 5 **tunnel source** *interface-type interface-number*

Example:

```
Device(config-if)# tunnel source 120.1.1.1
```

Specifies the tunnel source as a loopback interface.

Step 6 **tunnel destination** *ip-address*

Example:

```
Device(config-if)# tunnel destination 120.1.1.2
```

Identifies the IP address of the tunnel destination.

Step 7 **tunnel protection IPsec profile** *profile-name*

Example:

```
Device(config-if)# tunnel protection IPsec profile ipsec-prof
```

Associates a tunnel interface with an IPsec profile.

Step 8 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode. Returns to privileged EXEC mode.

GRE over IPsec configuration examples

The following sections provide configuration examples for GRE over IPsec.

Example: configuring GRE over IPsec

This section provides annotated examples for configuring a GRE tunnel protected by IPsec using IKEv2 on Cisco devices. Each example shows a logical step in the configuration workflow, with commands and brief explanations to help you reference or adapt the process.

The following examples show how to configure a GRE over IPsec tunnel.

The following example shows how to configure an Internet Key Exchange Version 2 (IKEv2) key ring with symmetric preshared keys based on an IP address:

```
Device(config)# crypto ikev2 keyring ikev2_key
Device(config-ikev2-keyring)# peer mypeer
Device(config-ikev2-keyring-peer)# address 0.0.0.0 0.0.0.0
Device(config-ikev2-keyring-peer)# pre-shared-key cisco123
```

The following example shows how to configure an IKEv2 profile:

```
Device(config)# crypto ikev2 profile ikev2_prof
Device(config-ikev2-profile)# match identity remote address 120.1.1.2
Device(config-ikev2-profile)# authentication remote pre-share
Device(config-ikev2-profile)# authentication local pre-share
Device(config-ikev2-profile)# keyring local ikev2_key
Device(config-ikev2-profile)# dpd 10 2 periodic
end
```

The following example shows how to attach an IKEv2 profile to an IPSec profile:

```
Device(config)# crypto ipsec transform-set tfs esp-aes esp-sha-hmac
```

```
esn
Device(cfg-crypto-tran)# mode tunnel
end
Device(cfg-crypto-tran)# crypto ipsec profile ipsec_prof
Device(ipsec-profile)# set transform-set tfs
Device(ipsec-profile)# set ikev2-profile ikev2_prof
end
```

The following example shows how to create a tunnel interface and configure a tunnel source and tunnel destination under the tunnel interface:

```
Device(config)# interface Tunnel100
Device(config-if)# ip address 128.1.1.1 255.255.255.0
Device(config-if)# tunnel source 120.1.1.1
Device(config-if)# tunnel destination 120.1.1.2
Device(config-if)# tunnel protection ipsec profile ipsec_prof
end
```

Example: configuring VRF aware GRE over IPsec

The following examples show how to configure a VRF aware GRE over IPsec tunnel.

The following example shows how to configure the FVRF instance and the IVRF instance:

```
Device# configure terminal
Device(config)# ip vrf fvrf
Device(config-vrf)# vrf definition CLIENT-VRF
Device(config-vrf)#address-family ipv4
Device(config-ipv4)# exit-address-family

Device# configure terminal
Device(config)# ip vrf ivrf
Device(config-vrf)# vrf definition WAN-VRF
Device(config-vrf)#address-family ipv4
Device(config-ipv4)# exit-address-family
```

The following example shows how to configure an Internet Key Exchange Version(IKEv2) key ring with symmetric preshared keys based on IP address:

```
Device# configure terminal
Device(config)# crypto ikev2 keyring ikev2_key
Device(config-ikev2-keyring)# peer mypeer
Device(config-ikev2-keyring-peer)# address 0.0.0.0 0.0.0.0
Device(config-ikev2-keyring-peer)# pre-shared-key cisco123
Device(config-ikev2-keyring-peer)# end
```

The following example shows how to configure an IKEv2 profile :

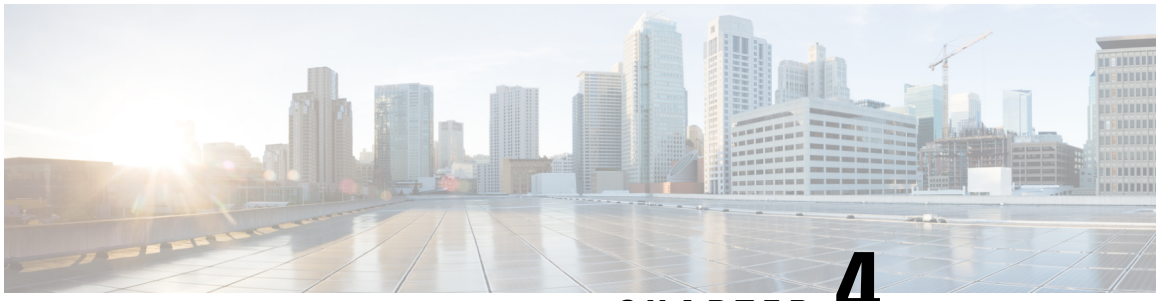
```
Device# configure terminal
Device(config)# crypto ikev2 profile ikev2_prof
Device(config-ikev2-profile)# match fvrf WAN-VRF
Device(config-ikev2-profile)#match identity remote address 130.1.1.1 255.255.255.255
Device(config-ikev2-profile)# authentication remote pre-share
Device(config-ikev2-profile)# authentication local pre-share
Device(config-ikev2-profile)# keyring local ikev2_key
Device(config-ikev2-profile)# dpd 10 2 periodic
Device(config-ikev2-profile)# end
```

The following example shows how to attach an IKEv2 profile to an IPSec profile:

```
Device# configure terminal
Device(config)# crypto ipsec transform-set tfs esp-aes esp-sha-hmac
 esn
Device(cfg-crypto-tran)# mode tunnel
Device(cfg-crypto-tran)# end

Device# configure terminal
Device(cfg-crypto-tran)# crypto ipsec profile ipsec_prof
Device(ipsec-profile)# set transform-set tfs
Device(ipsec-profile)# set ikev2-profile ikev2_prof
 responder-only
Device(ipsec-profile)# end
```

The following example shows how to configure a VRF aware GREoverIPSEC tunnel:

```
Device# configure terminal
Device(config)# interface Tunnel80
Device(config-if)# vrf forwarding CLIENT-VRF
Device(config-if)# ip address 172.16.1.1 255.255.255.0
Device(config-if)# tunnel source 130.1.1.2
Device(config-if)# tunnel destination 130.1.1.1
Device(config-if)# tunnel vrf WAN-VRF
Device(config-if)# tunnel protection ipsec profile ipsec_prof
Device(config-if)# end
```



CHAPTER 4

Configuring Multicast Routing over GRE Tunnel

Feature history for multicast routing over GRE tunnel

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature Name and Description	Supported Platform
Cisco IOS XE 17.18.1	Multicast Routing over GRE Tunnel: Multicast routing over GRE tunnel allows IP multicast traffic to be sent from a source to a multicast group, over an area where IP multicast is not supported.	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

Prerequisites for multicast routing over GRE tunnels

Before configuring multicast routing over a GRE tunnel, ensure that you understand IP multicast routing technology, are familiar with GRE tunneling concepts.

Restrictions for configuring multicast routing over GRE tunnel

The following are the restrictions for configuring multicast routing over GRE tunnel:

- IPv6 multicast over GRE tunnel is not supported.

- The total number of supported multicast routes (mroutes) is 32000, across all tunnels.
- Use the formula 8000/(((Number of tunnels)/4) + 1) to derive the number of mroutes.
- Bidirectional PIM is not supported.
- Multicast routing should be configured on the first hop router (FHR), the rendezvous point (RP) and the last hop router (LHR) to support multicast over the GRE tunnel.
- The tunnel source can be a loopback, physical, or L3 EtherChannel interface.
- No feature interactions such as IPSec, ACL, Tunnel counters, Crypto support, Fragmentation, Cisco Discovery Protocol (CDP), QoS, GRE keepalive, Multipoint GRE, etc. are supported on the GRE Tunnel.
- Tunnel source cannot be a subinterface.

Multicast routing over GRE tunnels

A multicast routing over GRE tunnel is a tunneling mechanism that

- transports IP multicast packets between network areas that do not support IP multicast natively,
- enables multicast traffic from source to multicast groups across incompatible segments, and
- supports both sparse mode and PIM SSM mode, as well as static RP and auto-RP configurations.

Benefits of tunneling to connect non-IP multicast areas

Allows multicast traffic to reach group members even if direct IP multicast routing is unavailable between source and destination.

Configure a GRE tunnel for IP multicast packets

Enable the transport of IP multicast packets between devices across a medium that does not support multicast routing.

Use this task to configure a GRE tunnel on a Cisco device to encapsulate and transport IP multicast traffic.

Before you begin

- Have the required tunnel source and destination IP addresses or interfaces.
- Ensure you have privileged EXEC access to the device.

Procedure

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

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip multicast-routing** **Example:** `Device(config)# ip multicast-routing`	Enables IP multicast routing.
Step 4	**interface tunnel** *number* **Example:** `Device(config)# interface tunnel 0`	Enters tunnel interface configuration mode.
Step 5	**ip address ip_address** *subnet_mask* **Example:** `Device(config-if)# ip address 192.168.24.1 255.255.255.252`	Configures IP address and IP subnet.
Step 6	**ip pim sparse-mode** **Example:** `Device(config-if)# ip pim sparse-mode`	Enables sparse mode of operation of Protocol Independent Multicast (PIM) on the tunnel interface with one of the following mode of operation:
Step 7	**tunnel source** { *ip-address* | *interface-name* } **Example:** `Device(config-if)# tunnel source 100.1.1.1`	Configures the tunnel source.
Step 8	**tunnel destination** { *hostname* | *ip-address* } **Example:** `Device(config-if)# tunnel destination 100.1.5.3`	Configures the tunnel destination.
Step 9	**end** **Example:** `Device(config-if)# end`	Ends the current configuration session and returns to privileged EXEC mode.

show interface *type number*

```
Device# show interface tunnel 0
```

Displays tunnel interface information.

Configure multicast routing over GRE tunnel

This section provides steps for configuring multicast routing over GRE tunnel.

Configuring a GRE Tunnel to Connect Non-IP Multicast Areas

To configure a GRE tunnel to transport IP multicast packets between a source and destination that are connected by a medium that does not support multicast routing.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip multicast-routing** **Example:** `Device(config)# ip multicast-routing`	Enables IP multicast routing.
Step 4	**interface tunnel** *number* **Example:** `Device(config)# interface tunnel 0`	Enters tunnel interface configuration mode.
Step 5	**ip address ip_address** *subnet_mask* **Example:** `Device(config-if)# ip address 192.168.24.1 255.255.255.252`	Configures IP address and IP subnet.
Step 6	**ip pim sparse-mode** **Example:** `Device(config-if)# ip pim sparse-mode`	Enables sparse mode of operation of Protocol Independent Multicast (PIM) on the tunnel interface with one of the following mode of operation:
Step 7	**tunnel source** { *ip-address* | *interface-name* } **Example:**	Configures the tunnel source.

	Command or Action	Purpose
	`Device(config-if)# tunnel source 100.1.1.1`	
Step 8	**tunnel destination** { *hostname* | *ip-address* } **Example:** `Device(config-if)# tunnel destination 100.1.5.3`	Configures the tunnel destination.
Step 9	**end** **Example:** `Device(config-if)# end`	Ends the current configuration session and returns to privileged EXEC mode.
Step 10	**show interface** *type number* **Example:** `Device# show interface tunnel 0`	Displays tunnel interface information.

What to do next

•

Tunneling to connect non-IP multicast areas

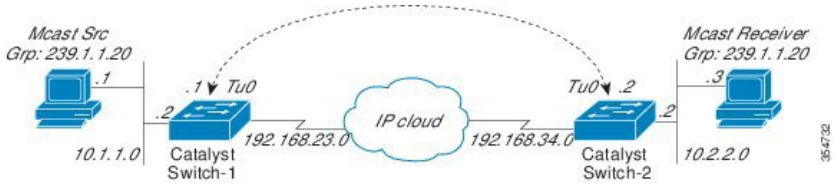
This topic provides tunneling examples to connect non-IP multicast areas.

Tunneling to connect non-IP multicast areas is a network interoperability technique that

- uses GRE tunneling to transport multicast routing packets between devices across non-multicast-enabled IP clouds,
- enables multicast group communication between separated network segments, and
- leverages PIM sparse mode to efficiently forward multicast traffic through tunnel interfaces.

The following example shows multicast-routing between a Catalyst switch through a GRE tunnel.

Figure 2: Tunnel Connecting Non-IP Multicast Areas



In the figure above, the multicast source (10.1.1.1) is connected to Catalyst Switch-1 and is configured for multicast group 239.1.1.20. The multicast receiver (10.2.2.3) is connected to Catalyst Switch-2 and is configured

to receive multicast packets for group 239.1.1.20. Separating Switch-1 and Switch-2 is an IP cloud, which is not configured for multicast routing.

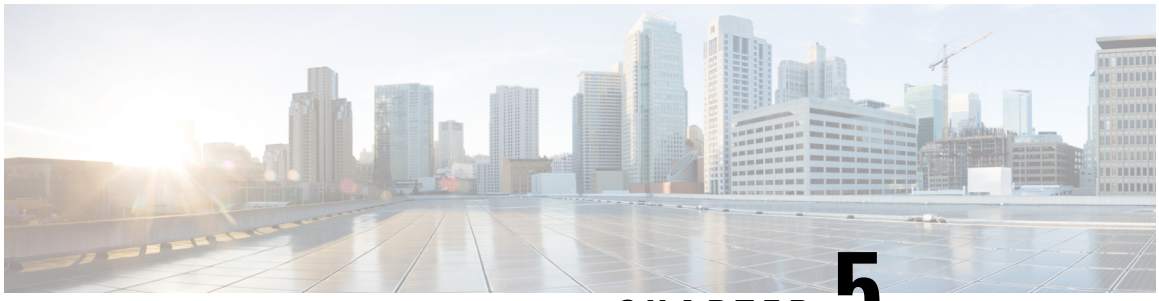
A GRE tunnel is configured between Switch-1 to Switch-2 sourced with their loopback interfaces. Multicast-routing is enabled on Switch-1 and Switch-2. The **ip pim sparse-mode** command is configured on tunnel interfaces to support PIM in the sparse mode. Sparse mode configuration on the tunnel interfaces allows sparse-mode packets to be forwarded over the tunnel depending on rendezvous point (RP) configuration for the group.

Switch-1 Configuration:

```
Device(config)# ip multicast-routing
Device(config)# interface Loopback0   //Tunnel source interface
Device(config-if)# ip address 2.2.2.2 255.255.255.255

Device(config)# interface Tunnel 10     //Tunnel interface configured for PIM
traffic
Device(config-if)# ip address 192.168.24.1 255.255.255.252
Device(config-if)# ip pim sparse-mode
Device(config-if)# ip nhrp map 192.168.24.3  4.4.4.4  //NHRP may optionally be
configured to dynamically discover tunnel end points.
Device(config-if)# ip nhrp map multicast 4.4.4.4
Device(config-if)# ip nhrp network-id 1
Device(config-if)# ip nhrp nhs 192.168.24.3
Device(config-if)# tunnel source Loopback0
Device(config-if)# tunnel destination 4.4.4.4

Device(config)# interface GigabitEthernet 0/0/0       //Source interface
Device(config-if)# ip address 10.1.1.2 255.255.255.0
Device(config-if)# ip pim sparse-mode
```

Switch-2 Configuration:

```
Device(config)# ip multicast-routing
Device(config)# interface Loopback0    //Tunnel source interface
Device(config-if)# ip address 4.4.4.4 255.255.255.255

Device(config)# interface Tunnel 10     //Tunnel interface configured for PIM
traffic
Device(config-if)# ip address 192.168.24.2 255.255.255.252
Device(config-if)# ip nhrp map 192.168.24.4 2.2.2.2  //NHRP may optionally be
configured to dynamically discover tunnel end points.
Device(config-if)# ip nhrp map multicast 2.2.2.2
Device(config-if)# ip nhrp network-id 1
Device(config-if)# ip nhrp nhs 192.168.24.4
Device(config-if)# ip pim sparse-mode
Device(config-if)# tunnel source Loopback0
Device(config-if)# tunnel destination 2.2.2.2

Device(config)# interface GigabitEthernet 0/0/0     //Receiver interface
Device(config-if)# ip address 10.2.2.2 255.255.255.0
Device(config-if)# ip pim sparse-mode
```



CHAPTER 5

Configuring Unicast & Multicast over point-to-multipoint GRE

Feature history for unicast and multicast over point-to-multipoint GRE

This table provides release and platform support information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature Name and Description	Supported Platform
Cisco IOS XE 17.18.1	Unicast and Multicast over Point-to-Multipoint GRE: The Unicast and Multicast over Point-to-Multipoint GRE feature allows to configure mGRE at the hub site and normal point-to-point GRE configuration at the spokes.	Cisco C9610 Series Smart Switches

Unicast and multicast over point-to-multipoint GRE

A point-to-multipoint GRE tunnel is a tunneling mechanism that

- supports both unicast and multicast traffic over a single GRE interface,
- enables dynamic routing protocol support across multiple remote endpoints, and
- allows efficient distribution of traffic in hub-and-spoke network topologies.

Unicast transmission sends data from one sender to one receiver, whereas multicast transmission sends data from one sender to multiple receivers simultaneously. Point-to-multipoint GRE tunnels can encapsulate both types of traffic, enabling flexible and scalable communication in large, distributed networks.

Next Hop Resolution Protocol (NHRP)

A Next Hop Resolution Protocol (NHRP) is a network protocol that

- dynamically maps non-broadcast multiaccess (NBMA) network endpoints,
- enables systems to learn the NBMA physical addresses of other systems on the network, and
- allows direct communication between NBMA-attached systems, avoiding manual tunnel configuration.

Next Hop Server (NHS): The hub that maintains the NHRP database of spoke public interface addresses.

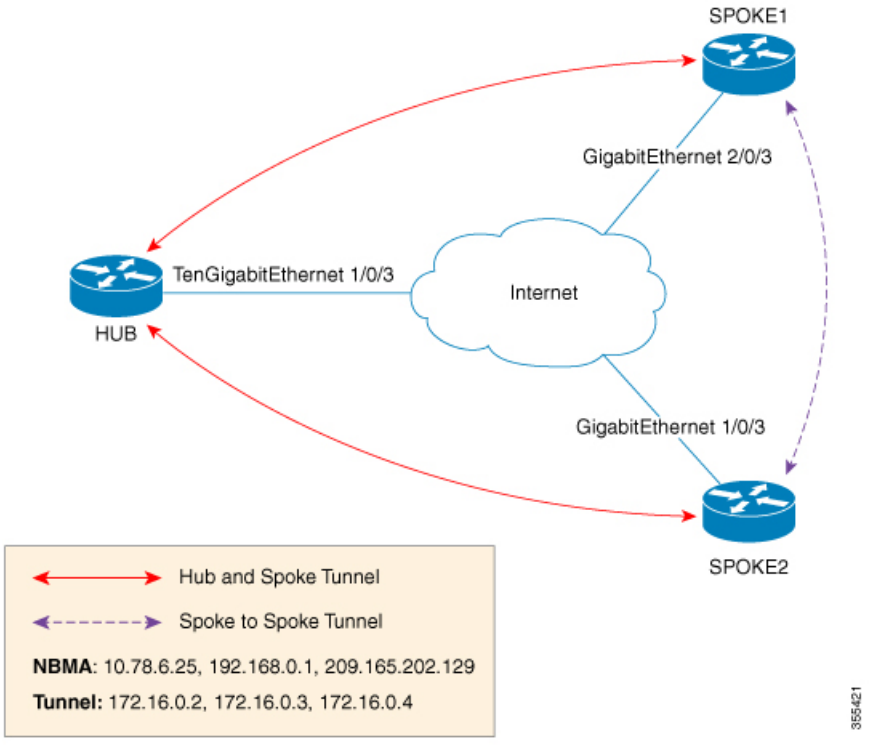
Next Hop Client (NHC): The spokes that register and query addresses with the NHS.

Multipoint GRE tunnels

A multipoint GRE tunnel (mGRE tunnel) is a type of VPN tunnel interface that

- enables communication between a central hub and multiple remote spokes across an IP network,
- allows multiple tunnel destinations to share a single logical interface, and
- supports both static and dynamic mapping methods for scaling large topologies.

Figure 3: Sample mGRE Configuration at Hub and Spokes



There are two different ways to configure mGRE on the hub and leave a normal GRE configuration on spokes:

- Static NHRP mapping statements on the hub router
- Dynamic NHRP mapping on the hub router

In static mappings, the hub router is manually configured with the spoke IP in the NHRP configuration and spokes are configured as point-to-point GRE tunnels. But if there are several branch routers, the configuration on the hub router becomes lengthy, and dynamic NHRP is used on the hub router. When using dynamic NHRP, the hub router requires that each of the spoke routers be configured to register with a Next Hop Server (NHS), which would also typically be the hub router. This NHS keeps track of the NHRP mappings so that the hub device knows where to send traffic (sent to multiple tunnel destinations). For this configuration to work correctly the IP address of the NHS server must also be statically mapped on spoke routers.

With the above hub-spoke topology, the only available way for spokes to send traffic to other spokes is to forward traffic through the hub. This requires an extra hop that may not be required when forwarding traffic. Each of the spokes has the ability to forward traffic directly to each other on the underlying IP network. When this happens, it will be more efficient for the spoke-to-spoke traffic to be routed directly between the spokes without having to jump through the hub router.

If both the hub and spokes are configured to use mGRE then the ability to set up dynamic spoke-to-spoke tunnels is permitted. With this configuration, each spoke still use the hub as an NHS which allows the hub to keep track of each of the spoke sites. It also allows mGRE and NHRP to work together to inform the spokes what the forwarding information is for the other spokes. This information can then be used for each of the spokes to dynamically set up mGRE tunnels between each of the other spokes, as required.

Configure unicast and multicast over point-to-multipoint GRE

To provide configurational information about unicast and multicast over point-to-multipoint GRE.

Configure unicast mGRE for a hub

Deploy a hub router interface using unicast multipoint GRE (mGRE) tunneling for scalable dynamic VPN connectivity.

Use this task when setting up a hub node in a DMVPN network using unicast mGRE to enable multiple dynamic spokes.

Before you begin

- Ensure you have administrative (privileged EXEC) access to the router.
- Plan your tunnel interface number, source interface, and address assignments.

Procedure

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

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Device(config)#interface tunnel 1`	Configures an interface and enters interface configuration mode.
Step 4	**tunnel mode gre multipoint** **Example:** `Device(config-if)#tunnel mode gre multipoint`	Configures multipoint GRE as the tunnel mode.
Step 5	**ip ospf network point-to-multipoint** **Example:** `Device(config-if)#ip ospf network point-to-multipoint`	If the underlying protocol is OSPF, execute this command to set the network type to point-to-multipoint.
Step 6	**ip address** *address mask* **Example:** `Device(config-if)#ip address 10.1.1.1 255.255.255.255`	Configures the IP address of the tunnel.
Step 7	**ipv6 address** *address prefix* **Example:** `Device(config-if)#ipv6 address 2001:DB8:1::1`	Configures the IPv6 address of the tunnel.
Step 8	**tunnel source** *address* **Example:** `Device(config-if)#tunnel source TenGigabitEthernet1/0/3`	Configures the source IP address of the tunnel.
Step 9	**{ip | ipv6} nhrp network-id** *id* **Example:** `Device(config-if)#ip nhrp network-id 1`	Defines the NHRP domain which differentiates if multiple NHRP domains (GRE tunnel interfaces) are available on the same NHRP router.
Step 10	**{ip | ipv6} nhrp registration timeout** *seconds* **Example:** `Device(config-if)#ip nhrp registration timeout 30`	Changes the interval that NHRP NHCs take to send NHRP registration requests to configured NHRP NHSs.

	Command or Action	Purpose
Step 11	**{ip | ipv6} nhrp holdtime** *seconds* **Example:** `Device(config-if)#ip nhrp holdtime 400`	Changes the number of seconds that NHRP NBMA addresses are advertised as valid in positive NHRP responses.
Step 12	**{ip | ipv6} nhrp authentication** *string* **Example:** `Device(config-if)#ip nhrp authentication DMVPN`	Specifies an authentication string.
Step 13	**ip pim nbma-mode** **Example:** `Device(config-if)#ip pim nbma-mode`	Configures a multiaccess WAN interface to be in non-broadcast multiaccess (NBMA) mode.
Step 14	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns to priviledged EXEC mode.

Configure unicast mGRE at a spoke

Configure a spoke router for unicast mGRE to enable dynamic IPsec connectivity with hub routers.

Unicast mGRE (Multipoint Generic Routing Encapsulation) enables dynamic and scalable VPN connectivity between hub and spoke routers in a DMVPN (Dynamic Multipoint VPN) topology. At the spoke, you must map the NBMA (Non-Broadcast Multi-Access) addresses of the hubs and configure NHRP (Next Hop Resolution Protocol) settings. This configuration allows the spoke to register with the hub and participate in secure communications.

Before you begin

- Ensure you have administrative access (privileged EXEC mode) to the spoke device.
- Obtain the NBMA and tunnel IP addresses of the hub router(s).
- Determine the tunnel interface number to use or confirm which tunnel interface will be configured.
- Confirm that the device is running software that supports mGRE and NHRP.
- (Optional) Back up existing router configurations.

Procedure

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

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Device(config)#interface tunnel 1`	Configures an interface and enters interface configuration mode.
Step 4	**ip nhrp map** *ip-address nbma-address* **Example:** `Device(config-if)#ip nhrp map 10.0.0.1 192.0.0.1`	Configures static IP-to-NBMA address mapping of a hub router on the spoke.
Step 5	**{ip | ipv6} nhrp map multicast** *nbma-address* **Example:** `Device(config-if)#ip nhrp map multicast 10.0.0.2`	Enables IP multicast and broadcast packets (example: routing protocol information) to be sent from the spoke to the hub.
Step 6	**ip nhrp nhs** *nhs-address* **Example:** `Device(config-if)#ip nhrp nhs 192.0.2.1`	Enables the spoke to send NHRP registration request to the hub. • Here *nhs-address* is the tunnel address of the hub.
Step 7	**ipv6 nhrp nhs** *nhs-address* **Example:** `Device(config-if)#ipv6 nhrp nhs 2001:DB8:1::2`	Enables the spoke to send an NHRP registration request to the hub. Here *nhs-address* is the IPv6 address of the hub tunnel.
Step 8	**ipv6 nhrp map** *address/prefix nbma address* **Example:** `Device(config-if)#ipv6 nhrp map 2001:DB8:1::3 192.0.2.2`	Configures static IPv6-to-NBMA address mapping of the hub on the spoke.
Step 9	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configure unicast mGRE at the hub

Set up unicast mGRE on the hub router to enable spoke-to-spoke communication.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Device(config)#interface tunnel 1`	Configures an interface and enters interface configuration mode.
Step 4	**{ip | ipv6} nhrp map multicast dynamic** **Example:** `Device(config-if)#ip nhrp map multicast dynamic`	Enables the NHRP server (hub) to create a broadcast/multicast mapping for the spoke when spoke routers register their unicast NHRP mapping with the hub.
Step 5	**{ip | ipv6} next-hop-self eigrp** *number* **Example:** `Device(config-if)#ip next-hop-self eigrp 10`	Enables the hub to use the next received hop while sending routing protocol updates of one spoke to another, so that hosts behind hosts can be reached directly.
Step 6	**{ip | ipv6} split-horizon eigrp** *number* **Example:** `Device(config-if)#ip split-horizon eigrp 10`	Enables routing protocol updates of one spoke to be sent to another spoke.
Step 7	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configure multicast mGRE

Enable multicast on an mGRE tunnel interface by configuring PIM in NBMA and sparse mode.

Perform these steps after you have configured unicast mGRE.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Device(config)#interface tunnel 1`	Configures an interface and enters interface configuration mode.
Step 4	**ip pim nbma-mode** **Example:** `Device(config-if)#ip pim nbma-mode`	Configures a multiaccess WAN interface to be in NBMA mode.
Step 5	**ip pim sparse-mode** **Example:** `Device(config-if)#ip pim sparse-mode`	Enables IPv4 Protocol Independent Multicast (PIM) sparse mode on an interface.
Step 6	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns to priviledged EXEC mode.

Verifying the mGRE configuration

To verify the following configuration and operation of multipoint GRE (mGRE) interfaces on Cisco devices, using the CLI commands.

Use the **show ip nhrp** command to display IPv4 Next Hop Resolution Protocol (NHRP) mapping information.

```
Spoke2#show ip nhrp 10.0.0.1

10.0.0.1/32 via 10.0.0.1
   Tunnel0 created 00:03:13, expire 00:06:47
   Type: dynamic, Flags: router used nhop
   NBMA address: 192.0.0.1

Spoke2#show ip nhrp 10.0.0.3

10.0.0.3/32 via 10.0.0.3
   Tunnel0 created 22:57:58, never expire
```

```
   Type: static, Flags: used
   NBMA address: 192.0.0.3
```

Use the **show ipv6 nhrp** command to display IPv6 Next Hop Resolution Protocol (NHRP) mapping information.

```
HUB#show running-config | interface tunnel6

Building configuration...

Current configuration : 255 bytes
!
interface Tunnel6
 no ip address
 no ip redirects
 ipv6 address 2001:DB8:1::1/64
 ipv6 eigrp 10
 no ipv6 next-hop-self eigrp 10
 no ipv6 split-horizon eigrp 10
 ipv6 nhrp network-id 1
 tunnel source FortyGigabitEthernet1/0/19
 tunnel mode gre multipoint
end

HUB#show ipv6 nhrp

2001:DB8:1::5/128 via 2001:DB8:1::5
   Tunnel6 created 02:37:30, expire 00:07:29
   Type: dynamic, Flags: registered nhop
   NBMA address: 192.168.0.2
2001:DB8:1::2A7:42FF:FE83:CEA0/128 via 2001:DB8:1::5
   Tunnel6 created 02:37:30, expire 00:07:29
   Type: dynamic, Flags: registered
   NBMA address: 192.168.0.2

HUB#

Spoke1#show running-config | interface tunnel6

Building configuration...

Current configuration : 292 bytes
!
interface Tunnel6
 no ip address
 no ip redirects
 ipv6 address 2001::5/64
 ipv6 eigrp 10
 ipv6 nhrp map multicast 192.168.0.3
 ipv6 nhrp map 2001:DB8:1::1/64 192.168.0.3
 ipv6 nhrp network-id 1
 ipv6 nhrp nhs 2001:DB8:1::1
 tunnel source FortyGigabitEthernet1/0/7
 tunnel mode gre multipoint
end

Spoke1#show ipv6 nhrp

2001:DB8:1::/64 via 2001:DB8:1::1
   Tunnel6 created 02:46:17, never expire
   Type: static, Flags:
   NBMA address: 192.168.0.3
```

```
2001:DB8:1::2A7:42FF:FE83:CFE0/128 via 2001:DB8:1::2A7:42FF:FE83:CFE0
   Tunnel6 created 02:45:39, never expire
   Type: static, Flags: nhs-ll
   NBMA address: 192.168.0.3
Spoke1#
```

Use the **show ipv6 route** command to display IPv6 content of the routing table.

```
Spoke1#show ipv6 route 2001:DB8:1::/64

Routing entry for 2001:DB8:1::/64
  Known via "eigrp 10", distance 90, metric 27008000, type internal
  Route count is 1/1, share count 0
  Routing paths:
     2001:DB8:1::2A7:42FF:FE83:CFE0, Tunnel6
      From 2001:DB8:1::2A7:42FF:FE83:CFE0
      Last updated 00:03:07 ago
Spoke1#

HUB#show ipv6 route 2001:DB8:1::/64

Routing entry for 2001:DB8:1::/64
  Known via "eigrp 10", distance 90, metric 27008000, type internal
  Route count is 1/1, share count 0
  Routing paths:
    2001:DB8:1::2A7:42FF:FE83:CEA0, Tunnel6
      From 2001:DB8:1::2A7:42FF:FE83:CEA0
      Last updated 00:01:29 ago
HUB#
```

Use the **debug nhrp detail** command to display NHRP registration and packet related information.

Use the **debug tunnel** command to display tunnel state changes and packet related information.

Configuration examples for unicast and multicast over point-to-multipoint GRE

The following sections provide configuration examples for unicast and multicast over point-to-multipoint GRE.

Example: configuring unicast mGRE for hub

The show an example of how to do a basic configuration for a unicast multipoint GRE (mGRE) tunnel interface for a hub.

```
Device>enable
Device#configure terminal
Device(config)#interface tunnel 1
Device(config-if)#tunnel mode gre multipoint
Device(config-if)#ip ospf network point-to-multipoint
Device(config-if)#ip address 10.1.1.1 255.255.255.255
```

```
Device(config-if)#ipv6 address 2001:DB8:1::1
Device(config-if)#tunnel source TenGigabitEthernet1/0/3
Device(config-if)#ip nhrp network-id 1
Device(config-if)#ip nhrp registration timeout 30
Device(config-if)#ip nhrp holdtime 400
Device(config-if)#ip nhrp authentication DMVPN
Device(config-if)#ip pim nbma-mode
Device(config-if)#end
```

Example: configuring unicast mGRE at spoke

To show how to configure unicast mGRE at a spoke.

```
Device>enable
Device#configure terminal
Device(config)#interface tunnel 1
Device(config-if)#ip nhrp map 10.0.0.1 192.0.0.1
Device(config-if)#ip nhrp map multicast 10.0.0.2
Device(config-if)#ip nhrp nhs 192.0.2.1
Device(config-if)#ipv6 nhrp nhs 2001:DB8:1::2
Device(config-if)#ipv6 nhrp map 2001:DB8:1::3 192.0.2.2
Device(config-if)#end
```

Example: configuring unicast mGRE for hub

The show an example of how to do a basic configuration for a unicast multipoint GRE (mGRE) tunnel interface for a hub.

```
Device>enable
Device#configure terminal
Device(config)#interface tunnel 1
Device(config-if)#tunnel mode gre multipoint
Device(config-if)#ip ospf network point-to-multipoint
Device(config-if)#ip address 10.1.1.1 255.255.255.255
Device(config-if)#ipv6 address 2001:DB8:1::1
Device(config-if)#tunnel source TenGigabitEthernet1/0/3
Device(config-if)#ip nhrp network-id 1
Device(config-if)#ip nhrp registration timeout 30
Device(config-if)#ip nhrp holdtime 400
Device(config-if)#ip nhrp authentication DMVPN
Device(config-if)#ip pim nbma-mode
Device(config-if)#end
```

Example: configuring multicast mGRE

To display an example on how to configure multicast mGRE.

```
Device>enable
Device#configure terminal
Device(config)#interface tunnel 1
Device(config-if)#ip pim nbma-mode
Device(config-if)#ip pim sparse-mode
Device(config-if)#end
```

Example: configuring mGRE at hub and spokes

To displays examples for configuring mGRE at hub and spokes.

Configuration at hub:

```
Device(config)#interface Tunnel0
Device(config-if)#ip address 172.16.0.2 255.255.255.0
Device(config-if)#no ip redirects
Device(config-if)#ip nhrp authentication DMVPN
Device(config-if)#ip nhrp network-id 1
Device(config-if)#ip nhrp registration timeout 30
Device(config-if)#no ip next-hop-self eigrp 10
Device(config-if)#no ip split-horizon eigrp 10
Device(config-if)#tunnel source TenGigabitEthernet1/0/3
Device(config-if)#tunnel mode gre multipoint
Device(config-if)#tunnel key 4
Device(config-if)#end
Device(config)#interface TenGigabitEthernet1/0/3
Device(config-if)#no switchport
Device(config-if)#ip address 10.78.6.25. 255.255.255.0
Device(config-if)#end
```

Configuration at spoke1:

```
Device(config)#interface Tunnel0
Device(config-if)#ip address 172.16.0.4 255.255.255.0
Device(config-if)#no ip redirects
Device(config-if)#ip nhrp authentication DMVPN
Device(config-if)#ip nhrp map 172.16.0.2 10.78.6.25
Device(config-if)#ip nhrp map multicast 10.78.6.25
Device(config-if)#ip nhrp network-id 1
Device(config-if)#ip nhrp nhs 172.16.0.2
Device(config-if)#ip nhrp registration timeout 30
Device(config-if)#tunnel source GigabitEthernet2/0/3
Device(config-if)#tunnel mode gre multipoint
Device(config-if)#tunnel key 4
Device(config-if)#end
Device(config)#interface GigabitEthernet2/0/3
Device(config-if)#no switchport
Device(config-if)#ip address 209.165.202.129 255.255.255.0
Device(config-if)#end
```

Configuration at spoke2:

```
Device(config)#interface Tunnel0
Device(config-if)#ip address 172.16.0.3 255.255.255.0
Device(config-if)#no ip redirects
Device(config-if)#ip nhrp authentication DMVPN
Device(config-if)#ip nhrp map 172.16.0.2 10.78.6.25
Device(config-if)#ip nhrp map multicast 10.78.6.25
Device(config-if)#ip nhrp network-id 1
Device(config-if)#ip nhrp nhs 172.16.0.2
Device(config-if)#ip nhrp registration timeout 30
Device(config-if)#tunnel source GigabitEthernet1/0/3
Device(config-if)#tunnel mode gre multipoint
Device(config-if)#tunnel key 4
```

```
Device(config-if)#end
Device(config)#interface GigabitEthernet1/0/3
Device(config-if)#no switchport
Device(config-if)#ip address 192.168.0.1 255.255.255.0
Device(config-if)#end
```