

GRE Tunnel Keepalives

(Cisco Systems)

Introduction

This document explains what GRE keepalives are and how they work. The GRE tunnel keepalives are not supported in conjunction with the tunnel protection ipsec profile command. This document discusses this issue. Note: GRE keepalives are not supported together with IPSec tunnel protection under any circumstances.

GRE Tunnel Keepalive Mechanism GRE Tunnels

GRE tunnels are designed to be completely stateless. This means that each tunnel end-point does not keep any information about the state or availability of the remote tunnel end-point. A consequence of this is that the local tunnel end-point router does not have the ability to bring the line protocol of the GRE Tunnel interface down if the remote end of the tunnel is unreachable. The ability to mark an interface as down when the remote end of the link is not available is used in order to remove any routes (specifically static routes) in the routing table that use that interface as the outbound interface. Specifically, if the line protocol for an interface is changed to down, then any static routes that point out that interface are removed from the routing table. This allows for the installation of an alternate (floating) static route or for Policy Based Routing (PBR) to select an alternate next-hop or interface.

Normally, a GRE Tunnel interface comes up as soon as it is configured and it stays up as long as there is a valid tunnel source address or interface which is up. The tunnel destination IP address must also be routable. This is true even if the other side of the tunnel has not been configured. This means that a static route or PBR forwarding of packets via the GRE tunnel interface remains in effect even though the GRE tunnel packets do not reach the other end of the tunnel.

Before GRE keepalives were implemented, there were only three reasons for a GRE tunnel to shut down:

- There is no route to the tunnel destination address.
- The interface that anchors the tunnel source is down.
- The route to the tunnel destination address is through the tunnel itself.

These three rules (missing route, interface down and mis-routed tunnel destination) are problems local to the router at the tunnel endpoints and do not cover problems in the intervening network. For example, these rules do not cover the case in which the GRE tunneled packets are successfully forwarded, but are lost before they reach the other end of the tunnel. This causes data packets that go through the GRE tunnel to be "black holed", even though an alternate route that uses PBR or a floating static route via another interface is potentially available. Keepalives on the GRE tunnel interface are used in order to solve this issue in the same way as keepalives are used on physical interfaces.

With Cisco IOS® Software Release 12.2(8)T, it is possible to configure keepalives on a point-to-point GRE tunnel interface. With this change, the tunnel interface dynamically shuts down if the keepalives fail for a certain period of time. In order to better understand how GRE tunnel keepalives work, these sections discuss some other common keepalive mechanisms.

Common Keepalive Mechanisms

Keepalive messages are sent by one network device via a physical or virtual circuit to inform another network device that the circuit between them still functions. The keepalive interval is the period of time between each keepalive message that is sent by a network device. The keepalive retries is the number of times that the device continues to send keepalive packets without response before the interface is brought down.

GRE Tunnel Keepalives

(Cisco Systems)

Ethernet Keepalives

On broadcast media like an Ethernet, keepalives are slightly different. Since there are many possible neighbors on the Ethernet, the keepalive is not designed to determine if the path to any one particular neighbor on the wire is available. It is only designed to check that the local system has read and write access to the Ethernet wire itself. The router produces an Ethernet packet with itself as the source and destination MAC-address and a special Ethernet type code of 0x9000. The Ethernet hardware sends this packet onto the Ethernet wire and then immediately receives this packet back again. This checks the sending and receiving hardware on the Ethernet adapter and the basic integrity of the wire.

Source MAC 00-00-0C-04-EF-04	Destination MAC 00-00-0C-04-EF-04	Protocol Type 9000	Data 0000 0100	Layer-2 Padding 0000 ... 0000
---------------------------------	--------------------------------------	-----------------------	-------------------	----------------------------------

HDLC Keepalives

Another well known keepalive mechanism is serial keepalives for HDLC. Serial keepalives are sent back and forth between two routers and the keepalives are acknowledged. With the use of sequence numbers, each router keeps track of the keepalive packets sent and acknowledged. In this way, the remote routers look at each others keepalives and track if the keepalives they send are received.

As an illustration of how serial keepalives work, Router 1 and Router 2 are directly connected via Serial0/0 and Serial2/0 respectively. In the Router 1 output, Serial 0/0 is shut down purposely. This causes Router 2 to miss three keepalives in order to illustrate how this failure causes Router 2 to shut down Serial2/0 when keepalives are missed.

This is sample output from the debug serial interface command for an HDLC connection when keepalives are enabled. When the difference in the values in the myseq and mineseen fields exceeds 3 on Router 2, the line goes down and the interface is reset.

```
Router 1
17:21:09.685: Serial0/0: HDLC myseq 0, mineseen 0*, yourseen 1, line up
17:21:19.725: Serial0/0: HDLC myseq 1, mineseen 1*, yourseen 2, line up
17:21:29.753: Serial0/0: HDLC myseq 2, mineseen 2*, yourseen 3, line up
17:21:39.773: Serial0/0: HDLC myseq 3, mineseen 3*, yourseen 4, line up
17:21:49.805: Serial0/0: HDLC myseq 4, mineseen 4*, yourseen 5, line up
17:21:59.837: Serial0/0: HDLC myseq 5, mineseen 5*, yourseen 6, line up
17:22:09.865: Serial0/0: HDLC myseq 6, mineseen 6*, yourseen 7, line up
17:22:19.905: Serial0/0: HDLC myseq 7, mineseen 7*, yourseen 8, line up
17:22:29.945: Serial0/0: HDLC myseq 8, mineseen 8*, yourseen 9, line up
Router1 (config-if)#shut
17:22:39.965: Serial0/0: HDLC myseq 9, mineseen 9*, yourseen 10, line up
17:22:42.225: %LINK-5-CHANGED: Interface Serial0/0, changed state
to administratively down
17:22:43.245: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0,
changed state to down
```

```
*Sep 24 17:21:04.929: Serial2/0: HDLC myseq 0, mineseen 0, yourseen 0, line up
*Sep 24 17:21:14.941: Serial2/0: HDLC myseq 1, mineseen 1*, yourseen 1, line up
*Sep 24 17:21:24.961: Serial2/0: HDLC myseq 2, mineseen 2*, yourseen 2, line up
*Sep 24 17:21:34.981: Serial2/0: HDLC myseq 3, mineseen 3*, yourseen 3, line up
*Sep 24 17:21:45.001: Serial2/0: HDLC myseq 4, mineseen 4*, yourseen 4, line up
*Sep 24 17:21:55.021: Serial2/0: HDLC myseq 5, mineseen 5*, yourseen 5, line up
*Sep 24 17:22:05.041: Serial2/0: HDLC myseq 6, mineseen 6*, yourseen 6, line up
*Sep 24 17:22:15.061: Serial2/0: HDLC myseq 7, mineseen 7*, yourseen 7, line up
*Sep 24 17:22:25.081: Serial2/0: HDLC myseq 8, mineseen 8*, yourseen 8, line up
*Sep 24 17:22:35.101: Serial2/0: HDLC myseq 9, mineseen 9*, yourseen 9, line up
*Sep 24 17:22:45.113: Serial2/0: HDLC myseq 10, mineseen 10*, yourseen 10, line
```

GRE Tunnel Keepalives (Cisco Systems)

```
up*Sep 24 17:22:55.133: Serial2/0: HDLC myseq 11, mineseen 10, yourseen 10, line
up *Sep 24 17:23:05.153: HD(0): Reset from 0x203758 *Sep 24 17:23:05.153: HD(0):
Asserting DTR *Sep 24 17:23:05.153: HD(0): Asserting DTR and RTS *Sep 24
17:23:05.153: Serial2/0: HDLC myseq 12, mineseen 10, yourseen 10, line up *Sep 24
17:23:15.173: HD(0): Reset from 0x203758 *Sep 24 17:23:15.173: HD(0): Asserting
DTR *Sep 24 17:23:15.173: HD(0): Asserting DTR and RTS *Sep 24 17:23:15.173:
Serial2/0: HDLC myseq 13, mineseen 10, yourseen 10, line down
17:23:16.201: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed
state to down Router2#
```

```
Router 2
*Sep 24 17:21:04.929: Serial2/0: HDLC myseq 0, mineseen 0, yourseen 0, line up
*Sep 24 17:21:14.941: Serial2/0: HDLC myseq 1, mineseen 1*, yourseen 1, line up
*Sep 24 17:21:24.961: Serial2/0: HDLC myseq 2, mineseen 2*, yourseen 2, line up
*Sep 24 17:21:34.981: Serial2/0: HDLC myseq 3, mineseen 3*, yourseen 3, line up
*Sep 24 17:21:45.001: Serial2/0: HDLC myseq 4, mineseen 4*, yourseen 4, line up
*Sep 24 17:21:55.021: Serial2/0: HDLC myseq 5, mineseen 5*, yourseen 5, line up
*Sep 24 17:22:05.041: Serial2/0: HDLC myseq 6, mineseen 6*, yourseen 6, line up
*Sep 24 17:22:15.061: Serial2/0: HDLC myseq 7, mineseen 7*, yourseen 7, line up
*Sep 24 17:22:25.081: Serial2/0: HDLC myseq 8, mineseen 8*, yourseen 8, line up
*Sep 24 17:22:35.101: Serial2/0: HDLC myseq 9, mineseen 9*, yourseen 9, line up
*Sep 24 17:22:45.113: Serial2/0: HDLC myseq 10, mineseen 10*, yourseen 10, line up
*Sep 24 17:22:55.133: Serial2/0: HDLC myseq 11, mineseen 10, yourseen 10, line up
*Sep 24 17:23:05.153: HD(0): Reset from 0x203758
*Sep 24 17:23:05.153: HD(0): Asserting DTR
*Sep 24 17:23:05.153: HD(0): Asserting DTR and RTS
*Sep 24 17:23:05.153: Serial2/0: HDLC myseq 12, mineseen 10, yourseen 10, line up
*Sep 24 17:23:15.173: HD(0): Reset from 0x203758
*Sep 24 17:23:15.173: HD(0): Asserting DTR
*Sep 24 17:23:15.173: HD(0): Asserting DTR and RTS
*Sep 24 17:23:15.173: Serial2/0: HDLC myseq 13, mineseen 10, yourseen 10, line down
17:23:16.201: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0,
changed state to down
Router2#
```

```
*Sep 24 17:21:04.929: Serial2/0: HDLC myseq 0, mineseen 0, yourseen 0, line up
*Sep 24 17:21:14.941: Serial2/0: HDLC myseq 1, mineseen 1*, yourseen 1, line up
*Sep 24 17:21:24.961: Serial2/0: HDLC myseq 2, mineseen 2*, yourseen 2, line up
*Sep 24 17:21:34.981: Serial2/0: HDLC myseq 3, mineseen 3*, yourseen 3, line up
*Sep 24 17:21:45.001: Serial2/0: HDLC myseq 4, mineseen 4*, yourseen 4, line up
*Sep 24 17:21:55.021: Serial2/0: HDLC myseq 5, mineseen 5*, yourseen 5, line up
*Sep 24 17:22:05.041: Serial2/0: HDLC myseq 6, mineseen 6*, yourseen 6, line up
*Sep 24 17:22:15.061: Serial2/0: HDLC myseq 7, mineseen 7*, yourseen 7, line up
*Sep 24 17:22:25.081: Serial2/0: HDLC myseq 8, mineseen 8*, yourseen 8, line up
*Sep 24 17:22:35.101: Serial2/0: HDLC myseq 9, mineseen 9*, yourseen 9, line up
*Sep 24 17:22:45.113: Serial2/0: HDLC myseq 10, mineseen 10*, yourseen 10, line up
*Sep 24 17:22:55.133: Serial2/0: HDLC myseq 11, mineseen 10, yourseen 10, line up
*Sep 24 17:23:05.153: HD(0): Reset from 0x203758
*Sep 24 17:23:05.153: HD(0): Asserting DTR
*Sep 24 17:23:05.153: HD(0): Asserting DTR and RTS
*Sep 24 17:23:05.153: Serial2/0: HDLC myseq 12, mineseen 10, yourseen 10, line up
*Sep 24 17:23:15.173: HD(0): Reset from 0x203758
*Sep 24 17:23:15.173: HD(0): Asserting DTR
*Sep 24 17:23:15.173: HD(0): Asserting DTR and RTS
*Sep 24 17:23:15.173: Serial2/0: HDLC myseq 13, mineseen 10, yourseen 10, line down
17:23:16.201: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0,
changed state to down
Router2#
```

GRE Tunnel Keepalives

The GRE tunnel keepalive mechanism is slightly different than for Ethernet or serial interfaces. It gives the ability for one side to originate and receive keepalive packets to and from a remote router even if the remote router does not support GRE keepalives. Since GRE is a packet tunneling mechanism for tunneling IP inside IP, a GRE IP tunnel packet can be built inside another GRE IP tunnel packet. For GRE keepalives, the sender pre-builds the keepalive response packet inside the original keepalive request packet so that the remote end only needs to do standard GRE decapsulation of the outer GRE IP header and then forward the inner IP GRE packet. This mechanism causes the keepalive response to forward out the physical interface rather than the tunnel interface. This means that the GRE keepalive response packet is not affected by any output features on the tunnel interface, such as 'tunnel protection ...', QoS, and so forth.).

GRE Tunnel Keepalives (Cisco Systems)

Note: If an inbound ACL on the GRE tunnel interface is configured, then the GRE tunnel keepalive packet that the opposite device sends must be permitted. If not, the opposite device's GRE tunnel will be down. (access-list <number> permit gre host <tunnel-source> host <tunnel-destination>)

Another attribute of GRE tunnel keepalives is that the keepalive timers on each side are independent and do not have to match. The problem with the configuration of keepalives only on one side of the tunnel is that only the router that has keepalives configured marks its tunnel interface as down if the keepalive timer expires. The GRE tunnel interface on the other side, where keepalives are not configured, remains up even if the other side of the tunnel is down. The tunnel can become a black-hole for packets directed into the tunnel from the side that did not have keepalives configured. In a large hub-and-spoke GRE tunnel network, it might be appropriate to only configure GRE keepalives on the spoke side and not on the hub side. This is because it is often more important for the spoke to discover that the hub is unreachable and therefore switch to a backup path (Dial Backup for example).

How Tunnel Keepalives Work

A GRE tunnel is a logical interface on a Cisco router that provides a way to encapsulate passenger packets inside a transport protocol. It is an architecture designed to provide the services to implement a point-to-point encapsulation scheme. These packets illustrate the IP tunneling concepts where GRE is the encapsulation protocol and IP is the transport protocol. The passenger protocol is also IP (although it can be another protocol like Decnet, IPX or Appletalk).

Normal Packet



Tunnel Packet

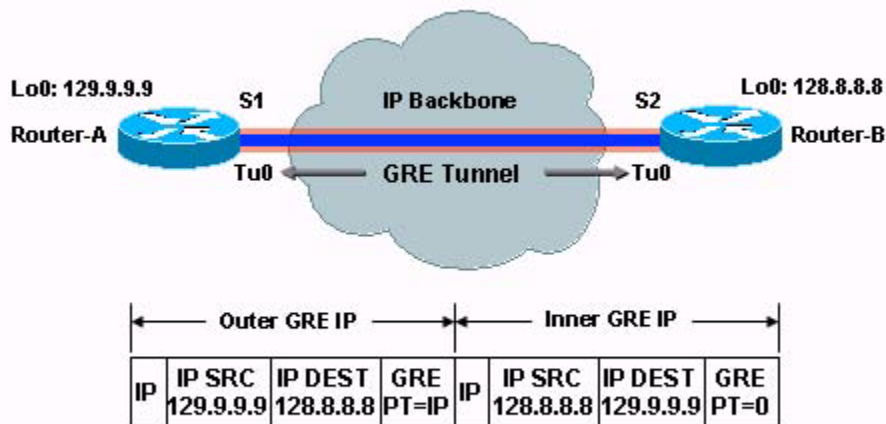


- IP is the transport protocol.
- GRE is the encapsulation protocol.
- IP is the passenger protocol.

In order to better understand how the tunnel keepalive mechanism works, consider this example tunnel topology and configuration. The physical interfaces of Router A and Router B are S1 and S2 respectively and the tunnel interfaces are called Tu0. There is an IP backbone network in between the two GRE tunnel endpoint routers.

This is an example of a keepalive packet that originates from Router A and is destined for Router B. The keepalive response that Router B returns to Router A is already inside the Inner IP Header. Router B simply decapsulates the keepalive packet and sends it back out the physical interface (S2). It processes the GRE keepalive packet just like any other GRE IP data packet.

GRE Tunnel Keepalives (Cisco Systems)



This output shows the commands you use in order to configure keepalives on GRE tunnels.

```
Router# configure terminal Router(config)#interface tunnel0 Router(config-if)#keepalive 5 4
```

!--- The syntax of this command is `keepalive [seconds [retries]]`.

!--- Keepalives are sent every 5 seconds and 4 retries. !--- Keepalives must be missed before the tunnel is shut down. !--- The default values are 10 seconds for the interval and 3 retries.

Note: GRE tunnel keepalives are only supported on point-to-point GRE tunnels. Tunnel keepalives are configurable on multipoint GRE (mGRE) tunnels but have no effect.

This table shows the configuration of Router A and Router B with tunnel keepalive configured on both routers.

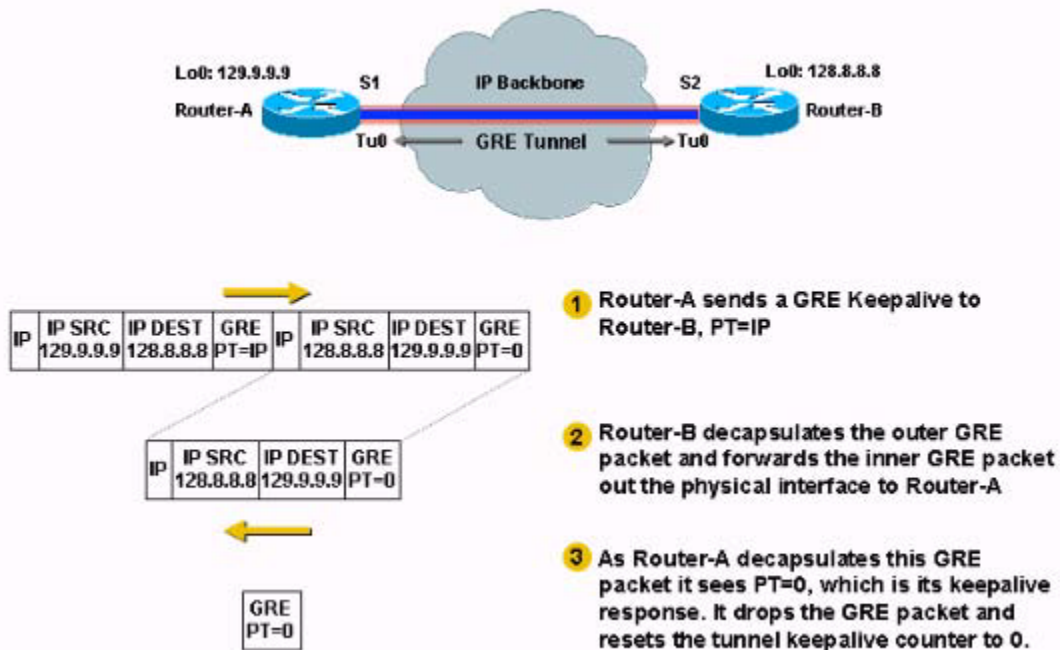
Hostname Router-A	Hostname Router-B
<pre>interface loopback 0 ip address 129.9.9.9 255.255.255.255 interface tunnel 0 ip address 1.1.1.1 255.255.255.240 tunnel source loopback0 tunnel destination 128.8.8.8 keepalive 5 4</pre>	<pre>interface loopback 0 ip address 128.8.8.8 255.255.255.255 interface tunnel 0 ip address 1.1.1.2 255.255.255.240 tunnel source loopback0 tunnel destination 129.9.9.9 keepalive 5 4</pre>

When you enable keepalives on the tunnel endpoint of Router A, the router at every <period> interval constructs the inner IP header and GRE header with a Protocol Type (PT) of 0. It then sends that packet out its tunnel interface, which results in the encapsulation of the packet with the outer IP header and a GRE header with PT = IP. Router A increments the tunnel keepalive counter by one. With the assumption that there is a way to reach the far end tunnel endpoint and the tunnel line protocol is not down due to other reasons, the packet arrives on Router B. It is then matched against Tunnel 0, becomes decapsulated, and is forwarded to the destination IP which is the tunnel source IP address on Router A. Upon arrival on Router A, the packet becomes decapsulated and the check of the PT results in 0. This signifies that this is a keepalive packet. The tunnel keepalive counter is then reset to 0 and the packet is discarded.

GRE Tunnel Keepalives (Cisco Systems)

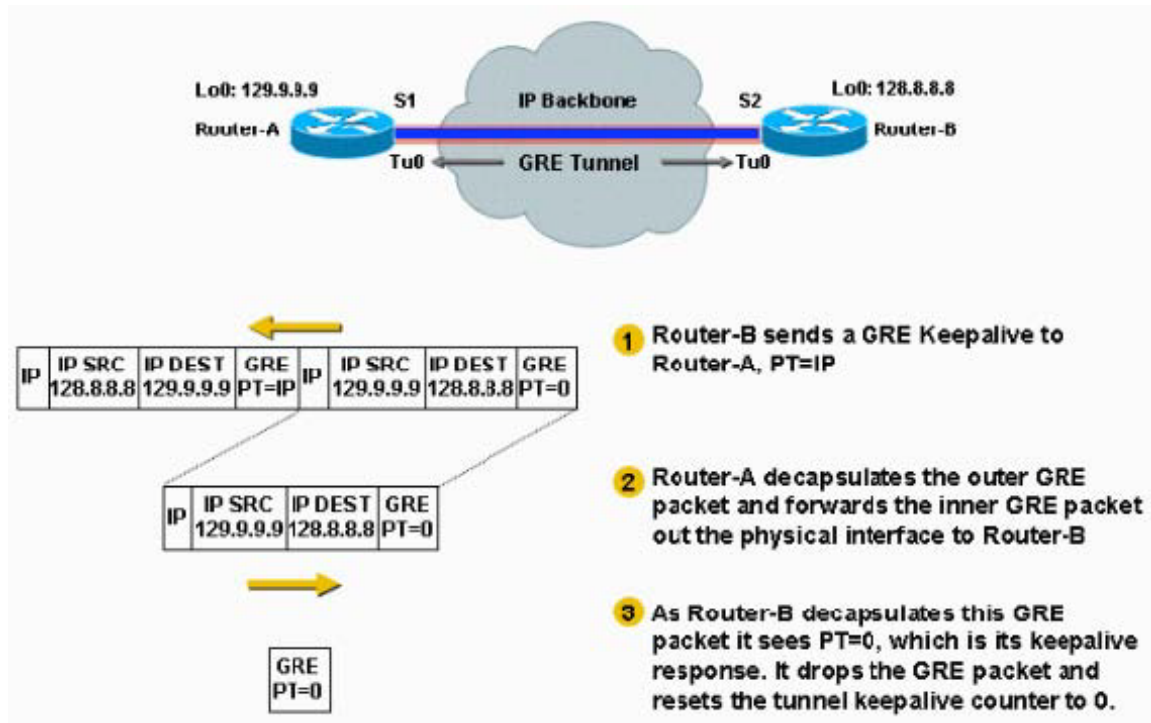
In the case when Router B is unreachable, Router A continues to construct and send keepalive packets as well as normal traffic. If the keepalives do not come back, the tunnel line protocol stays up as long as the tunnel keepalive counter is less than the number of <retries>. If that condition is not true, then the next time Router A attempts to send a keepalive to Router B, the line protocol is brought down.

In the up/down state, the tunnel does not forward or process any data traffic. However, it does continue to send keepalive packets. On the reception of a keepalive response, with the implication that the tunnel endpoint is again reachable, the tunnel keepalive counter is reset to 0 and the line protocol on the tunnel comes up. This diagram shows a sample scenario of what happens when Router A sends a GRE keepalive to Router B:



This diagram shows a sample scenario of what happens when Router B sends a GRE keepalive to Router A:

GRE Tunnel Keepalives (Cisco Systems)

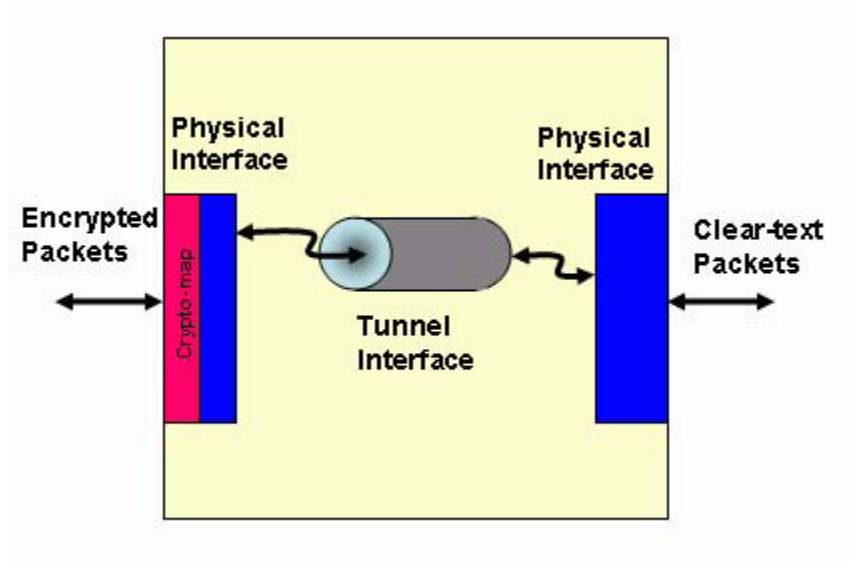


IPSec and GRE Keepalives GRE Tunnels with IPSec

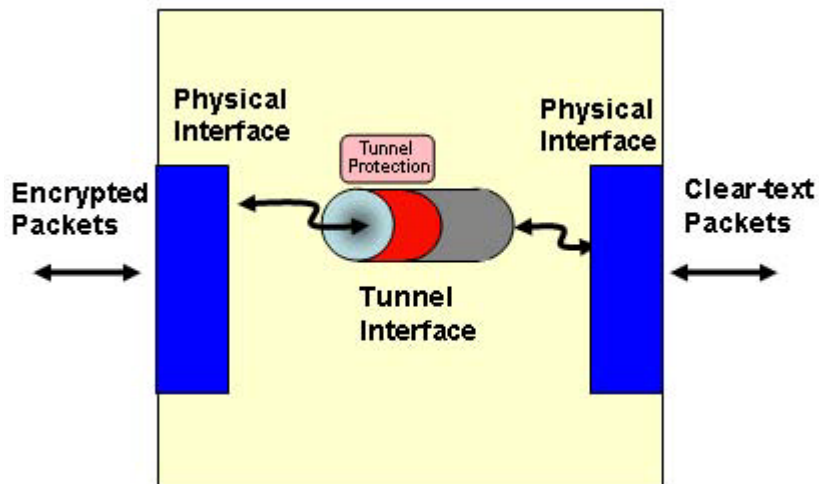
GRE tunnels are sometimes combined with IPSec because IPSec does not support IP multicast packets. This prevents dynamic routing protocols from running successfully over an IPSec VPN network. Since GRE tunnels do support IP multicast, a dynamic routing protocol can be run over a GRE tunnel. Then, you can encrypt the GRE IP unicast packets by IPSec. There are two different ways that IPSec can encrypt GRE packets. One way is with the use of a crypto-map and the other is to use tunnel protection. Both methods specify that IPSec encryption is performed after the addition of the GRE encapsulation. When a crypto-map is used, it is applied to the outbound physical interface(s) for the GRE tunnel packets. When tunnel protection is used, it is configured on the GRE tunnel interface. The tunnel protection command became available in Cisco IOS Software Release 12.2(13)T.

Note: GRE keepalives are not compatible with tunnel protection. This diagram shows encrypted packets that come into a router via a GRE tunnel interface. The router has a crypto-map applied on the physical interface. Once the packets are de-crypted and de-encapsulated, they continue on to their IP destination as clear-text.

GRE Tunnel Keepalives (Cisco Systems)



This diagram shows what happens when you configure tunnel protection on the GRE tunnel interface. The packets come into the router encrypted via the tunnel interface and are de-crypted and de-capsulated before they continue towards their destination as clear-text.



There are two key differences between when you use a crypto-map and when you use tunnel protection:

- The IPsec crypto-map is tied to the physical interface and is checked as packets are forwarded out the physical interface. Note: The GRE tunnel has already GRE encapsulated the packet by this point.
- Tunnel protection ties the encryption functionality to the GRE tunnel and is checked after the packet is GRE encapsulated but before the packet is handed to the physical interface.