Overview of Keepalive Mechanisms on Cisco IOS

Document ID: 118390

Contributed by Atri Basu and Michael Sullenberger, Cisco TAC Engineers.
Dec 17, 2014

Contents

Introduction

This document describes the various keepalive mechanisms on Cisco IOS®.

Background Information

Keepalive messages are sent by one network device via a physical or virtual circuit in order to inform another network device that the circuit between them still functions. For keepalives to work there are two essential factors:

- The keepalive interval is the period of time between each keepalive message that is sent by a network device. This is always configurable.
- The keepalive retries is the number of times that the device continues to send keepalive packets without response before the state is changed to "down". For some types of keepalives this is configurable, while for others there is a default value that cannot be changed.

Interface Keepalive Mechanisms

Ethernet Interfaces

On broadcast media such as an Ethernet, keepalives are slightly unique. 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	Destination MAC	Protocol Type	Data	Layer-2 Padding
00-00-0C-04-EF-04	00-00-0C-04-EF-04	9000	0000 0100	0000 ... 0000

Serial Interfaces

Serial interfaces can have different types of encapsulations and each encapsulation type determines the kind of keepalives that will be used.

Enter the ***keepalive*** command in interface configuration mode in order to set the frequency at which a router sends ECHOREQ packets to its peer:

- In order to restore the system to the default keepalive interval of 10 seconds, enter the ***keepalive*** command with the ***no*** keyword.
- In order to disable keepalives, enter the ***keepalive disable*** command.

Note: The ***keepalive*** command applies to serial interfaces that use High−Level Data Link Contol (HDLC) or PPP encapsulation. It does not apply to serial interfaces that use Frame Relay encapsulation.

Note: For both PPP and HDLC encapsulation types, a keepalive of zero disables keepalives and is reported in the ***show running−config*** command output as ***keepalive disable***.

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 to track each keepalive, each device is able to confirm if it's HDLC peer received the keepalive it sent. For HDLC encapsulation, three ignored keepalives causes the interface to be brought down.

Enable the ***debug serial interface*** command for an HDLC connection in order to allow the user to see keepalives that are generated and sent:

```
Sample Output:
17:21:09.685: Serial0/0: HDLC myseq 0, mineseen 0*, yourseen 1, line up
```

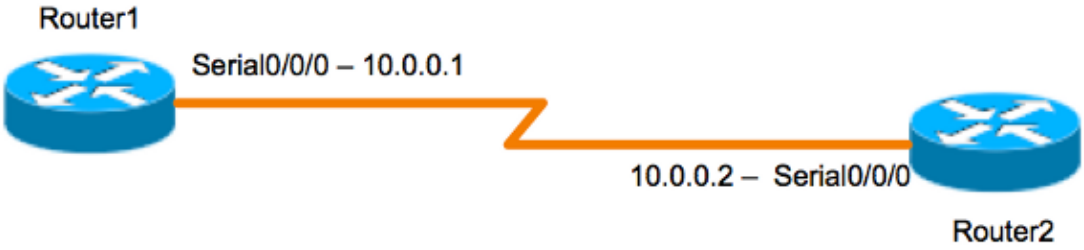
HDLC keepalives contain three pieces in order to determine it works:

- The "myseq" which is our own incrementing number.
- The "mineseen" which is actually an acknowledgement from the other side (incremented) which says they expect this number from us.
- The "yourseen" which is our acknowledgement to the other side.

Note: When the difference in the values in the myseq and mineseen fields exceeds three on Router 2, the line goes down and the interface is reset.

Since HDLC keepalives are ECHOREQ type keepalives, the keepalive frequency is important and it is recommended that they match up exactly on both sides. If the timers are out of sync, the sequence numbers start to get out of order. For example, if you set one side to 10 seconds and the other to 25 seconds, it will still allow the interface to remain up as long as the difference in frequency is not sufficient to cause the sequence numbers to be off by a difference of three.

As an illustration of how HDLC keepalives work, Router 1 and Router 2 are directly connected via Serial0/0 and Serial2/0 respectively. In order to illustrate how failed HDCL keepalives are used to track the interface states, Serial 0/0 will be shut down on Router 1.



Router 1

```
Router1#show interfaces serial 0/0/0
Serial0/0/0 is up, line protocol is up (connected)
Hardware is HD64570
Internet address is 10.0.0.1/8
MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
Encapsulation HDLC, loopback not set, keepalive set (10 sec)
    [output is omited]

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
```

Router 2

```
Router2#show interfaces serial 0/0/0
Serial0/0/0 is up, line protocol is up (connected)
Hardware is HD64570
Internet address is 10.0.0.2/8
MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
Encapsulation HDLC, loopback not set, keepalive set (10 sec)
    [output is omited]


17:21:04.929: Serial2/0: HDLC myseq 0, mineseen 0, yourseen 0, line up
17:21:14.941: Serial2/0: HDLC myseq 1, mineseen 1*, yourseen 1, line up
17:21:24.961: Serial2/0: HDLC myseq 2, mineseen 2*, yourseen 2, line up
17:21:34.981: Serial2/0: HDLC myseq 3, mineseen 3*, yourseen 3, line up
17:21:45.001: Serial2/0: HDLC myseq 4, mineseen 4*, yourseen 4, line up
17:21:55.021: Serial2/0: HDLC myseq 5, mineseen 5*, yourseen 5, line up
17:22:05.041: Serial2/0: HDLC myseq 6, mineseen 6*, yourseen 6, line up
17:22:15.061: Serial2/0: HDLC myseq 7, mineseen 7*, yourseen 7, line up
17:22:25.081: Serial2/0: HDLC myseq 8, mineseen 8*, yourseen 8, line up
17:22:35.101: Serial2/0: HDLC myseq 9, mineseen 9*, yourseen 9, line up
17:22:45.113: Serial2/0: HDLC myseq 10, mineseen 10*, yourseen 10, line up
17:22:55.133: Serial2/0: HDLC myseq 11, mineseen 10, yourseen 10, line up
17:23:05.153: HD(0): Reset from 0x203758
```

```
17:23:05.153: HD(0): Asserting DTR
17:23:05.153: HD(0): Asserting DTR and RTS
17:23:05.153: Serial2/0: HDLC myseq 12, mineseen 10, yourseen 10, line up
17:23:15.173: HD(0): Reset from 0x203758
17:23:15.173: HD(0): Asserting DTR
17:23:15.173: HD(0): Asserting DTR and RTS
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#
17:23:25.193: Serial2/0: HDLC myseq 14, mineseen 10, yourseen 10, line down
```

PPP Keepalives

PPP keepalives are a little bit different from HDLC keepalives. Unlike HDLC, PPP keepalives are more like pings. Both sides can ping each other at their leisure. The proper negotiated move is to ALWAYS respond to this "ping". So for PPP keepalives, the frequency or the timer value are only locally relevant and have no impact on the other side. Even if one side turns the keepalives off, it will still RESPOND to those echo requests from the side that does have a keepalive timer. However, it will never initiate any of its own.

Enable the ***debug ppp packet*** command for a PPP connection in order to allow the user to see the PPP keepalives that are sent:

```
17:00:11.412: Se0/0/0 LCP-FS: I ECHOREQ [Open] id 32 len 12 magic 0x4234E325
```

and responses that are received:

```
17:00:11.412: Se0/0/0 LCP-FS: O ECHOREP [Open] id 32 len 12 magic 0x42345A4D
```

PPP keepalives contain three pieces:

- ID number − used to identify which ECHOREQ the peer responds to.
- Keepalive type − ECHOREQ are keepalives sent by the originating device and ECHOREP are responses sent by the peer.
- Magic numbers − the notifications include the magic numbers of both the server and the remote client. The peer validates the magic number in the LCP Echo−Request packet, and transmits the corresponding LCP Echo−Reply packet that contains the magic number negotiated by the router.

For PPP encapsulation, five ignored keepalives causes the interface to be brought down

GRE Tunnel Interfaces

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. For more details on the working of GRE tunnel keepalives, see How GRE Keepalives Work.

Crypto Keepalives

IKE Keepalives

Internet Key Exchange (IKE) keepalives are a mechanism used to determine if a VPN peer is up and able to receive encrypted traffic. Separate crypto keepalives are required in addition to interface keepalives because VPN peers are generally never connected back to back, so interface keepalives do not provide enough information about state of the VPN peer.

On Cisco IOS devices, IKE keepalives are enabled by the use of a proprietary method called Dead Peer Detection (DPD). In order to allow the gateway to send DPDs to the peer, enter this command in global configuration mode:

crypto isakmp keepalive *seconds [retry−seconds]* [***periodic*** **|** ***on−demand***]

In order to disable keepalives, use the "no" form of this command. For more information on what each keyword in this command does, see crypto isakmp keepalive. For more granularity, the keepalives can also be configured under the ISAKMP profile. For more details, see ISAKMP Profile Overview [Cisco IOS IPsec].

NAT Keepalives

In case of scenarios where one VPN peer is behind a Network Address Translation (NAT), NAT−Traversal is used for encryption. However, during idle periods it is possible that the NAT entry on the upstream device might time out. This can cause problems when you bring up the tunnel and NAT is not bidirectional. NAT keepalives are enabled in order to keep the dynamic NAT mapping alive during a connection between two peers. NAT keepalives are UDP packets with an unencrypted payload of one byte. Although the current DPD implementation is similar to NAT keepalives, there is a slight difference − DPD is used to detect peer status while NAT keepalives are sent if the IPsec entity did not send or receive the packet at a specified period of time. The valid range is between 5 to 3600 seconds.

Tip: If NAT keepalives are enabled (through the ***crypto isamkp nat keepalive*** command), users should ensure that the idle value is shorter than the NAT mapping expiration time of 20 seconds.

For more information on this feature, see IPsec NAT Transparency.

Updated: Dec 17, 2014 Document ID: 118390