

Understanding and Troubleshooting Gatekeeper TTL and Aging out Process

Document ID: 18732

Introduction

Prerequisites

Requirements

Components Used

Conventions

Time to Live Tips

Cisco Gatekeeper Debugs

Related Information

Introduction

This document describes how the Cisco Gatekeeper ages out the endpoints using the Time to Live (TTL) value in different cases. Debugs and **show** commands are used to show how the TTL works in various ways.

Prerequisites

Requirements

Readers of this document should be knowledgeable of these topics:

- Cisco H.323 implementation, including Cisco Gatekeeper. For a basic understanding of H.323 Gatekeepers, refer to Understanding H.323 Gatekeepers.

Components Used

The information in this document is based on these software and hardware versions.

- For the purpose of this document, Cisco IOS® Software Release 12.3(9) is used to collect the information.

Ensure that you use Cisco IOS with the H.323 Gatekeeper functionality feature. This is denoted with an **x** in the Cisco IOS image name. For example, a valid Cisco IOS for the Cisco 3640 to act as a gatekeeper is c3640-ix-mz.123-9.bin.

- Cisco Gatekeeper (all platforms).

Note: NetMeeting was used as an H.323 endpoint in the example in this document because it does not provide a TTL value.

For information on how to configure NetMeeting with Cisco IOS Gateways, refer to How-to Configure Microsoft NetMeeting with Cisco IOS Gateways.

The information presented in this document was created from devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If you are working in a live network, ensure that you understand the potential impact of any command before using it.

Conventions

For more information on document conventions, see the Cisco Technical Tips Conventions.

Time to Live Tips

These are some tips on how TTL works on a Cisco Gatekeeper and how the aging process works for the endpoints in different cases.

- The Cisco Gatekeeper expects to hear from the endpoint periodically via the Registration Request (RRQ) messages (either lightweight or full).
- For endpoint timeouts, the Cisco Gatekeeper pays attention to the TTL value supplied by the endpoint in the RRQ message. There is a hard coded default of 1800 seconds (thirty minutes) for the endpoint timeout. This value can be changed with the Cisco Gatekeeper CLI command **endpoint ttl** *<time_value>* . This changes the behavior for all H.323 v1 endpoints, or H.323 v2 and later endpoints that do not include the TTL value in the RRQ message.
- The Cisco Gatekeeper runs an "endpoint aging process" periodically. This process varies based on current CPU load from every one minute to every five minutes. For every twenty percent of CPU utilization, the aging interval increases by one minute, up to a maximum of five minutes. In order not to overload the CPU when there are many endpoints, the aging process only runs through fifty endpoints per pass. If there are more, then those are deferred until the next timer pop. This can be from one to five minutes.
- If the timeToLive field is included in the RRQ Registration, Admission and Status (RAS) message, the gatekeeper uses that field's value to override the system default or the value configured using the **endpoint ttl** *<time_value>* gatekeeper CLI command. Once that time period elapses without hearing from the endpoint, the next timer pop goes through the cleanup process for that endpoint. The worst case is the TTL sent by the endpoint plus five minutes (if the Cisco Gatekeeper is consistently under heavy CPU load). A more likely worst case scenario is the TTL timeout plus one minute.
- When the endpoint does not include the timeToLive field in the RRQ message, the Cisco Gatekeeper treats the endpoint as if it does not support TTL. In this case, once the gatekeeper no longer receives RRQs from that endpoint, it does the TTL timeout (either the default of 1800 seconds, or the value specified in the **endpoint ttl** command). Then it sends out three Information Requests (IRQs) with intervals of anywhere from one to five minutes each (based on the CPU load of the gatekeeper). After it sends three IRQs, and does not receive any response, the Cisco Gatekeeper finally removes the endpoint.
- If the endpoint is in an active call, it is not aged out until the call is terminated.
- The Cisco Gatekeeper expects to hear from the endpoint(s) involved in a call with the Information Response (IRR) message. If the Cisco Gatekeeper does not receive periodic IRR messages that contain references to the "guid" for a call, the gatekeeper waits for four minutes, and then sends out an IRQ to the endpoint(s) the call is for. If, after eight more minutes, the Cisco Gatekeeper still has not heard anything about that call, the call is cleaned up and the gatekeeper sends Disengage Requests (DRQs) to the endpoints. A total of approximately twelve minutes elapses before a "dangling" call is cleaned up (and the bandwidth freed). This call timer is not configurable.
- Endpoints that are owned by an alternate Cisco Gatekeeper are not aged out directly (since this gatekeeper does not actually "own" the endpoint).
- Static endpoints (created with the configuration command **alias static xxxxx**) are not aged out.

Cisco Gatekeeper Debugs

These are some of the **show** and **debug** commands you can use to verify how the TTL works in various ways:

- **show gatekeeper endpoints**
- **debug ras**

- **debug h225 asn1**

These two sections discuss two cases where Cisco Gatekeeper ages out the endpoints using different TTL values.

Case 1

This output is from the **debug ras** and **debug h225 asn1** commands and is taken from a Cisco Gatekeeper. In the debug, the gateway has a TTL value of 60 seconds. The Cisco Gatekeeper confirms and accepts this in its Registration Confirmation (RCF) message, no matter what its default or configured endpoint TTL value is. This is because the endpoint includes a TTL value.

```
Mar  2 23:52:50.797: RAS INCOMING ENCODE BUFFER ::= 0E 400FD206 0008914A
00030000 0100AC10 0D2AE26A 00040067 006B0062 0
02D0032 00B50000 12128F00 02003B01 80211E00 36003100 36004600 32004400
43004300 30003000 30003000 30003000 30003101 00
0180
Mar  2 23:52:50.797:
Mar  2 23:52:50.797: RAS INCOMING PDU ::=
```

```
value RasMessage ::= registrationRequest :
{
  requestSeqNum 4051
  protocolIdentifier { 0 0 8 2250 0 3 }
  discoveryComplete FALSE
  callSignalAddress
  {
  }
  rasAddress
  {
    ipAddress :
    {
      ip 'AC100D2A'H
      port 57962
    }
  }
  terminalType
  {
    mc FALSE
    undefinedNode FALSE
  }
  gatekeeperIdentifier {"gkb-2"}
  endpointVendor
  {
    vendor
    {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
    }
  }
  timeToLive 60

  !--- TTL value.

  keepAlive TRUE
  endpointIdentifier {"616F2DCC00000001"}
  willSupplyUUIEs FALSE
  maintainConnection TRUE
}
```

```
Mar  2 23:52:50.805: RRQ (seq# 4051) rcvd
```

```
Mar 2 23:52:50.805: RAS OUTGOING PDU ::=
```

```
value RasMessage ::= registrationConfirm :
{
  requestSeqNum 4051
  protocolIdentifier { 0 0 8 2250 0 3 }
  callSignalAddress
  {
  }
  gatekeeperIdentifier {"gkb-2"}
  endpointIdentifier {"616F2DCC00000001"}
  alternateGatekeeper
  {
    {
      rasAddress ipAddress :
      {
        ip 'AC100D29'H
        port 1719
      }
      gatekeeperIdentifier {"gkb-1"}
      needToRegister TRUE
      priority 0
    }
  }
  timeToLive 60
  willRespondToIRR FALSE
  maintainConnection TRUE
}
```

```
Mar 2 23:52:50.813: RAS OUTGOING ENCODE BUFFER ::= 12 400FD206 0008914A
00030008 0067006B 0062002D 00321E00 36003100 3
6004600 32004400 43004300 30003000 30003000 30003000 3000310F 8A140140
AC100D29 06B70800 67006B00 62002D00 31800200 3B
010001 80
```

```
Mar 2 23:52:50.813:
```

```
Mar 2 23:52:50.817: IPSOCK_RAS_sendto: msg length 86 from
172.16.13.16:1719 to 172.16.13.42: 57962
```

```
Mar 2 23:52:50.817: RASLib::RASsendRCF: RCF (seq# 4051) sent to 172.16.13.42
```

Case 2

This is another example where an endpoint that did not send a TTL value in its RRQ message has been notified to send a lightweight RRQ before 120 seconds, which is the value configured on the gatekeeper. You can see in this output how the Cisco Gatekeeper does not delete the endpoint until three unanswered IRQ messages, even though an Unregistration Request (URQ) message was received. The times between the IRQ are between one and five minutes.

```
gka-1#show logging
```

```
Syslog logging: enabled (0 messages dropped, 0 messages rate-limited, 0 flushes, 0 overruns)
Console logging: disabled
Monitor logging: level debugging, 1076 messages logged
Buffer logging: level debugging, 4257 messages logged
Logging Exception size (4096 bytes)
Trap logging: level informational, 60 message lines logged
```

```
Log Buffer (9999999 bytes):
```

```
Mar 14 06:28:31.771: RAS INCOMING ENCODE BUFFER ::= 0C 80000006 0008914A
00020001 00AB4555 BF06B801 00AB4555 BF05C502 00014007 006B0065 00740070
00610074 0065006C 60B50053 4C164D69 63726F73 6F6674AE 204E6574 4D656574
696E67AE 0003332E 3000
```

Mar 14 06:28:31.783:
Mar 14 06:28:31.787: RAS INCOMING PDU ::=

```
value RasMessage ::= registrationRequest :
{
  requestSeqNum 1
  protocolIdentifier { 0 0 8 2250 0 2 }
  discoveryComplete FALSE
  callSignalAddress
  {
    ipAddress :
    {
      ip 'AB4555BF'H
      port 1720
    }
  }
  rasAddress
  {
    ipAddress :
    {
      ip 'AB4555BF'H
      port 1477
    }
  }
  terminalType
  {
    terminal
    {
    }
    mc FALSE
    undefinedNode FALSE
  }
  terminalAlias
  {
    h323-ID : {"ketpatel"}
  }
  endpointVendor
  {
    vendor
    {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 21324
    }
    productId '4D6963726F736F6674AE204E65744D656574696E...'H
    versionId '332E3000'H
  }
}
```

Mar 14 06:28:31.811: RAS OUTGOING PDU ::=

```
value RasMessage ::= registrationConfirm :
{
  requestSeqNum 1
  protocolIdentifier { 0 0 8 2250 0 3 }
  callSignalAddress
  {
  }
  terminalAlias
  {
    h323-ID : {"ketpatel"}
  }
  gatekeeperIdentifier {"gka-1"}
  endpointIdentifier {"81F6A89800000001"}
```

```

alternateGatekeeper
{
}
timeToLive 120
willRespondToIRR FALSE
maintainConnection FALSE
}

```

```

Mar 14 06:28:31.823: RAS OUTGOING ENCODE BUFFER ::= 12 C0000006 0008914A
00030001 4007006B 00650074 00700061 00740065 006C0800 67006B00 61002D00
311E0038 00310046 00360041 00380039 00380030 00300030 00300030 00300030
00310F8A 01000200 77010001 00

```

gka-1#show gatekeeper endpoints

```

GATEKEEPER ENDPOINT REGISTRATION
=====
CallSignalAddr  Port  RASignalAddr  Port  Zone Name          Type  Flags
-----
171.69.85.191   1720 171.69.85.191  1477  gka-1              TERM
H323-ID: ketpatel
Total number of active registrations = 1

```

```

Mar 14 06:28:31.835:
Mar 14 06:28:31.835: RAS OUTGOING PDU ::=

```

```

value RasMessage ::= infoRequest :
{
  requestSeqNum 70
  callReferenceValue 0
  callIdentifier
  {
    guid '00000000000000000000000000000000'H
  }
}

```

```

Mar 14 06:28:31.839: RAS OUTGOING ENCODE BUFFER ::= 56 00004500 000B0011
00000000 00000000 00000000 00000000 00
Mar 14 06:28:31.843:
Mar 14 06:28:31.847: RAS INCOMING ENCODE BUFFER ::= 58 80004502 03C00038
00310046 00360041 00380039 00380030 00300030 00300030 00300030 003100AB
4555BF05 C50100AB 4555BF06 B8024007 006B0065 00740070 00610074 0065006C
4007006B 00650074 00700061 00740065 006C
Mar 14 06:28:31.859:
Mar 14 06:28:31.859: RAS INCOMING PDU ::=

```

```

value RasMessage ::= infoRequestResponse :
{
  requestSeqNum 70
  endpointType
  {
    terminal
    {
    }
    mc FALSE
    undefinedNode FALSE
  }
  endpointIdentifier {"81F6A89800000001"}
  rasAddress ipAddress :
  {
    ip 'AB4555BF'H
  }
}

```

```

    port 1477
  }
  callSignalAddress
  {
    ipAddress :
    {
      ip 'AB4555BF'H
      port 1720
    }
  }
  endpointAlias
  {
    h323-ID : {"ketpatel"},
    h323-ID : {"ketpatel"}
  }
}

```

Mar 14 06:30:42.208: RAS OUTGOING PDU ::=

```

value RasMessage ::= infoRequest :
{
  requestSeqNum 71
  callReferenceValue 0
  callIdentifier
  {
    guid '00000000000000000000000000000000'H
  }
}

```

Mar 14 06:30:42.212: RAS OUTGOING ENCODE BUFFER ::= 56 00004600 000B0011
00000000 00000000 00000000 00000000 00

Mar 14 06:30:42.216:

Mar 14 06:30:42.216: RAS INCOMING ENCODE BUFFER ::= 58 80004602 03C00038
00310046 00360041 00380039 00380030 00300030 00300030 00300030 003100AB
4555BF05 C50100AB 4555BF06 B8024007 006B0065 00740070 00610074 0065006C
4007006B 00650074 00700061 00740065 006C

Mar 14 06:30:42.228:

Mar 14 06:30:42.232: RAS INCOMING PDU ::=

```

value RasMessage ::= infoRequestResponse :
{
  requestSeqNum 71
  endpointType
  {
    terminal
    {
    }
    mc FALSE
    undefinedNode FALSE
  }
  endpointIdentifier {"81F6A89800000001"}
  rasAddress ipAddress :
  {
    ip 'AB4555BF'H
    port 1477
  }
  callSignalAddress
  {
    ipAddress :
    {
      ip 'AB4555BF'H
      port 1720
    }
  }
}

```

```

    }
  }
  endpointAlias
  {
    h323-ID : {"ketpatel"},
    h323-ID : {"ketpatel"}
  }
}

```

```

Mar 14 06:32:05.938: RAS INCOMING ENCODE BUFFER ::= 19 40000101 00AB4555
BF06B802 4007006B 00650074 00700061 00740065 006C4007 006B0065 00740070
00610074 0065006C 1E003800 31004600 36004100 38003900 38003000 30003000
30003000 30003000 31
Mar 14 06:32:05.950:
Mar 14 06:32:05.950: RAS INCOMING PDU ::=

```

```

value RasMessage ::= unregistrationRequest :
{
  requestSeqNum 2
  callSignalAddress
  {
    ipAddress :
    {
      ip 'AB4555BF'H
      port 1720
    }
  }
  endpointAlias
  {
    h323-ID : {"ketpatel"},
    h323-ID : {"ketpatel"}
  }
  endpointIdentifier {"81F6A89800000001"}
}

```

```

Mar 14 06:32:05.962: RAS OUTGOING PDU ::=

```

```

value RasMessage ::= unregistrationConfirm :
{
  requestSeqNum 2
}

```

```

Mar 14 06:32:05.962: RAS OUTGOING ENCODE BUFFER ::= 1C 0001
Mar 14 06:32:05.966:

```

```

gka-1#show gatekeeper endpoints

```

```

GATEKEEPER ENDPOINT REGISTRATION
=====
CallSignalAddr  Port  RASSignalAddr  Port  Zone Name      Type  Flags
-----
171.69.85.191  1720  171.69.85.191  1477  gka-1          TERM
Total number of active registrations = 1

```

```

Mar 14 06:33:42.223: RAS OUTGOING PDU ::=

```

```

value RasMessage ::= infoRequest :
{
  requestSeqNum 72
  callReferenceValue 0
}

```

```

    callIdentifier
    {
      guid '00000000000000000000000000000000'H
    }
  }

```

```

Mar 14 06:33:42.227: RAS OUTGOING ENCODE BUFFER ::= 56 00004700 000B0011
00000000 00000000 00000000 00000000 00

```

```

Mar 14 06:33:42.231:

```

```

Mar 14 06:34:42.234: RAS OUTGOING PDU ::=

```

```

value RasMessage ::= infoRequest :
  {
    requestSeqNum 73
    callReferenceValue 0
    callIdentifier
    {
      guid '00000000000000000000000000000000'H
    }
  }

```

```

Mar 14 06:34:42.238: RAS OUTGOING ENCODE BUFFER ::= 56 00004800 000B0011
00000000 00000000 00000000 00000000 00

```

```

Mar 14 06:34:42.242:

```

```

Mar 14 06:35:42.244: RAS OUTGOING PDU ::=

```

```

value RasMessage ::= infoRequest :
  {
    requestSeqNum 74
    callReferenceValue 0
    callIdentifier
    {
      guid '00000000000000000000000000000000'H
    }
  }

```

```

Mar 14 06:35:42.248: RAS OUTGOING ENCODE BUFFER ::= 56 00004900 000B0011
00000000 00000000 00000000 00000000 00

```

```

Mar 14 06:35:42.252:

```

```

gka-1#

```

```

gka-1#show gatekeeper endpoints

```

```

GATEKEEPER ENDPOINT REGISTRATION

```

```

=====

```

CallSignalAddr	Port	RASignalAddr	Port	Zone Name	Type	Flags

```

Total number of active registrations = 0

```

Related Information

- [Cisco High-Performance Gatekeeper](#)
- [H.323 Version 2 Support](#)
- [Troubleshooting Gatekeeper Registration Issues](#)
- [Voice Technology Support](#)
- [Voice and Unified Communications Product Support](#)
- [Recommended Reading: Troubleshooting Cisco IP Telephony](#)
- [Technical Support – Cisco Systems](#)

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2008 – 2009 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Feb 02, 2006

Document ID: 18732
