

# Gateway to Gatekeeper (H.235) and Gatekeeper to Gatekeeper (IZCT) Security Troubleshooting Guide

## Contents

### [Introduction](#)

#### [Intradomain Gateway to Gatekeeper Security](#)

- [Time Stamp Passed in the Tokens](#)
- [How Cisco Implements the H.235 Recommendation](#)
- [How to Configure the Security Levels](#)
- [H.235 Usage on a Per-call Level without IVR](#)
- [Major Issues](#)
- [Debugs and Call Flow for the Different Levels](#)
- [Gateway IOS Problem](#)
- [Security with Alternate Endpoints](#)
- [OSP Token Support](#)
- [Different Levels of Security for each Endpoint or Zone](#)

#### [Interdomain Gatekeeper to Gatekeeper Security](#)

- [Implementing Gatekeeper to Gatekeeper Security](#)
- [Gatekeeper Configuration](#)
- [IZCT Call Flow](#)
- [Call Flow with Debugs](#)

### [Related Information](#)

## Introduction

H.323 networks have different kinds of configurations and call flows. This document attempts to document most of the the security concerns with H.323 networks that involve gatekeepers. This document summarizes the way each feature works and how to troubleshoot it with an explanation to most of the debugs. This document does not address the overall security of Voice over Internet Protocol (VoIP).

This document covers the following features:

- **Intradomain Gateway to Gatekeeper Security** This security is based on H.235, in which H.323 calls are authenticated, authorized, and routed by a gatekeeper. The gatekeeper is considered a known and trusted entity in a sense that the gateway does not authenticate it when the gateway tries to register with it.
- **Interdomain Gatekeeper to Gatekeeper Security** This security covers authenticating and authorizing of H.323 calls between the administrative domains of Internet Telephone Service Providers (ITSPs) using InterZone Clear Token (IZCT). This document covers only the portion where the terminating gatekeeper sends a token in its location confirmation (LCF) message so that it authenticates the answerCall Admission Request (ARQ). Location request (LRQ) validation is not included in this feature. LRQ validation is a feature scheduled for a future Cisco IOS release.

### Definitions

Acronym	Definition
ARQ	Admission Request A RAS message sent from an H.323 endpoint to a gatekeeper requesting an admission to establish a call.

ACF	Admission Confirm A RAS message sent from the gatekeeper to the endpoint confirming the acceptance of a call.
ARJ	Admission Rejection A RAS message from the gatekeeper to the endpoint rejecting the admission request.
CAT	Cisco Access Token It is the H.235 Clear Token.
CHAP	Challenge Handshake Authentication Protocol An authentication protocol where a challenge is used.
GCF	Gatekeeper Confirm A RAS message sent from a gatekeeper to the H.323 endpoint confirming the discovery of the gatekeeper.
GRQ	Gatekeeper Request A RAS message sent from an H.323 endpoint to discover the gatekeeper.
H.235	ITU recommendation for security and encryption for H-Series (H.323 and other H.245-based) multimedia terminals.
IZCT	InterZone Clear Token
LRQ	Location Request A RAS message sent from gatekeeper to the next hop gatekeeper or call leg to trace and route the call
RAS	Registration, Admission, and Status A protocol that allows a gatekeeper to perform registration, admission, and status check of the endpoint.
RCF	Registration Confirm A RAS message sent from the gatekeeper to the endpoint confirming the registration.
RRJ	Registration Reject A RAS message sent from the gatekeeper rejecting the registration request.
RRQ	Registration Request A RAS message sent from the endpoint to the gatekeeper requesting to register with it.
RIP	Request In Progress A RAS message sent from a gatekeeper to the sender stating the call is in progress

## Intradomain Gateway to Gatekeeper Security

H.323 is an ITU recommendation that addresses securing real-time communication over insecure networks which generally involves two major areas of concern: authentication and privacy. There are two types of authentication, according to H.235:

- Symmetric encryption-based authentication that requires no prior contact between the communicating entities
- Based on the ability to have some prior shared secret (further referenced as subscription based), two forms of subscription-based authentication are provided:
  - password
  - certificate

## Time Stamp Passed in the Tokens

A time stamp is used to prevent replay attacks. Therefore, it is needed for a mutually acceptable reference to time (from which to derive time stamps). The amount of acceptable time skew is a local implementation matter.

## How Cisco Implements the H.235 Recommendation

Cisco uses a CHAP-like authentication scheme as the basis for its gateway to its gatekeeper H.235 implementation. This allows you to leverage Authentication, Authorization, and Accounting (AAA), using existing functionality to perform the actual authentication. It also means that the gatekeeper is not required to have access to a database of gateway IDs, user account numbers, passwords, and PINs. The scheme is based on H.235, section 10.3.3 and is described as subscription-based password with hashing.

However, instead of using H.225 cryptoTokens, this method uses H.235 clearTokens with fields populated appropriately for use with RADIUS. This token is referred to as the Cisco Access Token. (CAT). You can always perform authentication locally on the gatekeeper instead of using a RADIUS server.

The use of cryptoTokens would require that the gatekeeper either maintains or has some way of acquiring passwords for all of the users and gateways. This is because the token field of the cryptoToken is specified such that the authenticating entity needs the password to generate its own token against which to compare the received one.

So Cisco gatekeepers totally ignore cryptoTokens. However, vocalTek gatekeepers and others who support H.235 section 10.3.3, use the cryptoTokens to authenticate the gateway. Cisco gatekeepers use the CAT to authenticate the gateway. Since the gateway doesn't know what type of gatekeeper he's talking to, it sends both in the Registration Request (RRQ). The authenticationCapability in the Gatekeeper Request (GRQ) are for the cryptoToken and indicate that the MD5 hashing is the authentication mechanism (although the CAT uses MD5 as well).

For detailed information, refer to [Cisco H.323 Gateway Security and Accounting Enhancements](#).

## How to Configure the Security Levels

How to Configure the Security Levels

- Endpoint- or Registration-level Security

With Registration Security enabled on the gatekeeper, the gateway is required to include a CAT in all heavyweight RRQ messages. The CAT, in this case, contains information which authenticates the gateway itself to the gatekeeper. The gatekeeper formats a message to a RADIUS server which authenticates the information contained in the token. It responds back to the gatekeeper with either an Access-Accept or Access-Reject, which in turn responds to the gateway with either a Response Confirmation (RCF) or an Response Reject (RRJ).

If a call is then placed from a successfully authenticated gateway, that gateway generates a new CAT upon receipt of an Admission Confirm (ACF) from the gatekeeper using the gateway password. This CAT is identical to the one generated during registration except for the time stamp. It is placed in the outgoing SETUP message. The destination gateway extracts the token from the SETUP message and places it in the destination side ARQ. The gatekeeper uses RADIUS as before to authenticate the originating gateway before sending the destination side ACF. This prevents a non-authenticated endpoint that knows a gateway's address from using it to circumvent the security scheme and accessing the Public Switched Telephone Network (PSTN).

Therefore, in this level, there is no need to include any tokens in the originating ARQs.

To configure the gatekeeper, from the gatekeeper command-line interface (CLI) type `router(config-gk)# [no] security token required-for registration`. The `no` option of the command causes the gatekeeper to no longer check for tokens in Registration, Admission, and Status (RAS) messages.

To configure the gateway, from the gateway CLI, type `router(config-gw)# [no] security password <PASSWORD> level endpoint`. The `no` option of the command causes the gateway to no longer generate tokens for RAS messages.

- Per-call Level Security

Per-call security builds upon registration-level security. In addition to meeting registration security requirements, a gateway is also required to include Access Tokens in all originating side ARQ messages when Per-call security is enabled on the gatekeeper. The token in this case contains information which identifies the user of the gateway to the gatekeeper. This information is obtained using an Interactive Voice Response (IVR) script on the gateway which prompts users to enter their User ID and PIN from the keypad before placing a call.

The CAT contained in the originating ARQ is authenticated by RADIUS the same as described above in Endpoint- or

Registration-level Security. After receiving the ACF, the gateway generates a new CAT using its password and sends it within the H.225 SETUP message to the terminating gateway.

To configure the gatekeeper, from the gatekeeper CLI, type `router(config-gk)# [no] security token required-for all`. The `no` option of the command causes the gatekeeper to no longer check for tokens in RAS messages.

To configure the gateway, from the gateway CLI, type `router(config-gw)# [no] security password <PASSWORD> level per-call`. The `no` option of the command causes the gateway to no longer generate tokens for RAS messages.

- All Level Security

This allows the gateway to include a CAT in all RAS messages needed for registration and for calls, so it is a combination of the above two levels. With this option, the validation of CAT in ARQ messages is based on an the account number and PIN of the user making a call. The validation of the CAT sent in all of the other RAS messages is based on the password configured for the gateway. Therefore, it is similar to the Per-call level.

To configure the gatekeeper, from the gatekeeper CLI, type `router(config-gk)# [no] security token required-for all`. The `no` option of the command causes the gatekeeper to no longer check for tokens in RAS messages.

To configure the gateway, from the gateway CLI, type `router(config-gw)# [no] security password <PASSWORD> level all`. The `no` option of the command causes the gateway to no longer generate tokens for RAS messages.

## H.235 Usage on a Per-call Level without IVR

H.235 cannot be used on a per-call level without IVR. If is no IVR to collect an account and PIN, the gateway should send the ARQ without a Clear Token (but with a crypto token). And since the Cisco gatekeeper only accepts Clear Tokens, the call will be rejected by the gatekeeper with a reason of security denial.

The following example shows the `h225 asn1` debugs collected from an originating gateway (OGW) that was not configured for an IVR to collect the account and PIN. You can see that the RRQ message has a Clear Token, but the ARQ does not, and an Admission Rejection (ARJ) message was sent back to the gateway.

```
Mar 4 01:31:24.358: H235 OUTGOING ENCODE BUFFER ::= 61 000100C0
2B955BEB 08003200 32003200 32000006 006F0067 00770000
Mar 4 01:31:24.358:
Mar 4 01:31:24.358: RAS OUTGOING PDU ::=
value RasMessage ::= registrationRequest :
{
  requestSeqNum 29
  protocolIdentifier { 0 0 8 2250 0 3 }
  discoveryComplete FALSE
  callSignalAddress
  {
  }
  rasAddress
  {
    ipAddress :
    {
      ip 'AC100D0F'H
      port 57514
    }
  }
  terminalType
  {
    mc FALSE
    undefinedNode FALSE
  }
  gatekeeperIdentifier {"ogk1"}
  endpointVendor
  {
    vendor
    {
      t35CountryCode 181
      t35Extension 0
    }
  }
}
```

```

    manufacturerCode 18
  }
}
timeToLive 60
tokens

```

*!--- Clear Token is included in the RRQ message*

```

{
  {
    tokenOID { 1 2 840 113548 10 1 2 1 }
    timeStamp 731208684
    challenge 'F57C3C65B59724B9A45C93F98CCF9E45'H
    random 12
    generalID {"ogw"}
  }
}
cryptoTokens
{
  cryptoEPPwdHash :
  {
    alias h323-ID : {"ogw"}
    timeStamp 731208684
    token
    {
      algorithmOID { 1 2 840 113549 2 5 }
      paramS
      {
      }
      hash "D7F85666AF3B881ADD876DD61C20D5D9"
    }
  }
}
keepAlive TRUE
endpointIdentifier {"81F5E24800000001"}
willSupplyUUIEs FALSE
maintainConnection TRUE
}

```

```

Mar 4 01:31:24.370: RAS OUTGOING ENCODE BUFFER::= 0E 40001C06 0008914A
00030000 0100AC10 0D0FE0AA 0003006F 0067006B 003100B5 00001212 EF000200
3B2F014D 000A2A86 4886F70C 0A010201 C02B955B EB10F57C 3C65B597 24B9A45C
93F98CCF 9E45010C 06006F00 67007700 002A0104 02006F00 670077C0 2B955BEB
082A8648 86F70D02 05008080 D7F85666 AF3B881A DD876DD6 1C20D5D9 0180211E
00380031 00460035 00450032 00340038 00300030 00300030 00300030 00300031
01000180

```

```

Mar 4 01:31:24.378: h323chan_dgram_send:Sent UDP msg. Bytes sent: 173 to
172.16.13.35:1719

```

```

Mar 4 01:31:24.378: RASLib::GW_RASSendRRQ:
3640-1#debug RRQ (seq# 29) sent to 172.16.13.35

```

```

Mar 4 01:31:24.462: h323chan_chn_process_read_socket

```

```

Mar 4 01:31:24.462: h323chan_chn_process_read_socket: fd (2) of type CONNECTED has
data

```

```

Mar 4 01:31:24.462: h323chan_chn_process_read_socket: h323chan accepted/connected

```

```

Mar 4 01:31:24.462: h323chan_dgram_recvdata:rcvd from [172.16.13.35:1719] on so
ck[2]

```

```

Mar 4 01:31:24.466: RAS INCOMING ENCODE BUFFER::= 12 40001C06 0008914A
00030006 006F0067 006B0031 1E003800 31004600 35004500 32003400 38003000
30003000 30003000 30003000 310F8A01 0002003B 01000180

```

```

Mar 4 01:31:24.466:

```

```
Mar 4 01:31:24.466: RAS INCOMING PDU ::=
value RasMessage ::= registrationConfirm :
```

```
{
  requestSeqNum 29
  protocolIdentifier { 0 0 8 2250 0 3 }
  callSignalAddress
  {
  }
  gatekeeperIdentifier {"ogk1"}
  endpointIdentifier {"81F5E24800000001"}
  alternateGatekeeper
  {
  }
  timeToLive 60
  willRespondToIRR FALSE
  maintainConnection TRUE
}
```

```
Mar 4 01:31:24.470: RCF (seq# 29) rcvd
```

```
Mar 4 01:32:00.220: H225 NONSTD OUTGOING PDU ::=
```

```
value ARQnonStandardInfo ::=
```

```
{
  sourceAlias
  {
  }
  sourceExtAlias
  {
  }
  callingOctet3a 129
  interfaceSpecificBillingId "ISDN-VOICE"
}
```

```
Mar 4 01:32:00.220: H225 NONSTD OUTGOING ENCODE BUFFER::= 80 000008A0
01810B12 4953444E 2D564F49 4345
```

```
Mar 4 01:32:00.220:
```

```
Mar 4 01:32:00.220: H235 OUTGOING ENCODE BUFFER::= 61 000100C0
2B955C0F 08003200 32003200 32000006 006F0067 00770000
```

```
Mar 4 01:32:00.224:
```

```
Mar 4 01:32:00.224: RAS OUTGOING PDU ::=
```

```
value RasMessage ::= admissionRequest :
```

```
{
  requestSeqNum 30
  callType pointToPoint : NULL
  callModel direct : NULL
  endpointIdentifier {"81F5E24800000001"}
  destinationInfo
  {
    e164 : "3653"
  }
  srcInfo
  {
    e164 : "5336",
    h323-ID : {"ogw"}
  }
  bandwidth 1280
  callReferenceValue 5
  nonStandardData
  {
    nonStandardIdentifier h221NonStandard :
    {
      t35CountryCode 181
      t35Extension 0
    }
  }
}
```

```

    manufacturerCode 18
  }
  data '80000008A001810B124953444E2D564F494345'H
}
conferenceID 'E1575DA6175611CC8014A6051561649A'H
activeMC FALSE
answerCall FALSE
canMapAlias TRUE
callIdentifier
{
  guid 'E1575DA6175611CC8015A6051561649A'H
}
cryptoTokens

```

*!--- Only cryptoTokens included, no clear one*

```

{
  cryptoEPPwdHash :
  {
    alias h323-ID : {"ogw"}
    timeStamp 731208720
    token
    {
      algorithmOID { 1 2 840 113549 2 5 }
      paramS
      {
      }
      hash "105475A4C0A833E7DE8E37AD3A8CDDFF"
    }
  }
}
willSupplyUUIEs FALSE
}

```

```

Mar 4 01:32:00.236: RAS OUTGOING ENCODE BUFFER ::= 27 88001D00 F0003800
31004600 35004500 32003400 38003000 30003000 30003000 30003000 31010180
69860201 80866940 02006F00 67007740 05000005 40B50000 12138000 0008A001
810B1249 53444E2D 564F4943 45E1575D A6175611 CC8014A6 05156164 9A056120
01801100 E1575DA6 175611CC 8015A605 1561649A 2A010402 006F0067 0077C02B
955C0F08 2A864886 F70D0205 00808010 5475A4C0 A833E7DE 8E370AD3 A8CDDFF01 00

```

```

Mar 4 01:32:00.240: h323chan_dgram_send:Sent UDP msg. Bytes sent: 170 to
172.16.13.35:1719

```

```

Mar 4 01:32:00.240: RASLib::GW_RASSendARQ: ARQ (seq# 30) sent to 172.16.13.35

```

```

Mar 4 01:32:00.312: h323chan_chn_process_read_socket

```

```

Mar 4 01:32:00.312: h323chan_chn_process_read_socket: fd (2) of type CONNECTED has
data

```

```

Mar 4 01:32:00.312: h323chan_chn_process_read_socket: fd (2) of type CONNECTED has
data

```

```

Mar 4

```

```

3640-1#01:32:00.312: h323chan_chn_process_read_socket: h323chan accepted/connected

```

```

Mar 4 01:32:00.312: h323chan_dgram_recvddata:rcvd from [172.16.13.35:1719] on so
ck[2]

```

```

Mar 4 01:32:00.312: RAS INCOMING ENCODE BUFFER ::= 2C 001D8001 00

```

```

Mar 4 01:32:00.312:

```

```

Mar 4 01:32:00.312: RAS INCOMING PDU ::=
value RasMessage ::= admissionReject :

```

*!--- ARQ was rejected with security denial reason*

```

{
  requestSeqNum 30

```

```

    rejectReason securityDenial : NULL
  }
Mar 4 01:32:00.312: ARJ (seq# 30) rcvd

```

## Major Issues

The major issues you should be concerned with include:

- Configuration of the gateway and gatekeeper
- RADIUS configuration on the gatekeeper and the RADIUS server
- NTP You must have the same time on all gateways and gatekeepers
- Software failure due to a bug

## Debugs and Call Flow for the Different Levels

Since the all level security covers both registration and per-call cases, the lab was set up with that level of security for this exercise. The call flows for the registration part and a normal VoIP call are explained in the configuration below.

**Note:** The configuration below is not complete, more commands will follow in between the debug outputs. It is designed to show what problem could happen if you do not check all things such as configuration, NTP, and RADIUS. In addition, the gateway was authenticated on the gatekeeper locally so that you can see what values are set for the gateway ID and password. The following configuration was snipped so that only the related configuration is shown.

```

!
interface Ethernet0/0
 ip address 172.16.13.15 255.255.255.224
 half-duplex
 h323-gateway voip interface
 h323-gateway voip id gka-1 ipaddr 172.16.13.35 1718

!--- Gatekeeper name is gka-1

h323-gateway voip h323-id gwa-1@cisco.com

!--- Gateway H323-ID is gwa-1@cisco.com

h323-gateway voip tech-prefix 1#
!
!
gateway
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0 4
 exec-timeout 0 0
 password ww
 logging synchronous
end

!--- No NTP configured
!--- The snipped Gatekeeper configuration is as follows:

!
aaa new-model
aaa authentication login default local
aaa authentication login h323 local
aaa authorization exec default local

```



```

aaa authorization exec h323 local
aaa accounting connection h323 start-stop group radius
!
username gwa-1 password 0 2222
username gwa-2 password 0 2222
!
gatekeeper
  zone local gka-1 cisco.com 172.16.13.35
  security token required-for all

!--- The gatekeeper is configured for the "All level security"

no shutdown
!
!
line con 0
  exec-timeout 0 0
line aux 0
line vty 0 4
  password ww
line vty 5 15
!
no scheduler max-task-time
no scheduler allocate
ntp master

!--- This gatekeeper is set as an NTP master

!
end

```

In the example below, the following debugs were turned on:

- debug ras
- debug h225 asn1
- debug radius
- debug aaa authentication
- debug aaa authorization

The first thing that occurs is the gateway sends a GRQ to the gatekeeper and the gatekeeper sends a Gatekeeper Confirm (GCF) to the gateway. The gateway then sends an RRQ and waits for either an RCF or RRJ.

In the above configuration, the gateway is not set for any level of security so that its GRQ would carry no **authenticationCapability** that would be needed for the tokens. But still, the gatekeeper would send back a GCF as shown below.

```

*Mar 2 13:32:45.413: RAS INCOMING ENCODE BUFFER ::= 00 A0000006
0008914A 000200AC 100D0FD2 C6088001 3C050401 00204002 00006700 6B006100
2D003102 400E0067 00770061 002D0031 00400063 00690073 0063006F 002E0063
006F006D 0080CC
*Mar 2 13:32:45.421:
*Mar 2 13:32:45.425: RAS INCOMING PDU ::=

value RasMessage ::= gatekeeperRequest :
{
  requestSeqNum 1
  protocolIdentifier { 0 0 8 2250 0 2 }
  rasAddress ipAddress :
  {
    ip 'AC100D0F'H

```

```

    port 53958
  }
  endpointType
  {
    gateway
    {
      protocol
      {
        voice :
        {
          supportedPrefixes
          {
            {
              prefix e164 : "1#"
            }
          }
        }
      }
    }
  }
  mc FALSE
  undefinedNode FALSE
}
gatekeeperIdentifier {"gka-1"}
endpointAlias
{
  h323-ID : {"gwa-1@cisco.com"},

```

*!--- The H323-ID of the gateway is gwa-1@cisco.com*

```

    e164 : "99"
  }
}

```

```

*Mar 2 13:32:45.445: RAS OUTGOING PDU ::=
value RasMessage ::= gatekeeperConfirm :

```

```

{
  requestSeqNum 1
  protocolIdentifier { 0 0 8 2250 0 3 }
  gatekeeperIdentifier {"gka-1"}
  rasAddress ipAddress :
  {
    ip 'AC100D23'H
    port 1719
  }
}

```

*!--- The gateway sends an RRQ message after that to the gatekeeper with the  
 !--- ip address sent in the GCF. This RRQ will not carry any Token information  
 !--- because security is not configured on the gateway.*

```

*Mar 2 13:32:45.477: RAS INCOMING ENCODE BUFFER ::= 0E C0000106 0008914A
00028001 00AC100D 0F06B801 00AC100D 0FD2C608 80013C05 04010020 40000240
0E006700 77006100 2D003100 40006300 69007300 63006F00 2E006300 6F006D00
80CC0800 67006B00 61002D00 3100B500 00120E8A 02003B01 000100

```

```

*Mar 2 13:32:45.489:

```

```

*Mar 2 13:32:45.493: RAS INCOMING PDU ::=
value RasMessage ::= registrationRequest :

```

```

{
  requestSeqNum 2

```

```
protocolIdentifier { 0 0 8 2250 0 2 }
discoveryComplete TRUE
callSignalAddress
{
  ipAddress :
  {
    ip 'AC100D0F'H
    port 1720
  }
}
rasAddress
{
  ipAddress :
  {
    ip 'AC100D0F'H
    port 53958
  }
}
terminalType
{
  gateway
  {
    protocol
    {
      voice :
      {
        supportedPrefixes
        {
          {
            prefix e164 : "1#"
          }
        }
      }
    }
  }
}
mc FALSE
undefinedNode FALSE
}
terminalAlias
{
  h323-ID : {"gwa-1@cisco.com"},
  e164 : "99"
}
gatekeeperIdentifier {"gka-1"}
endpointVendor
{
  vendor
  {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
  }
}
timeToLive 60
keepAlive FALSE
willSupplyUUIEs FALSE
}
```

*!--- Since the gateway did not include any tokens and the gatekeeper is set to authenticate*

*!--- all endpoints and calls, the gatekeeper will reject the gateway request, sending an RRJ*

*!--- with **securityDenial** reason.*

\*Mar 2 13:32:45.525: RAS OUTGOING PDU ::=

```
value RasMessage ::= registrationReject :
{
  requestSeqNum 2
  protocolIdentifier { 0 0 8 2250 0 3 }
  rejectReason securityDenial : NULL
```

*!--- Gatekeeper rejects the RRQ with security denial reason*

```
  gatekeeperIdentifier {"gka-1"}
}
```

\*Mar 2 13:32:45.529: RAS OUTGOING ENCODE BUFFER::= 14 80000106

0008914A 00038301 00080067 006B0061 002D0031

\*Mar 2 13:32:45.533:

*!--- We'll now configure the security password 2222 level all command on the gateway.*

*!--- The configuration of the gateway will appear as follows:*

```
!
gateway
  security password 0356095954 level all
```

*!--- The password will be hashed*

!

*!--- As soon as you make this change the gateway will send a new GRQ to the gatekeeper.*

*!--- This time you will see the **authenticationCapability** and **algorithmOIDs**.*

\*Mar 2 13:33:15.551: RAS INCOMING ENCODE BUFFER::= 02 A0000206

0008914A 000200AC 100D0FD2 C6088001 3C050401 00204002 00006700 6B006100

2D003102 400E0067 00770061 002D0031 00400063 00690073 0063006F 002E0063

006F006D 0080CC0C 30020120 0A01082A 864886F7 0D0205

\*Mar 2 13:33:15.563:

\*Mar 2 13:33:15.567: RAS INCOMING PDU ::=

```
value RasMessage ::= gatekeeperRequest :
{
  requestSeqNum 3
  protocolIdentifier { 0 0 8 2250 0 2 }
  rasAddress ipAddress :
  {
    ip 'AC100D0F'H
    port 53958
  }
  endpointType
  {
    gateway
    {
      protocol
      {
        voice :
        {
```



```

    {
      ip 'AC100D0F'H
      port 1720
    }
  }
  rasAddress
  {
    ipAddress :
    {
      ip 'AC100D0F'H
      port 53958
    }
  }
  terminalType
  {
    gateway
    {
      protocol
      {
        voice :
        {
          supportedPrefixes
          {
            {
              prefix e164 : "1#"
            }
          }
        }
      }
    }
  }
  mc FALSE
  undefinedNode FALSE
}
terminalAlias
{
  h323-ID : {"gwa-1@cisco.com"},
  e164 : "99"
}
gatekeeperIdentifier {"gka-1"}
endpointVendor
{
  vendor
  {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
  }
}
timeToLive 60
tokens

```

*!--- Clear Token, or what is called CAT*

```

{
  {
    tokenOID { 1 2 840 113548 10 1 2 1 }
    timeStamp 731030839
    challenge 'B9D84DAE58F6CB4B5EE5DFB6B92DD281'H
    random 1
  }
}

```

```

    generalID {"gwa-1@cisco.com"}
  }
}
cryptoTokens

```

*!--- CryptoToken field*

```

{
  cryptoEPPwdHash :
  {
    alias h323-ID : {"gwa-1@cisco.com"}
    timeStamp 731030839
    token
    {
      algorithmOID { 1 2 840 113549 2 5 }
      paramS
      {
      }
      hash "2B21B94F3980ED12116C56B79F4B4CDB"
    }
  }
}
keepAlive FALSE
willSupplyUUIEs FALSE
}

```

Before discussing the response, we'll explain some of the related fields in the above RRQ message. First, you can see there are two types of tokens: a clear token or what we call Cisco Access Token (CAT), and a crypto token.

As mentioned before, Cisco gatekeepers will ignore the crypto tokens but still the gateway would send both because it does not know to what type of gatekeeper it is talking. And since other vendors might use crypto tokens, the gateway would send both.

The following explains the ClearToken syntax.

- tokenOID The Object ID to identify the token.
- timeStamp The current UTC time of the gateway. seconds since 00:00 1/1/1970 UTC.
- challenge A 16-byte MD5 message digest generated by the gateway using the following fields:

challenge = [ random + GW/User Password + timeStamp ] MD5 Hash

RADIUS performs the following calculation (since it knows the random number, the gateway password, and the CHAP challenge) to determine what the challenge should be: CHAP Response = [ CHAP ID + UserPassword + CHAP Challenge ] MD5 Hash

- random A one-byte value used by RADIUS to identify this particular request.
- generalID The gateway H323-ID or the user account number depending on the level of security and the type of RAS message.

The gateway increments a variable modulo 256 for each authentication request to satisfy this RADIUS requirement.

**Note:** All of these fields are important but we need to give more attention to both the time stamp and the generalID. In our case, the time stamp is **731030839** and the generalID is **gwa-1@cisco.com**.

The cryptoToken in the RRQ contains information about the gateway that generated the token, including the gateway ID (which is the H.323 ID configured on the gateway) and the gateway password.

This field contains one of the cryptoToken types defined for the CryptoH323Token field specified in H.225. Currently, the only type of cryptoToken supported is the cryptoEPPwdHash.

The following fields are contained within the cryptoEPPwdHash field:

- alias The gateway alias, which is the H.323 ID of the gateway
- timestamp The current time stamp

- token The Message Digest 5 (MD5)-encoded PwdCertToken. This field contains the following:
  - timestamp The same as the timestamp of the cryptoEPPwdHash
  - password The gateway's password
  - generalID The same gateway alias as the one included in the cryptoEPPwdHash
  - tokenID The object ID

Now we'll view the response from the gatekeeper in the following example.

```
*Mar 2 13:33:15.723: RAS OUTGOING PDU ::=
```

```
value RasMessage ::= registrationReject :
{
  requestSeqNum 4
  protocolIdentifier { 0 0 8 2250 0 3 }
  rejectReason securityDenial : NULL
```

```
!--- The gatekeeper rejected the RRQ with securityDenial reason
```

```
  gatekeeperIdentifier {"gka-1"}
}
```

```
*Mar 2 13:33:15.727: RAS OUTGOING ENCODE BUFFER ::= 14 80000306 0008914A
00038301 00080067 006B0061 002D0031
```

```
*Mar 2 13:33:15.731:
```

As you see above, the RRQ was rejected by the gatekeeper, but why? If you noticed in the configuration of the gateway, there was no Network Time Protocol (NTP) set. The gatekeeper checks the time stamp of the token to see if it is within an acceptable window relative to its own time. Currently this window is +/- 30 seconds around the gatekeeper's UTC time.

A token outside of this window causes the gatekeeper to discard this message. This will prevent replay attacks from someone trying to reuse a snooped token. In practice, all gateways and gatekeepers should use NTP to avoid this time skew problem. The gatekeeper found that the time stamp in the token is within the acceptable window of its time so it did not even bother to check with RADIUS to authenticate the gateway.

The gateway was then configured for NTP pointing to the gatekeeper as the NTP master, so that both the gateway and gatekeeper would have the same time. When this occurred, the gateway sent a new RRQ and this time the gatekeeper replied back to the new RRQ with an RRJ.

The following debugs are from the gatekeeper. Remember that debugs is running to see if the gatekeeper will go to the authentication phase.

```
Mar 2 13:57:41.313: RAS INCOMING ENCODE BUFFER ::= 0E C0005906 0008914A
00028001 00AC100D 0F06B801 00AC100D 0FD2C608 80013C05 04010020 40000240
0E006700 77006100 2D003100 40006300 69007300 63006F00 2E006300 6F006D00
80CC0800 67006B00 61002D00 3100B500 00120EEA 02003B47 014D000A 2A864886
F70C0A01 0201C02B 9367D410 7DD4C637 B6DD4E34 0883A7E5 E12A2B78 012C1E00
67007700 61002D00 31004000 63006900 73006300 6F002E00 63006F00 6D000042
01040E00 67007700 61002D00 31004000 63006900 73006300 6F002E00 63006F00
6DC02B93 67D4082A 864886F7 0D020500 8080ED73 946B13E9 EAED6F4D FED13478
A6270100 0100
```

```
Mar 2 13:57:41.345:
```

```
Mar 2 13:57:41.349: RAS INCOMING PDU ::=
```

```
value RasMessage ::= registrationRequest :
{
  requestSeqNum 90
  protocolIdentifier { 0 0 8 2250 0 2 }
  discoveryComplete TRUE
  callSignalAddress
  {
    ipAddress :
    {
```



```

    ip 'AC100D0F'H
    port 1720
  }
}
rasAddress
{
  ipAddress :
  {
    ip 'AC100D0F'H
    port 53958
  }
}
terminalType
{
  gateway
  {
    protocol
    {
      voice :
      {
        supportedPrefixes
        {
          {
            prefix e164 : "1#"
          }
        }
      }
    }
  }
}
mc FALSE
undefinedNode FALSE
}
terminalAlias
{
  h323-ID : {"gwa-1@cisco.com"},
  e164 : "99"
}
gatekeeperIdentifier {"gka-1"}
endpointVendor
{
  vendor
  {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
  }
}
timeToLive 60
tokens
{
  {
    tokenOID { 1 2 840 113548 10 1 2 1 }
    timeStamp 731080661
    challenge '7DD4C637B6DD4E340883A7E5E12A2B78'H
    random 44
    generalID {"gwa-1@cisco.com"}
  }
}
}

```

```

cryptoTokens
{
  cryptoEPPwdHash :
  {
    alias h323-ID : {"gwa-1@cisco.com"}
    timeStamp 731080661
    token
    {
      algorithmOID { 1 2 840 113549 2 5 }
      paramS
      {
      }
      hash "ED73946B13E9EAED6F4DFED13478A627"
    }
  }
}
keepAlive FALSE
willSupplyUUIEs FALSE
}

```

```

Mar 2 13:57:41.401: AAA: parse name=<no string> idb type=-1 tty=-1
Mar 2 13:57:41.405: AAA/MEMORY: create_user (0x81416060) user='gwa-1@cisco.com'
ruser='NULL' ds0=0port='NULL' rem_addr='NULL' authen_type=CHAP
service=LOGIN priv=0 initial_task_id='0'
Mar 2 13:57:41.405: AAA/AUTHEN/START (2845574558): port='' list='h323'
action=LOGIN service=LOGIN
Mar 2 13:57:41.405: AAA/AUTHEN/START (2845574558): found list h323
Mar 2 13:57:41.405: AAA/AUTHEN/START (2845574558): Method=LOCAL
Mar 2 13:57:41.405: AAA/AUTHEN (2845574558): User not found, end of method list
Mar 2 13:57:41.405: AAA/AUTHEN (2845574558): status = FAIL

```

*!--- Authentication failed, user not found on list.*

```

Mar 2 13:57:41.405: voip_chapstyle_auth: astruct.status = 2
Mar 2 13:57:41.405: AAA/MEMORY: free_user (0x81416060) user='gwa-1@cisco.com'
ruser='NULL' port='NULL' rem_addr='NULL' authen_type=CHAP service=LOGIN priv=0
Mar 2 13:57:41.409: RAS OUTGOING PDU ::=
value RasMessage ::= registrationReject :
{
  requestSeqNum 90
  protocolIdentifier { 0 0 8 2250 0 3 }
  rejectReason securityDenial : NULL
  gatekeeperIdentifier {"gka-1"}
}

```

As you can see in the example above, after configuring NTP, the gatekeeper still rejects the RRQ. This time, however, it went through the authentication process for that gateway. The gatekeeper rejects the RRQ because the user (here the gateway ID) was not found on the list of valid users on the RADIUS. Remember in the configuration of the gatekeeper, we are authenticating the gateway locally. On the user list you can see gwa-1 but that is not right user since the user in the RRQ is gwa-1@cisco.com. You should be aware of such an issue.

Also, once we configured the username gwa-1@cisco.com password 0 2222 command on the gatekeeper, the gatekeeper confirmed the RRQ and the gateway was registered.

In this lab, another gateway (gwa-2) was registered to the same gatekeeper (gka-1) and we made a call from gwa-1@cisco.com to gwa-2 just to see how the ARQ, ACF, and setup messages will look.

The following debugs collected are from the originating and terminating gateway (gwa-2).

- debug h225 asn1
- debug ras
- debug voip ccapi inout

An explanation of some of the debug messages is included.

Before we print the debugs from the originating/terminating gateway, the following text explains the call flow:

1. Upon receiving a SETUP message from the PSTN, the gateway sends an ARQ to and receives an ACF from the gatekeeper.
2. When the gateway receives the ACF, the gateway generates a CAT using the gateway password, H323-ID alias, and current time. The token is placed in the Call Control Block (CCB).
3. When the gateway sends the SETUP message to the terminating gateway it retrieves the access token from the CCB and places it in the nonStandardParameter field of a clearToken within the SETUP message.
4. The terminating gateway removes the token from the SETUP message, converts it from a nonStandardParameter into a CAT, and places it into the ARQ.
5. The gatekeeper checks the time stamp of the token to see if it is within an acceptable window relative to its own time. Currently this window is +/- 30 seconds around the gatekeeper's UTC time. A token outside of this window causes the gatekeeper to discard this message causing the call to be rejected.
6. If the token is acceptable, the gatekeeper formats a RADIUS Access Request packet, filling in the appropriate attributes to verify a CHAP Challenge and sends it to a RADIUS server.
7. Assuming that the gateway's alias is known at the server, the server locates the password associated with this alias and generates its own CHAP response using the alias, password, and the CHAP Challenge from the gatekeeper. If its CHAP response matches the one received from the gatekeeper the server sends an Access Accept packet to the gatekeeper. If they don't match, or if the gateway's alias is not in the servers database, the server sends an Access Reject packet back to the gatekeeper.
8. The gatekeeper responds to the gateway with an ACF if it received an Access Accept, or an ARJ with cause code **securityDenial** if it received an Access Reject. If the gateway receives an ACF, the call is connected.

The following example shows the debug from the originating gateway.

**Note:** The **h225 asn1** debugs for the setup is not in this example since it will be the same as seen in the terminating gateway example that follows the originating gateway example.

```

Mar 2 19:39:07.376: cc_api_call_setup_ind (vdbPtr=0x6264AB2C,
callInfo={called=3653,called_oct3=0x81,calling=,calling_oct3=0x81,calling_oct3a=0x0,
calling_xlated=false,subscriber_type_str=RegularLine,fdest=1,peer_tag=5336,
prog_ind=3},callID=0x61DDC2A8)
Mar 2 19:39:07.376: cc_api_call_setup_ind type 13 , prot 0
Mar 2 19:39:07.376: cc_process_call_setup_ind (event=0x6231F0C4)
Mar 2 19:39:07.380: >>>CCAPI handed cid 30 with tag 5336 to app "DEFAULT"
Mar 2 19:39:07.380: sess_appl: ev(24=CC_EV_CALL_SETUP_IND), cid(30), disp(0)
Mar 2 19:39:07.380: sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(30), disp(0)
Mar 2 19:39:07.380: ssaCallSetupInd
Mar 2 19:39:07.380: ccCallSetContext (callID=0x1E, context=0x6215B5A0)
Mar 2 19:39:07.380: ssaCallSetupInd cid(30), st(SSA_CS_MAPPING),oldst(0),
ev(24)ev->e.evCallSetupInd.nCallInfo.finalDestFlag = 1
Mar 2 19:39:07.380: ssaCallSetupInd finalDest cllng(1#5336), cllcd(3653)
Mar 2 19:39:07.380: ssaCallSetupInd cid(30), st(SSA_CS_CALL_SETTING),oldst(0),
ev(24)dpMatchPeersMoreArg result= 0
Mar 2 19:39:07.380: ssaSetupPeer cid(30) peer list: tag(3653) called number (3653)
Mar 2 19:39:07.380: ssaSetupPeer cid(30), destPat(3653), matched(4), prefix(),
peer(62664554), peer->encapType (2)
Mar 2 19:39:07.380: ccCallProceeding (callID=0x1E, prog_ind=0x0)
Mar 2 19:39:07.380: ccCallSetupRequest (Inbound call = 0x1E, outbound peer =3653,
dest=, params=0x62327730 mode=0, *callID=0x62327A98, prog_ind = 3)
Mar 2 19:39:07.380: ccCallSetupRequest numbering_type 0x81
Mar 2 19:39:07.380: ccCallSetupRequest encapType 2 clid_restrict_disable 1
null_orig_clg 1 clid_transparent 0 callingNumber 1#5336
Mar 2 19:39:07.380: dest pattern 3653, called 3653, digit_strip 0
Mar 2 19:39:07.380: callingNumber=1#5336, calledNumber=3653, redirectNumber=
display_info= calling_oct3a=0
Mar 2 19:39:07.384: accountNumber=, finalDestFlag=1,
guid=6aef.3a87.165c.11cc.8040.d661.b74f.9390

```

```

Mar 2 19:39:07.384: peer_tag=3653
Mar 2 19:39:07.384: ccIFCallSetupRequestPrivate: (vdbPtr=0x621B2360, dest=,
callParams={called=3653,called_oct3=0x81, calling=1#5336,calling_oct3=0x81,
calling_xlated=false, subscriber_type_str=RegularLine, fdest=1,
voice_peer_tag=3653},mode=0x0) vdbPtr type = 1
Mar 2 19:39:07.384: ccIFCallSetupRequestPrivate: (vdbPtr=0x621B2360, dest=,
callParams={called=3653, called_oct3 0x81, calling=1#5336,calling_oct3
0x81, calling_xlated=false, fdest=1, voice_peer_tag=3653}, mode=0x0, xltrc=-5)
Mar 2 19:39:07.384: ccSaveDialpeerTag (callID=0x1E, dialpeer_tag=0xE45)
Mar 2 19:39:07.384: ccCallSetContext (callID=0x1F, context=0x621545DC)
Mar 2 19:39:07.384: ccCallReportDigits (callID=0x1E, enable=0x0)
Mar 2 19:39:07.384: cc_api_call_report_digits_done (vdbPtr=0x6264AB2C,
callID=0x1E, disp=0)
Mar 2 19:39:07.384: sess_appl: ev(52=CC_EV_CALL_REPORT_DIGITS_DONE), cid(30),disp(0)
Mar 2 19:39:07.384: cid(30)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_REPORT_DIGITS_DONE)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(1)
Mar 2 19:39:07.384: -cid2(31)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_MAPPING)
Mar 2 19:39:07.384: ssaReportDigitsDone cid(30) peer list: (empty)
Mar 2 19:39:07.384: ssaReportDigitsDone callid=30 Reporting disabled.
Mar 2 19:39:07.388: H225 NONSTD OUTGOING PDU ::=
value ARQnonStandardInfo ::=
{
  sourceAlias
  {
  }
  sourceExtAlias
  {
  }
  interfaceSpecificBillingId "ISDN-VOICE"
}
Mar 2 19:39:07.388: H225 NONSTD OUTGOING ENCODE BUFFER ::= 80 00000820
0B124953 444E2D56 4F494345
Mar 2 19:39:07.388:
Mar 2 19:39:07.388: H235 OUTGOING ENCODE BUFFER ::= 61 000100C0 2B93B7DA
08003200 32003200 3200001E 00670077 0061002D 00310040 00630069 00730063
006F002E 0063006F 006D0000
Mar 2 19:39:07.392:
Mar 2 19:39:07.392: RAS OUTGOING PDU ::=
value RasMessage ::= admissionRequest :

```

*!--- ARQ is sent to the gatekeeper*

```

{
  requestSeqNum 549
  callType pointToPoint : NULL
  callModel direct : NULL
  endpointIdentifier {"8155346000000001"}
  destinationInfo
  {
    e164 : "2#3653"
  }
  srcInfo
  {
    e164 : "1#5336",
    h323-ID : {"gwa-1@cisco.com"}
  }
  bandwidth 640
  callReferenceValue 15
  nonStandardData
  {

```

```

nonStandardIdentifier h221NonStandard :
{
  t35CountryCode 181
  t35Extension 0
  manufacturerCode 18
}
data '80000008200B124953444E2D564F494345'H
}
conferenceID '6AEF3A87165C11CC8040D661B74F9390'H
activeMC FALSE
answerCall FALSE
canMapAlias TRUE
callIdentifier
{
  guid '6AEF3A87165C11CC8041D661B74F9390'H
}
tokens

```

*!--- Token is included since we have an **all** level of security*

```

{
  {
    tokenOID { 1 2 840 113548 10 1 2 1 }
    timeStamp 731101147
    challenge '1CADDBA948A8291C1F134035C9613E3E'H
    random 246
    generalID {"gwa-1@cisco.com"}
  }
}
cryptoTokens
{
  cryptoEPPwdHash :
  {
    alias h323-ID : {"gwa-1@cisco.com"}
    timeStamp 731101147
    token
    {
      algorithmOID { 1 2 840 113549 2 5 }
      params
      {
      }
      hash "5760B7B4877335B7CD24BD24E4A2AA89"
    }
  }
}
willSupplyUUIEs FALSE
}

```

```

Mar 2 19:39:07.408: RAS OUTGOING ENCODE BUFFER::= 27 88022400 F0003800
31003500 35003300 34003600 30003000 30003000 30003000 30003000 31010280
50698602 02804086 69400E00 67007700 61002D00 31004000 63006900 73006300
6F002E00 63006F00 6D400280 000F40B5 00001211 80000008 200B1249 534444E2D
564F4943 456AEF3A 87165C11 CC8040D6 61B74F93 9004E320 01801100 6AEF3A87
165C11CC 8041D661 B74F9390 48014D00 0A2A8648 86F70C0A 010201C0 2B93B7DA
101CADDB A948A829 1C1F1340 35C9613E 3E0200F6 1E006700 77006100 2D003100
40006300 69007300 63006F00 2E006300 6F006D00 00420104 0E006700 77006100
2D003100 40006300 69007300 63006F00 2E006300 6F006DC0 2B93B7DA 082A8648
86F70D02 05008080 5760B7B4 877335B7 CD24BD24 E4A2AA89 0100
Mar 2 19:39:07.412: h323chan_dgram_send:Sent UDP msg. Bytes sent: 291 to

```

```
172.16.13.35:1719
```

```
Mar 2 19:39:07.416: RASLib::GW_RASSendARQ: ARQ (seq# 549) sent to 172.16.13.35
Mar 2 19:39:07.432: h323chan_dgram_rcvdata:rcvd from [172.16.13.35:1719] on sock[1]
```

```
Mar 2 19:39:07.432: RAS INCOMING ENCODE BUFFER ::= 2B 00022440 028000AC
100D1706 B800EF1A 00C00100 020000
Mar 2 19:39:07.432:
Mar 2 19:39:07.432: RAS INCOMING PDU ::=
value RasMessage ::= admissionConfirm :
```

*!--- Received from the gatekeeper with no tokens*

```
{
  requestSeqNum 549
  bandwidth 640
  callModel direct : NULL
  destCallSignalAddress ipAddress :
  {
    ip 'AC100D17'H
    port 1720
  }
  irrFrequency 240
  willRespondToIRR FALSE
  uuiesRequested
  {
    setup FALSE
    callProceeding FALSE
    connect FALSE
    alerting FALSE
    information FALSE
    releaseComplete FALSE
    facility FALSE
    progress FALSE
    empty FALSE
  }
}
```

```
Mar 2 19:39:07.436: ACF (seq# 549) rcvd
```

The following example shows the debugs from the terminating gateway (TGW). Notice that the TGW has set up the second leg since it got ACF, and the call will be connected.

```
Mar 2 19:39:07.493: PDU DATA = 6147C2BC
```

```
value H323_UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body setup :
    {
      protocolIdentifier { 0 0 8 2250 0 2 }
      sourceAddress
      {
        h323-ID : {"gwa-1@cisco.com"}
      }
    }
  }
}
```

*!--- Setup sent from gwa-1@cisco.com gateway*

```
}
sourceInfo
{
```

```

gateway
{
  protocol
  {
    voice :
    {
      supportedPrefixes
      {
        {
          prefix e164 : "1#"
        }
      }
    }
  }
}
mc FALSE
undefinedNode FALSE
}
activeMC FALSE
conferenceID '6AEF3A87165C11CC8040D661B74F9390'H
conferenceGoal create : NULL
callType pointToPoint : NULL
sourceCallSignalAddress ipAddress :
{
  ip 'AC100D0F'H
  port 11032
}
callIdentifier
{
  guid '6AEF3A87165C11CC8041D661B74F9390'H
}
tokens

```

*!--- Setup includes the Clear Token (CAT)*

```

{
  {
    tokenOID { 1 2 840 113548 10 1 2 1 }
    timeStamp 731101147
    challenge 'AFBAAFDF79446B9D8CE164DB8C111A87'H
    random 247
    generalID {"gwa-1@cisco.com"}
    nonStandard
    {
      nonStandardIdentifier { 0 1 2 4 }
      data '2B93B7DBAFBAAFDF79446B9D8CE164DB8C111A87...'H
    }
  }
}
fastStart
{
  '0000000C6013800A04000100AC100D0F4673'H,
  '400000060401004C6013801114000100AC100D0F...'H
}
mediaWaitForConnect FALSE
canOverlapSend FALSE
}
h245Tunneling TRUE
nonStandardControl

```

```

{
  {
    nonStandardIdentifier h221NonStandard :
    {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
    }
    data 'E001020001041504039090A31803A983811E0285...'H
  }
}
}
}

```

RAW\_BUFFER::=

```
E0 01020001 04150403 9090A318 03A98381 1E028583 70058133 36353302 80060004
00000003
```

Mar 2 19:39:07.509: PDU DATA = 6147F378

value H323\_UU\_NonStdInfo ::=

```

{
  version 2
  protoParam qsigNonStdInfo :
  {
    iei 4
    rawMesg '04039090A31803A983811E028583700581333635...'H
  }
  progIndParam progIndIEinfo :
  {
    progIndIE '00000003'H
  }
}

```

PDU DATA = 6147F378

value ARQnonStandardInfo ::=

```

{
  sourceAlias
  {
  }
  sourceExtAlias
  {
  }
}

```

RAW\_BUFFER::=

```
00 0000
```

Mar 2 19:39:07.517: RAW\_BUFFER::=

```
61 000100C0 2B93B7DA 08003200 32003200 3200000A 00670077 0061002D
00320000
```

Mar 2 19:39:07.517: PDU DATA = 6147C2BC

value RasMessage ::= **admissionRequest** :

*!--- An answer ARQ is sent to the gatekeeper to authenticate the caller*

```

{
  requestSeqNum 22

```



```

callType pointToPoint : NULL
callModel direct : NULL
endpointIdentifier {"81F5989C00000002"}
destinationInfo
{
  e164 : "2#3653"
}
srcInfo
{
  e164 : "1#5336"
}
srcCallSignalAddress ipAddress :
{
  ip 'AC100D0F'H
  port 11032
}
bandWidth 640
callReferenceValue 2
nonStandardData
{
  nonStandardIdentifier h221NonStandard :
  {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
  }
  data '000000'H
}
conferenceID '6AEF3A87165C11CC8040D661B74F9390'H
activeMC FALSE
answerCall TRUE
canMapAlias FALSE
callIdentifier
{
  guid '6AEF3A87165C11CC8041D661B74F9390'H
}
tokens

```

*!--- CAT is included*

```

{

  {
    tokenOID { 0 4 0 1321 1 2 }
    timeStamp 731101147
    challenge 'AFBAAFDF79446B9D8CE164DB8C111A87'H
    random 247
    generalID {"gwa-1@cisco.com"}
  }
}
cryptoTokens
{
  cryptoEPPwdHash :
  {
    alias h323-ID : {"gwa-2"}
    timeStamp 731101147
    token
    {
      algorithmOID { 1 2 840 113549 2 5 }
    }
  }
}

```

```

    params
    {
    }
    hash "8479E7DE63AC17C6A46E9E19659568"
  }
}
willSupplyUUIEs FALSE
}
RAW_BUFFER::=
27 98001500 F0003800 31004600 35003900 38003900 43003000 30003000 30003000
30003000 32010280 50698601 02804086 6900AC10 0D0F2B18 40028000 0240B500
00120300 00006AEF 3A87165C 11CC8040 D661B74F 939044E3 20010011 006AEF3A
87165C11 CC8041D6 61B74F93 9044014D 00060400 8A290102 C02B93B7 DA10AFBA
AFDF7944 6B9D8CE1 64DB8C11 1A870200 F71E0067 00770061 002D0031 00400063
00690073 0063006F 002E0063 006F006D 00002E01 04040067 00770061 002D0032
C02B93B7 DA082A86 4886F70D 02050080 808479E7 0DE63AC1 7C6A46E9 E1965905
680100
Mar 2 19:39:07.533: h323chan_dgram_send:Sent UDP msg. Bytes sent: 228
to 172.16.13.35:1719

Mar 2 19:39:07.533: RASLib::GW_RASSendARQ: ARQ (seq# 22) sent to 172.16.13.35
Mar 2 19:39:07.549: h323chan_dgram_recvddata:rcvd from [172.16.13.35:1719]
on sock[1]
RAW_BUFFER::=
2B 00001540 028000AC 100D1706 B800EF1A 00C00100 020000
Mar 2 19:39:07.549: PDU DATA = 6147C2BC
value RasMessage ::= admissionConfirm :
!--- ACF is received from the gatekeeper.
{
  requestSeqNum 22
  bandwidth 640
  callModel direct : NULL
  destCallSignalAddress ipAddress :
  {
    ip 'AC100D17'H
    port 1720
  }
  irrFrequency 240
  willRespondToIRR FALSE
  uuiesRequested
  {
    setup FALSE
    callProceeding FALSE
    connect FALSE
    alerting FALSE
    information FALSE
    releaseComplete FALSE
    facility FALSE
    progress FALSE
    empty FALSE
  }
}
}

Mar 2 19:39:07.553: ACF (seq# 22) rcvd
Mar 2 19:39:07.553: cc_api_call_setup_ind (vdbPtr=0x61BC92EC,
callInfo={called=2#3653,called_oct3=0x81,calling=1#5336,calling_oct3=0x81,
calling_oct3a=0x0,subscriber_type_str=Unknown, fdest=1 peer_tag=5336,
prog_ind=3},callID=0x6217CC64)

```

```

Mar 2 19:39:07.553: cc_api_call_setup_ind type 0 , prot 1
Mar 2 19:39:07.553: cc_api_call_setup_ind (vdbPtr=0x61BC92EC,
callInfo={called=2#3653, calling=1#5336, fdest=1 peer_tag=5336},
callID=0x6217CC64)
Mar 2 19:39:07.553: cc_process_call_setup_ind (event=0x61E1EAFc)
Mar 2 19:39:07.553: >>>CCAPI handed cid 9 with tag 5336 to app "DEFAULT"
Mar 2 19:39:07.553: sess_appl: ev(25=CC_EV_CALL_SETUP_IND), cid(9), disp(0)
Mar 2 19:39:07.553: sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(9), disp(0)
Mar 2 19:39:07.553: ssaCallSetupInd
Mar 2 19:39:07.553: ccCallSetContext (callID=0x9, context=0x62447A28)
Mar 2 19:39:07.553: ssaCallSetupInd cid(9), st(SSA_CS_MAPPING),oldst(0),
ev(25)ev->e.evCallSetupInd.nCallInfo.finalDestFlag = 1
Mar 2 19:39:07.553: ssaCallSetupInd finalDest cllng(1#5336), cllcd(2#3653)
Mar 2 19:39:07.553: ssaCallSetupInd cid(9), st(SSA_CS_CALL_SETTING),oldst(0),
ev(25)dpMatchPeersMoreArg result= 0
Mar 2 19:39:07.557: ssaSetupPeer cid(9) peer list: tag(3653)
called number (2#3653)
Mar 2 19:39:07.557: ssaSetupPeer cid(9), destPat(2#3653), matched(5),
prefix(21), peer(620F1EF0), peer->encapType (1)
Mar 2 19:39:07.557: ccCallProceeding (callID=0x9, prog_ind=0x0)
Mar 2 19:39:07.557: ccCallSetupRequest (Inbound call = 0x9, outbound peer
=3653, dest=, params=0x61E296C0 mode=0, *callID=0x61E299D0, prog_ind = 3)
Mar 2 19:39:07.557: ccCallSetupRequest numbering_type 0x81
Mar 2 19:39:07.557: dest pattern 2#3653, called 2#3653, digit_strip 1
Mar 2 19:39:07.557: callingNumber=1#5336, calledNumber=2#3653,
redirectNumber=display_info= calling_oct3a=0
Mar 2 19:39:07.557: accountNumber=, finalDestFlag=1,
guid=6aef.3a87.165c.11cc.8040.d661.b74f.9390
Mar 2 19:39:07.557: peer_tag=3653
Mar 2 19:39:07.557: ccIFCallSetupRequestPrivate: (vdbPtr=0x61E4473C, dest=,
callParams={called=2#3653,called_oct3=0x81, calling=1#5336,calling_oct3=0x81,
subscriber_type_str=Unknown, fdest=1, voice_peer_tag=3653},mode=0x0) vdbPtr
type = 6
Mar 2 19:39:07.557: ccIFCallSetupRequestPrivate: (vdbPtr=0x61E4473C, dest=,
callParams={called=2#3653, called_oct3 0x81, calling=1#5336,calling_oct3 0x81,
fdest=1, voice_peer_tag=3653}, mode=0x0, xltrc=-4)
Mar 2 19:39:07.557: ccSaveDialpeerTag (callID=0x9, dialpeer_tag=
Mar 2 19:39:07.557: ccCallSetContext (callID=0xA, context=0x6244D9EC)
Mar 2 19:39:07.557: ccCallReportDigits (callID=0x9, enable=0x0)

```

## Gateway IOS Problem

In the same lab, IOS image 12.2(6a) was loaded on the OGW. When a call was made, it was noticed that the OGW still sends a Clear Token based on its password even though the gateway was not configured for IVR to collect the Account/PIN. In addition, the gatekeeper configured for the all level accepted that call. This is documented in the bug ID CSCdw43224.

## Security with Alternate Endpoints

As mentioned above, we can provide end-to-end call security with the use of access tokens that is sent in the clearTokens field in the RAS/H225 messages. When enabling such security, the source gateway forwards the access token received from the gatekeeper in an ACF to the destination H.323 endpoint in the H.225 SETUP message. This destination H.323 endpoint then itself forwards the access token received in the SETUP message to the gatekeeper in its admission request. By doing this, it gives the remote gatekeeper the ability to admit calls based on the validity of the access token. The contents of the access token are up to the entity generating it. To minimize security holes and to guard against man-in-the-middle attacks, gatekeepers may encode destination specific information in the access token. This means that when alternateEndpoints are provided in an ACF, the gatekeeper may provide a separate access token for each alternateEndpoint specified.

When first attempting to establish a connection, the Cisco gateway sends the access token it has received in the clearToken field of the ACF with the address in the destCallSignalAddress field. If this attempt is unsuccessful and the Cisco gateway proceeds to attempt connections with an alternate endpoint, it uses the associated access token (if it is available) from the alternateEndpoints list. If the alternateEndpoints list received in the ACF does not include access tokens, but the ACF itself included an access token, the Cisco gateway includes this access

token in all attempts to connect with an alternate endpoint.

## OSP Token Support

Currently the Open Settlement Protocol (OSP) and its tokens are only supported on Cisco gateways; there is no support on the gatekeeper. The gateway recognizes OSP tokens received from a settlement server and inserts them into the Q.931 setup message to a terminating gateway.

## Different Levels of Security for each Endpoint or Zone

Currently you cannot configure different levels of security for each endpoint or zone. The security level is for all zones managed by that gatekeeper. A feature request can be opened for such issue.

## Interdomain Gatekeeper to Gatekeeper Security

Interdomain gatekeeper to gatekeeper security provides the ability to validate intradomain and interdomain gatekeeper-to-gatekeeper requests on a per-hop basis. This means that the destination gatekeeper will terminate the CAT and generate a new one if the gatekeeper decides to forward the LRQ onwards. If the gatekeeper detects an invalid LRQ signature it responds by sending a Location Reject (LRJ).

### Implementing Gatekeeper to Gatekeeper Security

The originating gatekeeper generates an Inter-zone ClearToken (IZCT) when a location request (LRQ) is initiated or an ACF is about to be sent in case of an intra-zone call. This token is traversed through its routing path. Along the path, each gatekeeper updates the destination gatekeeper ID and/or source gatekeeper ID, if necessary, to reflect the zone information. The terminating gatekeeper generates a token with its password, and this token is carried back in the location confirmation (LCF) messages and passed to OGW. The OGW includes this token in the H.225 SETUP message. When the TGW receives the token, it is forwarded in the ARQ answerCall and validated by the terminating gatekeeper (TGW) without any need for a RADIUS server.

The authentication type is based on password with hashing as described in ITU H.235. Specifically, the encryption method is MD5 with password hashing.

The purpose of the IZCT is to know if the LRQ has arrived from a foreign domain, from which zone, from which carrier. It is also used to pass a token to the OGW in the LCF from the TGK. Within the IZCT format, the following information is required:

- srcCarrierID -Source carrier identification
- dstCarrierID - Destination carrier identification
- intCarrierID - Intermediate carrier identification
- srcZone - Source zone
- dstZone - Destination zone
- interzone type
  - INTRA\_DOMAIN\_CISCO
  - INTER\_DOMAIN\_CISCO
  - INTRA\_DOMAIN\_TERM\_NOT\_CISCO
  - INTER\_DOMAIN\_ORIG\_NOT\_CISCO

This feature will work fine without any need for carrier ID from the gateway or a Carrier Sensitive Routing (CSR) server. In such case, the fields about the carrier ID will be empty. The examples below do not include any carrier ID. For a detailed call flow, release and platform support, and configurations, refer to the [Interdomain Gatekeeper Security Enhancement](#) document.

## Gatekeeper Configuration

The IZCT feature requires the following configuration on the gatekeeper.

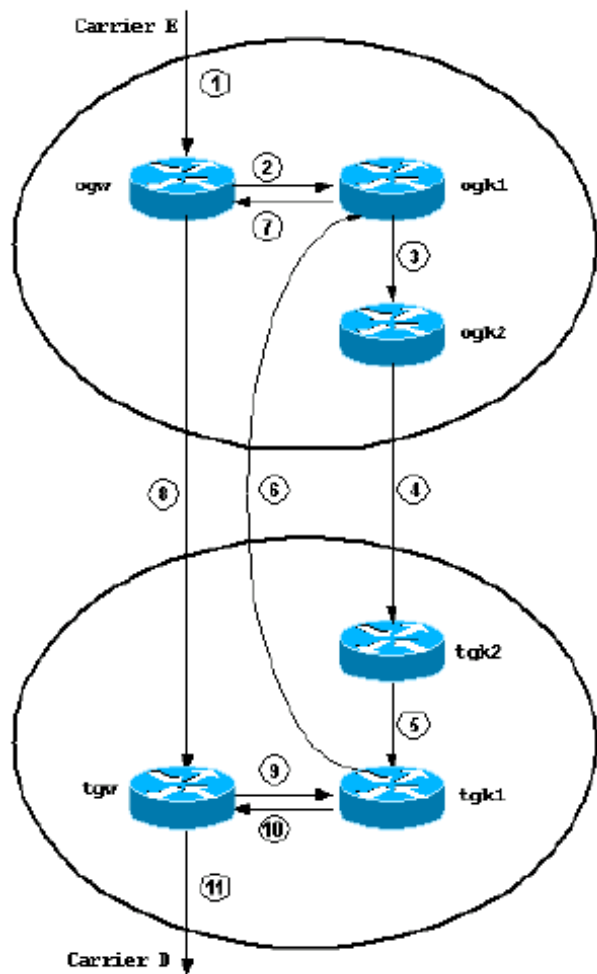
```
Router(gk-config)#[no] security izct password <PASSWORD>
```

The password should be six to eight characters. You will also need to identify which zone is in a foreign domain as follows:

```
Router(config-gk)#zone remote other-gatekeeper-name other-domain-name
other-ip-address
[port-number] [cost cost-value [priority priority-value]] [foreign-domain]
```

## IZCT Call Flow

The following diagram shows the IZCT flow.



In the following configuration, the names of the gateways and gatekeepers is the same as those used in the above diagram but with lower case. The call flow is explained after the configuration, with debug explanations.

To explain the IZCT feature and call flow, the first example does not have the intradomain gateway to gatekeeper security. After that, there will be examples where the TGW was not able to generate the IZCT so that the TGK1 would reject the call. This is to show that the feature is working as designed. All of these setups will be based on the topology in the above diagram.

### Example 1: Call flow for gatekeeper to gatekeeper security only

The following example shows the related configurations of all gateways and gatekeepers.

#### OGW Configuration

#### TGW Configuration

<pre> ! hostname ogw !controller E1 3/0   pri-group timeslots 1-2,16 ! interface Ethernet0/0   ip address 172.16.13.15 255.255.255.224   half-duplex   h323-gateway voip interface   h323-gateway voip id ogk1 ipaddr 172.16.13.35 1718   h323-gateway voip h323-id ogw   h323-gateway voip tech-prefix 1# ! voice-port 3/0:15 ! dial-peer voice 5336 pots   incoming called-number .   destination-pattern 5336   direct-inward-dial   port 3/0:15   prefix 21 ! dial-peer voice 3653 voip   incoming called-number .   destination-pattern 3653   session target ras   dtmf-relay h245-alphanumeric   codec g711ulaw ! gateway ! ntp clock-period 17178791 ntp server 172.16.13.35 end </pre>	<pre> hostname tgw ! controller E1 0   clock source line primary   ds0-group 0 timeslots 1-2 type r2-digital r2-compelled ! interface Ethernet0   ip address 172.16.13.23 255.255.255.224   h323-gateway voip interface   h323-gateway voip id tgk1 ipaddr 172.16.13.41 1718   h323-gateway voip h323-id tgw   h323-gateway voip tech-prefix 2# ! voice-port 0:0   compand-type a-law ! dial-peer voice 3653 pots   application test1   incoming called-number .   destination-pattern 3653   port 0:0   prefix 21 ! dial-peer voice 5336 voip   incoming called-number .   destination-pattern 5336   session target ras   dtmf-relay h245-alphanumeric   codec g711ulaw ! gateway ! ntp clock-period 17179814 ntp server 172.16.13.35 end </pre>
---	--

**OGK1 Configuration****TGK1 Configuration**

```

!
hostname ogk1
!
interface Ethernet0/0
  ip address 172.16.13.35
255.255.255.224
  half-duplex
!
gatekeeper
  zone local ogk1 domainA.com
172.16.13.35
  zone remote ogk2 domainA.com
172.16.13.14 1719
  zone prefix ogk2 36*
  zone prefix ogk1 53*
  security izct password 111222
  gw-type-prefix 1#* default-
technology
no shutdown
!

```

```

!
hostname tgk1
!
interface Ethernet0/0
  ip address 172.16.13.41
255.255.255.224
  ip directed-broadcast
  half-duplex
!
gatekeeper
  zone local tgk1 domainB.com
172.16.13.41
  zone remote tgk2 domainB.com
172.16.13.16 1719
  zone prefix tgk1 36*
  zone prefix tgk2 53*
  security izct password 111222
  gw-type-prefix 2#* default-
technology

```

!	no shutdown
no scheduler max-task-time	!
no scheduler allocate	ntp clock-period 17179797
ntp master	ntp server 172.16.13.35
!	!
end	end

**OGK2 Configuration**

```
!
hostname ogk2
!
interface Ethernet0/0
 ip address 172.16.13.14
 255.255.255.224
 full-duplex
!
gatekeeper
 zone local ogk2 domainA.com
 zone remote ogk1 domainA.com
 172.16.13.35 1719
 zone remote tgk2 domainB.com
 172.16.13.16 1719 foreign-domain
 zone prefix tgk2 36*
 zone prefix ogk1 53*
 security izct password 111222
 lrq forward-queries
 no shutdown
!
ntp clock-period 17208242
ntp server 172.16.13.35
!
end
```

**TGK2 Configuration**

```
!
hostname tgk2
!
interface Ethernet0/0
 ip address 172.16.13.16
 255.255.255.224
 half-duplex
!
gatekeeper
 zone local tgk2 domainB.com
 zone remote tgk1 domainB.com
 172.16.13.41 1719
 zone remote ogk2 domainA.com
 172.16.13.14 1719 foreign-domain
 zone prefix tgk1 36*
 zone prefix ogk2 53*
 security izct password 111222
 lrq forward-queries
 no shutdown
!
ntp clock-period 17179209
ntp server 172.16.13.35
!
end
```

## Call Flow with Debugs

The following examples use the debugs to explain the call flow.

1. A user on carrier E calls a user on carrier D.

```
Mar 4 15:31:19.989: cc_api_call_setup_ind (vdbPtr=0x6264ADF0, callInfo={called=3653,
called_oct3=0x80, calling=4085272923, calling_oct3=0x21, calling_oct3a=0x80
calling_xlated=false, subscriber_type_str=RegularLine, fdest=1, peer_tag=5336,
prog_ind=0}, callID=0x6219F9F0)
Mar 4 15:31:19.993: cc_api_call_setup_ind type 13 , prot 0
Mar 4 15:31:19.993: cc_process_call_setup_ind (event=0x6231A6B4)
Mar 4 15:31:19.993: >>>CCAPI handed cid 7 with tag 5336 to app "DEFAULT"
Mar 4 15:31:19.993: sess_appl: ev(24=CC_EV_CALL_SETUP_IND), cid(7), disp(0)
Mar 4 15:31:19.993: sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(7), disp(0)
Mar 4 15:31:19.993: ssaCallSetupInd
Mar 4 15:31:19.993: ccCallSetContext (callID=0x7, context=0x621533F0)
Mar 4 15:31:19.997: ssaCallSetupInd cid(7), st(SSA_CS_MAPPING), oldst(0),
ev(24) ev->e.evCallSetupInd.nCallInfo.finalDestFlag = 1
Mar 4 15:31:19.997: ssaCallSetupInd finalDest cllng(4085272923), cllcd(3653)
Mar 4 15:31:19.997: ssaCallSetupInd cid(7), st(SSA_CS_CALL_SETTING), oldst(0),
ev(24)dpMatchPeersMoreArg result= 0
Mar 4 15:31:19.997: ssaSetupPeer cid(7) peer list: tag(3653) called number (3653)
Mar 4 15:31:19.997: ssaSetupPeer cid(7), destPat(3653), matched(4), prefix(),
peer(626640B0), peer->encapType (2)
Mar 4 15:31:19.997: ccCallProceeding (callID=0x7, prog_ind=0x0)
Mar 4 15:31:19.997: ccCallSetupRequest (Inbound call = 0x7, outbound peer=3653,
dest=,
```

```

params=0x62327730 mode=0, *callID=0x62327A98, prog_ind = 0)
Mar 4 15:31:19.997: ccCallSetupRequest numbering_type 0x80
Mar 4 15:31:19.997: ccCallSetupRequest encapType 2 clid_restrict_disable 1 null
_orig_clg 0 clid_transparent 0 callingNumber 4085272923
Mar 4 15:31:19.997: dest pattern 3653, called 3653, digit_strip 0
Mar 4 15:31:19.997: callingNumber=4085272923, calledNumber=3653, redirectNumber
= display_info= calling_oct3a=80
Mar 4 15:31:19.997: accountNumber=, finalDestFlag=1,
guid=221b.686c.17cc.11cc.8010.a049.e052.4766
Mar 4 15:31:19.997: peer_tag=3653
Mar 4 15:31:19.997: ccIFCallSetupRequestPrivate: (vdbPtr=0x621B2360, dest=,
callParams={called=3653,called_oct3=0x80, calling=4085272923,calling_oct3=0x21,
calling_xlated=false, subscriber_type_str=RegularLine, fdest=1, voice_peer_tag=365
3},mode=0x0) vdbPtr type = 1
Mar 4 15:31:19.997: ccIFCallSetupRequestPrivate: (vdbPtr=0x621B2360, dest=,
callParams={called=3653, called_oct3 0x80, calling=4085272923,calling_oct3 0x21,
calling_xlated=false, fdest=1, voice_peer_tag=3653}, mode=0x0, xltrc=-5)
Mar 4 15:31:20.001: ccSaveDialpeerTag (callID=0x7, dialpeer_tag=0xE45)
Mar 4 15:31:20.001: ccCallSetContext (callID=0x8, context=0x6215388C)
Mar 4 15:31:20.001: ccCallReportDigits (callID=0x7, enable=0x0)

```

2. Since the originating gateway's dialpeer (tag=3653) is configured for RAS, it sends an ARQ to OGK1.

```

Mar 4 15:31:20.001: H225 NONSTD OUTGOING PDU ::=

```

```

value ARQnonStandardInfo ::=
{
  sourceAlias
  {
  }
  sourceExtAlias
  {
  }
  callingOctet3a 128
  interfaceSpecificBillingId "ISDN-VOICE"
}

```

```

Mar 4 15:31:20.005: H225 NONSTD OUTGOING ENCODE BUFFER ::= 80 000008A0
01800B12 4953444E 2D564F49 4345
Mar 4 15:31:20.005:
Mar 4 15:31:20.005: RAS OUTGOING PDU ::=

```

```

value RasMessage ::= admissionRequest :

```

```

!--- ARQ is sent out to ogk1

```

```

{
requestSeqNum 1109
callType pointToPoint : NULL
callModel direct : NULL
endpointIdentifier {"81567A4000000001"}
destinationInfo
{
  e164 : "3653"
}
srcInfo
{
  e164 : "4085272923",
  h323-ID : {"ogw"}
}
bandwidth 640

```



```

callReferenceValue 4
nonStandardData
{
  nonStandardIdentifier h221NonStandard :
  {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
  }
  data '80000008A001800B124953444E2D564F494345'H
}
conferenceID '221B686C17CC11CC8010A049E0524766'H
activeMC FALSE
answerCall FALSE
canMapAlias TRUE
callIdentifier
{
  guid '221B686C17CC11CC8011A049E0524766'H
}
willSupplyUUIEs FALSE
}

```

```

Mar 4 15:31:20.013: RAS OUTGOING ENCODE BUFFER ::= 27 88045400 F0003800
31003500 36003700 41003400 30003000 30003000 30003000 30003000 31010180
69860204 8073B85A 5C564002 006F0067 00774002 80000440 B5000012 13800000
08A00180 0B124953 444E2D56 4F494345 221B686C 17CC11CC 8010A049 E0524766
04E02001 80110022 1B686C17 CC11CC80 11A049E0 52476601 00
Mar 4 15:31:20.017: h323chan_dgram_send:Sent UDP msg. Bytes sent: 130 to
172.16.13.35:1719

```

```

Mar 4 15:31:20.017: RASLib::GW_RASSendARQ: ARQ (seq# 1109) sent to
172.16.13.35

```

3. Upon receiving the ARQ, the OGK1 determines that the destination is serviced by the remote zone OGK2. It then identifies that an IZCT is needed (via CLI: **security izct password <pwd>**). OGK1 proceeds to create the IZCT before the LRQ is sent. It then sends the IZCT and the LRQ out to OGK2 and sends a RIP message back to OGW

```

Mar 4 15:31:19.927: H225 NONSTD OUTGOING PDU ::=
value LRQnonStandardInfo ::=
{
  ttl 6
  nonstd-callIdentifier
  {
    guid '221B686C17CC11CC8011A049E0524766'H
  }
  callingOctet3a 128
  gatewaySrcInfo
  {
    e164 : "4085272923",
    h323-ID : {"ogw"}
  }
}

```

```

Mar 4 15:31:19.935: H225 NONSTD OUTGOING ENCODE BUFFER ::= 82 86B01100
221B686C 17CC11CC 8011A049 E0524766 01801002 048073B8 5A5C5640
02006F00 670077
Mar 4 15:31:19.939:
Mar 4 15:31:19.939: PDU ::=

```

```

value IZCToken ::=

```

*!--- The gatekeeper creates and sends out the IZCT*

```

{
  izctInterZoneType intraDomainCisco : NULL

  !--- The destination is in the same domain, so it is intraDomainCisco type

  izctSrcZone "ogk1"

  !--- The source Zone is ogk1

)

```

```

Mar 4 15:31:19.943: ENCODE BUFFER ::= 07 00C06F67 6B310473 72630464
73740469 6E74

```

```

Mar 4 15:31:19.947:

```

```

Mar 4 15:31:19.947: RAS OUTGOING PDU ::=

```

```

value RasMessage ::= locationRequest :

```

```

!--- LRQ is sent out to ogk2

```

```

{
  requestSeqNum 2048
  destinationInfo
  {
    e164 : "3653"
  }
  nonStandardData
  {
    nonStandardIdentifier h221NonStandard :
    {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
    }
    data '8286B01100221B686C17CC11CC8011A049E05247...'H
  }
  replyAddress ipAddress :
  {
    ip 'AC100D23'H
    port 1719
  }
  sourceInfo
  {
    h323-ID : {"ogk1"}
  }
  canMapAlias TRUE
  tokens

```

```

!--- The IZCT is included

```

```

{
  {
    tokenOID { 1 2 840 113548 10 1 0 }
    nonStandard
    {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '0700C06F676B31047372630464737404696E74'H
    }
  }

```

```

    }
  }
}

```

```

Mar 4 15:31:19.967: RAS OUTGOING ENCODE BUFFER ::= 4A 8007FF01 01806986
40B50000
12288286 B0110022 1B686C17 CC11CC80 11A049E0 52476601 80100204 8073B85A
5C56400
2 006F0067 007700AC 100D2306 B70BA00B 01400300 6F006700 6B003101
802B0100 80092A
86 4886F70C 0A010009 2A864886 F70C0A01 00130700 C06F676B 31047372
63046473 74046
96E 74

```

```
Mar 4 15:31:19.983:
```

```
Mar 4 15:31:19.987: IPSOCK_RAS_sendto: msg length 122 from 172.16.13.35:1719
to 172.16.13.14: 1719
```

```
Mar 4 15:31:19.987: RASLib::RASSendLRQ: LRQ (seq# 2048) sent to 172.16.13.14
```

```
Mar 4 15:31:19.987: RAS OUTGOING PDU ::=
```

```
value RasMessage ::= requestInProgress :
```

```
!--- RIP message is sent back to OGW
```

```

{
  requestSeqNum 1109

  delay 9000
}

```

```
Mar 4 15:31:19.991: RAS OUTGOING ENCODE BUFFER ::= 80 05000454 2327
```

```
Mar 4 15:31:19.991:
```

```
Mar 4 15:31:19.991: IPSOCK_RAS_sendto: msg length 7 from 172.16.13.35:1719 to
172.16.13.15: 57076
```

```
Mar 4 15:31:19.991: RASLib::RASSendRIP: RIP (seq# 1109) sent to 172.16.13.15
```

4. When OGK2 receives the LRQ, it checks the IZCT, and from the configuration it finds that the LRQ should also contain an IZCT. OGK2 then creates a new IZCT by changing the izctSrcZone and izctDstZone to ogk2 and forwards the LRQ to TGK2. After sending out the LRQ to TGK2, it sends back a RIP message to OGK1.

If the gatekeepers are part of a cluster, the cluster name will be used for the SrcZone or DstZone.

```
Mar 4 15:31:20.051: RAS OUTGOING PDU ::=
```

```
value RasMessage ::= requestInProgress :
```

```
!--- RIP message sent back to ogk1
```

```

{
  requestSeqNum 2048
  delay 6000
}

```

```
Mar 4 15:31:20.055: RAS OUTGOING ENCODE BUFFER ::= 80 050007FF 176F
```

```
Mar 4 15:31:20.055:
```

```
Mar 4 15:31:20.055: IPSOCK_RAS_sendto: msg length 7 from 172.16.13.14:1719 to
172.16.13.35: 1719
```

```
Mar 4 15:31:20.059: RASLib::RASSendRIP: RIP (seq# 2048) sent to 172.16.13.35
```

```
Mar 4 15:31:20.059: H225 NONSTD OUTGOING PDU ::=
```

```

value LRQnonStandardInfo ::=
{
  ttl 5
  nonstd-callIdentifier
  {
    guid '221B686C17CC11CC8011A049E0524766'H
  }
  callingOctet3a 128
  gatewaySrcInfo
  {
    e164 : "4085272923",
    h323-ID : {"ogw"}
  }
}

```

```

Mar 4 15:31:20.063: H225 NONSTD OUTGOING ENCODE BUFFER::= 82 06B01100
221B686C 17CC11CC 8011A049 E0524766 01801002 048073B8 5A5C5640 02006F00 670077
Mar 4 15:31:20.072:
Mar 4 15:31:20.072: PDU ::=

```

```

value IZCToken ::=
{
  izctInterZoneType intraDomainCisco : NULL

```

*!--- Still intraDomain since message OGK1 is not a foreign domain*

```

  izctSrcZone "ogk2"

```

*!--- SrcZone and DstZone become ogk2*

```

  izctDstZone "ogk2"
}

```

```

Mar 4 15:31:20.076: ENCODE BUFFER::= 47 00C06F67 6B32066F 676B3204 73726304
64737404 696E74
Mar 4 15:31:20.080:
Mar 4 15:31:20.080: RAS OUTGOING PDU ::=

```

```

value RasMessage ::= locationRequest :

```

*!--- The LRQ is forwarded to TGK2*

```

{
  requestSeqNum 2048
  destinationInfo
  {
    e164 : "3653"
  }
  nonStandardData
  {
    nonStandardIdentifier h221NonStandard :
    {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
    }
    data '8206B01100221B686C17CC11CC8011A049E05247...'H
  }
  replyAddress ipAddress :
  {

```

```

    ip 'AC100D23'H
    port 1719
  }
  sourceInfo
  {
    h323-ID : {"ogk1"}
  }
  canMapAlias TRUE
  tokens

```

*!--- IZCT is included*

```

{
  {
    tokenOID { 1 2 840 113548 10 1 0 }
    nonStandard
    {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '4700C06F676B32066F676B320473726304647374...'H
    }
  }
}

```

```

Mar 4 15:31:20.104: RAS OUTGOING ENCODE BUFFER ::= 4A 8007FF01
01806986 40B50000 12288206 B0110022 1B686C17 CC11CC80 11A049E0 52476601
80100204 8073B85A 5C564002 006F0067 007700AC 100D2306 B70BA00B 01400300
6F006700 6B003101 80300100 80092A86 4886F70C 0A010009 2A864886 F70C0A01
00184700 C06F676B 32066F67 6B320473 72630464 73740469 6E74

```

```
Mar 4 15:31:20.120:
```

```
Mar 4 15:31:20.120: IPSOCK_RAS_sendto: msg length 127 from 172.16.13.14:1719
to 172.16.13.16: 1719
```

```
Mar 4 15:31:20.124: RASLib::RASSendLRQ: LRQ (seq# 2048) sent to 172.16.13.16
```

5. TGK2 determines that the LRQ came from a foreign domain. It updates the IZCT's dstZone with its own ID and interZoneType as INTER\_DOMAIN\_CISCO. It then creates a new CAT and passes the updated IZCT and the LRQ to TGK1.

TGK2 treats the zone from which an LRQ is received as a foreign-domain zone in either of the following two scenarios:

- o The TGK2's remote zone list does not contain the zone from which an LRQ is received.
- o The TGK2's remote zone list contains the zone from which an LRQ is received and the zone is marked with a foreign-domain flag.

It would then send a RIP message back to OGK1.

```
Mar 4 15:31:20.286: RAS OUTGOING PDU ::=
value RasMessage ::= requestInProgress :
```

*!--- RIP message sent back to OGK1 since lrq-forward queries are configured on OGK2 and TGK2*

```

{
  requestSeqNum 2048
  delay 6000
}

```

```
Mar 4 15:31:20.286: RAS OUTGOING ENCODE BUFFER ::= 80 050007FF 176F
```

```
Mar 4 15:31:20.286:
```

```
Mar 4 15:31:20.286: IPSOCK_RAS_sendto: msg length 7 from 172.16.13.16:1719 to
172.16.13.35: 1719
```

```
Mar 4 15:31:20.286: RASLib::RASSendRIP: RIP (seq# 2048) sent to 172.16.13.35
```

```
Mar 4 15:31:20.286: H225 NONSTD OUTGOING PDU ::=
```

```

value LRQnonStandardInfo ::=
{
  ttl 4
  nonstd-callIdentifier
  {
    guid '221B686C17CC11CC8011A049E0524766'H
  }
  callingOctet3a 128
  gatewaySrcInfo
  {
    e164 : "4085272923",
    h323-ID : {"ogw"}
  }
}

```

```

Mar 4 15:31:20.290: H225 NONSTD OUTGOING ENCODE BUFFER::= 81 86B01100
221B686C 17CC11CC 8011A049 E0524766 01801002 048073B8 5A5C5640 02006F00 670077
Mar 4 15:31:20.290:
Mar 4 15:31:20.290: PDU ::=
value IZCToken ::=

```

*!--- The IZCT information*

```

{
  izctInterZoneType interDomainCisco : NULL

```

*!--- The zone type is interDomain since the OGK2 in a foreign domain as configured in TGK2*

```

  izctSrcZone "ogk2"

```

*!--- SrcZone is still ogk2*

```

  izctDstZone "tgk2"

```

*!--- DstZone changed to tgk2*

```

}

```

```

Mar 4 15:31:20.294: ENCODE BUFFER::= 47 20C06F67 6B320674 676B3204
73726304 64737404 696E74
Mar 4 15:31:20.294:
Mar 4 15:31:20.294: RAS OUTGOING PDU ::=
value RasMessage ::= locationRequest :

```

*!--- LRQ is sent to TGK1*

```

{
  requestSeqNum 2048
  destinationInfo
  {
    e164 : "3653"
  }
  nonStandardData
  {
    nonStandardIdentifier h221NonStandard :
    {
      t35CountryCode 181
      t35Extension 0
    }
  }
}

```

```

    manufacturerCode 18
  }
  data '8186B01100221B686C17CC11CC8011A049E05247...'H
}
replyAddress ipAddress :
{
  ip 'AC100D23'H
  port 1719
}
sourceInfo
{
  h323-ID : {"ogk1"}
}
canMapAlias TRUE
tokens

```

*!--- The IZCT is included*

```

{
  {
    tokenOID { 1 2 840 113548 10 1 0 }
    nonStandard
    {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '4720C06F676B320674676B320473726304647374...'H
    }
  }
}
}

```

```

Mar 4 15:31:20.302: RAS OUTGOING ENCODE BUFFER ::= 4A 8007FF01 01806986
40B50000 12288186 B0110022 1B686C17 CC11CC80 11A049E0 52476601 80100204
8073B85A 5C564002 006F0067 007700AC 100D2306 B70BA00B 01400300 6F006700
6B003101 80300100 80092A86 4886F70C 0A010009 2A864886 F70C0A01 00184720
C06F676B 32067467 6B320473 72630464 73740469 6E74

```

```
Mar 4 15:31:20.306:
```

```
Mar 4 15:31:20.306: IPSOCK_RAS_sendto: msg length 127 from 172.16.13.16:1719
to 172.16.13.41: 1719
```

```
Mar 4 15:31:20.306: RASLib::RASSendLRQ: LRQ (seq# 2048) sent to 172.16.13.41
```

6. Normally TGK1 updates the IZCT's dstCarrierID to Carrier E, which is determined by the routing process. However, since no carriers are used you will not see that. TGK1 generates a hash token with the IZCT's password, and sends an LCF with the updated IZCT in it to OGK1. This izctHash is used to authenticate the answerCall ARQ that TGK1 receives from the TGW when the later receives the VoIP setup message from OGW.

```
Mar 4 15:31:20.351: PDU ::=
value IZCToken ::=
```

*!--- IZCT with a hash is generated to be sent back to TGK2*

```

{
  izctInterZoneType interDomainCisco : NULL
  izctSrcZone "ogk2"
  izctDstZone "tgk2"
  izctTimestamp 731259080
  izctRandom 3
  izctHash '5A7D5E18AA658A6A4B4709BA5ABEF2B9'H
}

```

```
Mar 4 15:31:20.355: ENCODE BUFFER ::= 7F 20C06F67 6B320674 676B32C0 2B9620C7
0103105A 7D5E18AA 658A6A4B 4709BA5A BEF2B904 73726304 64737404 696E74
```

```

Mar 4 15:31:20.355:
Mar 4 15:31:20.355: RAS OUTGOING PDU ::=
value RasMessage ::= locationConfirm :

```

*!--- LCF is sent back to OGK1 since lrq-forward queries are configured on OGK2 and TGK2*

```

{
requestSeqNum 2048
callSignalAddress ipAddress :
{
  ip 'AC100D17'H
  port 1720
}
rasAddress ipAddress :
{
  ip 'AC100D17'H
  port 55762
}
nonStandardData
{
  nonStandardIdentifier h221NonStandard :
  {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
  }
  data '000140020074006700770600740067006B003101...'H
}
destinationType
{
  gateway
  {
    protocol
    {
      voice :
      {
        supportedPrefixes
        {
        }
      }
    }
  }
  mc FALSE
  undefinedNode FALSE
}
tokens

```

*!--- The IZCT is included*

```

{
{
tokenOID { 1 2 840 113548 10 1 0 }
nonStandard
{
  nonStandardIdentifier { 1 2 840 113548 10 1 0 }
  data '7F20C06F676B320674676B32C02B9620C7010310...'H
}
}
}

```



}

```
Mar 4 15:31:20.367: RAS OUTGOING ENCODE BUFFER ::= 4F 07FF00AC
100D1706 B800AC10 0D17D9D2 40B50000 122F0001 40020074 00670077 06007400
67006B00 31011001 40020074 00670077 00AC100D 1706B800 00000000 00000000
00104808 0880013C 05010000 48010080 092A8648 86F70C0A 0100092A 864886F7
0C0A0100 307F20C0 6F676B32 0674676B 32C02B96 20C70103 105A7D5E 18AA658A
6A4B4709 BA5ABEF2 B9047372 63046473 7404696E 74
```

```
Mar 4 15:31:20.371:
```

```
Mar 4 15:31:20.371: IPSOCK_RAS_sendto: msg length 154 from 172.16.13.41:1719 to
172.16.13.35: 1719
```

```
Mar 4 15:31:20.371: RASLib::RASSendLCF: LCF (seq# 2048) sent to 172.16.13.35
```

7. OGK1 extracts the IZCT from the LCF and sends it in an ACF to the OGW.

```
Mar 4 15:31:20.316: PDU ::=
```

```
value IZCToken ::=
```

```
!--- The extracted IZCT
```

```
{
  izctInterZoneType interDomainCisco : NULL
  izctSrcZone "ogk2"
  izctDstZone "tgk2"
  izctTimestamp 731259080
  izctRandom 3
  izctHash '5A7D5E18AA658A6A4B4709BA5ABEF2B9'H
}
```

```
Mar 4 15:31:20.324: ENCODE BUFFER ::= 7F 20C06F67 6B320674 676B32C0 2B9620C7 0103105A
7D5E18AA 658A6A4B 4709BA5A BEF2B904 73726304 64737404 696E74
```

```
Mar 4 15:31:20.328:
```

```
Mar 4 15:31:20.332: RAS OUTGOING PDU ::=
```

```
value RasMessage ::= admissionConfirm :
```

```
!--- ACF is sent back to OGW with the hashed IZCToken
```

```
{
  requestSeqNum 1109
  bandwidth 640
  callModel direct : NULL
  destCallSignalAddress ipAddress :
  {
    ip 'AC100D17'H
    port 1720
  }
  irrFrequency 240
  tokens
```

```
!--- The IZCT included
```

```
{
  {
    tokenOID { 1 2 840 113548 10 1 0 }
    nonStandard
    {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '7F20C06F676B320674676B32C02B9620C7010310...'H
    }
  }
}
willRespondToIRR FALSE
uuiesRequested
```

```

    {
      setup FALSE
      callProceeding FALSE
      connect FALSE
      alerting FALSE
      information FALSE
      releaseComplete FALSE
      facility FALSE
      progress FALSE
      empty FALSE
    }
  }
}

```

```

Mar 4 15:31:20.352: RAS OUTGOING ENCODE BUFFER ::= 2B 00045440 028000AC 100D1706
B800EF1A 08C04801 0080092A 864886F7 0C0A0100 092A8648 86F70C0A 0100307F 20C06F67
6B320674 676B32C0 2B9620C7 0103105A 7D5E18AA 658A6A4B 4709BA5A BEF2B904 73726304
64737404 696E7401 00020000

```

```

Mar 4 15:31:20.364:

```

```

Mar 4 15:31:20.364: IPSOCK_RAS_sendto: msg length 97 from 172.16.13.35:1719 to
172.16.13.15: 57076

```

```

Mar 4 15:31:20.368: RASLib::RASSendACF: ACF (seq# 1109) sent to 172.16.13.15

```

#### 8. The OGW sends the IZCT to the TGW in the H.225 SETUP message.

```

Mar 4 15:31:20.529: H225.0 OUTGOING PDU ::=
value H323_UserInformation ::=

```

```

{
  h323-uu-pdu
  {
    h323-message-body setup :

```

*!--- H.225 SETUP message sent to TGW*

```

{
  protocolIdentifier { 0 0 8 2250 0 2 }
  sourceAddress
  {
    h323-ID : {"ogw"}
  }
  sourceInfo
  {
    gateway
    {
      protocol
      {
        voice :
        {
          supportedPrefixes
          {
            {
              prefix e164 : "1#"
            }
          }
        }
      }
    }
  }
  mc FALSE
  undefinedNode FALSE
}
activeMC FALSE
conferenceID '221B686C17CC11CC8010A049E0524766'H

```

```

conferenceGoal create : NULL
callType pointToPoint : NULL
sourceCallSignalAddress ipAddress :
{
  ip 'AC100D0F'H
  port 11003
}
callIdentifier
{
  guid '221B686C17CC11CC8011A049E0524766'H
}
tokens

```

*!--- The hashed IZCT info is included in the setup message*

```

{
  {
    tokenOID { 1 2 840 113548 10 1 0 }
    nonStandard
    {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '7F20C06F676B320674676B32C02B9620C7010310...'H
    }
  }
}
fastStart
{
  '0000000C6013800A04000100AC100D0F4125'H,
  '400000060401004C6013801114000100AC100D0F...'H
}
mediaWaitForConnect FALSE
canOverlapSend FALSE
}
h245Tunneling TRUE
nonStandardControl
{
  {
    nonStandardIdentifier h221NonStandard :
    {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
    }
    data '6001020001041F04038090A31803A983816C0C21...'H
  }
}
}
}

```

9. The TGW passes the IZCT to the TGK1 in an ARQ answerCall.

```

Mar 4 15:31:20.613:
Mar 4 15:31:20.613: RAS OUTGOING PDU ::=
value RasMessage ::= admissionRequest :

```

*!--- ARQ answerCall type is sent to TGK1*

```

{
requestSeqNum 78
callType pointToPoint : NULL
callModel direct : NULL
endpointIdentifier {"617D829000000001"}

```

```

destinationInfo
{
  e164 : "3653"
}
srcInfo
{
  e164 : "4085272923",
  h323-ID : {"ogw"}
}
srcCallSignalAddress ipAddress :
{
  ip 'AC100D0F'H
  port 11003
}
bandWidth 1280
callReferenceValue 3
nonStandardData
{
  nonStandardIdentifier h221NonStandard :
  {
    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
  }
  data '80000008800180'H
}
conferenceID '221B686C17CC11CC8010A049E0524766'H
activeMC FALSE
answerCall TRUE
canMapAlias TRUE
callIdentifier
{
  guid '221B686C17CC11CC8011A049E0524766'H
}
tokens

```

*!--- The hashed IZCToken info is included*

```

{
  {
    tokenOID { 1 2 840 113548 10 1 0 }
    nonStandard
    {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '7F20C06F676B320674676B32C02B9620C7010310...'H
    }
  }
}
willSupplyUUIEs FALSE
}

```

10. TGK1 authenticates the destination IZCT successfully, because TGK1 generated the hash in the IZCT and it sends back an ACF to the TGW.

```

Mar 4 15:31:20.635:
Mar 4 15:31:20.635: PDU ::=
value IZCToken ::=

```

*!--- The extracted IZCT from the ARQ to be validated*

```

{
  izctInterZoneType interDomainCisco : NULL
}

```

```

    izctSrcZone "ogk2"
    izctDstZone "tgk2"
    izctTimestamp 731259080
    izctRandom 3
    izctHash '5A7D5E18AA658A6A4B4709BA5ABEF2B9'H
}

```

```

Mar 4 15:31:20.639: RAS OUTGOING PDU ::=
value RasMessage ::= admissionConfirm :

```

*!--- After the IZCT is validated, ACF is sent back to TGW*

```

{
  requestSeqNum 78
  bandwidth 1280
  callModel direct : NULL
  destCallSignalAddress ipAddress :
  {
    ip 'AC100D17'H
    port 1720
  }
  irrFrequency 240
  willRespondToIRR FALSE
  uuiesRequested
  {
    setup FALSE
    callProceeding FALSE
    connect FALSE
    alerting FALSE
    information FALSE
    releaseComplete FALSE
    facility FALSE
    progress FALSE
    empty FALSE
  }
}

```

11. The TGW establishes the call toward Carrier D after receiving the ACF.

### **Example 2: Call failed because TGW was not able to extract the IZCT from the received setup message**

The following example is based on the same topology and configuration as Example 1. In this example, the software of the TGW was changed to a version where the IZCT is not supported. In such a case, the TGW is not able to extract the IZCT from the setup message, which causes the TGK1 to reject the call with a disconnect reason of security denial.

This example shows only show the setup message, ARQ, and the ARJ on the TGW since the call flow is the same as Example 1.

```

Mar 4 19:50:32.346: PDU DATA = 6147C2BC
value H323_UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body setup :

```

*!--- H.225 SETUP message is received with a token included*

```

{
  protocolIdentifier { 0 0 8 2250 0 2 }
  sourceAddress
  {
    h323-ID : {"ogw"}
  }
  sourceInfo

```

```

{
  gateway
  {
    protocol
    {
      voice :
      {
        supportedPrefixes
        {
          {
            prefix e164 : "1#"
          }
        }
      }
    }
  }
  mc FALSE
  undefinedNode FALSE
}
activeMC FALSE
conferenceID '56CA67C817F011CC8014A049E0524766'H
conferenceGoal create : NULL
callType pointToPoint : NULL
sourceCallSignalAddress ipAddress :
{
  ip 'AC100D0F'H
  port 11004
}
callIdentifier
{
  guid '56CA67C817F011CC8015A049E0524766'H
}

```

**tokens**

*!--- Hashed IZCT is included*

```

{
  {
    tokenOID { 1 2 840 113548 10 1 0 }
    nonStandard
    {
      nonStandardIdentifier { 1 2 840 113548 10 1 0 }
      data '7F20C06F676B320674676B32C02B965D85010410...'H
    }
  }
}
fastStart
{
  '0000000C6013800A04000100AC100D0F45D9'H,
  '400000060401004C6013801114000100AC100D0F...'H
}
mediaWaitForConnect FALSE
canOverlapSend FALSE
}
h245Tunneling TRUE
nonStandardControl
{
  {
    nonStandardIdentifier h221NonStandard :
    {

```

```

    t35CountryCode 181
    t35Extension 0
    manufacturerCode 18
  }
  data '6001020001041F04038090A31803A983816C0C21...'H
}
}
}
}
}

```

```

RAW_BUFFER::=
60 01020001 041F0403 8090A318 03A98381 6C0C2180 34303835 32373239 32337005
80333635 33
Mar 4 19:50:32.362: PDU DATA = 6147F378
value H323_UU_NonStdInfo ::=
{
  version 2
  protoParam qsigNonStdInfo :
  {
    iei 4
    rawMesg '04038090A31803A983816C0C2180343038353237...'H
  }
}

```

```

PDU DATA = 6147F378
value ARQnonStandardInfo ::=
{
  sourceAlias
  {
  }
  sourceExtAlias
  {
  }
  callingOctet3a 128
}

```

```

RAW_BUFFER::=
80 00000880 0180
Mar 4 19:50:32.366: PDU DATA = 6147C2BC
value RasMessage ::= admissionRequest :

```

*!--- ARQ is sent out, in it there is no token*

```

{
  requestSeqNum 23
  callType pointToPoint : NULL
  callModel direct : NULL
  endpointIdentifier {"617D829000000001"}
  destinationInfo
  {
    e164 : "3653"
  }
  srcInfo
  {
    e164 : "4085272923"
  }
  srcCallSignalAddress ipAddress :
  {

```

```

    ip 'AC100D0F'H
    port 11004
  }
  bandwidth 640
  callReferenceValue 1
  nonStandardData
  {
    nonStandardIdentifier h221NonStandard :
    {
      t35CountryCode 181
      t35Extension 0
      manufacturerCode 18
    }
    data '80000008800180'H
  }
  conferenceID '56CA67C817F011CC8014A049E0524766'H
  activeMC FALSE
  answerCall TRUE
  canMapAlias FALSE
  callIdentifier
  {
    guid '56CA67C817F011CC8015A049E0524766'H
  }
  willSupplyUUIEs FALSE
}

```

```
RAW_BUFFER::=
```

```

27 98001600 F0003600 31003700 44003800 32003900 30003000 30003000 30003000
30003000 31010180 69860104 8073B85A 5C5600AC 100D0F2A FC400280 000140B5
00001207 80000008 80018056 CA67C817 F011CC80 14A049E0 52476644 E0200100
110056CA 67C817F0 11CC8015 A049E052 47660100

```

```
Mar 4 19:50:32.374: h323chan_dgram_send:Sent UDP msg. Bytes sent: 117 to
172.16.13.41:1719
```

```
Mar 4 19:50:32.374: RASLib::GW_RASSendARQ: ARQ (seq# 23) sent to 172.16.13.41
```

```
Mar 4 19:50:32.378: h323chan_dgram_rcvdata:rcvd from [172.16.13.41:1719]
on sock[1]
```

```
RAW_BUFFER::=
```

```
2C 00168001 00
```

```
Mar 4 19:50:32.378: PDU DATA = 6147C2BC
```

```
value RasMessage ::= admissionReject :
```

```
!--- ARJ is received with a reason of security denial
```

```

{
  requestSeqNum 23
  rejectReason securityDenial : NULL
}

```


```
Mar 4 19:50:32.378: ARJ (seq# 23) rcvd
```

---

## Related Information

- [Inter-Domain Gatekeeper Security Enhancement](#)
- [Cisco IOS Debug Command Reference, Release 12.3](#)
- [Voice Technologies](#)
- [Voice, Telephony & Messaging Devices](#)
- [Voice Software](#)



- [Voice, Telephony and Messaging TAC eLearning Solutions](#)
- **Recommended Reading: Troubleshooting Cisco IP Telephony** [Troubleshooting Cisco IP Telephony](#)  , Cisco Press, ISBN 1587050757
- [Technical Support - Cisco Systems](#)

---

<a href="#">Home</a>	<a href="#">How to Buy</a>	<a href="#">Login</a>	<a href="#">Profile</a>	<a href="#">Feedback</a>	<a href="#">Site Map</a>	<a href="#">Help</a>
----------------------	----------------------------	-----------------------	-------------------------	--------------------------	--------------------------	----------------------

All contents are Copyright © 1992-2003 Cisco Systems, Inc. All rights reserved. [Important Notices](#) and [Privacy Statement](#).