Cisco Unified Border Element (CUBE)

Call recording solution Application Note

**Created By:**

**Pashmeen Mistry**

**Technical Marketing Engineer, Cisco Systems**

**TABLE OF CONTENTS**

[Introduction](#_Introduction) 3

[Prerequisites](#_Prerequisites_2) 4

Conventions 4

[Topology Used for this Application Note](#_Topology_Used_for): 5

[Configuration related to the CUBE Recording Feature](#_Configuration_related_to): 5

[Complete configuration of CUBE as used in the above topology:](#_Complete_configuration_of) 7

[Call Flows Supported:](#_Call_Flows_Supported:) 10

[High-Availability Options of Call Recording Solution:](#_High-Availability_Options_of) 11

[Verify:](#_Verify:) 17

[Troubleshoot:](#_Troubleshoot_2) 15

# Introduction

The Cisco Unified Border Element (CUBE) can be used in a network-based recording solution to fork active SIP-SIP calls traversing through it.

The media forking on CUBE supports Cisco’s Open Recording Architecture (ORA). CUBE at the network layer would act as a Recording Client and Cisco MediaSense will act as a Recording Server at the Services Layer.

The functionalities of the Recording Client (CUBE in this case) are:

* To set up a separate SIP dialog with Cisco MediaSense
* To fork the active calls traversing through it, and to act as the source of the recorded media, sending it to the Recording Server (Cisco MediaSense).
* During the initial SIP dialog setup, CUBE sends information to the server, which would help the server associate the call with the media streams and identify the participants of the call. This information is called “metadata”
* This recording can then be consumed via any 3rd party partner application.



**Figure 1. MediaSense Architecture**

This document outlines how to configure the Cisco Unified Border Element for this feature. This capability is supported starting Cisco IOS release 15.2(1)T and later.

# Prerequisites

Ensure that you meet these requirements before you attempt this configuration:

* Basic knowledge of how to configure and use Cisco IOS® voice
* Basic knowledge of how to configure and use CUBE

The basic requirements for setting up CUBE ISR G2 Media-forking feature include:

* ISR G2 equipped with the UC Technology Package, configured as a CUBE in Flow-through mode and with Cisco IOS release 15.2(1)T or later
* SIP-SIP call flows
* All ISR-G2 platforms are supported (2901, 2911, 2921, 2951, 3945, 3945E)
* Cisco MediaSense requirements:
	+ Cisco MediaSense 8.5.3 or later
	+ Cisco UCS server with directly attached disk drives up to 4TB capacity
		- C-series server, with directly attached hard disk drives
		- B-series with fiber-attached SAN drives
		- UCS Express (SRE910 with 8GB RAM, MediaSense 8.5.4)

VMWare ESXi version 5.0 (4.0 or 4.1 supported for backwards compatibility) with LRO disabled. (Note: Hypervisor is required. Cisco MediaSense is not designed to run on bare metal hardware)

* + CUCM 8.5.1 or later (for AXL authentication only)

 For more information see:

Cisco MediaSense SRND:

[http://www.cisco.com/en/US/docs/voice\_ip\_comm/cust\_contact/contact\_center/mediasense/853/srnd/cms853srnd.pdf](http%253A%252F%252Fwww.cisco.com%252Fen%252FUS%252Fdocs%252Fvoice_ip_comm%252Fcust_contact%252Fcontact_center%252Fmediasense%252F853%252Fsrnd%252Fcms853srnd.pdf)

Cisco Developer Network for MediaSense API:

[http://developer.cisco.com/web/mediasense](http%25253A%25252F%25252Fdeveloper.cisco.com%25252Fweb%25252Fmediasense)

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

### Conventions

Refer to the [Cisco Technical Tips Conventions](http%25253A%25252F%25252Fwww.cisco.com%25252Fen%25252FUS%25252Ftech%25252Ftk801%25252Ftk36%25252Ftechnologies_tech_note09186a0080121ac5.shtml) for more information on document conventions.

# Topology Used for this Application Note:



Note: CUBE and MediaSense can be on the same or different subnet. Only requirement is to have IP connectivity between the two.

# Configuration related to the CUBE Recording Feature:

There are 2 methods to configure CUBE for this feature:

**Method 1:**

**CUBE (config)# media profile recorder 100001**

**CUBE (config-profile)# media-recording 20**

**CUBE (config)# media class 32**

**CUBE (config-media)# recorder profile 10000**

**1** The media profile recorder tag stores the dial-peer tag that is used to point to the recording server. In this example, dial-peer 20 will be used to point to the recording server.

**2** The media class tag will then be used to store the recorder profile tag created above.

**CUBE (config)#dial-peer voice 20 voip3**

**CUBE (config-dial-peer)#description Dial-peer pointing to the recording server**

**CUBE (config-dial-peer)#destination-pattern 999994**

**CUBE (config-dial-peer)#session protocol sipv2**

**CUBE (config-dial-peer)#session target ipv4:10.194.118.33**

**CUBE (config-dial-peer)#session transport tcp**

**CUBE (config)#dial-peer voice 1 voip5**

**CUBE (config-dial-peer)#description Dial-peer for the call that needs to be recorded**

**CUBE (config-dial-peer)#media-class 3**

**3** This dial-peer will be used to set up the SIP dialog with the recording server. The IP address used as the session target is the IP address of the recording server.

**4** This can be any dummy destination-pattern. It is not used to match this dial-peer.

**5** The media class tag will then be applied under the dial-peer used for the call that needs to be recorded.

Alternatively, you can use Method 2 to configure CUBE for media forking.

**Method 2:**

**CUBE (config)# media class 36**

**CUBE (config-media)# recorder parameter**

**CUBE (config-media-recorder)# media-recording 20**

**CUBE (config)#dial-peer voice 20 voip**

**CUBE (config-dial-peer)#description Dial-peer pointing to the recording server**

**CUBE (config-dial-peer)#destination-pattern 99999**

**CUBE (config-dial-peer)#session protocol sipv2**

**CUBE (config-dial-peer)#session target ipv4:10.194.118.33**

**CUBE (config)#dial-peer voice 1 voip7**

**CUBE (config-dial-peer)#description Dial-peer for the call that needs to be recorded**

**CUBE (config-dial-peer)#media-class 3**

**6** The media class tag will store the dial-peer tag that is used to point to the recording server under “recorder parameter” CLI

**7** This media class tag will then be applied under the dial-peer used for the call that needs to be recorded.

# Complete configuration of CUBE as used in the above topology:

Note: The configuration related to this feature is highlighted in yellow.

! Last configuration change at 23:38:31 UTC Fri Mar 11 2011

version 15.2

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname CUBE

!

boot-start-marker

boot system flash1:c3900-universalk9-mz.SSA.152-0.0.2.PIA16

boot-end-marker

!

!

logging buffered 10000000

no logging console

!

no aaa new-model

!

no ipv6 cef

ip cef

!

!

!

!

!

multilink bundle-name authenticated

!

!

!

!

!

crypto pki token default removal timeout 0

!

!

voice-card 0

!

!

!

voice service voip

 address-hiding

 mode border-element

 allow-connections sip to sip

 fax protocol t38 version 0 ls-redundancy 0 hs-redundancy 0 fallback none

 sip

!

!

!

!

!

!

**media profile recorder 10000**

 **media-recording 20**

**!**

**media class 3**

 **recorder profile 10000**

!

license udi pid C3900-SPE150/K9 sn FOC14173A1D

license boot module c3900 technology-package uck9

hw-module pvdm 0/0

!

!

!

!

redundancy

!

!

!

!

!

!

interface Embedded-Service-Engine0/0

 no ip address

 shutdown

!

interface GigabitEthernet0/0

 ip address 2.1.1.1 255.255.255.0

 duplex auto

 speed auto

!

interface GigabitEthernet0/1

 ip address 1.1.1.1 255.255.255.0

 duplex auto

 speed auto

!

interface GigabitEthernet0/2

 ip address 172.19.153.38 255.255.255.0

 duplex auto

 speed auto

!

ip forward-protocol nd

!

no ip http server

no ip http secure-server

!

ip route 0.0.0.0 0.0.0.0 172.19.153.1

ip route 10.1.0.0 255.255.0.0 GigabitEthernet0/1

ip route 20.5.0.0 255.255.0.0 GigabitEthernet0/0

!

!

nls resp-timeout 1

cpd cr-id 1

!

!

control-plane

!

!

!

!

mgcp profile default

!

!

dial-peer voice 2 voip

 description Inbound Dial-peer for the call that needs to be recorded

 destination-pattern 100.

 session protocol sipv2

 session target ipv4:20.5.1.1

 codec g711ulaw

 no vad

!

**dial-peer voice 1 voip**

 **description Outbound Dial-peer for the call that needs to be recorded**

 **destination-pattern 200.**

 **session protocol sipv2**

 **session target ipv4:10.1.210.1**

 **media-class 3**

 **codec g711ulaw**

 **no vad**

!

**dial-peer voice 20 voip**

 **description Dial-peer pointing to recording server**

 **destination-pattern 99999**

 **session protocol sipv2**

 **session target ipv4:10.194.118.33**

!

!

!

!

gatekeeper

 shutdown

!

!

!

line con 0

line aux 0

line 2

 no activation-character

 no exec

 transport preferred none

 transport input all

 transport output pad telnet rlogin lapb-ta mop udptn v120 ssh

 stopbits 1

line vty 0 4

 login

 transport input all

!

scheduler allocate 20000 1000

end

**Limitations:**

* CUBE can fork any negotiated/supported audio codec towards the recording server. Cisco MediaSense 8.5.4 supports G711ulaw, G711alaw, G729r8, G729br8, and G.722.
* Currently, the codec used for the recording session should be the same as the negotiated codec of the active call that traverses through CUBE.

# Call Flows Supported:

* Basic Calls (Early Offer-Early Offer, Delay Offer – Early Offer, Delay Offer – Delay Offer)
* Mid-call RE-INVITE/UPDATEs received on the primary active call with media change, will result in update indication to the Recording server with a RE-INVITE
* Hold/Resume on primary call will result in Hold/Resume in the forked call
* For sRTP-RTP calls, media recording can be configured only under the dial-peer corresponding to the RTP leg.
* For IPv4-IPv6 calls, media recording can be configured only under the dial-peer corresponding to IPv4 addresses.
* Low density transcoded primary calls are supported
* High Density transcoded primary calls are not supported
* For fax calls (T.38 and Fax passthru) – RE-INVTE will be sent to Recording server to stop recording when the call switches from voice to fax and a new RE-INVITE will be sent to start the recording when the call switches from fax to voice call. No recording will take place during Fax transmission through CUBE.
* DTMF forking is also supported. More specifically, RTP-NTE to RTP-NTE, In-band to In-band, and In-band to RTP-NTE DTMF interworking is supported. Out-of-band to Out-of-band DTMF digits do not get forked.
* If CUBE is in flow-around or SDP passthru mode, media forking is not supported
* For video calls, only audio portion of the media will get forked.
* Dynamic payload type interworking for audio codec and DTMF will be supported
* Out of Dialog OPTIONS ping message will be supported. Forking call leg will also use this to check the health of the recording server.

# High-Availability for Call Recording Solution:

For certain business critical enterprises, its important to achieve High-Availability of this call recording solution. The following mechanisms are in place to achieve this:

* Out of dialog OPTIONS ping feature should be configured under the dial-peer pointing to the recording server. If the recording server is down, the dial-peer will be marked as DOWN and the next dial-peer in an UP state will be used. This dial-peer can be used to point to a different recording server. Thus ensuring High Availability of this solution.

In the below configuration, 2 media-recording tags (20 & 21) are applied under the recorder profile 10000. The same media-recording tags are used to create 2 dial-peers pointing to two separate recording servers with different preference. OPTIONS message is configured under both these dial-peers. If the recording server fails to respond to this keep-alive, that dial-peer will be marked down and the next dial-peer will be used.

**Configuration Snippet for this option:**

media profile recorder 10000

 media-recording 20 21

!

media class 3

 recorder profile 10000

!

!

!

dial-peer voice 1 voip

 description Outbound Dial-peer for the call that needs to be recorded

 destination-pattern 200.

 session protocol sipv2

 session target ipv4:10.1.210.1

 media-class 3

 codec g711ulaw

 no vad

!

dial-peer voice 20 voip

 description Dial-peer pointing to recording server

 destination-pattern 99999

 session protocol sipv2

 preference 1

 session target ipv4:10.194.118.33

 voice-class sip options-keepalive

dial-peer voice 21 voip

 description Dial-peer pointing to back-up recording server

 destination-pattern 99999

 session protocol sipv2

 preference 2

 session target ipv4:10.195.119.34

 voice-class sip options-keepalive

* Media forking can also be configured on both the call legs of the CUBE. Thus 2-recorded streams will be sent to 2 different recording servers for the same active call. This is achieved by configuring the “media-class” command on both the inbound and outbound dial-peer for the call that needs to be recorded.

This will ensure CUBE will fork a total of 4 RTP streams for the same active call. Thus creating 2 exact copies of the same call.



**NOTE:** Please keep in mind that the number of active calls that goes through CUBE will be reduced. Please contact your Cisco Sales Representative to know the performance numbers of CUBE for this solution.

In the below example, 2 separate media class profiles are created – media class 3 and media class 4, which are applied to the incoming & outgoing call leg of CUBE respectively to create 2 duplicate streams for the same call.

**Configuration Snippet for this option:**

media profile recorder 10000

 media-recording 20

!

media class 3

 recorder profile 10000

!

media profile recorder 20000

 media-recording 21

!

media class 4

 recorder profile 20000

dial-peer voice 2 voip

 description Inbound dial-peer for the call that needs to be recorded

 destination-pattern 100.

 session protocol sipv2

 session target ipv4:20.5.1.1

 media-class 3

 codec g711ulaw

 no vad

!

dial-peer voice 1 voip

 description Outbound Dial-peer for the call that needs to be recorded

 destination-pattern 200.

 session protocol sipv2

 session target ipv4:10.1.210.1

 media-class 4

 codec g711ulaw

 no vad

!

dial-peer voice 20 voip

 description Dial-peer pointing to recording server

 destination-pattern 99999

 session protocol sipv2

 session target ipv4:10.194.118.33

dial-peer voice 21 voip

 description Second Dial-peer pointing to recording server

 destination-pattern 88888

 session protocol sipv2

 session target ipv4:10.195.119.34

* Load balancing of MediaSense servers can also be achieved using the same mechanism. For that, a list of dial-peers needs to be configured with the same destination-pattern, one for each MediaSense server. CUBE will then randomly load balance media forking across these MediaSense servers. For that it’s important the dial-peers have the same preference.

**Configuration Snippet for this option:**

media profile recorder 10000

 media-recording 20 21

!

media class 3

 recorder profile 10000

!

!

!

dial-peer voice 1 voip

 description Outbound Dial-peer for the call that needs to be recorded

 destination-pattern 200.

 session protocol sipv2

 session target ipv4:10.1.210.1

 media-class 3

 codec g711ulaw

 no vad

!

dial-peer voice 20 voip

 description Dial-peer pointing to recording server to MediaSense server 1

 destination-pattern 99999

 session protocol sipv2

 session target ipv4:10.194.118.33

 preference 1

 voice-class sip options-keepalive

dial-peer voice 21 voip

 description Dial-peer pointing to recording server to MediaSense server 2

 destination-pattern 99999

 session protocol sipv2

 session target ipv4:10.195.119.34

 preference 1

 voice-class sip options-keepalive

# Recording Solution for TDM Trunks using CUBE:

The following section will provide detailed guidelines on how to leverage this recording solution for calls on TDM trunks.

CUBE, being an integrated platform can provide TDM trunk connectivity and act as a session border controller at the same time.



For this solution to work, calls from the PSTN are looped back to itself thus creating an inbound VoIP SIP leg to the CUBE. It then sends the call to the call agent in the enterprise network creating an outbound VoIP SIP leg. Thus, the gateway is used to terminate the TDM leg, and originate an IP leg towards the call agent.

**NOTE:** Please keep in mind that the number of active calls that goes through CUBE will be reduced. Please contact your Cisco Sales Representative to know the performance numbers of CUBE for this solution.

**Configuration Snippet for this solution:**

!

voice translation-rule 1
 rule 1 /\(.\*\)/ /888\1/
Voice translation-rule 2
 Rule 2 /\^888\(.\*\)/ /\1/
voice translation-profile prefix
 translate called 1
Voice translation-profile strip
 Translate called 2
!
Interface Loopback0
 Ip address 1.1.1.1 255.255.255.255

media profile recorder 10000
 media-recording 20
!
media class 3
 recorder profile 10000
!
Dial-peer voice 1 pots
 Incoming called-number 200.

 Translation-profile incoming prefix
 Port 0/0/0:23
!
Dial-peer voice 2 voip
 Description “To loop incoming PSTN calls back to itself”
 Destination-pattern 888200.
 Session protocol sipv2
 Session target ipv4:1.1.1.1   <<< IP address of the Router

 Codec g711ulaw
 No vad

Dial-peer voice 3 voip
 Description Incoming DP for the looped call
 Incoming called-number 888200.

 Translation-profile incoming strip
 Session protocol sipv2
 Codec g711ulaw
 No vad

Dial-peer voice 4 voip
 Description outgoing DP for the looped call to PBX
 Destination-pattern 200.
 Session protocol sipv2
 Session target ipv4:10.1.210.1
 Codec g711ulaw
 Media-class 3
 No vad

Dial-peer voice 5 voip
 Description “Incoming DP for the outgoing PSTN calls”
 Translation-profile incoming prefix

 incoming called-number 100.
 Session protocol sipv2
 Codec g711ulaw
 No vad

Dial-peer voice 6 voip
 Description “To loop outgoing PSTN calls”
 Destination-pattern 888100.
 Session protocol sipv2
 Session target ipv4:1.1.1.1   <<< IP address of the Router

 Media-class 3
 Codec g711ulaw
 No vad

Dial-peer voice 7 voip
 Description “Incoming DP for the looped call”

 translation-profile incoming strip
 incoming called-number 888100.
 Session protocol sipv2
 Codec g711ulaw
 No vad

Dial-peer voice 8 pots
 Destination-pattern 100.
 Port 0/0/0:23

dial-peer voice 20 voip
 description “For recording leg to MediaSense server”
 destination-pattern 99999
 session protocol sipv2
 preference 1
 session target ipv4:10.194.118.33

In the above example, the call flow is as follows:

For Incoming PSTN call:

1. Calls will match inbound POTS dial-peer 1. The translation-profile “prefix” applied under dial-peer 1 will change the called number to 888200x.
2. Due to the prefixed called number, it will then match outbound VoIP dial-peer 2
3. Since the session target on the dial-peer 2 is pointing back to the same router, the call will loop and match inbound VoIP dial-peer 3 and outbound VoIP dial-peer 4.
4. Since both inbound & outbound dial-peers are VoIP, the router acts as a CUBE. The media class configuration applied under dial-peer 4 triggers this call leg to be forked.
5. The translation profile applied to dial-peer 3, will strip the prefixed digits 888 and thus matching dial-peer 4 based on the original called number.

For Outbound PSTN calls:

1. Calls will match inbound VoIP dial-peer 5. This dial-peer will prefix 888 to the called number.
2. Due to the prefixed called number, the call will match outbound VoIP dial-peer 6. This dial-peer has been configured for recording using the media class configuration. Hence this call leg will get recorded. This dial-peer also loops back the call to itself.
3. The looped call will now match incoming VoIP dial-peer 7. This dial-peer will strip the digits we prefixed above.
4. The call will now match to outbound POTS dial-peer 8 sending the call to the PSTN.

# Verify:

Use the CLI below to verify the Media Forking configuration is correct and working.

**CUBE#show voip rtp connections**

VoIP RTP active connections :

No. CallId dstCallId LocalRTP RmtRTP LocalIP RemoteIP

1 36 37 18358 19362 2.1.1.1 20.5.1.1

2 37 36 17294 17690 1.1.1.1 10.1.210.1

3 39 38 19812 42196 172.19.153.38 10.194.118.33

4 40 38 24230 60234 172.19.153.38 10.194.118.33

Found 4 active RTP connections

In the above example, the call between the 2 phones has resulted into 2 RTP streams (1 and 2). The 2 RTP streams 3 & 4 are the recorded streams being sent to the recording server (10.194.118.33 in this example). The call recording server will be muxing these 2 RTP streams into 1 stream which will represent the recorded call.

**CUBE#sh voip recmsp session**

RECMSP active sessions:

**MSP Call-ID** AnchorLeg Call-ID ForkedLeg Call-ID

**143**  141 145

Found 1 active sessions

**CUBE#show voip recmsp session detail call-id <the value specified in the above o/p>**

**CUBE#show voip recmsp session detail call-id 143**

RECMSP active sessions:

Detailed Information

=========================

Recording MSP Leg Details:

Call ID: 143

GUID : 7C5946D38ECD

AnchorLeg Details:

Call ID: 141

Forking Stream type: voice-nearend

Participant: 2001

Non-anchor Leg Details:

Call ID: 140

Forking Stream type: voice-farend

Participant: 1001

Forked Leg Details:

Call ID: 145

Near End Stream CallID 145

Stream State ACTIVE

Far End stream CallID 146

Stream State ACTIVE

Found 1 active sessions

Stream State---This will show the state of the call – can be either in ACTIVE or HOLD state

Anchor Leg Call-id: This is the call-id of the anchor leg (Dial-peer where forking is enabled) which in also internal to the system. The output in brief describes the participant number and stream type as voice-near end, which is called party side.

Non-Anchor Call-id: This is the call-id of non-anchor leg (Dial-peer where forking is not enabled).

Forked Call-id: This forking leg call-id will show near-end and far-end stream call-id details with state of the Stream.

**CUBE#show voip rtp forking**

VoIP RTP active forks:

 Fork 1

 stream type voice-only (0): count 0

 stream type voice+dtmf (1): count 0

 stream type dtmf-only (2): count 0

 stream type voice-nearend (3): count 1

 **remote ip 10.194.118.33, remote port 38526**, local port 18648

 **codec g711ulaw**, logical ssrc 0x53

 **packets sent 29687**, packets received 0

 stream type voice+dtmf-nearend (4): count 0

 stream type voice-farend (5): count 1

 **remote ip 10.194.118.33, remote port 50482**, local port 17780

 **codec g711ulaw**, logical ssrc 0x55

 **packets sent 29686**, packets received 0

 stream type voice+dtmf-farend (6): count 0

 stream type video (7): count

Remote ip/ Port is the recording server ip and port address. Codec will indicate which codec is negotiated for recording call leg. Packets sent indicate the number of packets sent to recording server from each stream.

# Troubleshoot

This section provides information you can use to troubleshoot your configuration. **Note:**Refer to [Important Information on Debug Commands](http%25253A%25252F%25252Fwww.cisco.com%25252Fen%25252FUS%25252Ftech%25252Ftk801%25252Ftk379%25252Ftechnologies_tech_note09186a008017874c.shtml) before you use **debug** commands.

 Debug ccsip messages

 Debug voip recmsp all

 Debug voip ccapi all

 Debug voip application all

Note: Do not turn on a large number of debugs on a system carrying a high volume of active call traffic.

# Additional Resources

You can refer to the following links for more information about this feature:

**Cisco MediaSense documentation**

* Developer Guide (available at [http://developer.cisco.com/web/mediasense](http%253A%252F%252Fdeveloper.cisco.com%252Fweb%252Fmediasense))
* Sample client code (available at [http://developer.cisco.com/web/mediasense](http%253A%252F%252Fdeveloper.cisco.com%252Fweb%252Fmediasense))
* Solution Reference Network Design (SRND)  http://www.cisco.com/en/US/docs/voice\_ip\_comm/cust\_contact/contact\_center/mediasense/853/srnd/cms853srnd.pdf
* Installation and Adminstration Guide [http://www.cisco.com/en/US/docs/voice\_ip\_comm/cust\_contact/contact\_center/mediasense/853/inst\_admin/cms853iag.pdf](http%253A%252F%252Fwww.cisco.com%252Fen%252FUS%252Fdocs%252Fvoice_ip_comm%252Fcust_contact%252Fcontact_center%252Fmediasense%252F853%252Finst_admin%252Fcms853iag.pdf)
* Release Notes  [http://www.cisco.com/en/US/docs/voice\_ip\_comm/cust\_contact/contact\_center/mediasense/853/releasenotes/cms853rn.pdf](http%253A%252F%252Fwww.cisco.com%252Fen%252FUS%252Fdocs%252Fvoice_ip_comm%252Fcust_contact%252Fcontact_center%252Fmediasense%252F853%252Freleasenotes%252Fcms853rn.pdf)
* Troubleshooting tips [http://docwiki.cisco.com/wiki/Troubleshooting\_Tips\_for\_Cisco\_MediaSense\_8.5](http%253A%252F%252Fdocwiki.cisco.com%252Fwiki%252FTroubleshooting_Tips_for_Cisco_MediaSense_8.5)

Draft versions of these documents covering 8.5(4) can be supplied.

Cisco Unified Border Element Product Page – [www.cisco.com/go/cube](http%25253A%25252F%25252Fwww.cisco.com%25252Fgo%25252Fcube)

Network-Based Recording Using Cisco Unified Border Element:

http://www.cisco.com/en/US/docs/ios-xml/ios/voice/cube\_proto/configuration/15-2mt/cube-network-based.html