Mediant™ 1000 Series

Multi-Service Business Router

Session Border Controller

IP Media Reference Guide

Version 6.8

February 2015

Document # LTRT-28642

HD VoIP Sounds Better

AudioCodes

Table of Contents

List of Figures

List of Tables

Reader's Notes

Notice

This document is an IP Media Reference Guide for AudioCodes Mediant 1000 product line.

Information contained in this document is believed to be accurate and reliable at the time of printing. However, due to ongoing product improvements and revisions, AudioCodes cannot guarantee accuracy of printed material after the Date Published nor can it accept responsibility for errors or omissions. Before consulting this document, check the corresponding Release Notes regarding feature preconditions and/or specific support in this release. In cases where there are discrepancies between this document and the Release Notes, the information in the Release Notes supersedes that in this document. Updates to this document and other documents as well as software files can be downloaded by registered customers at http://www.audiocodes.com/downloads.

Date Published: February-04-2015

Trademarks

AudioCodes, AC, AudioCoded, Ardito, CTI2, CTI², CTI Squared, HD VoIP, HD VoIP Sounds Better, InTouch, IPmedia, Mediant, MediaPack, NetCoder, Netrake, Nuera, Open Solutions Network, OSN, Stretto, TrunkPack, VMAS, VoicePacketizer, VoIPerfect, VoIPerfectHD, What's Inside Matters, Your Gateway To VoIP and 3GX are trademarks or registered trademarks of AudioCodes Limited. All other products or trademarks are property of their respective owners. Product specifications are subject to change without notice.

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

Customer Support

Customer technical support and service are generally provided by AudioCodes' Distributors, Partners, and Resellers from whom the product was purchased. For technical support for products purchased directly from AudioCodes, or for customers subscribed to AudioCodes Customer Technical Support (ACTS), contact support@audiocodes.com.

Abbreviations and Terminology

Each abbreviation, unless widely used, is spelled out in full when first used.

Throughout this manual, unless otherwise specified, the term *device* refers to the Mediant 1000B MSBR.

Related Documentation

Manual Name
SIP CPE Release Notes
Mediant 1000B MSBR Hardware Installation Manual
MSBR Series CLI Reference Guide for Data-Router Functionality
MSBR Series CLI Reference Guide for System & VoIP Functionality
SBC Design Guidel
DConvert User's Guide
CPTWizard User's Guide

Notes and Warnings

Note: The scope of this document does not fully cover security aspects for deploying the device in your environment. Security measures should be done in accordance with your organization's security policies. For basic security guidelines, you should refer to AudioCodes *Recommended Security Guidelines* document.

Note: Before configuring the device, ensure that it is installed correctly as instructed in the *Hardware Installation Manual*.

Note: This device is considered an INDOOR unit and must be installed ONLY indoors.

Note: The terms IP-to-Tel and Tel-to-IP refer to the direction of the call relative to the device. IP-to-Tel refers to calls received from the IP network and destined to the PSTN/PBX (i.e., telephone connected directly or indirectly to the device); Tel-to-IP refers to calls received from the PSTN/PBX and destined for the IP network.

Notes:

- FXO (Foreign Exchange Office) is the interface replacing the analog telephone and connects to a Public Switched Telephone Network (PSTN) line from the Central Office (CO) or to a Private Branch Exchange (PBX). The FXO is designed to receive line voltage and ringing current, supplied from the CO or the PBX (just like an analog telephone). An FXO VoIP device interfaces between the CO/PBX line and the Internet.
- FXS (Foreign Exchange Station) is the interface replacing the Exchange (i.e., the CO or the PBX) and connects to analog telephones, dial-up modems, and fax machines. The FXS is designed to supply line voltage and ringing current to these telephone devices. An FXS VoIP device interfaces between the analog telephone devices and the Internet.

Note: OPEN SOURCE SOFTWARE. Portions of the software may be open source software and may be governed by and distributed under open source licenses, such as the terms of the GNU General Public License (GPL), the terms of the Lesser General Public License (LGPL), BSD and LDAP, which terms are located at: http://www.audiocodes.com/support and all are incorporated herein by reference. If any open source software is provided in object code, and its accompanying license requires that it be provided in source code as well, Buyer may receive such source code by contacting AudioCodes, by following the instructions available on AudioCodes website.

Legal Notice:

- By default, the device supports export-grade (40-bit and 56-bit) encryption due to US government restrictions on the export of security technologies. To enable 128-bit and 256-bit encryption on your device, contact your AudioCodes sales representative.
- This device includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).
- This device includes cryptographic software written by Eric Young (eay@cryptsoft.com).

Customer Information

1. This equipment, RMX module of Mediant 1000 MSBR complies with Part 68 of the FCC Rules and the requirements adopted by the ACTA. On the front panel of the module is a label that contains among other information, a product identifier in the format US:6NPDDNANRMX. If requested, this number must be provided to the telephone company.
2. This equipment is designed to be connected to the telephone network using RJ-48C connector, which complies with Part 68 rules and requirements adopted by ACTA. The service order codes (SOC) are 6.0F for digital interface and the Facility interface codes (FIC) are: 04DU9.1SN, 04DU9.1KN, 04DU9.BN, 04DU9.DN.
3. Should the product cause harm to the telephone network, the telephone company will notify you in advance that temporary discontinuance of service may be required. If advance notice is not practical, you will be notified as soon as possible. Also, you will be advised of your right to file a complaint with the FCC if it is necessary.
4. The telephone company may make changes in its facilities, equipment, operations or procedures that could affect the operation of the equipment. If this happens, the telephone company will provide advance notice in order for you to make necessary modifications to maintain uninterrupted service.
5. If trouble is experienced with this equipment, for repair or warranty information please contact AudioCodes Inc., 2099 Gateway Place, Suite 500, San Jose, CA, 95110, phone number (408) 441 1175, URL:www.audiocodes.com. If the equipment is causing harm to the telephone network, the telephone company may request to disconnect the equipment until the problem is resolved.
6. Installation is described in the product's Installation Manual. Connection to telephone company provided coin service is prohibited. Connection to party lines service is subject to state tariffs.

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our Web site at http://www.audiocodes.com/downloads

1 Overview

This document is an IP Media Reference Guide for AudioCodes Mediant 1000 product line. The device conference, transcoding, announcement and media server applications can be used separately, each on a different platform, or all on the same device. The SIP URI name in the INVITE message is used to identify the resource (media server, conference or announcement) to which the SIP session is addressed.

The number of DSP channels that are allocated for IP conferences, transcoding and IP announcements is determined by the parameter MediaChannels. Other DSP channels can be used for PSTN media server.

The device's SIP implementation is based on the decomposition model described in the following IETF Internet-Drafts:

- "A Multi-party Application Framework for SIP" (draft-ietf-sipping-cc-framework-06)
- "Models for Multi Party Conferencing in SIP" (draft-ietf-sipping-conferencing-framework-05)
- "A Framework for Conferencing with the Session Initiation Protocol (SIP)" (RFC 4353)
- "Basic Network Media Services with SIP" (RFC 4240)
- "Media Server Control Markup Language (MSCML) and Protocol" (draft-vandyke-mscml-06)

Note: To use the device's advanced Announcement capabilities, it's essential that the *ini* file parameter AMSProfile be set to 1.

Some of the IP media functionalities are configured in the IP Media settings page.

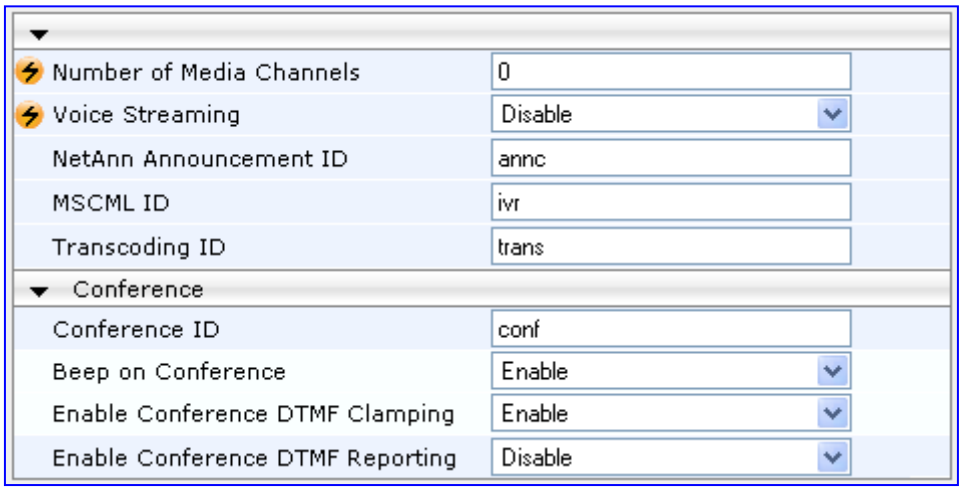
Notes:

- This page appears only if your device is installed with the relevant Software License Key.
- For a detailed description of parameters used in this document, refer to the *User's Manual*.

➢ **To access the IP Media Settings page:**

- Open the IP Media Settings page (**Configuration** tab > **VoIP** menu > **IP Media** submenu > **IP Media Settings**).

Figure 1-1: IP Media Settings Page



2 Transcoding using Third-Party Call Control

The device supports transcoding using a third-party call control Application server. This support is provided by the following:

- Using RFC 4117 (see 'Using RFC 4117' on page 13)
- Using RFC 4240 - NetAnn Conferencing (see Using RFC 4240 - NetAnn 2-Party Conferencing on page 14)

Note: Transcoding can also be implemented using the IP-to-IP application and SBC application.

2.1 Using RFC 4117

The device supports RFC 4117 - Transcoding Services Invocation in the Session Initiation Protocol (SIP) Using Third Party Call Control (3pcc) - providing transcoding services (i.e., acting as a transcoding server). This is used in scenarios where two SIP User Agents (UA) would like to establish a session, but do not have a common coder or media type. When such incompatibilities are found, the UAs need to invoke transcoding services to successfully establish the session. Note that transcoding can also be performed using NetAnn, according to RFC 4240.

To enable the RFC 4117 feature, the parameter EnableRFC4117Transcoding must be set to 1 (and the device must be reset).

The 3pcc call flow is as follows: The device receives from one of the UAs, a single INVITE with an SDP containing two media lines. Each media represents the capabilities of each of the two UAs. The device needs to find a match for both of the media, and opens two channels with two different media ports to the different UAs. The device performs transcoding between the two voice calls.

In the example below, an Application Server sends a special INVITE that consists of two media lines to perform transcoding between G.711 and G.729:

```
m=audio 20000 RTP/AVP 0
c=IN IP4 A.example.com
m=audio 40000 RTP/AVP 18
c=IN IP4 B.example.com
```

2.2 Using RFC 4240 - NetAnn 2-Party Conferencing

Transcoding bridges (or translates) between two remote *network* locations, each of which uses a different coder and/or a different DTMF and fax transport types. The device supports IP-to-IP transcoding. It creates a transcoding call that is similar to a dial-in, two-party conference call. The SIP URI in the INVITE message is used as a transcoding service identifier. The transcoding identifier is configured by the 'Transcoding ID' parameter (TranscodingID), located in the IP Media Settings page (**Configuration** tab > **VoIP** menu > **IP Media** > **IP Media Settings**).

It is assumed that the device is controlled by a third-party, Application server (or any SIP user agent) that instructs the device to start an IP transcoding call by sending two SIP INVITE messages with SIP URI that includes the transcoding identifier name. For example:

```
Invite sip:trans123@audiocodes.com SIP/2.0
```

The left part of the SIP URI includes the transcoding ID ('the default string is 'trans') and is terminated by a unique number (123). The device immediately sends a 200 OK message in response to each INVITE.

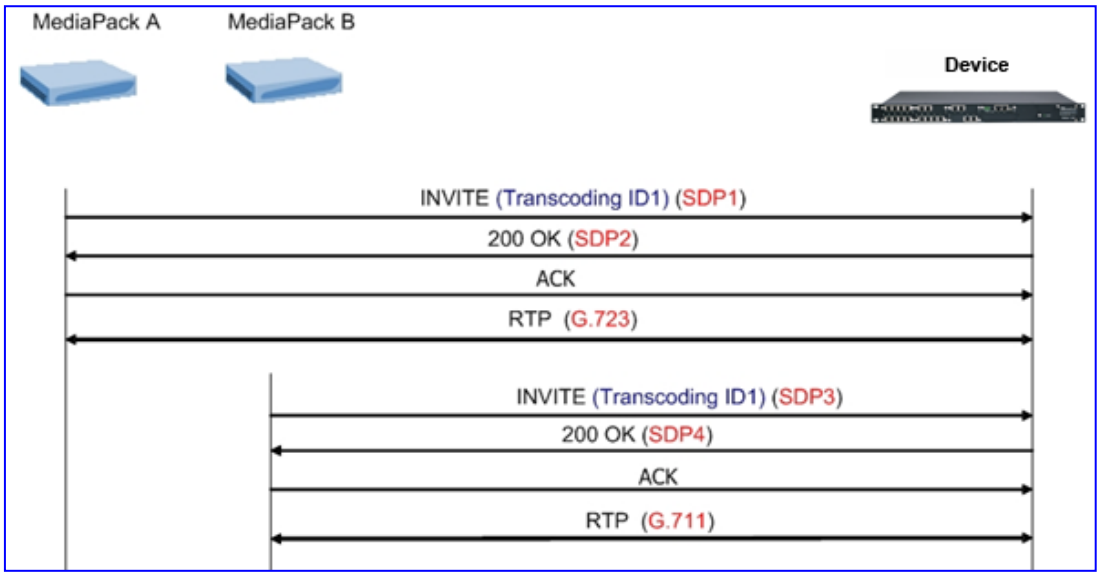
Each of the transcoding SIP call participants can use a different VoIP coder and a different DTMF transport type, negotiated with the device using common SIP negotiation.

Sending a BYE request to the device by any of the participants, terminates the SIP session and removes it from the Transcoding session. The second BYE from the second participant ends the transcoding session and releases its resources.

The device uses two media (DSP) channels for each call, thereby reducing the number of available transcoding sessions to half of the defined value for MediaChannels. To limit the number of resources for transcoding, use the 'Number of Media Channels' parameter (MediaChannels), located in the IP Media Settings page (**Configuration** tab > **VoIP** menu > **IP Media** > **IP Media Settings**). For example, if 'Number of Media Channels' is set to "40", only 20 transcoding sessions are available.

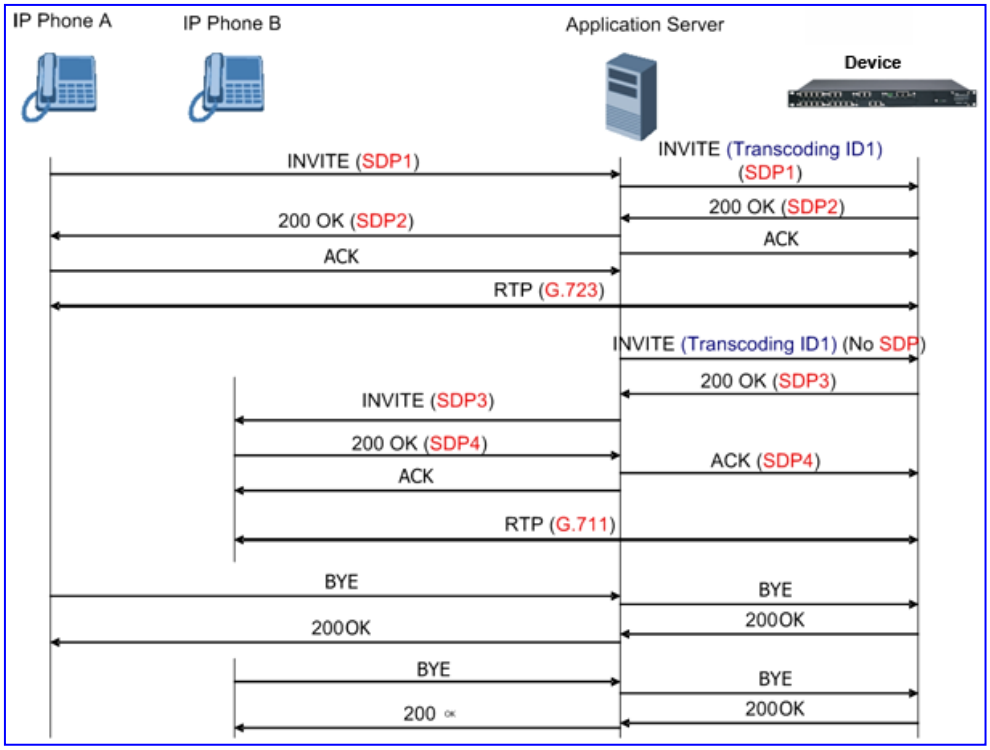
The figure below illustrates an example of a direct connection to a device:

Figure 2-1: Direct Connection (Example)



The figure below illustrates an example of implementing an Application server:

Figure 2-2: Implementing an Application Server (Example)



Reader's Notes

3 Conference Server

The device supports dial-in, multi-party conferencing. In conference applications, the device functions as a centralized conference bridge. In ad-hoc or prearranged conferences, users 'invite' the conference bridge. The conference bridge mixes the media and sends it to all participants.

The device supports the following interfaces for conferencing:

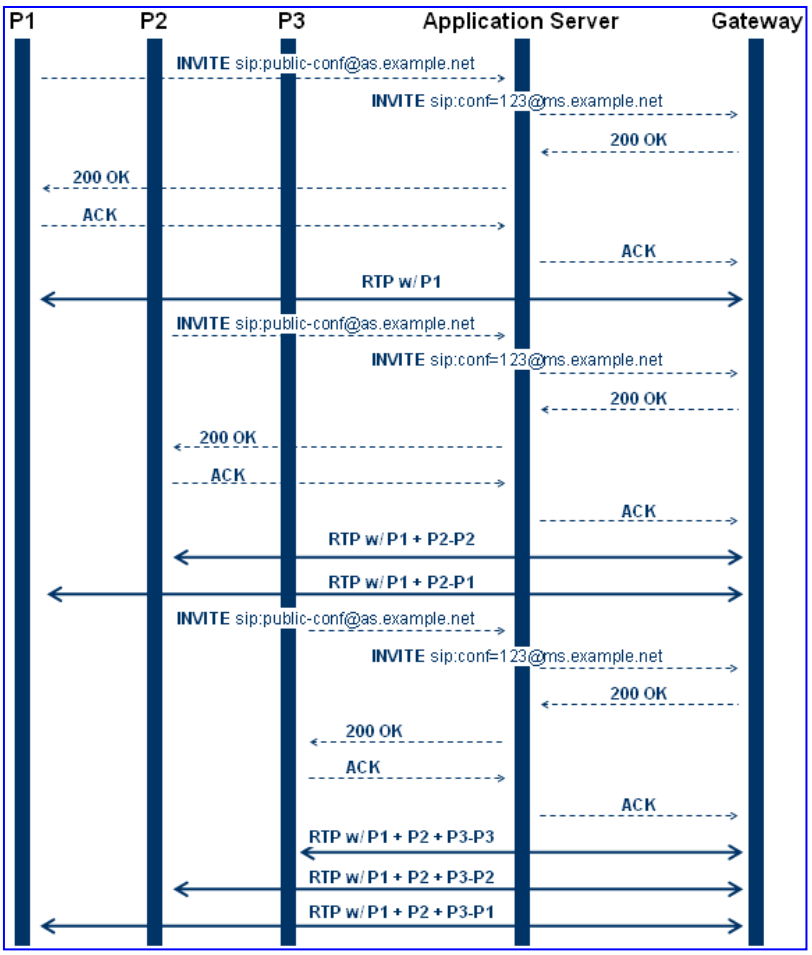
- Simple, according to NetAnn (see 'Simple Conferencing (NetAnn)' on page 18)
- Advanced, according to MSCML (see 'Advanced Conferencing (MSCML)' on page 20)

3.1 Simple Conferencing (NetAnn)

3.1.1 SIP Call Flow

A SIP call flow for simple conferencing is shown below:

Figure 3-1: SIP Call Flow



3.1.2 Creating a Conference

The device creates a conference call when the first user joins the conference. To create a conference, the Application Server sends a regular SIP INVITE message to the device. The User Part of that Request-URI includes both the Conference Service Identifier (indicating that the requested Media Service is a Conference) and a Unique Conference Identifier (identifying a specific instance of a conference).

```
INVITE sip: conf100@audiocodes.com SIP/2.0
```

By default, a request to create a conference reserves three resources on the device. It is possible to reserve a larger number of resources in advance by adding the number of required participants to the User Part of the Request-URI. For example, '6conf100' reserves six resources for the duration of the conference. If the device can allocate the requested number of resources, it responds with a 200 OK.

The Conference Service Identifier can be configured using the 'Conference ID' parameter (ConferenceID), located in the IP Media Settings page (**Configuration** tab > **VoIP** menu > **IP Media** > **IP Media Settings**). By default, it is set to 'conf'.

3.1.3 Joining a Conference

To join an existing conference, the Application Server sends a SIP INVITE message with the same Request-URI as the one that created the conference. Each conference participant can use a different coder negotiated with the device using usual SIP negotiation.

If more than the initially requested number of participants try to join the conference (i.e., four resources were reserved and a fifth INVITE is received) and the device has an available resource, the request is granted.

If an INVITE to join an existing conference is received with a request to reserve a larger number of participants than initially requested, it is granted if the device has available resources. A request for a smaller number of participants is not granted as this may create a situation where existing legs would need to be disconnected.

The maximal number of participants in a single conference is 60. The maximal number of participants that actually participate in the mix at a given time is three (the loudest legs).

The Application Server can place a participant on Hold/Un-hold by sending the appropriate SIP Re-INVITE on that participant dialog.

3.1.4 Terminating a Conference

The device never disconnects an existing conference leg. If a BYE is received on an existing leg, it is disconnected, but the resource is still saved if the same leg (or a different one) wants to re-join the conference. This logic occurs only for the initial number of reserved legs.

For example:

1. INVITE reserves three legs.
2. A, B, and C join the conference.
3. A disconnects.
4. A joins (guaranteed).
5. D joins.
6. A disconnects.
7. A joins (not guaranteed).

Sending a BYE request to the device terminates the participant's SIP session and removes it from the conference. The final BYE from the last participant ends the conference and releases all conference resources.

3.1.5 PSTN Participants

Adding PSTN participants is done by performing a loopback from the IP side (the device's IP address is configured in the Outbound IP Routing Table). If the destination phone number in the incoming call from the PSTN is equal to the Conference Service Identifier and Unique Conference Identifier, the participant joins the conference.

A PSTN participant uses two DSP channels (caused by the IP loopback).

3.2 Advanced Conferencing (MSCML)

3.2.1 Creating a Conference

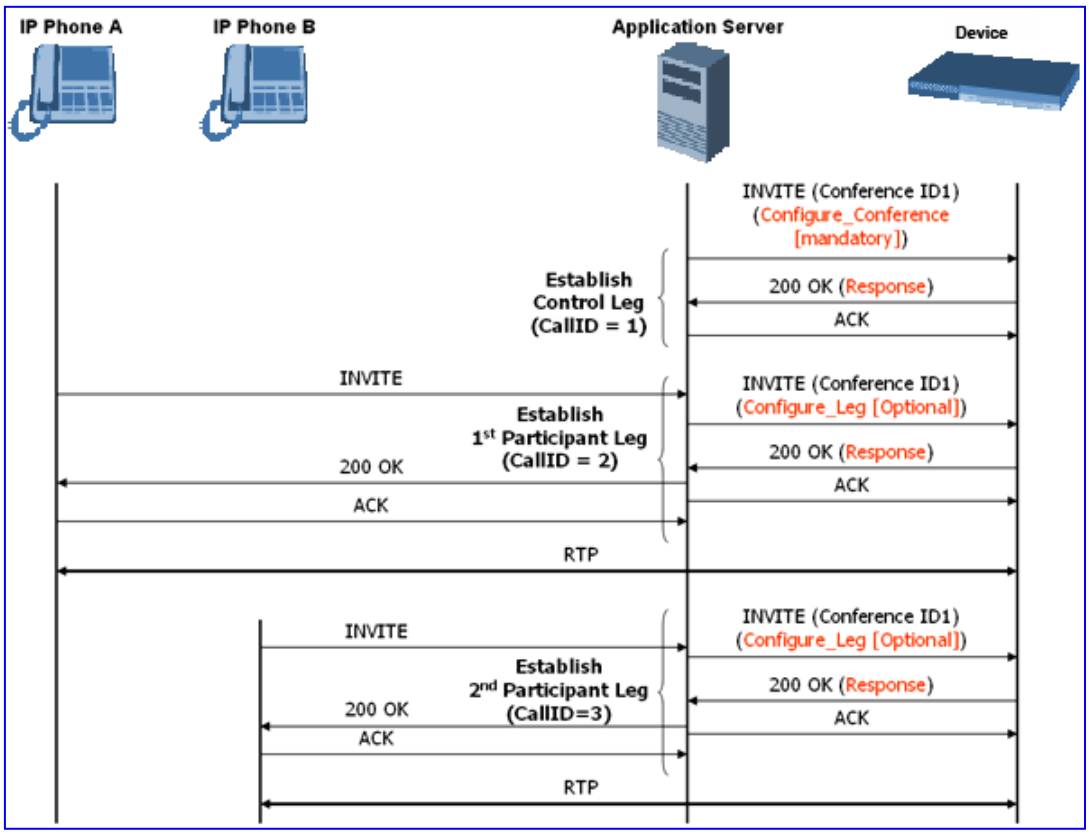
The device creates a conference call when the first INVITE is received from the Application Server (same as NetAnn). The Unique Conference Identifier is used to join participants to the same conference. This first INVITE must include a <configure_conference> MSCML request body. If this body is not included, a simple conference is established. This first leg is the Control Leg, which is different from a regular Participant Leg. The Control Leg is used to perform operations for the whole conference.

The MSCML response to the first INVITE is sent in the 200 OK SIP response. If no error occurs, the response is:
<response request="configure_conference" code="200" text="OK"/>.

The <configure_conference> can include the following attributes:

- **Id:** Identification number of the MSCML request. This is used to correlate between MSCML requests and responses.
- **Reservedtalkers:** Defines the maximum number of talker legs. As the device does not support “listener only” legs, this actually sets the maximum number of participants in the conference. The device reserves this number of participants for the entire duration of the conference. If a participant leg decides to leave the conference by issuing a BYE, the resource is not freed, thereby allowing that same leg (or a new one) to join at any stage.
- **Reserveconfmedia:** Determines if Media Services such as Play or Record can be applied to the conference. If set to Yes, the device reserves the necessary amount of resources to play an announcement to the whole conference or record the whole conference. The Application Server can change the value of reserveconfmedia during an existing conference. By default, reserveconfmedia is set to Yes.

Figure 3-2: Creating a Conference



3.2.2 Joining a Conference

To join an existing conference, the Application Server sends a SIP INVITE message with the same Request-URI as the one that created the conference. The INVITE message may include a <configure_leg> MSCML request body. If not included, defaults are used for that leg attributes.

The <configure_leg> can include the following attributes:

- **Id:** identification number of the MSCML request. This is used to correlate between MSCML requests and responses.
- **Type:** Talker / Listener. If set to Listener, the incoming RTP from that leg does not participate in the conference mix. The default is Talker.
- **Mixmode:**
  - Full: RTP from this leg participates in the mix (default).
  - Mute: RTP from this leg is not participating in the mix.
  - Private: RTP from this leg can only hear participants within a conference team (<teammate>) to which it belongs (see below).

The <configure_team> element enables clients to create personalized mixes for scenarios where the standard mixmode settings do not provide sufficient control. The <configure_team> element is a child of <configure_leg>. The <configure_team> element, containing one or more <teammate> elements, specifies those participants that should be present in this participant's personalized mix. The <configure_team> element supports several commands: set, add, remove, and query.

The participants are identified in the <teammate> elements by their IDs that are assigned in their <configure_leg> element. The team configuration is implicitly symmetric, i.e. if participant A defines participant B as its team member, implicitly participant B defines participant A as its team member.

A "coaching" example:

Table 3-1: MSCML Conferencing with Personalized Mixes

Participant	ID	Team Members	Mixmode	Hears
Supervisor	"supervisor"	Agent	Private	Customer and Agent
Agent	"agent"	Supervisor	Full	Customer and Supervisor
Customer	"customer"		Full	Agent

This scenario is established as follows:

1. Conference is created on the control leg with <configure_conference>.
2. Coach leg joins and issues:

```
<configure_leg id="supervisor" mixmode="private"/>
```

3. Agent leg joins and issues:

```
<configure_leg id="agent">
         <configure_team action="set">
           <teammate id="supervisor"/>
         </configure_team>
       </configure_leg>
```

4. Customer joins and issues:

```
<configure_leg id="customer"/>
```

3.2.3 Modifying a Conference

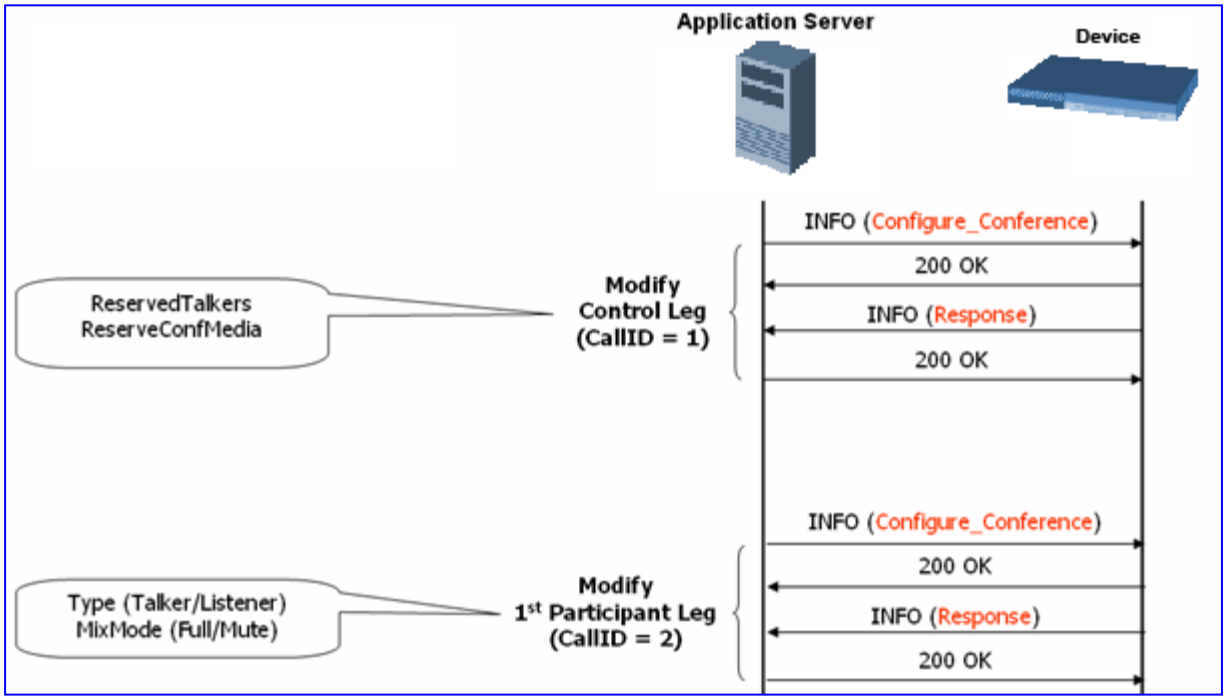
To modify an existing conference, INFO messages are used. Each INFO message carries an MSCML request. The MSCML response is included in an INFO message back from the device to the Application Server. It is possible to modify an entire conference (by issuing requests on the Control Leg) or only a certain participant (by issuing requests on that specific leg).

To modify the entire conference, a <configure_conference> MSCML request body is sent in an INFO message on the Control Leg SIP dialog. Using this request, the Application Server can modify the following attributes:

- **Reservedtalkers:** If the Application Server sets a number that is lower than the initial number requested in the INVITE, then the request is not granted. If the number is higher than the initial number, the device sends a success response in the response INFO.
- **Reserveconfmedia:** If the necessary resources for applying Media Services on the entire conference were reserved in advance, then by setting reserveconfmedia to Yes, it is reserved. If set to No, the device can free the resource.

To modify a certain Participant Leg, a <configure_leg> MSCML request body is sent in an INFO message on that leg SIP dialog. Using this request, the Application Server can modify any of the attributes defined for the <configure_leg> request.

Figure 3-3: Modifying a Conference

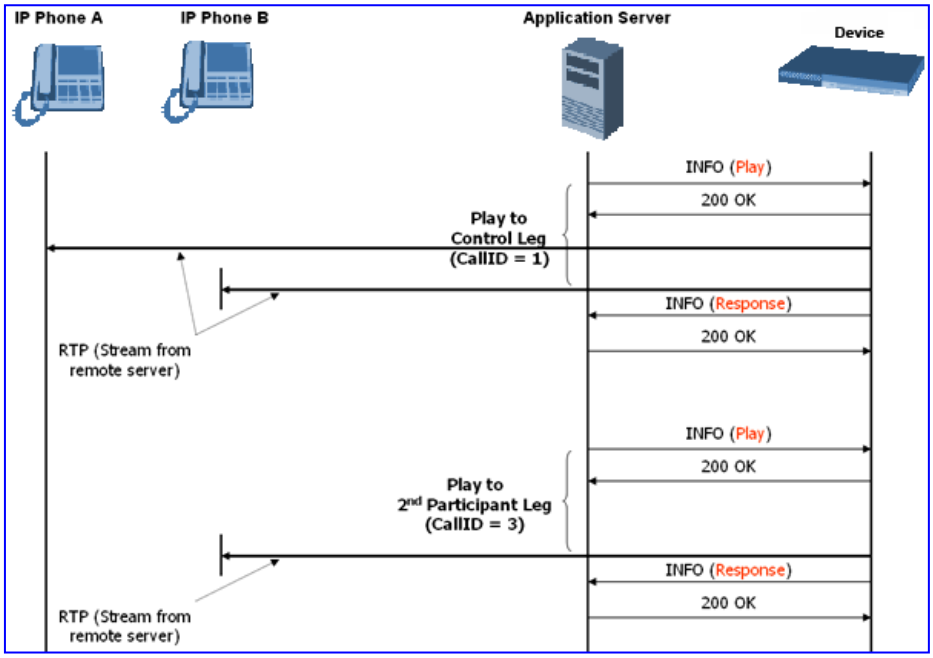


3.2.4 Applying Media Services on a Conference

The Application Server can issue a Media Service request (<play>, <playcollect>, or <playrecord>) on either the Control Leg or a specific Participant Leg. For a Participant Leg, all three requests are applicable. For the Control Leg, the <playcollect> is not applicable as there is no way to collect digits from the whole conference.

When issuing a Media Service on the Control Leg, it affects all Participant Legs in the conference, e.g., play an announcement. When issuing a Media Service on a Participant Leg, it affects the specific leg only.

Figure 3-4: Applying Media Services on a Conference



3.2.5 Active Speaker Notification

After an advanced conference is established, the Application Server can subscribe to the device to receive notifications of the current set of active speakers in a conference at any given moment. This feature is referred to as *Active Speaker Notification* (ASN) and is designed according to the MSCML standard. Notifications provide information on the number of active participants and their details.

The notifications are sent unsolicited at specific intervals requested by the application and only when a change in the number of active conference speakers occurs. If a change in the speakers list occurs, the server issues an INVITE to the specific SIP UA, and then transfers the call to the UA.

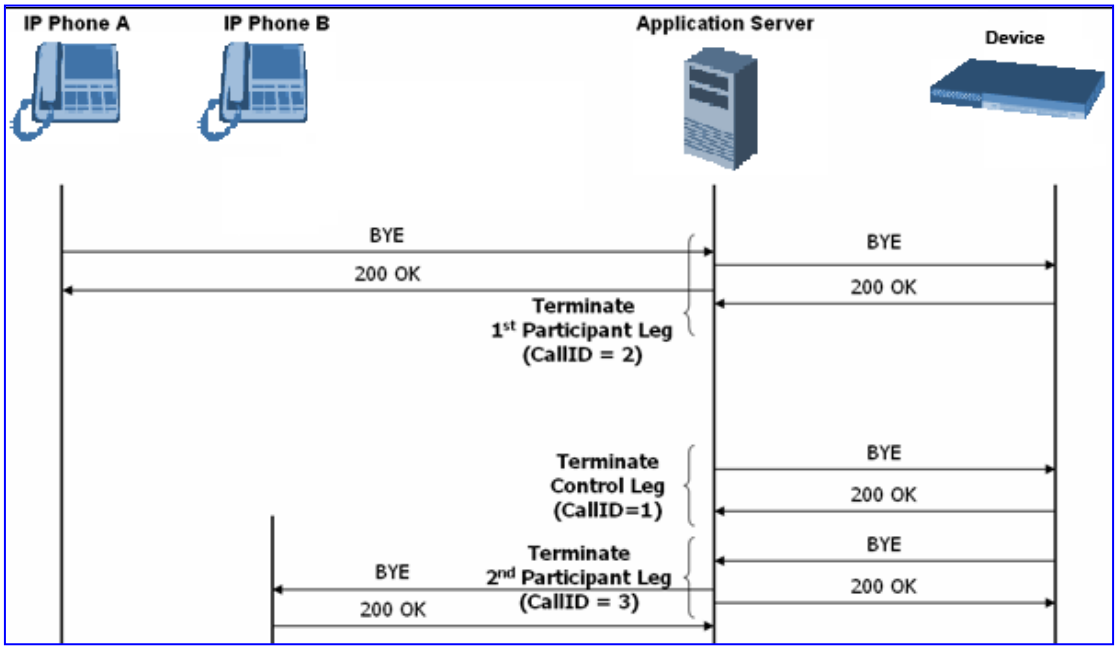
Event notifications are sent in SIP INFO messages, as shown in the example below of XML Response Generated for ASN:

```
<?xml version="1.0" encoding="utf-8"?>
<MediaServerControl version="1.0">
<notification>
<conference uniqueID="3331" numtalkers="1">
<activetalkers>
<talker callID="98142661715120001936l9@10.8.27.118"/>
</activetalkers>
</conference>
</notification>
</MediaServerControl>
```

3.2.6 Terminating a Conference

To remove a leg from a conference, the Application Server issues a SIP BYE request on the selected dialog representing the conference leg. The Application Server can terminate all legs in a conference by issuing a SIP BYE request on the Control Leg. If one or more participants are still in the conference when the device receives a SIP BYE request on the Control Leg, the device issues SIP BYE requests on all of the remaining conference legs to ensure a clean up of the legs.

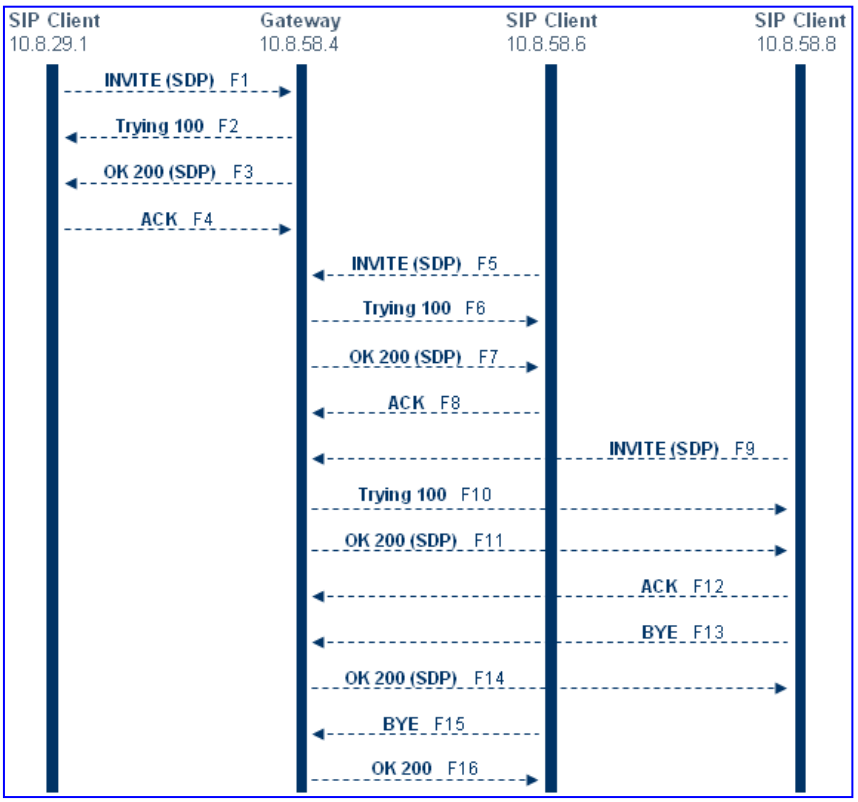
Figure 3-5: Terminating a Conference



3.3 Conference Call Flow Example

The call flow, shown in the following figure, describes SIP messages exchanged between the device (10.8.58.4) and three conference participants (10.8.29.1, 10.8.58.6 and 10.8.58.8).

Figure 3-6: Conference Call Flow Example



1. **SIP MESSAGE 1:** 10.8.29.1:5060 -> 10.8.58.4:5060

```
INVITE sip:conf100@10.8.58.4;user=phone SIP/2.0
Via: SIP/2.0/UDP 10.8.29.1;branch=z9hG4bKacRHmJhMj
Max-Forwards: 70
From: <sip:100@10.8.8.10>;tag=1c352329022
To: <sip:conf100@10.8.58.4;user=phone>
Call-ID: 1792526528qlax@10.8.29.1
CSeq: 1 INVITE
Contact: <sip:100@10.8.29.1>
Supported: em,100rel,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Type: application/sdp
Content-Length: 216
v=0
o=AudiocodesGW 663410 588654 IN IP4 10.8.29.1
s=Phone-Call
c=IN IP4 10.8.29.1
t=0 0
m=audio 6000 RTP/AVP 8 96
a=rtpmap:8 pcma/8000
a=rtpmap:96 telephone-event/8000
```

```
a=fmtp:96 0-15
a=ptime:20
a=sendrecv
```

2. **SIP MESSAGE 2:** 10.8.58.4:5060() -> 10.8.29.1:5060()

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 10.8.29.1;branch=z9hG4bKacRHmJhMj
From: <sip:100@10.8.8.10>;tag=1c352329022
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c222574568
Call-ID: 1792526528qlax@10.8.29.1
CSeq: 1 INVITE
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Length: 0
```

3. **SIP MESSAGE 3:** 10.8.58.4:5060 -> 10.8.29.1:5060

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.8.29.1;branch=z9hG4bKacRHmJhMj
From: <sip:100@10.8.8.10>;tag=1c352329022
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c222574568
Call-ID: 1792526528qlax@10.8.29.1
CSeq: 1 INVITE
Contact: <sip:10.8.58.4>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Type: application/sdp
Content-Length: 216
v=0
o=AudiocodesGW 820775 130089 IN IP4 10.8.58.4
s=Phone-Call
c=IN IP4 10.8.58.4
t=0 0
m=audio 7160 RTP/AVP 8 96
a=rtpmap:8 pcma/8000
a=rtpmap:96 telephone-event/8000
a=fmtp:96 0-15
a=ptime:20
a=sendrecv
```

4. **SIP MESSAGE 4:** 10.8.29.1:5060 -> 10.8.58.4:5060

```
ACK sip:10.8.58.4 SIP/2.0
Via: SIP/2.0/UDP 10.8.29.1;branch=z9hG4bKacbUrWtRo
Max-Forwards: 70
From: <sip:100@10.8.8.10>;tag=1c352329022
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c222574568
Call-ID: 1792526528qlax@10.8.29.1
CSeq: 1 ACK
Contact: <sip:100@10.8.29.1>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Length: 0
```

5. **SIP MESSAGE 5:** 10.8.58.6:5060 -> 10.8.58.4:5060

```
INVITE sip:conf100@10.8.58.4;user=phone SIP/2.0
Via: SIP/2.0/UDP 10.8.58.6;branch=z9hG4bKacfowEuut
Max-Forwards: 70
From: <sip:600@10.8.8.10>;tag=1c201038291
To: <sip:conf100@10.8.58.4;user=phone>
Call-ID: 1008914574iYgW@10.8.58.6
CSeq: 1 INVITE
Contact: <sip:600@10.8.58.6>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.4.60A.005.009
Content-Type: application/sdp
Content-Length: 313
v=0
o=AudiocodesGW 702680 202680 IN IP4 10.8.58.6
s=Phone-Call
c=IN IP4 10.8.58.6
t=0 0
m=audio 6000 RTP/AVP 4 8 0 110 96
a=rtpmap:4 g723/8000
a=fmtp:4 annexa=no
a=rtpmap:8 pcma/8000
a=rtpmap:0 pcmu/8000
a=rtpmap:110 AMR/8000/1
a=rtpmap:96 telephone-event/8000
a=fmtp:96 0-15
a=ptime:30
a=sendrecv
```

6. **SIP MESSAGE 6:** 10.8.58.4:5060 -> 10.8.58.6:5060

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 10.8.58.6;branch=z9hG4bKacfowEuut
From: <sip:600@10.8.8.10>;tag=1c201038291
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c1673415884
Call-ID: 1008914574iYgW@10.8.58.6
CSeq: 1 INVITE
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
```

```
MSBR/v.5.40A.010.006
Content-Length: 0
```

7. **SIP MESSAGE 7:** 10.8.58.4:5060 -> 10.8.58.6:5060

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.8.58.6;branch=z9hG4bKacfowEuut
From: <sip:600@10.8.8.10>;tag=1c201038291
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c1673415884
Call-ID: 1008914574iYgW@10.8.58.6
CSeq: 1 INVITE Contact: <sip:conf100@10.8.58.4>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Type: application/sdp
Content-Length: 236
v=0 o=AudiocodesGW 886442 597756 IN IP4 10.8.58.4
s=Phone-Call
c=IN IP4 10.8.58.4
t=0 0
m=audio 7150 RTP/AVP 4 96
a=rtpmap:4 g723/8000
a=fmtp:4 annexa=no
a=rtpmap:96 telephone-event/8000
a=fmtp:96 0-15
a=ptime:30
a=sendrecv
```

8. **SIP MESSAGE 8:** 10.8.58.6:5060 -> 10.8.58.4:5060

```
ACK sip:conf100@10.8.58.4 SIP/2.0
Via: SIP/2.0/UDP 10.8.58.6;branch=z9hG4bKacRRRZPXN
Max-Forwards: 70
From: <sip:600@10.8.8.10>;tag=1c201038291
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c1673415884
Call-ID: 1008914574iYgW@10.8.58.6
CSeq: 1 ACK
Contact: <sip:600@10.8.58.6>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.4.60A.005.009
Content-Length: 0
```

9. **SIP MESSAGE 9:** 10.8.58.8:5060 -> 10.8.58.4:5060

```
INVITE sip:conf100@10.8.58.4;user=phone SIP/2.0
Via: SIP/2.0/UDP 10.8.58.8;branch=z9hG4bKaczJpxnnv
Max-Forwards: 70
From: <sip:800@10.8.58.8>;tag=1c2419012378
To: <sip:conf100@10.8.58.4;user=phone>
Call-ID: 150852731NDDC@10.8.58.8
CSeq: 1 INVITE
Contact: <sip:800@10.8.58.8>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.4.60A.005.009
Content-Type: application/sdp Content-Length: 236
```

```
v=0
o=AudiocodesGW 558246 666026 IN IP4 10.8.58.8
s=Phone-Call
c=IN IP4 10.8.58.8
t=0 0 m=audio 6000 RTP/AVP 4 96
a=rtpmap:4 g723/8000
a=fmtp:4 annexa=no
a=rtpmap:96 telephone-event/8000
a=fmtp:96 0-15
a=ptime:30
a=sendrecv
```

10. **SIP MESSAGE 10:** 10.8.58.4:5060 -> 10.8.58.8:5060

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 10.8.58.8;branch=z9hG4bKaczJpxnnv
From: <sip:800@10.8.58.8>;tag=1c2419012378
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c3203015250
Call-ID: 150852731NDDC@10.8.58.8
CSeq: 1 INVITE
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Length: 0
```

11. **SIP MESSAGE 11:** 10.8.58.4:5060 -> 10.8.58.8:5060

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.8.58.8;branch=z9hG4bKaczJpxnnv
From: <sip:800@10.8.58.8>;tag=1c2419012378
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c3203015250
Call-ID: 150852731NDDC@10.8.58.8
CSeq: 1 INVITE
Contact: <sip:conf100@10.8.58.4>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Type: application/sdp
Content-Length: 236
v=0
o=AudiocodesGW 385533 708665 IN IP4 10.8.58.4
s=Phone-Call
c=IN IP4 10.8.58.4
t=0 0
m=audio 7140 RTP/AVP 4 96
a=rtpmap:4 g723/8000
a=fmtp:4 annexa=no
a=rtpmap:96 telephone-event/8000
a=fmtp:96 0-15
a=ptime:30
a=sendrecv
```

12. **SIP MESSAGE 12:** 10.8.58.8:5060 -> 10.8.58.4:5060

```
ACK sip:conf100@10.8.58.4 SIP/2.0
Via: SIP/2.0/UDP 10.8.58.8;branch=z9hG4bKacisqqyow
Max-Forwards: 70
From: <sip:800@10.8.58.8>;tag=1c2419012378
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c3203015250
Call-ID: 150852731NDDC@10.8.58.8
CSeq: 1 ACK
Contact: <sip:800@10.8.58.8>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Length: 0
```

13. **SIP MESSAGE 13:** 10.8.58.8:5060 -> 10.8.58.4:5060

```
BYE sip:conf100@10.8.58.4 SIP/2.0
Via: SIP/2.0/UDP 10.8.58.8;branch=z9hG4bKackSIyGww
Max-Forwards: 70
From: <sip:800@10.8.58.8>;tag=1c2419012378
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c3203015250
Call-ID: 150852731NDDC@10.8.58.8
CSeq: 2 BYE
Contact: <sip:800@10.8.58.8>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Length: 0
```

14. **SIP MESSAGE 14:** 10.8.58.4:5060 -> 10.8.58.8:5060

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.8.58.8;branch=z9hG4bKackSIyGww
From: <sip:800@10.8.58.8>;tag=1c2419012378
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c3203015250
Call-ID: 150852731NDDC@10.8.58.8
CSeq: 2 BYE
Contact: <sip:conf100@10.8.58.4>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Length: 0
```

15. **SIP MESSAGE 15:** 10.8.58.6:5060 -> 10.8.58.4:5060

```
BYE sip:conf100@10.8.58.4 SIP/2.0
Via: SIP/2.0/UDP 10.8.58.6;branch=z9hG4bKacQypxnvl
Max-Forwards: 70
From: <sip:600@10.8.8.10>;tag=1c201038291
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c1673415884
Call-ID: 1008914574iYgW@10.8.58.6
CSeq: 2 BYE
Contact: <sip:600@10.8.58.6>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Length: 0
```

16. SIP MESSAGE 16: 10.8.58.4:5060 -> 10.8.58.6:5060

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.8.58.6;branch=z9hG4bKacQypxnvl
From: <sip:600@10.8.8.10>;tag=1c201038291
To: <sip:conf100@10.8.58.4;user=phone>;tag=1c1673415884
Call-ID: 1008914574iYgW@10.8.58.6
CSeq: 2 BYE
Contact: <sip:conf100@10.8.58.4>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40A.010.006
Content-Length: 0
```

Reader's Notes

4 Announcement Server

The device supports playing and recording of announcements (local Voice Prompts or HTTP streaming) and playing of Call Progress Tones over the IP network. Three different methods are available for playing and recording announcements:

- NetAnn for playing a single announcement (see 'NetAnn Interface' on page 33)
- MSCML for playing single or multiple announcements and collecting digits (see 'MSCML Interface' on page 35)

4.1 NetAnn Interface

The device supports playing announcements using NetAnn format (according to RFC 4240).

4.1.1 Playing a Local Voice Prompt

To play a single local Voice Prompt, the Application Server (or any SIP user agent) sends a regular SIP INVITE message with SIP URI that includes the NetAnn Announcement Identifier name. For example:

```
INVITE sip:annc@audiocodes.com; play=file:///12 SIP/2.0
```

The left part of the SIP URI includes the string 'annc'. In the example above, the device starts playing announcement number 12 from the internal Voice Prompts file. The following Voice Prompts file formats are supported:

- file:///
- file:////
- http://<localhost>

The NetAnn Announcement Identifier string is configured by the 'NetAnn Announcement ID' parameter (NetAnnAnncID), located in the IP Media Settings page (**Configuration** tab > **VoIP** menu > **IP Media** > **IP Media Settings**).

Sending a BYE request terminates the SIP session and stops the playing of the announcement. If the played Voice Prompt reaches its end, the device initiates a BYE message to notify the Application Server that the session has ended.

4.1.2 Playing using HTTP/NFS Streaming

To play a single announcement via HTTP or NFS streaming, the Application Server (or any SIP user agent) sends a regular SIP INVITE message with SIP URI that includes the NetAnn Announcement Identifier name. For example:

```
INVITE sip:annc@ac.com;
play=http://server.net/gem/Hello.wav   SIP/2.0
```

The left part of the SIP URI includes the string 'annc' terminated by the IP address of the HTTP server, and the name and path of the file to be played. In the example above, the device starts playing the 'Hello.wav' file that resides in the folder 'server.net/gem'. The NetAnn Announcement Identifier string is configured by the 'NetAnn Announcement ID' parameter (NetAnnAnncID), located in the IP Media Settings page (**Configuration** tab > **VoIP** menu > **IP Media** > **IP Media Settings**).

Sending a BYE request terminates the SIP session and stops the playing of the announcement. If the played announcement reaches its end, the device initiates a BYE message to notify the Application Server that the session is ended.

Notes:

- A 200 OK message is sent only after the HTTP connection is successfully established and the requested file is found. If the file isn't found, a 404 Not Found response is sent.
- To use NFS, the requested file system should be first mounted by using the NFS Servers table.

4.1.3 Supported Attributes

When playing announcements, the following attributes are available:

- **Repeat:** defines the number of times the announcement is repeated. The default is 1. The valid range is 1 to 1000, or -1 (i.e., repeats the message forever).
- **Delay:** defines the delay (in msec) between announcement repetitions. The default is 0. The valid range is 1 to 3,600,000.
- **Duration:** defines the total duration (in msec) the announcement(s) are played. The default is 0 (i.e., no limitation). The valid range is 1 to 3,600,000.

For example:

```
INVITE sip:annc@ac.com;
play=http://server.net/gem/Hello.wav; repeat=5;delay=10000 SIP/2.0
```

4.2 MSCML Interface

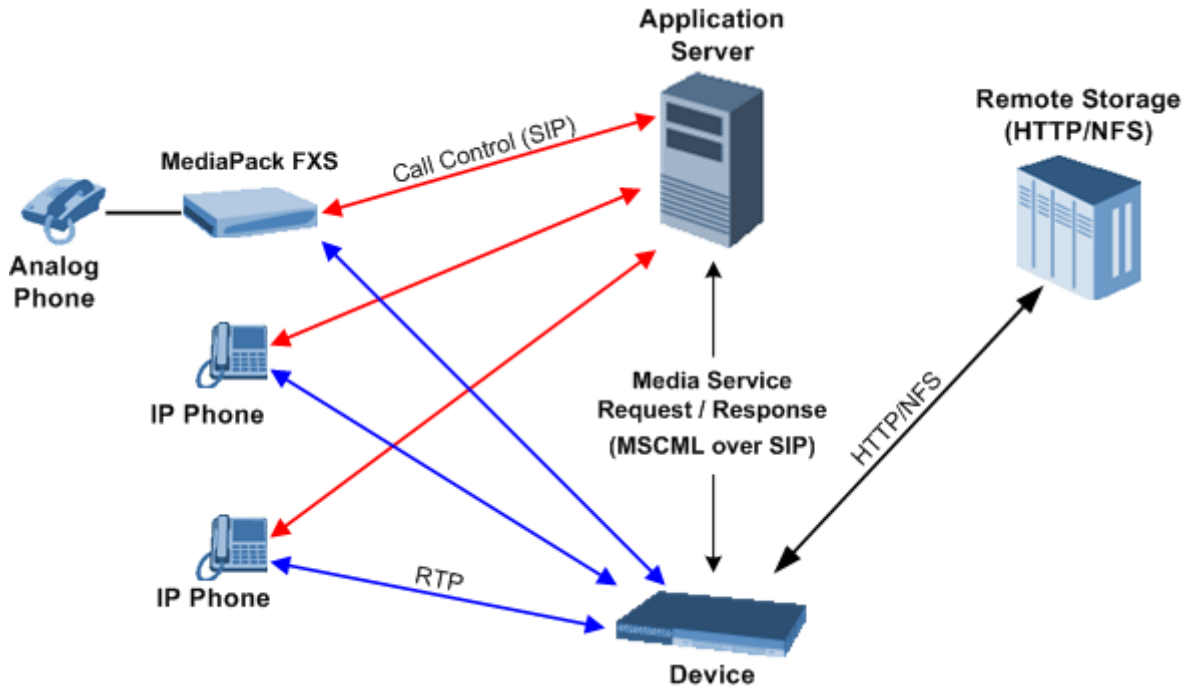
Media Server Control Markup Language (MSCML), according to IETF RFC 5022 is a protocol used in conjunction with SIP to provide advanced announcements handling. MSCML is implemented by adding an XML body to existing SIP INFO messages. Only a single message body (containing a single request or response) is allowed per message.

The device supports all the Interactive Voice Response (IVR) requirements for playing announcements, collecting digits, and recording (Play, PlayCollect, and PlayRecord).

Note: MSCML is only supported on devices operating with 128-MByte RAM size.

The following figure illustrates standard MSCML application architecture:

Figure 4-1: MSCML Interface



The architecture comprises the following components:

- **Device:** Operating independently, the device controls and allocates its processing resources to match each application's requirements. Its primary role is to handle requests from the Application server for playing announcements and collecting digits.
- **Application Server:** An application platform that controls the call signaling. It interfaces with the device using MSCML. It instructs the media server to play announcements, collect digits and record voice streams.
- **Remote Storage:** An HTTP server that contains less-frequently used voice prompts for playback and to which voice stream recording is performed.
- **IP Phones / MediaPack:** Client applications.

4.2.1 Operation

The Application server communicates with the device using MSCML Requests (sent by the Application server), as shown in the example below:

```
<?xml version="1.0" encoding="utf-8"?>
  <MediaServerControl version="1.0">
    <request>
      ... request body ...
    </request>
  </MediaServerControl>
```

The device uses MSCML Responses (i.e., sent by the device) to reply to the Application server, as shown in the example below:

```
<?xml version="1.0" encoding="utf-8"?>
  <MediaServerControl version="1.0">
    <response>
      ... response body ...
    </response>
  </MediaServerControl>
```

To start an MSCML IVR call, the Application server (or any SIP user agent) sends a regular SIP INVITE message with a SIP URI that includes the MSCML Identifier name. For example:

```
INVITE sip:ivr@audiocodes.com SIP/2.0
```

The left part of the SIP URI includes the MSCML Identifier string 'ivr', which is configured by the 'MSCML ID' parameter (MSCMLID), located in the IP Media Settings page (**Configuration** tab > **VoIP** menu > **IP Media** > **IP Media Settings**).

After a call is established, SIP INFO messages are used to carry MSCML requests and responses. An INFO message that carries an MSCML body is identified by its content-type header that is set to 'application/mediaservercontrol+xml'.

Note that IVR requests are not queued. Therefore, if a request is received while another is in progress, the device stops the first operation and executes the new request. The device generates a response message for the first request and returns any data collected up to that point. If an application is required to stop a request in progress, it issues a <Stop> request. This request also causes the device to generate a response message.

The device supports basic IVR functions of playing announcements, collecting DTMF digits, and voice stream recording. These services are implemented using the following Request and Response messages:

- <Play> for playing announcements
- <PlayCollect> for playing announcements and collecting digits
- <PlayRecord> for playing announcements and recording voice
- <Stop> for stopping the playing of an announcement

The device sends a Response to each Request that is issued by the Application server.

The <Play>, <PlayCollect>, and <PlayRecord> messages are composed of two sections: Attributes and a Prompt block (the request can contain several different Prompt blocks). The Attributes section includes several request-specific parameters. The Prompt block section itself is also composed of two sections:

- **Prompt-specific Parameter (optional):** *baseurl* - defines a URL address that functions as a prefix to all audio segment URLs in the Prompt block.
- **Physical Audio Segments:** These are physical audio files that are located either locally or on an external HTTP server. If the file is located locally, the reference to it is by using one of the following syntaxes:

  'file://x', 'file:///x', 'file:////x' or 'http://localhost/x'

  Where *x* denotes the file's URL in for HTTP streaming.

4.2.2 Playing Announcements

A <Play> request is used to play an announcement to the caller. Each <Play> request contains a single Prompt block and the following request-specific parameters:

- *id*: an optional random number used to synchronize request and response.
- *prompturl*: a specific audio file URL that is used in addition to the references in the Prompt block. This audio file is the first to be played.

An example of an MSCML <Play> Request that includes local and streaming audio files is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<MediaServerControl version="1.0">
    <request>
          <play id=”123”>
                <prompt>
                      <audio url="http://localhost/1"/>
                      <audio url="http://10.3.0.2/aa.wav"/>
                <audiourl="nfs://10.3.0.3/prov_data/bb.wav"/>
                </prompt>
          </play>
    </request>
</MediaServerControl>
```

4.2.3 Playing Announcements and Collecting Digits

The <PlayCollect> request is used to play an announcement to the caller and to then collect entered DTMF digits. The play part of the <PlayCollect> request is identical to the <Play> request. The collect part includes an expected digit map. The collected digits are continuously compared to the digit map. Once a match is found, the collected digits are sent in a <PlayCollect> response. The digit map should be in MGCP format (the type value must be set to ‘mgcpdigitmap’). For example:

```
<regex type="mgcpdigitmap" value="([0-1]xxx)">
</regex>
```

Each <PlayCollect> request contains the following request-specific parameters in addition to the Prompt block (all parameters are optional):

- *id*: an optional random number used to synchronize request and response.
- *prompturl:* a specific audio file URL that is used in addition to the references in the prompt block. This audio file is the first to be played.
- *barge*: if set to ‘NO’, DTMF digits received during announcement playback are ignored. If set to ‘YES’, DTMF digits received during announcement playback stop the playback and start the digit collection phase.
- *firstdigittimer:* defines the amount of time (in milliseconds) the user does not enter any digits, after which a response is sent indicating timeout.
- *interdigittimer:* defines the amount of time (in milliseconds) the user does not enter any digits after the first DTMF digit is received, after which a response is sent indicating timeout.

- *extradigittimer:* used to enable the following:
  - Detection of command keys (ReturnKey and EscapeKey).
  - Not report the shortest match. MGCP Digitmap searches for the shortest possible match. This means that if a digitmap of (123 | 1234) is defined, once the user enters 123, a match is found and a response is sent. If ExtraDigitTimer is defined, the match can also be 1234 because the device waits for the next digits. To use ExtraDigitTimer, it must be defined in the request and you must add a "T" to the Digitmap (for example, 'xxT'). The ExtraDigitTimer is only used when a match is found. Before a match is found, the timer used is the InterDigitTimer. Therefore, if the ExtraDigitTimer expires, a "match" response reason is reported -- never a "timeout".
- *maxdigits:* defines the maximum number of collected DTMF digits after which the device sends a response.
- *cleardigits:* defines whether or not the device clears the digit buffer between subsequent requests.
- *returnkey*: defines a specific digit (including '*' and '#') which (when detected during a collection) stops the collection and initiates a response (that includes all digits collected up to that point) to be sent.
- *escapekey:* defines a specific digit (including '*' and '#') which (when detected during a collection) stops the collection and initiates a response (with no collected digits) to be sent.

An example is shown below of an MSCML <PlayCollect> Request that includes a sequence with variables and an MGCP digit map:

```
<?xml version="1.0" encoding="utf-8"?>
<MediaServerControl version="1.0">
          <request>
                <playcollect id="6379" barge="NO" returnkey="#">
                      <prompt>
                            <audio url="http://localhost/1">
                      </prompt>
                      <regex type="mgcpdigitmap" value="([0-
1]xxx)">
                      </regex>
                </playcollect>
          </request>
</MediaServerControl>
```

An example is shown below of an MSCML <PlayCollect> Response:

```
<?xml version="1.0" encoding="utf-8"?>
<MediaServerControl version="1.0">
          <response request="playcollect" id="6478" code="200"
text="OK" digits="4563">
          </response>
</MediaServerControl>
```

4.2.4 Playing Announcements and Recording Voice

The <PlayRecord> request is used to play an announcement to the caller and to then record the voice stream associated with that caller. The play part of the <PlayRecord> request is identical to the <Play> request. The record part includes a URL to which the voice stream is recorded. This URL refers to anHTTP server.

Each <PlayRecord> request contains the following request-specific parameters in addition to the Prompt block (all parameters except 'recurl' are optional):

- *id*: an optional random number used to synchronize request and response.
- *prompturl:* a specific audio file URL that is used in addition to the references in the prompt block. This audio file is the first to be played.
- *barge*: if set to 'NO', DTMF digits received during announcement playback are ignored. If set to 'YES', DTMF digits received during announcement playback stop the playback and start the recording phase.
- *cleardigits:* defines whether or not the device clears the digit buffer between subsequent requests.
- *escapekey:* defines a specific digit (including '*' and '#') which (when detected during any phase) stops the request and initiates a response.
- *recurl*: the URL on the external storage server to which the RTP stream is sent for recording. This is a mandatory parameter.
- *mode:* defines if the recording 'overwrites' the existing file or 'appends' to it.
- *initsilence:* defines how long to wait for initial speech input before terminating the recording. This parameter may take an integer value in milliseconds.
- *endsilence:* defines how long the device waits after speech has ended to stop the recording. This parameter may take an integer value in milliseconds.
- *duration*: the total time in milliseconds for the entire recording. Once this time expires, recording stops and a response is generated.
- *recstopmask:* defines a digit pattern to which the device compares digits detected during the recording phase. If a match is found, recording stops and a response is sent.

An example is shown below of an MSCML <PlayRecord> Request:

```
<?xml version="1.0" encoding="utf-8"?>
<MediaServerControl version="1.0">
          <request>
                <playrecord id="75899" barge="NO"
Recurl=nfs://10.11.12.13/save/recordings/11.wav>
                      <prompt>
                            <audio url="nfs://100.101.102.103/45">
                      </prompt>
                </playrecord>
          </request>
</MediaServerControl>
```

An example is shown below of an MSCML <PlayRecord> Response:

```
<?xml version="1.0" encoding="utf-8"?>
<MediaServerControl version="1.0">
          <response request="playrecord" id="75899" code="200"
text="OK" reclength="15005">
          </response>
</MediaServerControl>
```

4.2.5 Stopping the Playing of an Announcement

The Application server issues a <stop> request when it requires that the device stops a request in progress and not initiate another operation. The only (optional) request-specific parameter is id.

The device refers to a SIP re-INVITE message with hold media (c=0.0.0.0) as an implicit <Stop> request. The device immediately terminates the request in progress and sends a response.

An example is shown below of an MSCML <Stop> Request:

```
<?xml version="1.0" encoding="utf-8"?>
<MediaServerControl version="1.0">
          <request>
                <stop id="123">
                </stop>
          </request>
</MediaServerControl>
```

4.2.6 Relevant Parameters

The following parameters (described in IP Media Parameters) are used to configure the MSCML:

- AmsProfile = 1 (default and mandatory setting)
- AASPackagesProfile = 3 (default and mandatory setting)
- VoiceStreamUploadMethod = 1 (mandatory)
- EnableVoiceStreaming = 1 (mandatory)
- MSCMLID (default=”ivr”)
- When using AutoUpdate:
  - VPFileURL
  - AutoUpdateFrequency / AutoUpdatePredefinedTime

4.2.7 Signal Events Notifications

The device supports Signal Events Notifications as defined in RFC 4722/5022 - MSCML. MSCML defines event notifications that are scoped to a specific SIP dialog or call leg. These events allow a client to be notified of various call progress signals. Subscriptions for call leg events are performed by sending an MSCML <configure_leg> request on the desired SIP dialog. Call leg events may be used with the MSCML conferencing or IVR services. Using the Signal Notifications, the device can report the following events:

Table 4-1: Reportable Events

Type	Subtype
AMD	▪ voice ▪ automata ▪ silence ▪ unknown
CPT	▪ SIT-NC ▪ SIT-IC ▪ SIT-VC ▪ SIT-RO ▪ busy ▪ reorder
FAX	▪ CED ▪ CNG ▪ modem

Below is an example:

```
<?xml version="1.0"?>
 <MediaServerControl version="1.0">
  <request>
   <configure_leg>
    <subscribe>
     <events>
      <signal type="amd" report="yes"/>
     </events>
    </subscribe>
   </configure_leg>
  </request>
 </MediaServerControl>

<?xml version="1.0"?>
 <MediaServerControl version="1.0">
  <notification>
   <signal type="amd" subtype="voice"/>
  </notification>
</MediaServerControl>
```

4.3 Voice Streaming

The voice streaming layer provides you with the ability to play and record different types of files while using an NFS or HTTP server.

4.3.1 Voice Streaming Features

The following subsections summarize the Voice Streaming features supported on HTTP and NFS servers, unless stated otherwise.

4.3.1.1 Basic Streaming Play

You may play a .wav, .au or .raw file from a remote server using G.711 coders.

4.3.1.2 Supported File Formats

The voice streaming layer provides support for .wav, .au, and .raw file formats. The maximum supported header size of the file is 150 bytes.

In .wav format, only mono mode and supported/known coders are supported. The maximum number of the non-data, non-fmt chunks can be up to 5.

4.3.1.3 Play from Offset

You may play a .wav, .au or .raw file from a given offset within the file. Offset can be both positive and negative relative to the files length. A negative offset relates to an offset from the end of the file.

4.3.1.4 Remote File Systems

You may configure up to 16 remote file systems to operate with the system through NFS mounting.

4.3.1.5 Using Proprietary Scripts

You may use cgi or servlet scripts released with the version for recording to a remote HTTP server using the POST or PUT method.

4.3.1.6 Dynamic HTTP URLs

Voice streaming supports dynamic HTTP URLs. The following terminology is used:

- **Static audio content:** Traditional audio file URLs containing references to specific files (.wav, .au or .raw). For example: http://10.50.0.2/qa/GOSSIP_ENG.wav
- **Dynamic audio content:** URLs referencing to cgi scripts or servlets. For example: http://10.50.0.2/cgi/getaudio.cgi?filename=DEFAULT_GREETING.raw&offset=0

In the case of dynamic URLs, the device performs the GET command with the supplied URL and as a result, the servlet or cgi script on the Web server is invoked. The Web server responds by sending a GET response containing the audio.

The URL format can be as follows (RFC 1738 URLs, section 3.3):

```
http://<host>:<port>/<path>?<searchpart>
```

where,

- **:<port>** is optional.
- **<path>** is a path to a server-side script.
- **<searchpart>** is of the form: key=value**[&key**=value]*

Note: At least one key=value pair is required.

Another example of a dynamic URL is shown below:

```
http://MyServer:8080/prompts/servlet?action=play&language=eng&file
=welcome.raw&format=1
```

(See also RFC 2396 URI: Generic Syntax.)

The servlet or cgi script can respond by sending a complete audio file or a portion of an audio file. The device skips any .wav or .au file header that it encounters at the beginning of the response. The device does not attempt to use any information in the header. For example, the device does not use the coder from the header. Note however, that the coder may be supplied through Web or *ini* file parameters.

4.3.1.7 Play LBR Audio File

You may play a file using low bit rate coders for .wav and .raw files.

Note: This feature is relevant for both NFS and HTTP.

4.3.1.8 Basic Record

You may record a .wav, .au or .raw files to a remote server using G.711 coders.

Note: This feature is relevant for both NFS and HTTP.

4.3.1.9 Remove DTMF Digits at End of Recording

You may configure a recording to remove the DTMF received at the end, indicating an end of a recording.

Note: This feature is relevant for both NFS and HTTP.

4.3.1.10 Record Files Using LBR

You may record a file using low bit rate coders for .wav and .raw files.

Notes: This feature is relevant for both NFS and HTTP.

4.3.1.11 Modifying Streaming Levels Timers

Several parameters enable the user to control streaming level timers for NFS and HTTP and also the number of data retransmission when using NFS as the application layer protocol:

- **General command timeout – ServerRespondTimeout:** Defines the maximal time a command or respond may be delayed. This relates both to HTTP commands (GET, PUT, POST, HEAD etc.) and to NFS commands (create, lookup, read, write etc.).
- **Recording packet overruns timer – StreamingRecordingOverRunTimeout:** An overrun condition is one in which the device sends data to the server but does not receive responds from the server acknowledging that it received the data. Overruns relate to recording data to a remote server and result with “holes” in the recording. The streaming level aborts sessions containing consecutive overruns as derived from this timer. You may set the timer to longer periods than the default value, thereby enabling the device to be more "tolerant" to overrun conditions.
- **Playing packet underruns – StreamingPlayingUnderRunTimeout:** An underrun condition is one in which the device does not supply the DSPs with sufficient data, thus "starving" the DSPs. Underruns relate to playing data from a server to the device where, due to environmental conditions (usually network problems), the data is not passed quickly enough. This condition results with broken data passed to the user. The streaming level aborts sessions containing consecutive underruns as derived from this timer. You may set the timer to longer periods than the default value, thereby enabling the device to be more "tolerant" to underrun conditions.
- **NFS command retransmission – NFSClientMaxRetransmission:** Defines the number of times an NFS command is retransmitted when the server side does not respond. By default, the value is set to 0 and not used - instead, the number of retransmissions is derived from the server response timeout parameter and the current Recovery Time Objective (RTO) of the system.

These parameters may be configured using the *ini* file, Web interface, or SNMP.

4.3.2 Using File Coders with Different Channel Coders

The tables in the following subsections describe the support for different combinations of file coders (used for recording or playing a file) and channel coders (used when opening a voice channel).

The following abbreviations are used in the subsequent tables:

- **LBR:** Low Bit Rate Coder
- **PCMU:** G.711 µ-law coder
- **PCMA:** G.711 A-law coder
- **WB:** Linear PCM 16KHZ Wide Band Coder

Note: When recording with an LBR type coder, it is assumed that the same coder is used both as the file coder and the channel coder. Combinations of different LBR coders are currently not supported.

4.3.2.1 Playing a File

The table below lists the device's support of channel coders and file coders for playing a file.

Table 4-2: Coder Combinations - Playing a File

File Coder	File Type											
	.wav				.au				.raw			
	Channel Coder				Channel Coder				Channel Coder			
	PCMA	PCMU	LBR	WB	PCMA	PCMU	LBR	WB	PCMA	PCMU	LBR	WB
PCMA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
PCMU	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
LBR	No	No	Yes	No	No	No	No	No	No	No	Yes	Yes

4.3.2.2 Recording a File

The table below lists the device's support of channel coders and file coders for recording a file.

Table 4-3: Coder Combinations - Recording a File

File Coder	File Type											
	WAV				AU				RAW			
	Channel Coder				Channel Coder				Channel Coder			
	PCMA	PCMU	LBR	WB	PCMA	PCMU	LBR	WB	PCMA	PCMU	LBR	WB
PCMA	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	No
PCMU	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	No
LBR	No	No	Yes	No	No	No	No	No	No	No	No	No

4.3.3 Maximum Concurrent Playing and Recording

For details on maximum concurrent playing and recording, refer to the *Release Notes*.

4.3.4 LBR Coders Support

The following table describes the different low bit rate (LBR) coders and their support for .wav, .au, and .raw files.

Note: Coder support depends on the specific DSP template version installed on the device.

Table 4-4: LBR Coders and File Extension Support

Coder	.wav file	.raw file	.au file
G.726 (Rate 16)	Yes	Yes	No
G.726 (Rate 24)	Yes	Yes	No
G.726 (Rate 32)	Yes	Yes	No
G.726 (Rate 40)	Yes	Yes	No
G.723.1 (Rate 5.3)	Yes	Yes	No
G.723.1 (Rate 6.3)	Yes	Yes	No
G.729	Yes	Yes	No
GSM FR	Yes	Yes	No
MS GSM	Yes	Yes	No
GSM EFR	Yes	Yes	No
AMR (Rate 4.75)	No	Yes	No
AMR (Rate 5.15)	No	Yes	No
AMR (Rate 5.9)	No	Yes	No
AMR (Rate 6.7)	No	Yes	No
AMR (Rate 7.4)	No	Yes	No
AMR (Rate 7.95)	No	Yes	No
AMR (Rate 10.2)	No	Yes	No
AMR (Rate 12.2)	No	Yes	No
QCELP (Rate 8)	No	Yes	No
QCELP (Rate 13)	No	Yes	No

4.3.5 HTTP Recording Configuration

The HTTP record method (PUT or POST) is configured using the following offline *ini* parameter:

```
// 0=post (default), 1=put
VoiceStreamUploadMethod = 1
```

The default is shown below:

```
VoiceStreamUploadPostUri =
"/audioupload/servlet/AcAudioUploadServlet"
```

Note: The PUT method disregards this string.

4.3.6 Supported HTTP Servers

The following is a list of HTTP servers that are known to be compatible with AudioCodes voice streaming under Linux™:

- **Apache:** cgi scripts are used for recording and supporting dynamic URLs.
- **Jetty:** servlets scripts are used for recording and supporting dynamic URLs.
- **Apache tomcat:** using servlets.

4.3.6.1 Tuning the Apache Server

It is recommended to perform the following modifications in the http.conf file located in the apache conf/ directory:

- Define PUT script location: Assuming the put.cgi file is included in this package, add the following line for defining the PUT script (script must be placed in the cgi-bin/ directory):

  ```
  Script PUT /cgi-bin/put.cgi
  ```

- Create the directory /the-apache-dir/perl (for example, /var/www/perl) and copy the CGI script to this directory. In the script, change the first line from c:/perl/bin/perl to your perl executable file (this step is required only if mod_perl is not included in your Apache installation).
- Keep-alive parameters: the following parameters must be set for correct operation with multiple POST requests:

  ```
  KeepAlive On
  MaxKeepAliveRequests 0 (unlimited amount)
  ```

- Using mode perl, fix the mod_perl to the following:

  ```
  <IfModule mod_perl.c>
  <Location /cgi-bin>
   SetHandler perl-script
     PerlResponseHandler ModPerl::Registry
     Options +ExecCGI
     PerlOptions +ParseHeaders
     Order allow,deny
     Allow from all
  </Location>
  </IfModule>
  ```

- Apache MPM worker: it is recommended to use the Multi-Processing Module implementing a hybrid multi-threaded multi-process Web server. The following configuration is recommended:

```
<IfModule worker.c>
ThreadLimit       64
StartServers         2
ServerLimit       20000
MaxClients        16384
MinSpareThreads    100
MaxSpareThreads     250
ThreadsPerChild     64
MaxRequestsPerChild  16384
</IfModule>
```

4.3.7 Common Troubleshooting

Always inspect the Syslog for any problem you may encounter; in many cases, the cause appears there.

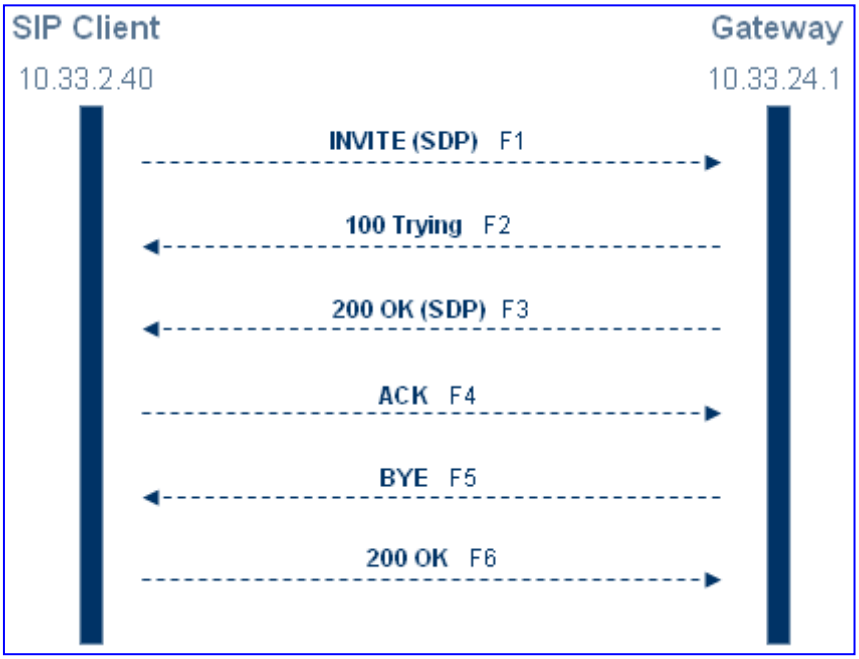
Table 4-5: Troubleshooting

Problem	Probable Cause	Corrective Action
General Voice Streaming Problems		
Attempts to perform voice streaming operations results in each Syslog containing the string: 'VS_STACK_NOT_ACTIVE'.	Voice streaming is not enabled.	Enable voice streaming by loading an *ini* file containing this entry: EnableVoiceStreaming = 1
HTTP Voice Streaming Problems		
The last half-second of an announcement is not played, or a record operation terminates abnormally and the Syslog displays the following: 'VSReceiveFromNetwork: VS_CONNECTION_WITH_SERVER_LOST'. (The problem has been experienced with Apache version 2.0.50 on Solaris 9.)	The Web server is closing the virtual circuit at unexpected times.	Increase the Apache KeepAliveTimeout config parameter. Try to increase it to 30 seconds or longer than the longest announcement or expected record session.

4.4 Announcement Call Flow Example

The call flow, shown in the following figure, describes SIP messages exchanged between the device (10.33.24.1) and a SIP client (10.33.2.40) requesting to play local announcement #1 (10.8.25.17) using AudioCodes proprietary method.

Figure 4-2: Announcement Call Flow Example



1. **SIP MESSAGE 1:** 10.33.2.40:5060 -> 10.33.24.1:5060

```
INVITE
sip:annc@10.33.24.1;play=http://10.3.0.2/hello.wav;repeat=2
SIP/2.0
Via: SIP/2.0/UDP 10.33.2.40;branch=z9hG4bKactXhKPQT
Max-Forwards: 70
From: <sip:103@10.33.2.40>;tag=1c2917829348
To: <sip:annc@10.33.24.1>
Call-ID: 1414622340oZZq@10.33.2.40
CSeq: 1 INVITE
Contact: <sip:103@10.33.2.40>
Supported: em,100rel,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-4.0 GA/v.4.0 GA
Content-Type: application/sdp
Content-Length: 215
v=0
o=AudiocodesGW 377662 728960 IN IP4 10.33.41.52
s=Phone-Call
c=IN IP4 10.33.41.52
t=0 0
m=audio 4030 RTP/AVP 4 0 8
a=rtpmap:4 g723/8000
a=rtpmap:0 pcmu/8000
a=rtpmap:8 pcma/8000
a=ptime:30
a=sendrecv
```

2. **SIP MESSAGE 2:** 10.33.24.1:5060 -> 10.33.2.40:5060

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 10.33.2.40;branch=z9hG4bKactXhKPQT
From: <sip:103@10.33.2.40>;tag=1c2917829348
To: <sip:annc@10.33.24.1>;tag=1c1528117157
Call-ID: 1414622340oZZq@10.33.2.40
CSeq: 1 INVITE
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40.010.006D
Content-Length: 0
```

3. **SIP MESSAGE 3:** 10.33.24.1:5060 -> 10.33.2.40:5060

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.33.2.40;branch=z9hG4bKactXhKPQT
From: <sip:103@10.33.2.40>;tag=1c2917829348
To: <sip:annc@10.33.24.1>;tag=1c1528117157
Call-ID: 1414622340oZZq@10.33.2.40
CSeq: 1 INVITE Contact: <sip:10.33.24.1>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40.010.006D
Content-Type: application/sdp
Content-Length: 165
v=0
o=AudiocodesGW 355320 153319 IN IP4 10.33.24.1
s=Phone-Call
c=IN IP4 10.33.24.1
t=0 0
m=audio 7170 RTP/AVP 0
a=rtpmap:0 pcmu/8000
a=ptime:20
a=sendrecv
```

4. **SIP MESSAGE 4:** 10.33.2.40:5060 -> 10.33.24.1:5060

```
ACK sip:10.33.24.1 SIP/2.0
Via: SIP/2.0/UDP 10.33.2.40;branch=z9hG4bKacnNUEeKP
Max-Forwards: 70
From: <sip:103@10.33.2.40>;tag=1c2917829348
To: <sip:annc@10.33.24.1>;tag=1c1528117157
Call-ID: 1414622340oZZq@10.33.2.40
CSeq: 1 ACK
Contact: <sip:103@10.33.2.40>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-4.0 GA/v.4.0 GA
Content-Length: 0
```

5. **SIP MESSAGE 5:** 10.33.24.1:5060 -> 10.33.2.40:5060

```
BYE sip:103@10.33.2.40 SIP/2.0
Via: SIP/2.0/UDP 10.33.24.1;branch=z9hG4bKacFhtFbFR
Max-Forwards: 70
From: <sip:annc@10.33.24.1>;tag=1c1528117157
To: <sip:103@10.33.2.40>;tag=1c2917829348
Call-ID: 1414622340oZZq@10.33.2.40
CSeq: 1 BYE
Contact: <sip:10.33.24.1>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
User-Agent: Audiocodes-Sip-Gateway-Mediant 1000
MSBR/v.5.40.010.006D
Content-Length: 0
```

6. **SIP MESSAGE 6:** 10.33.2.40:5060 -> 10.33.24.1:5060

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.33.24.1;branch=z9hG4bKacFhtFbFR
From: <sip:annc@10.33.24.1>;tag=1c1528117157
To: <sip:103@10.33.2.40>;tag=1c2917829348
Call-ID: 1414622340oZZq@10.33.2.40
CSeq: 1 BYE
Contact: <sip:103@10.33.2.40>
Supported: em,timer,replaces,path
Allow:REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFE
R,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-4.0 GA/v.4.0 GA
Content-Length: 0
```

Reader's Notes

AudioCodes

Reference Guide

www.audiocodes.com