ETSI TS 101 909-19-2 V1.1.1 (2002-03)

Technical Specification

Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 19: IPCablecom Audio Server Protocol Specification; Sub-part 2: MGCP option

Reference

DTS/AT-020020-19-2

Keywords

access, broadband, cable, IP, multimedia, PSTN

Contents

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Access and Terminals (AT).

The present document is part 19, sub-part 2 of a multi-part deliverable covering Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services. Full details of the entire series can be found in part 1 (TS 101 909-1 [15]).

Introduction

The present document describes the architecture and specifies the protocols that may be used for playing announcements in voice-over-IP (VoIP) IPCablecom networks.

Announcements are typically needed for calls that do not complete. Additionally, they may be used to provide enhanced information services to the caller. Different carrier service feature sets require different announcement sets and announcement formats. Announcements can be as basic as fixed-content announcements (e.g, all circuits busy) or as complex as those provided by intelligent IVR (Interactive Voice Response) systems. The IPCablecom service model requires that all announcements be provisioned and signalled in a standard manner for all supported call features and use case scenarios. The present document defines a set of signalling protocols that are used to provide announcement services within a cable network.

NOTE: An alternative audio server solution may be found in TS 101 909-19-1. Where alternative solutions for the same interface are being considered, interoperability issues between the various IPCablecom system components need to be addressed.

1 Scope

The present document describes the architecture and protocols that are required for playing announcements in voice over IP (VoIP) IPCablecom networks as an internal component of the IPCablecom system Announcements are typically needed for calls that do not complete. Additionally, they may be used to provide enhanced information services to the caller. Different carrier service feature sets require different announcement sets and announcement formats.

Announcements can be as basic as fixed-content announcements (e.g, all circuits busy) or as complex as those provided by intelligent IVR (Interactive Voice Response) systems. The IPCablecom service model requires that all announcements be provisioned and signalled in a standard manner for all supported call features and use case scenarios.

The present document identifies a set of signaling protocols that are used to provide announcement services within a cable network. For one of these protocols, the IPCablecom Network Call Signaling (NCS) protocol [1], the present document defines two new event packages:

- A Base Audio Package.
- An Advanced Audio Package.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] ETSI TS 101 909-2: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 2: Architectural framework for the delivery of time critical services over cable Television networks using cable modems".

[2] ETSI TS 101 909-4: "Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 4: Network Call Signalling Protocol".

[3] ETSI ETR 187: "Human Factors (HF); Recommendation of characteristics of telephone services tones when locally generated in telephony terminals".

[4] ETSI TS 101 909-13: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 13: Trunking Gateway Control Protocol".

[5] ETSI TS 101 909-3: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 3: Audio Codec Requirements for the Provision of Bi-Directional Audio Service over Cable Television Networks using Cable Modems".

[6] ETSI TS 101 909-11: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 11: Security".

[7] ETSI TS 101 909-10: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 10: Event Message Requirements for the Provision of Real Time Services over Cable Television Networks using Cable Modems".

[8] ITU-T Recommendation J.112: "Transmission systems for interactive cable television services".

[9] ISO 639-2: "Codes for the representation of names of languages - Part 2: Alpha-3 code".

[10] ISO 4217: "Codes for the representation of currencies and funds".

[11] ISO 8601: "Data elements and interchange formats - Information interchange - Representation of dates and times".

[12] IETF 2705 Version 1.0: "Media Gateway Control Protocol (MGCP)".

[13] IETF 2396: "Uniform Resource Identifiers (URI): Generic Syntax".

[14] IETF 2234: "Augmented BNF for Syntax Specifications: ABNF".

[15] ETSI TS 101 909-1: "Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 1: General".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

access node: Layer two termination device that terminates the network end of the ITU-T Recommendation J.112 connection.

NOTE: It is technology specific. In ITU-T Recommendation J.112 [8], annex A it is called the INA while in annex B it is the CMTS.

announcement: An announcement to be played.

NOTE: Consists of one or more audio segments.

announcement servers (also known as audio servers): network components that manage and play informational tones and messages in response to events that occur in the network

NOTE: Most announcements are media streams that originate from servers in the network. Some simple tones and short announcements can also reside at the MTA and in the MG.

append: If set to true, the audio recording will append to any existing content in the Recording ID.

cable modem: layer two termination device that terminates the customer end of the J.112 connection

cable modem termination system: device at a cable head-end which implements the DOCSIS RFI MAC protocol and connects to CMs over an HFC network

clear digit buffer: If set to true, clears the digit buffer before playing the initial prompt.

delete persistent audio: indicates that the specified persistent audio segment is to be deleted

digit map: As specified in IETF 2705, Media Gateway Control Protocol (MGCP) Version 1.0, which specifies one or more digit patterns to be collected.

duration: maximum amount of time to play and possibly replay an announcement

extra digit timer: amount of time to wait for a user to enter a final digit once the maximum expected amount of digits has been entered

failure announcement: played when all data entry attempts have failed

first digit timer: amount of time allowed for the user to enter the first digit

H.248: ITU/IETF protocol for media gateway control.

NOTE: Also known as MEGACO. See http://www.itu.org/ for details**.**

initial prompt: initial announcement prompting the user to either enter DTMF digits or to speak

inter digit timer: amount of time allowed for the user to enter each subsequent digit

interval: silence to be inserted between iterative plays

IPCablecom: ETSI working group project that includes an architecture and a series of specifications that enable the delivery of real time services (such as telephony) over the cable television networks using cable modems

iterations: maximum number of times an announcement is to be played

ManageAudio: performs audio management operations on persistent audio which typically not related to a current interaction with a user, e.g. "delete an audio segment"

MEGACO: IETF/ITU protocol for media gateway control

NOTE: Also known as H.248. See www.ietf.org for details.

no digits reprompt: played after the user has failed to enter a valid digit pattern during a PlayCollect event

no speech reprompt: played after the user has failed to speak during a PlayRecord event

NonInterruptible Play: If set to true, the initial prompt of the PlayCollect or PlayRecord event is not interruptible by either voice or digits.

number of attempts: number of attempts the user is allowed to enter a valid digit pattern or to make a recording

offset: specifies the offset into an announcement to start playing

off-net(work): voice call or data transmission session in which either the originating or terminating device is connected to an IPCablecom network which is interconnected to another network which is supporting the second terminal

Offset: specifies the offset into an announcement to start playing

on-net(work): voice call or data transmission session in which the originating and terminating devices are connected to a single IPCablecom network which may consist of one or more zones or domains

OperationComplete: detected upon the successful completion of a Play, PlayRecord, Play Collect, or ManageAudio signal

OperationFailed: detected upon the failure of a Play, PlayRecord, PlayCollect, or ManageAudio signal

PlayAnnouncement: plays an announcement in situations where there is no need for interaction with the user

PlayCollect: plays a prompt and collects DTMF digits entered by a user

PlayRecord: plays a prompt and records user speech

prespeech timer: amount of time to wait for the user to initially speak

postspeech timer: amount of silence necessary after the end of the last speech segment for the recording to be considered complete

Recording ID: URI to be assigned to the physical segment which is to be recorded by the PlayRecord event

recording length timer: maximum allowable length of the recording, not including pre or post speech silence

record persistent audio: If set to true, the recording that is made is persistent instead of temporary.

reinput key: defines a digit map that, if matched, has the following action: discard any digits collected or recordings in progress and resume digit collection or recording

reprompt: played after the user has made an error such as entering an invalid digit pattern or not speaking

restart key: defines a digit map that, if matched, has the following action: discard any digits collected or recording in progress, replay the prompt, and resume digit collection or recording

return key: defines a digit map that, if matched, has the following action: stop digit collection or recording

Service ID (SID): 14-bit number assigned by a CMTS to identify an upstream virtual circuit.

NOTE: Each SID separately requests and is granted the right to use upstream bandwidth.

Standalone Media Terminal Adapter (S-MTA): single node which contains an MTA and a non-J.112 MAC (e.g, Ethernet)

speed: relative playback speed of announcement specifiable as a positive or negative percentage of the original playback speed

success announcement: played when data collection has succeeded

volume: relative playback volume of announcement specifiable as a positive or negative decibel variation from the original playback volume

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ABNF	Augmented Backus-Naur Form
AN	Acess Node
ANP	ANouncement Player
AS	Audio Server
ASP	Audio Server Protocol
CMS	Call Management Server
CMTS	Cable Modem Termination System
DOCSIS	Data Over Cable Service Interface Specification
DTMF	Dual Tone Multi-Frequency
EDT	Extra Digit Timer
HFC	Hybrid Fibre Coaxial
ICT	Information and Communications Technologies
IVR	Interactive Voice Response system
MEGACO	MEdia GAteway COntrol
MG	Media Gateway
MGC	Media Gateway Controller
MGCP	Media Gateway Control Protocol
MP	Media Player
MPC	Media Player Controller
MTA	Media Terminal Adapter
NCS	Network-based Call Signalling
PSTN	Public Switched Telephone Network
RKS	Record Keeping Server
RTP	Real Time Protocol
SDP	Session Description Protocol
SID	Service ID
S-MTA	Standalone MTA
TGCP	Trunking Gateway Control Protocol
URI	Universal Resource Identifier
URL	Uniform Rosource locator
VoIP	Voice over IP

4 Technical overview

The IPCablecom Audio Server specification defines a suite of signaling protocols for providing announcement and media services in an IPCablecom network. This clause:

- defines the architectural requirements for providing IPCablecom announcement and media services,
- defines and categorizes announcement and media types,
- defines the components and their roles in the IPCablecom Audio Server Architecture, and
- describes the signaling and media interfaces in the IPCablecom Audio Server specification.

4.1 Architectural Requirements

The architectural requirements and assumptions for providing Audio and Media Services for an IPCablecom Network are listed below. These requirements are based upon the specifications and technical reports that define the IPCablecom architecture.

The reference architecture for the IPCablecom Network [1] is shown in figure 1.

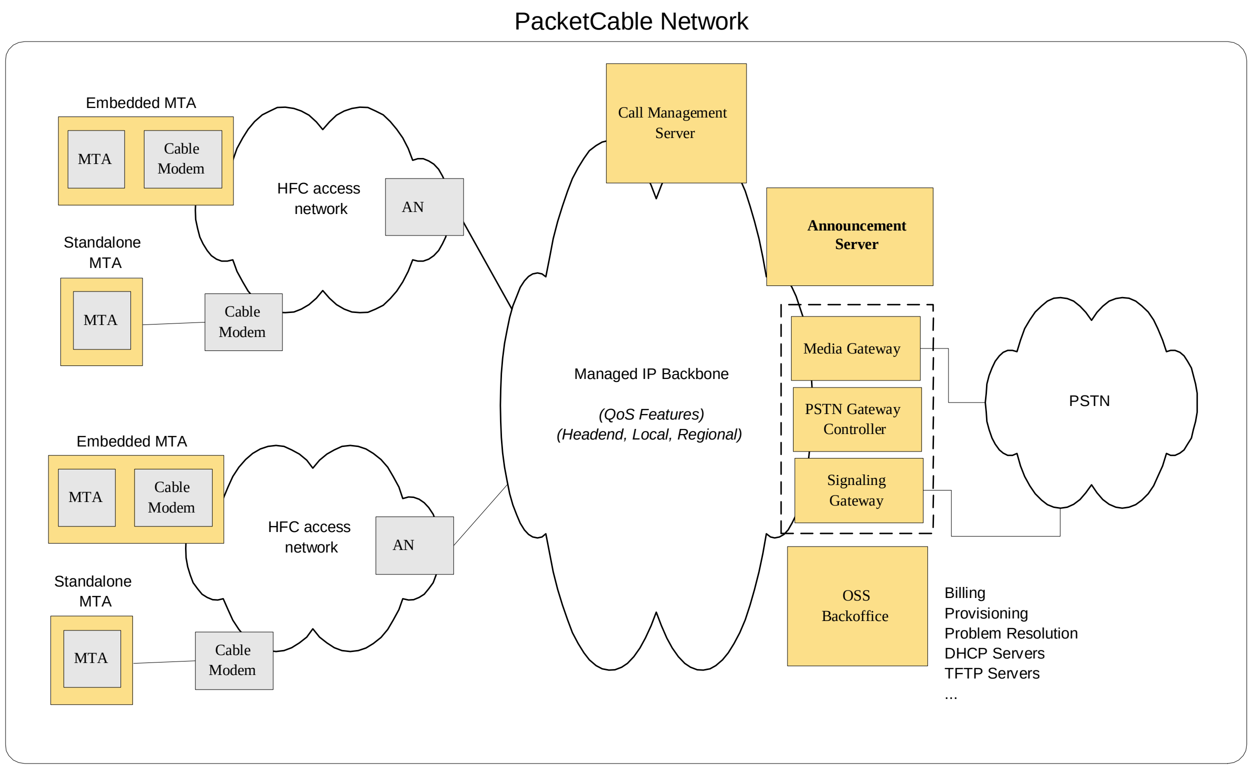


Figure 1: IPCablecom Network Component Reference Model

4.1.1 Call destination

The Audio Server Specification shall define how announcements are provided for IPCablecom on-net to off-net and on-net to on-net calls.

NOTE: Announcements for Off-net to on-net calls will usually be handled by the PSTN as a result of SS7 clearing messages. However when appropriate, they also may be played from the IPCablecomIPCablecom Media Gateway (MG).

4.1.2 Media formats

The required media formats for announcements are specified by the IPCablecom Codecs specification [5].

4.1.3 Security

Audio shall be signalled and played in a secure manner. Security protocols defined in the IPCablecom Security specification [6] shall be supportable in the IPCablecom Audio Server Specification [5].

4.1.4 Operational support systems

Audio Servers may be required to support the IPCablecom billing and event message protocols [7].

4.2 Announcement definitions

Announcements can be divided into four distinct categories: tones, fixed-content, variable content, and interactive announcements.

4.2.1 Tones

Includes tones such as reorder, busy, and ringback. See ETR 187 [3] for further examples.

4.2.2 Fixed-content announcements

Fixed-content Announcements consist of audio messages with fixed-content that require no user interaction. For example, "Your call did not go through. Please hang up and try your call again".

4.2.3 Variable content announcements

Variable Content Announcements are messages that contain a customizing parameter(s) yet require no user interaction. For example, "The number you have dialled, 321-9876, has been changed. The new number is 321-6789."

4.2.4 Interactive announcements

Interactive Announcements are announcements that require user interaction, DTMF (Dual Tone Multi-Frequency) or IVR. For example, "The number you have dialled, 541-321-9876, has been changed. The new number is 541-321-6789. To be connected to the new number, at a cost of thirty-five cents, please press 1."

4.2.5 Naming conventions for endpoint identifiers

A flat name space for endpoints is used, with audio ports indicated by the prefix *aud* and the port number, e.g, aud/12@audio-server-3.whatever.net. Wildcards ($,) may by used in place of the port numbers in accordance with standard NCS rules [2] for wild card use.

Systems that support announcements only (i.e, no digit collection, no recording, and no speech recognition ability) may use the prefix *ann* instead of *aud*.

Some systems may use one additional level in the naming scheme to support the identification of specific cards. In this case the naming would look like aud/<card number>/<port number>@audio-server-3.whatever.net

4.3 Architectural components

IPCablecom components responsible for providing announcement services are defined below. These components work together to provide the complete set of announcement services available from the IPCablecom network provider. There may be more than one of these components in the network. Figure 2 defines a logical architecture for providing announcement services and only where an interface is exposed is it expected to meet IPCablecom specification requirements.

4.3.1 Audio Server (AS)

An AS is a logical entity composed of a Media Player Controller (MPC) and a Media Player (MP).

4.3.1.1 Media Player Controller (MPC)

The MPC initiates and manages all announcement services provided by the Media Player. The MPC accepts requests from the CMS and arranges for the MP to provide the announcement in the appropriate stream so that the user hears the announcement. The MPC also serves as the termination for certain calls routed to it for IVR services. These might include, for example, calls in which the user dials a 'free phone' number to reach a credit-card calling service operated by the IPCablecom network operator. When the MP collects information from the end-user, the MPC is responsible for interpreting this information and manage the IVR session accordingly. Hence, the MPC will manage call state.

The MPC can be standalone, or it can be embedded within the CMS. See figures 2 and 3 for illustrations of standalone and embedded MPC configurations.

4.3.1.2 Media Player (MP)

The Media Player is a media resource server. It is responsible for receiving and interpreting commands from the MPC and for delivering the appropriate announcement(s) to the MTA. The MP provides the media streams with the announcement contents. The MP also is responsible for accepting and reporting user inputs (e.g, DTMF tones). The MP functions under the control of the MPC.

An MP can be standalone, or it can be embedded with the MPC in a Media Server. See figures 2 and 4 for respective illustrations of the stand-alone and embedded MP configurations.

4.3.2 Multimedia Terminal Adapter (MTA)

The MTA has the ability to provide tones and a limited set of fixed-content announcements to the user. The MTA accepts NCS [2] signaling requests from the CMS and plays the appropriate tones and announcements accordingly.

4.3.3 Media Gateway (MG)

The Media Gateway also has the ability to provide fixed-content announcements to PSTN end-users involved in off-net to on-net calls. The MG accepts TGCP [4] request to play announcement from the Media Gateway Controller (MGC) and provides the announcements accordingly.

4.3.4 Call Management Server (CMS)

The CMS determines when announcements should be played at the MTA, when to use the resources of a network MPC/MP complex, and when to play announcements to a PSTN end-user from the MG. This is based on the status of a call in progress. The CMS then signals the appropriate entity: MTA, MPC, or MGC to play tones or announcements to the end-user, accordingly.

4.4 IPCablecom Audio Server interface descriptions

The signaling interfaces to support Media Services are shown in figure 2 and are summarized in table 1.

Table 1: Announcement Interfaces

Interface	Signaling Components	Protocol
Ann-1	MTA/CMS, MGC/MG	NCS/TGCP with announcement packages
Ann-2	MPC/MP	NCS with announcement package
Ann-3	CMS/MPC, CMS/MGC	Undefined. See clause 4.4.3.
Ann-4	MP/MTA	RTP

The remainder of this clause provides an overview of the announcements interfaces introduced above.

4.4.1 Ann-1 Interface - CMS/MTA and MGC/MG announcement package

An announcement package based upon the IPCablecom Network Call Signaling (NCS) protocol has been defined that can be used for both the CMS-MTA and MGC-MG interfaces.

4.4.1.1 CMS/MTA interface

The CMS to MTA interface provides a mechanism for the CMS to signal the MTA to play locally stored announcements. Simple tones and some frequently used fixed-content announcements (e.g, network busy) may be stored in the MTA so they can be played to the IPCablecom subscriber without tying up network bandwidth or media resources. Furthermore, storing these announcements in the MTA allows for providing informative progress tones to the end user independently of the network state (e.g, congestion).

4.4.1.2 MGC/MG interface

The MGC to MG interface provides a mechanism for the MG to play fixed-content announcements to PSTN end-users involved in off-net to on-net calls. For example, MG announcements may be used to provide PSTN users call progress information for calls that cannot be completed to the IPCablecom network (all-lines-busy). Simple, fixed-content announcements (e.g, all-lines-busy) may be stored at the Media Gateway to provide announcements to PSTN users.

4.4.2 Ann-2 interface - MPC/MP announcement package

The MPC to MP protocol is based upon an NCS announcement package. Less frequently used tones and fixed-content announcements, as well as all variable content and interactive announcements are provided by MPC and MP complex.

When the CMS identifies a need for an AS-based announcement, it sends a request to the MPC over interface Ann-3. Upon receiving a request from the CMS, the MPC opens a session with the Media Player using the NCS package. The MP then interacts with the specified endpoint over interface Ann-4.

4.4.3 Ann-3 interface - CMS/MPC and CMS/MGC

The Ann-3 interface allows the CMS to request the MPC to establish announcement sessions between the MP and another endpoint. It also allows the CMS to request the MGC to have the MG play fixed-content announcements to a PSTN endpoint. This interface is currently undefined. It is expected that this signaling interface will be based upon the IPCablecom CMS/CMS signaling protocol (to be specified in TS 101 909-16). The protocol for the Ann-3 is for further study.

4.4.4 Ann-4 interface - MP/MTA

The interface Ann-4 defines the media stream format (RTP) for delivery of the announcement from the ANP to the MTA. The specifics of interface Ann-4 are not within the scope of the present document.

4.4.5 Audio Server physical vs logical configuration

It should be noted that the MPC and MP are logical components that may reside in the same physical entities. When logical components reside in the same physical entity, interfaces between these components are not exposed and become unspecified.

It should also be noted that standalone components using the Ann-2 and Ann-3 interfaces specified in the present document may be shared by many network entities.

Figure 2 depicts an example of a network where the CMS, MPC, and MP are implemented as separate physical entities communicating over the Ann-2 and Ann-3 interfaces.

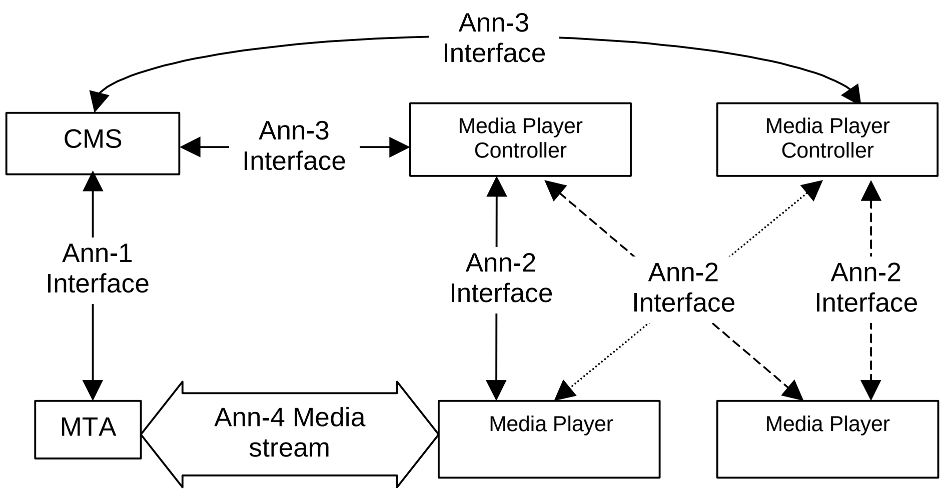


Figure 2: Standalone Components Configuration

The MPC maybe embedded with the CMS, as shown in figure 3. In this case, interface Ann-3 is not exposed and becomes unspecified.

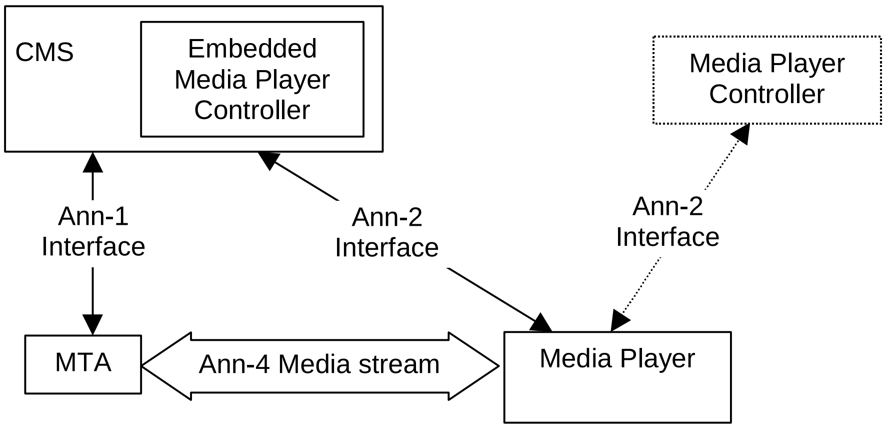


Figure 3: Embedded MPC Configuration

Similarly, the MP may be embedded with the MPC, as shown in figure 4 in which case the interface Ann-2 is not exposed and becomes unspecified.

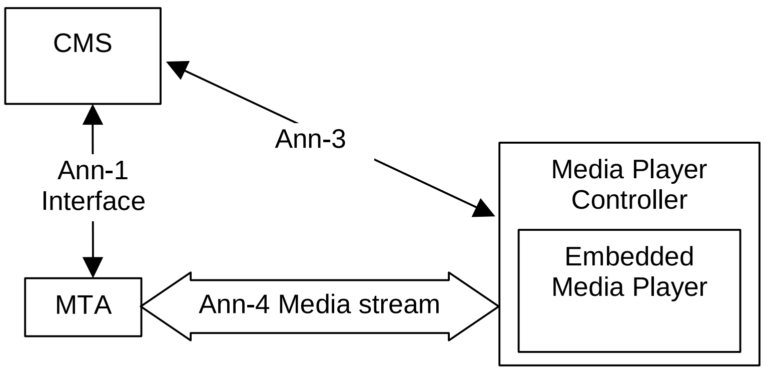


Figure 4: Embedded MP Configuration

Finally, the CMS and AS, (MPC and MP) may be embedded in the same physical entity, in which case the Ann-2 and Ann-3 interfaces are not exposed and become unspecified.

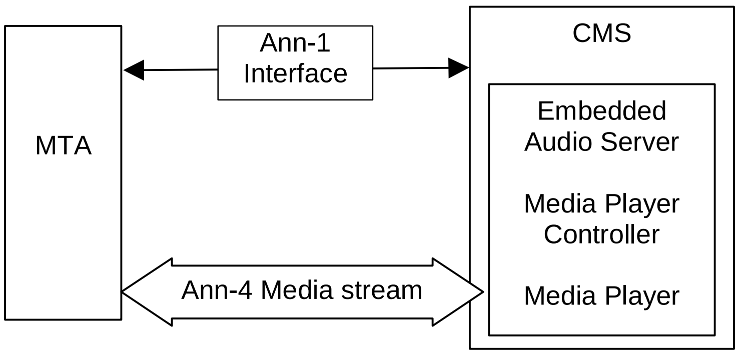


Figure 5: Embedded AS Configuration

4.5 Interface specifications

The IPCablecom Audio Server Specification defines a set of interfaces between the components responsible for providing audio services. Figure 6 illustrates the interfaces between these components. Only where an interface is exposed is it expected to meet IPCablecom specification requirements.

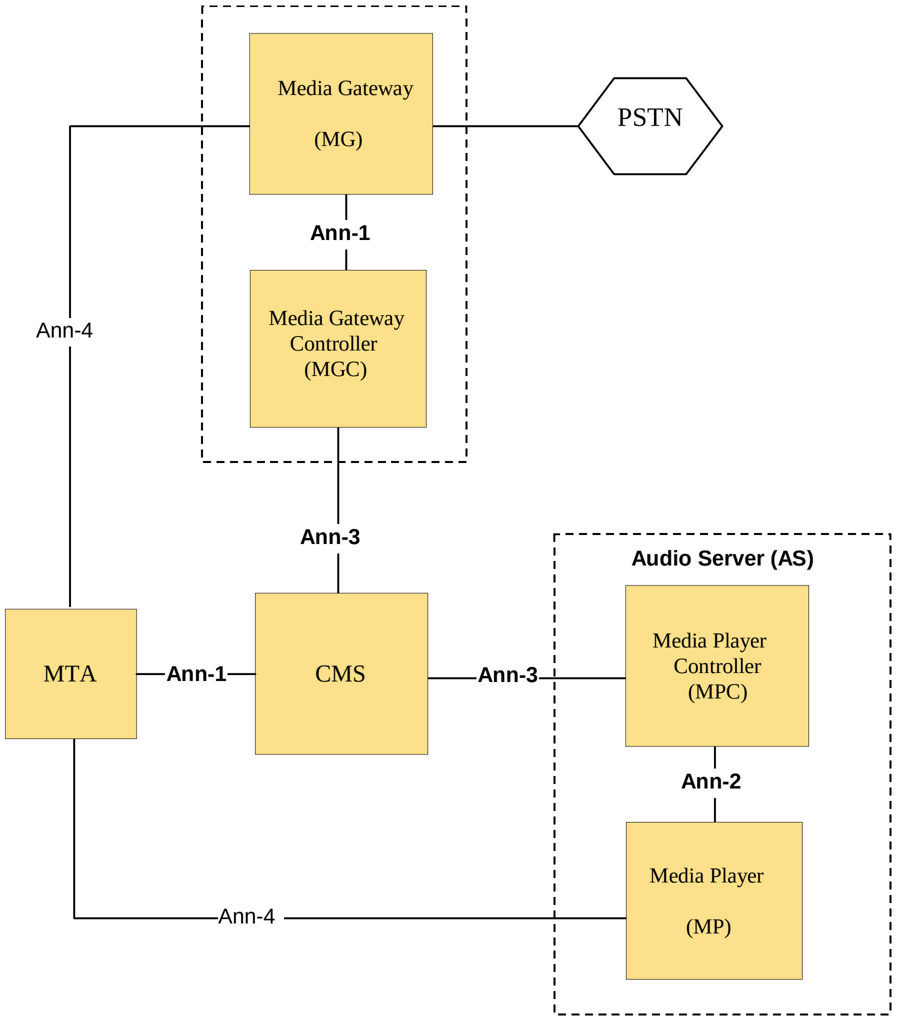


Figure 6: IPCablecom Audio Server Components and Interfaces

5 Ann-1 interface: CMS-MTA and MGC-MG

The CMS-MTA and MGC-MG announcement interfaces are implemented by the Legacy Audio Package of the NCS/TGCP protocol, which provides the playback of tones and fixed-content announcements to the end-users.

5.1 CMS-MTA interface

Each MTA in the network may store a predefined set of simple announcements locally. When an announcement is needed, the CMS will decide if it should instruct the MTA to play a local announcement or set up a connection between the MTA and a Network ANS and have the announcement played over the network. Playing simple announcements from the MTA saves network resources.

The MTA may store announcements in either static or dynamic memory. If announcements are stored in dynamic memory then the announcements will not be available until the MTA has accessed them from the network.

MTAs require the ability to be updated dynamically with announcements so that the same MTA can move from service provider to service provider without requiring complete firmware upgrades. This capability is for further study and will need to be worked jointly with the IPCablecom architecture, security, and provisioning teams.

5.1.1 Announcement list

Each MTA is required to store and play announcements a network-defined set of announcements for common network situations. Theses announcements may be played using the Announcement Server Package defined in IETF 2705 [12], using URI (Universal Resource Identifiers) to identify the announcements. Cached versions of all announcement URIs should be refreshed every time the MTA connects to the network. Other methods of propagating new announcements to MTAs, for instance while the MTA remains in service, is for further study.

5.2 MGC-MG interface

The MG announcement interface (Ann-1) allows for the MGC to request the MG play fixed-content announcements to PSTN end-users. The MGC/MG announcement interface package does not specify any standard announcements to be stored locally in the MG. All announcements are provisioned dynamically and are referenced accordingly.

This MG announcement provisioning capability is for further study and will need to be worked jointly with the IPCablecom architecture, security, and provisioning teams.

6 Ann-2 interface: MPC-MP

6.1 Introduction

An MP (Media Player) is a shared resource in the IPCablecom Network that can be instructed to provide media services to an end-user or terminal. These services include streaming fixed-content, variable content and interactive announcements to IPCablecom subscribers. For example, the MP is responsible for playing prompts and collecting digits when charging a call to a calling card.

The MP is controlled by an external element, the MPC (Media Player Controller). The MPC-MP Interface defines a two new NCS announcement packages used to control the Media Player. The Base Audio Package provides a standard set of IVR functions such as Play, PlayCollect, and PlayRecord. The Advanced Audio Package is a superset of the Base Audio Package and provides additional capabilities.

The MP is responsible for managing its own resources. When accepting a request, the MP SHALL make sure that the required resources are available before accepting request. When a single session involves multiple requests to the Media Player, the MP may experience a shortage of resources preventing it from accepting one given request belonging to that session. In this case, the MP user (i.e, the MPC) is responsible for re-sending the request or terminating the end-user session elegantly.

6.2 Audio package concepts

The base and advanced audio packages support both simple and complex audio structures. A simple audio structure might be a single announcement such as "Welcome to our automated directory assistance service." A more complex audio structure might consist of an announcement followed by voice variable followed by another announcement, for example "There are thirty seven minutes remaining on your prepaid calling card," where "There are" is a fixed string prompt, the number of minutes is a voice variable, and "minutes remaining on your prepaid calling card" is another fixed string prompt.

It is also possible to define complex audio structures that are qualified by user defined selectors such as language, audio file format, gender, accent, customer, or voice talent. For instance, if the above example were qualified by language and accent selectors, it would be possible to play "There are thirty seven minutes remaining on your prepaid calling card" in English spoken with a southern accent providing that the audio to support this had been provisioned.

There are two methods of specifying complex audio. The first is to directly reference the individual components. This requires a complete description of each component to be specified via the protocol. The second method is to provision the components on the Audio Server as a single entity and to export a reference to that entity to the call agent. In this case, only the reference (plus any dynamic data required, such as a variable data) is passed via the protocol, and no specification of individual components is necessary.

These packages provide significant functionality most of which is controlled via protocol parameters. Most parameters are optional, and where ever possible default to reasonable values. An audio application that references to complex provisioned audio structures can specify audio events using a minimum of syntax by taking advantage of optional parameters and parameter defaults.

6.2.1 Understanding audio segments

An audio segment is a reference that resolves to one or more audio recordings. There are four types of audio segments:

Physical: A physical segment is the simplest type of segment, a single recording. The recording could be a single word, such as "one", or an extended block of speech, such as "Our office is closed at this time. Please call back during normal business hours." Every physical segment is assigned a unique URI (Universal Resource Identifier) which among other things can be a hierarchical name, or a simple name or number.

Sequence: A sequence is a provisioned ordered list of audio segments. Every sequence is assigned a unique URI. A sequence can contain any of the four segment types (physical segments, other sequences, sets, and variables). On playback a sequence identifier is resolved to an ordered list of physical segments which are played in order.

Set: A set is a provisioned collection of semantically related audio segments and an associated selector. Each set is assigned a unique URI. A set can contain physical segments, sequences, other sets, or variables. At runtime the value of the selector is used to determine which element of the set is played.

Individual selector types are not defined in the syntax (except for the pre-defined language selector) and are instead defined by the provisioner. A provisioner could define one or more of the following selector types: language, accent, gender, accent, customer, or day of the week. For each selector type, the provisioner shall define a range of valid values. The provisioner may also choose to define a default value. At runtime if a selector value is not supplied the default value is used.

Variable: A voice variable represents a single semantic concept (such as date or number) and dynamically produces the appropriate speech based on information supplied at runtime. Each provisioned voice variable is assigned a unique URI. For example, if an application needs to play a date, rather than telling the AudioServer to play each individual component of the date (e.g, "March" "twenty" "second" "nineteen" "ninety" "nine"), it can specify a voice variable of type date with value "19990322". The variable then assembles and plays the component audio needed to speak the date. Specification of variables is considered in more detail in a later clause of the present document.

6.2.2 Segment identifiers

Provisioned segments and segments recorded at runtime are identified by URIs as defined in IETF 2396 [13].

A URI can be a simple name or it can be a URL. Three URL schemes are allowed: the file: scheme, the ftp:scheme, and the http: scheme. The file: scheme is used for audio local to the Audio Server. The ftp:scheme is used for audio remote to the Audio Server. The http: scheme can be used for audio local to the Audio Server using the http://local host convention or audio remote to the Audio Server. All audio references that require parameters encoded in the URL (e.g. set selectors) shall use the http: scheme. Table 2 shows some of the possibilities.

Table 2: Example URIs

Reference to local audio (flat file):
S: pa(an=file://welcome)
Reference to local audio (flat file):
S: pa(an=file://12354)
Reference to local audio:
S: pa(an=file://audio/xyztel/welcome)
Reference to remote audio:
S: pa(an=http://audio/xyztel/welcome)

6.2.3 Segment life

Physical segments may be provisioned or they may be recorded during the course of a call. A physical segment recorded during the course of a call can be either transient or persistent. A transient physical segment lasts only for the life of the call during which it was recorded. A persistent physical segment lasts beyond the live of the call during which it was recorded.

6.2.4 Nested sets and sequences

Nested definition of both sets and sequences are allowed, i.e. it is legal to define a set of sets or a sequence of sequences. In addition, intermixing sets and sequences may be used to specify audio structures and it is possible to specify a set of sequences or a sequence containing one or more set elements. Direct or transitive definition of a set or segment in terms of itself is not allowed.

Nesting of sets and sequences should be restricted to two or three levels.

6.2.5 Sequence example

In the following example, a provisioner has provisioned one physical segment and two variable segments and has provisioned a sequence, http:/mysegment, which is an ordered list of the three segments. The sequence when played speaks the following: "Today's date is <weekday> <date>".

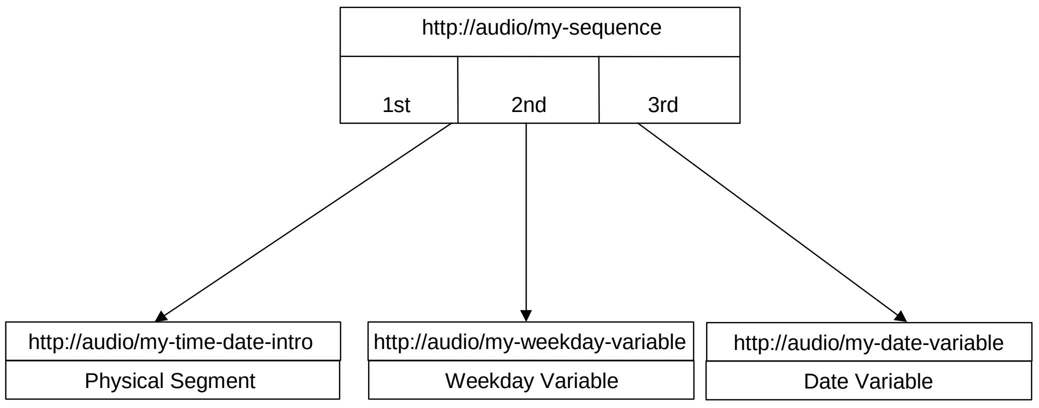
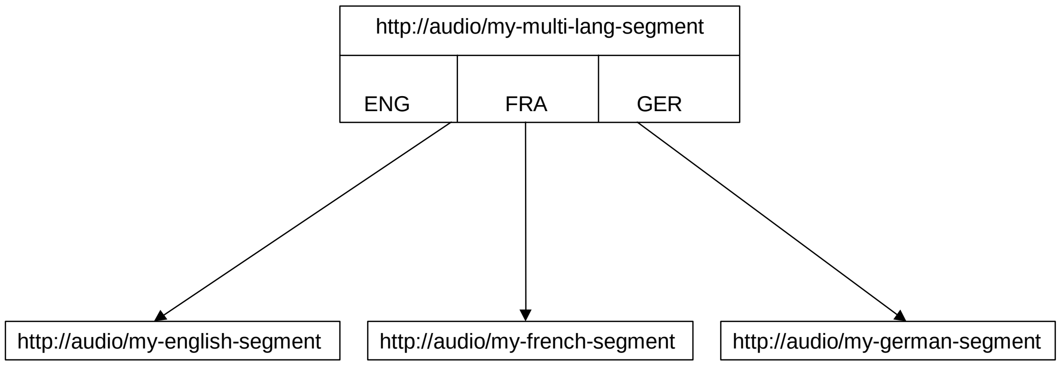

http://audio/my-sequence
1st
2nd
3rd
http://audio/my-time-date-intro
Physical Segment
http://audio/my-weekday-variable
Weekday Variable
http://audio/my-date-variable
Date Variable

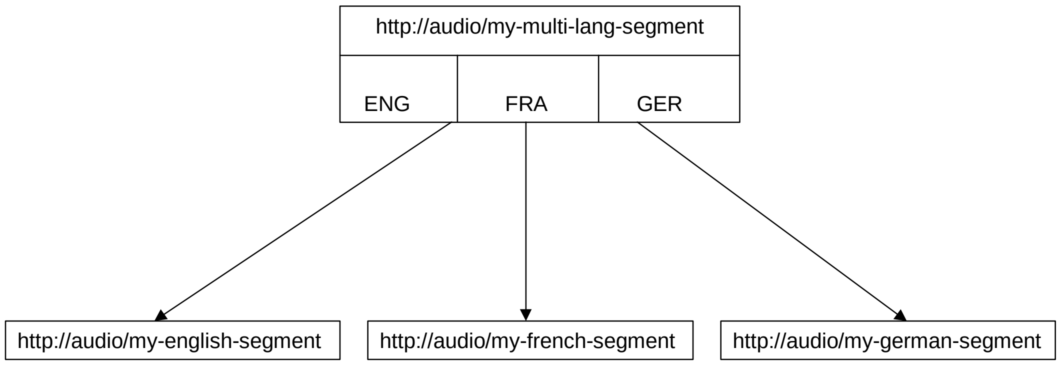
Figure 7: Sequence example

6.2.6 Set example

To support an application which plays a particular piece of audio in either English, French, or German, a provisioner could define a set with the pre-defined selector, "lang", and would use define three of the possible values for that selector, "eng", "fra", and "ger". The provisioner would provision three audio segments, one in each language, and would associate the English segment with the "eng" selector value, etc. The provisioner also could define a default value of the selector when no selector value is supplied, "eng" for instance. The entire set would be assigned a unique URI.

At runtime a reference to the set with the selector set to "fra" would result in the French version of the prompt being played. A reference to the set with no selector would result in the English version of the prompt being played since English has been set as the default selector value.


http://audio/my-multi-lang-segment
ENG
FRA
GER
http://audio/my-english-segment
http://audio/my-french-segment
http://audio/my-german-segment



http://audio/my-multi-lang-segment
ENG
FRA
GER
http://audio/my-english-segment
http://audio/my-french-segment
http://audio/my-german-segment

Figure 8: Set example

6.2.7 Set with nested sequence example

In this example, the provisioner has provisioned three physical segments, one in English, one in French, and one in German, and the provisioner has also provisioned three date variables. Using these six segments the provisioner has provisioned three sequences, each consisting of a physical segment followed by a date variable. Finally the provisioner has provisioned a set consisting of the three sequences and with language as the set selector.

At runtime a reference to the set with the selector set to "eng" and a variable value of "20001015" would result in the following being played in English: "Today's date is October 15th, 2000."

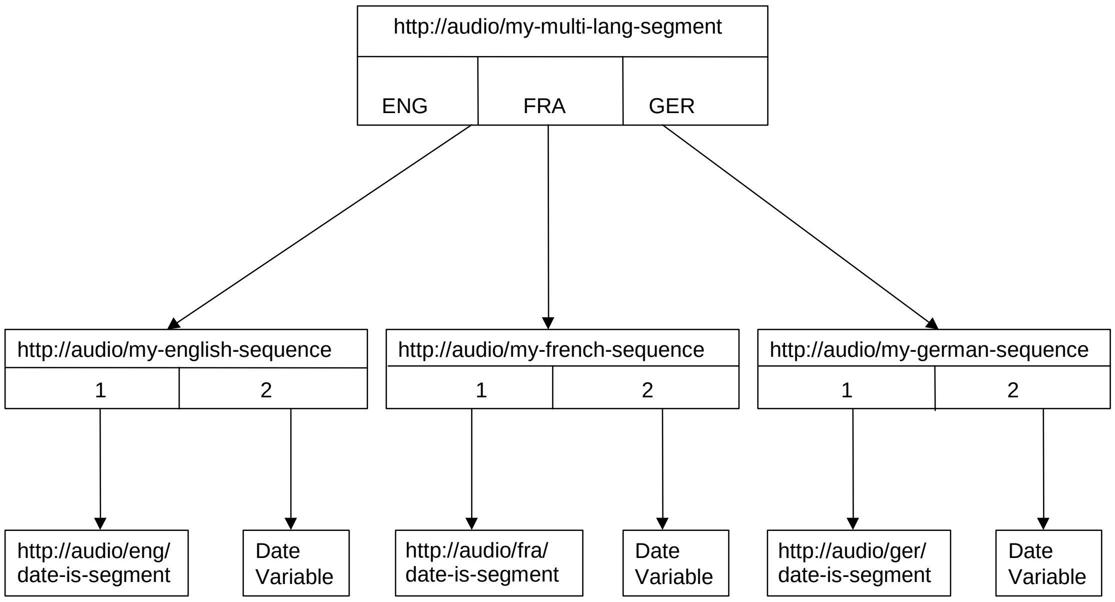

http://audio/my-multi-lang-segment
ENG
FRA
GER
http://audio/my-english-sequence
1
2
http://audio/my-french-sequence
1
2
http://audio/my-german-sequence
1
2
http://audio/eng/ date-is-segment
Date Variable
http://audio/fra/ date-is-segment
Date Variable
http://audio/ger/ date-is-segment
Date Variable

Figure 9: Set with nested sequence example

6.3 Base audio package

6.3.1 Abstract

This event package provides support for the standard IVR operations of PlayAnnouncement, PlayCollect, and PlayRecord. It supports direct references to simple audio as well as indirect references to simple and complex audio. It provides audio variables, control of audio interruptibility, digit buffer control, special key sequences, and support for reprompting during data collection.

Package Name: BAU

6.3.2 Events

Table 3: Events

Symbol	Definition	R	S	Duration
pa(parms)	PlayAnnouncement		TO	variable
pc(parms)	PlayCollect		TO	variable
pr(parms)	PlayRecord		TO	variable
ma(parms)	ManageAudio		BR	variable
oc	OperationComplete	x		
of(parms)	OperationFailed	x		

PlayAnnouncement: Plays an announcement in situations where there is no need for interaction with the user. Because there is no need to monitor the incoming media stream this event is an efficient mechanism for treatments, informational announcements, etc.

PlayCollect: Plays a prompt and collects DTMF digits entered by a user. If no digits are entered or an invalid digit pattern is entered, the user may be reprompted and given another chance to enter a correct pattern of digits. The following digits are supported: 0-9, *, and #. By default PlayCollect does not play an initial prompt, makes only one attempt to collect digits, and therefore functions as a simple Collect operation. Various special purpose keys, key sequences, and key sets can be defined for use during the PlayCollect operation.

PlayRecord: Plays a prompt and records user speech. If the user does not speak, the user may be reprompted and given another chance to record. By default PlayRecord does not play an initial prompt, makes only one attempt to record, and therefore functions as a simple Record operation. The call agent may specify a URI to be associated with the recording or the call agent may ask the Audio Server to allocate a URI and return it to the call agent as part of the OperationComplete event. Digits entered by the user during a recording that are not defined as special key sequences are ignored and become part of the recording.

ManageAudio: Performs audio management operations on persistent audio which typically not related to a current interaction with a user, e.g. "delete an audio segment".

OperationComplete: Detected upon the successful completion of a Play, PlayRecord, Play Collect, or ManageAudio signal.

OperationFailed: Detected upon the failure of a Play, PlayRecord, PlayCollect, or ManageAudio signal.

6.3.3 Signal interactions

If an audio package signal is active on an endpoint and another signal of the same type is applied, the two signals including parameters and parameter values will be compared. If the signals are identical, the signal in progress will be allowed to continue and the new signal will be discarded. Because of this behaviour the advanced audio package may not interoperate well with some other packages such as the line and trunk packages.

6.3.4 Parameters

The PlayAnnouncement, PlayRecord, and PlayCollect events may each be qualified by a string of parameters, most of which are optional. Where appropriate, parameters default to reasonable values. If a required parameter is not supplied an error is returned to the application.

These parameters are shown in table 4.

Table 4: Parameters

Symbol	Definition	pa	pc	pr	ma
an	announcement	O	F	F	F
ip	initial prompt	F	O	O	F
rp	reprompt	F	O	O	F
nd	no digits reprompt	F	O	F	F
ns	no speech reprompt	F	F	O	F
fa	failure announcement	F	O	O	F
sa	success announcement	F	O	O	F
off	offset	O	O	O	F
ni	non-interruptible play	F	O	O	F
it	iterations	O	F	F	F
iv	interval	O	F	F	F
du	duration	O	F	F	F
sp	speed	O	O	O	F
vl	volume	O	O	O	F
cb	clear digit buffer	F	O	O	F
dm	digit map	F	O	O	F
fdt	first digit timer	F	O	F	F
idt	inter digit timer	F	O	O	F
edt	extra digit timer	F	O	F	F
prt	prespeech timer	F	F	O	F
pst	postspeech timer	F	F	O	F
rlt	recording length timer	F	F	M	F
rsk	restart key	F	O	O	F
rik	reinput key	F	O	O	F
rtk	return key	F	O	O	F
na	number of attempts	F	O	O	F
ap	append	F	F	O	F
rid	recording id	F	F	M	F
rpa	record persistent audio	F	F	F	O
dpa	delete persistent audio	F	F	F	O
NOTE: O = Optional M = Mandatory F = Forbidden					

Announcement: An announcement to be played. Consists of one or more audio segments.

Initial Prompt: The initial announcement prompting the user to either enter DTMF digits or to speak. Consists of one or more audio segments. If not specified (the default), the event immediately begins digit collection or recording.

Reprompt: Played after the user has made an error such as entering an invalid digit pattern or not speaking. Consists of one or more audio segments. Defaults to the Initial Prompt.

No Digits Reprompt: Played after the user has failed to enter a valid digit pattern during a PlayCollect event. Consists of one or more audio segments. Defaults to the Reprompt.

No Speech Reprompt: Played after the user has failed to speak during a PlayRecord event. Consists of one or more audio segments. Defaults to the Reprompt.

Failure Announcement: Played when all data entry attempts have failed. Consists of one or more audio segments. No default.

Success Announcement: Played when data collection has succeeded. Consists of one or more audio segments. No default.

Offset: Specifies the offset into an announcement to start playing. Offset shall only be used with the initial prompt of the PlayCollect or PlayRecord events where the initial prompt is be a single physical segment. An offset shall be either positive or negative. A positive offset is the offset going forward from the beginning of the prompt. A negative offset is the offset going backwards from the end of the prompt. Offsets are specified in 10 millisecond units. Defaults to 0.

Offsets are useful when digit handling is done by the call agent, e.g. the user hits a DTMF key, the key is sent to the call agent, the call agent decides to ignore the key and tells the Audio Server to resume playing at the point of interrupt. Another application is to allow the user to skip back and forward in a physical segment.

NonInterruptible Play: If set to true, the initial prompt of the PlayCollect or PlayRecord event is not interruptible by either voice or digits. Defaults to false. Valid values are the text strings "true" and "false." Digits entered during a non-interruptible initial prompt are accumulated and are treated as they would if they had been entered during the second (collect or record) phase of the event.

Iterations: The maximum number of times an announcement is to be played. A value of minus one (-1) indicates the announcement is to be repeated forever. Defaults to one (1).

Interval: The interval of silence to be inserted between iterative plays. Specified in units of 100 ms. Defaults to 10 (one second).

Duration: The maximum amount of time to play and possibly replay an announcement. Takes precedence over iteration and interval. Specified in units of 100 ms. No default.

Speed: The relative playback speed of announcement specifiable as a positive or negative percentage of the original playback speed.

Volume: The relative playback volume of announcement specifiable as a positive or negative decibel variation from the original playback volume.

Clear Digit Buffer: If set to true, clears the digit buffer before playing the initial prompt. Defaults to false. Valid values are the text strings "true" and "false."

Digit Map: A digit map as specified in IETF 2705 [12], which specifies one or more digit patterns to be collected. Valid digits are 0-9, *, and #.

First Digit Timer: The amount of time allowed for the user to enter the first digit. The first digit starts after the announcements end. Specified in units of 100 ms. Defaults to 50 units (five seconds).

Inter Digit Timer: The amount of time allowed for the user to enter each subsequent digit. Specified units of 100 ms seconds. Defaults to 30 (three seconds).

Extra Digit Timer: The amount of time to wait for a user to enter a final digit once the maximum expected amount of digits have been entered. EDT Typically this timer is used to wait for a terminating key in applications where a specific key has been defined to terminate input. Specified in units of 100 ms. If not specified, this timer is not activated. If an extra digit is entered it is returned to the application along with the other collected digits.

The Extra Digit Timer can be used to implement a consistent human interface when collecting a variable number of digits where collection can be terminated by a Return Key, typically the # key. For example, suppose an application has asked for a minimum of three digits and a maximum of six. If the user consistently uses the # key to terminate collection following digit strings are acceptable: xxx#, xxxx#, xxxxx#, and xxxxxx. The inconsistency arises when the user enters six digits. Because the maximum number of digits have been entered the Audio Server returns the digits immediately without waiting for the # key. If the type ahead is allowed (the default Audio Server behaviour) and if user then enters the # key, the application has to decide whether the user meant the # key to terminate the six digits already collected or if the user meant to enter the # key to begin the next digit collection. The Extra Digit Timer tells the Audio Server to wait for an additional period of time after the maximum number of digits have been entered to see if the user is going to enter another key.

Prespeech Timer: The amount of time to wait for the user to initially speak. Specified in units of 100 ms. Defaults to 30 (three seconds).

Postspeech Timer: The amount of silence necessary after the end of the last speech segment for the recording to be considered complete. Specified in units of 100 ms. Defaults to 50 units (five seconds).

Recording Length Timer: The maximum allowable length of the recording, not including pre or post speech silence. Specified in units of 100 ms. This parameter is mandatory for the PlayRecord signal. A value of -1 (minus one) means there is no limit to recording length. In this case the recording is open ended, and it is up to the application to manage the storage resources for recordings.

Restart Key: Defines a digit map that, if matched, has the following action: discard any digits collected or recording in progress, replay the prompt, and resume digit collection or recording. No default.

The use of this key does not constitute an attempt to enter user input (i.e. it does not count against the number of attempts specified by the Number Of Attempts parameter). Restart Keys are handled locally by the Audio Server and are not returned to the call agent. During a recording, all digits except for the restart, reinput, and return keys (if defined) are ignored and become part of the recording.

Reinput Key: Defines a digit map that, if matched, has the following action: discard any digits collected or recordings in progress and resume digit collection or recording. No default.

The use of this key does not constitute an attempt to enter user input (i.e. it does not count against the number of attempts specified by the Number Of Attempts parameter). Reinput keys are handled locally by the Audio Server and are not returned to the call agent. During a recording, all digits except for the restart, reinput, and return keys (if defined) are ignored and become part of the recording.

Return Key: Defines a digit map that, if matched, has the following action: stop digit collection or recording. If the return key is hit during a PlayCollect event, all keys collected are returned to the call agent. If the return key is hit during a PlayRecord event, the recording is saved allo keys collected are returned, and a Recording ID is returned if appropriate. (See definition of RecordingID for details) No default.

Number Of Attempts: The number of attempts the user is allowed to enter a valid digit pattern or to make a recording. Defaults to 1. Also used as a return parameter to indicate the number of attempts the user made.

Append: If set to true, the audio recording will append to any existing content in the Recording ID. It may not be used with wildcarded Recording Ids. Valid values are "true" and "false".

Recording ID: A URI to be assigned to the physical segment which is to be recorded by the PlayRecord event. If this parameter is set to the ANY wildcard, "$", the Audio Server will allocate the URI, associate it with the newly recorded segment, and return it to the call agent with the OperationComplete event.

Record Persistent Audio: If set to true, the recording that is made is persistent instead of temporary. Defaults to false. Valid values are the text strings "true" and "false."

Delete Persistent Audio: Indicates that the specified persistent audio segment is to be deleted. This parameter is carried by the PlayRecord event, although nothing is either played or recorded in this case.

6.3.5 Type-ahead

The Audio Server supports type-ahead by default. Type-ahead is not supported for the Play event because by definition no digit collection is done during this event. Type-ahead can be turned off for all prompts associated with a PlayCollect or PlayRecord event by setting the Clear Digit Buffer parameter.

6.3.6 Return parameters

Each event has an associated set of possible return parameters which are returned with either the OperationComplete or OperationFailed events. These parameters are listed in table 5:

Table 5: Return Parameters

Symbol	Definition	pl	pc	pr	ma
ap	amount played	F	C	C	F
dc	digits collected	F	O	O	F
na	number of attempts	F	M	M	F
rc	return code	O	O	O	O
rl	recording length	F	F	M	F
rid	recording id	F	F	O	F
NOTE: O = Optional, M = Mandatory, F = Forbidden, C = Conditional (see expanded definition)					

Amount Played: The length played of an initial prompt if the prompt was interrupted, in 10 ms units. This parameter is mandatory if the prompt was interrupted, and forbidden otherwise.

Digits Collected: If returned with an oc event, this parameter contains the DTMF digits that were collected during a PlayCollect operation. If returned with an of event, this parameter contains the DTMF digits that were collected during an unsuccessful PlayCollect or PlayRecord operation up until the point of failure.

Number Of Attempts: The number of attempts the user actually needed to enter a valid digit pattern or to make a recording. Defaults to 1. Also used as an input parameter to specify the number of attempts the user will be allowed to enter a valid digit pattern or make a recording. This parameter is returned only if an na parameter was specified on the PlayCollect or PlayRecord.

Recording Length: The length of the recording, not including pre or post speech silence. Specified in units of 100 ms. This parameter is mandatory for the PlayRecord signal. In the case where the append operation was used, this is the length of the new recording, not the total length.

Recording ID: A URI assigned to physical segment recorded by the PlayRecord operation. This parameter is returned only if the RecordingID parameter to the PlayRecord event has been set to the ANY wildcard, "$". If this is the case the Audio Server allocates a unique URI, associates it with the newly recorded segment, and returns it to the call agent.

Return Code: A return code giving the final status of the operation.

Table 6: Return codes

Return Code	Meaning
600	Illegal syntax
601	Unknown segment ID
602	Variable type not supported
603	Variable subtype not supported
604	Invalid variable name
605	Variable value out of range
606	Inconsistent variable specification
607	Extra sequence data
608	Missing sequence data
609	Mismatch between play specification and provisioned data
610	Delete audio error
611	Unable to record temporary audio
612	Unable to delete temporary audio
613	Unable to record persistent audio
614	Unable to delete persistent audio
615	Unable to override non-existent segment id
616	Unable to remove override from non-existent segment id
617	Provisioning error
618	Hardware failure
619	Unspecified failure
620	No digits
621	No speech
622	Spoke too long
623	Digit map not matched
624	Max attempts exceeded
625	No free segment ids
626	Required parameter not set
627	Inconsistent parameter set
628	Value out of range
629	Invalid offset
630	Invalid digit map

EXAMPLES:

The PlayAnnouncement event completed successfully. Note that no return code is necessary:

O: BAU/oc

The PlayAnnouncement event failed the parameters supplied were inconsistent:

O: BAU/of(rc=633)

The PlayCollect event completed successfully on the user's second attempt when the user entered the digits 04375182:

O: BAU/oc(na=2 dc=04375182)

The PlayRecord event was successful on the user's first attempt; the id of the recording made by the user is 983:

O: BAU/oc(na=1 ri=983)

The PlayRecord event was successful on the user's first attempt; the id of the recording made by the user is 983 and the duration was 27,5 seconds:

O: BAU/oc(na=1 ri=983 rl=275)

6.3.7 Segment descriptors

Segment descriptors are used with the an, ip, rp, nd, ns, fa, and sa parameters (see table 4) to define the segments that make up an announcement. There are two kinds of segment descriptors.

Table 7: Segment Descriptors

Symbol	Definition
<URI>	segment identifier
vb	variable

Segment Identifier: A URI identifying a provisioned entity, i.e. a physical segment, sequence, or variable.

Variable: Specifies a voice variable by type, subtype, and value, and used when the application specifies a variable on the fly as opposed to referencing a provisioned variable. Does not apply to provisioned variables. Variables are more completely defined in a subsequent clause of the present document.

6.3.8 Variable syntax

The syntax supports two kinds of variables. Embedded variables are variables that have been provisioned as part of an audio segment. At runtime the call agent references the segment and specifies a value for the variable. Typically embedded variables are provisioned along with recorded speech, e.g. "A representative will be with you in approximately 5 minutes. If you would prefer to leave a voice message, press 1 now." where the variable is the number of minutes. Standalone variables are variables that are not provisioned and therefore shall be completely specified on the fly by the call agent. Variables are specified by the following parameters: type, subtype, and value. Variable types include Date, Money, Number, Time, etc. Subtype is a refinement of type. For example the variable type Money might have an associated range of subtypes such as Euro, Kroner, Pound, etc. Not all variables require a subtype, and for these variables the subtype parameter should be set to null.

For embedded variables, the type and subtype shall be provisioned. The value may be provisioned. If it is not provisioned it shall be specified as part of the variable reference. In a list of segments, an embedded variable value specification applies only to the segment that directly precedes it. If a segment has multiple embedded variables, the values shall be given in the order in which the variables are encountered when the segment is played.

EXAMPLES:

Standalone variable: S: pa(an=vb(mny,usd,1153))

Embedded variable: S: pa(an=file://ann1<1153>)

Not all variables, such as the date variable shown in the next example, require a subtype. In that case, the subtype is encoded with the value "null":

S: pa(an=vb(dat,null,101598))

In some cases it may be desirable to play an announcement that contains an embedded variable without playing the variable itself. To do this a single "null" is provided for the value:

S: pa(an=file://ann1<null>)

6.3.9 Variable definitions

Variable types and subtypes are specified in table 8.

Table 8: Variable types and subtypes

Type	Subtype	Definition
dat	mdy, dmy, etc.	Date
	mdy	Month-Day-Year
	dym	Day-Year-Month
dig	gen, edn	Digits
	gen	Generic
	edn	European DN
dur		Duration
mth		Month
mny	<ISO 4217 [10] three letter codes>	Money
num	crd, ord	Number
	crd	Cardinal
	ord	Ordinal
sil		Silence
str		String
tme	t12, t24	Time
	t12	Twelve hour format
	t24	Twenty four hour format
wkd		Weekday

Date: Speaks a date specified as YYYYMMDD (per ISO 8601 [11]). If the subtype is Month-Date-Year "20001015", for example would be spoken as "October Fifteenth Two Thousand." If the subtype is Date-Month_Year the same date would be spoken as "Fifteen October Two Thousand." Date subtypes may be extended as needed as long as they are patterned after the existing subtypes (i.e. they shall be a three letter combination of the letters m, d and y).

Digits: Speaks a string of digits one at a time. The possibility of subtypes in which pauses are inserted at various point in the spoken digit string is also allowed.

Duration: Duration is specified in seconds and is spoken in one or more units of time as appropriate, e.g. "3661" is spoken as "One hour, one minute, and one second", "3360" is spoken as "One hour and one minute", and "3600" is spoken as "One minute."

Money: Money is specified in the smallest units of a given currency and is spoken in one or more units of currency as appropriate, e.g. "1,10" in European euros would be spoken "one euro and ten cents." The three letter codes defined in ISO 4217 [10] Currency And Funds Code List, are used to specify the currency subtype. A small excerpt from ISO 4217 [10] follows.

Table 9: Sample currency codes

Code	Currency	Entity
EUR	Euro	Europe
GBP	British Pound	UK
DKK	Danish Kroner	Denmark

Money can be specified in negative or positive units of currency. In the above example "-"-1,10" would be spoken as "minus one euro and ten cents."

Month: Speaks the specified month, e.g. "10" is spoken as "October." Specification is in MM format with "01" denoting January, "02" denoting February, etc.

Number: Speaks a number in cardinal form or in ordinal form. For example, "100" is spoken as "one hundred" in cardinal form and "one hundredth" in ordinal form. Cardinal numbers can be specified as negative or positive.

Silence: Plays a specified period of silence. Specification is in 100 millisecond units.

String: Speaks each character of a string, e.g. "a34bc" is spoken "A, three, four, b, c." Valid characters are a-z, A-Z, 0-9, #, and *.

Time: Speaks a time in either twelve-hour format or twenty-four hour format depending on the specified subtype. For example "1700" is spoken as "Five p.m." in twelve hour format or as "Seventeen hundred hours" in twenty-four hour format. Specification is in HHMM format per ISO 8601 [11], International Data and Time Notation.

Tone: The tone variable is used to cause the audio player to generate a defined tone from any other standard package as part of the sequence of audio segments. If the package referenced in the request is not known to (or not supported by) the audio player an error code of *603 Variable subtype not supported* shall be returned.

Caution: Only tones of known duration should be used.

EXAMPLES:

vb(ton,L,ci(1942,3036619100,CableLabs))

vb(ton,D,2) -*or*- vb(ton,L,2)

vb(ton,SL,(D/1,D/5,D/7))

Weekday: Speaks the day of the week, e.g. "Monday." Weekdays are specified as single digits, with "1" denoting Sunday, "2" denoting Monday, etc.

6.3.10 Timers

Four timers are defined in this package:

- First digit timer (FDT).
- Interdigit timer (IDT).
- Interdigit critical timer (ICT).
- Extra digit timer (EDT).

Consistent implementation of the interaction between timers is important for applications that use audio servers that meet the present document. The following guidelines are strongly recommended:

1) There is no need for more than one of the timers to run at any given time in processing of a digit map.

2) The first digit timer (FDT) will be started on receipt of the collection request if no initial prompt is present, at completion of the playing of the initial prompt, and on completion of any reprompts. If a digit is collected during the playing of the initial prompt or reprompt, the FDT is not started.

3) The interdigit timer (IDT) will be started when the end of a tone is detected if there are no possible matches and there are still possible matches. The IDT will not run if the collected tone (digit) completes a match or if the collected tone (digit) completes a match except for a terminal "T".

4) The interdigit critical timer (ICT) will be started when the digit map includes a terminal "T" and the matched string is a subset of a longer string. If an additional digit/tone is detected during the running of the ICT, it is examined to determine whether it creates a possible match (or partial match) of an alternative in the digit map. Thus in the digit map "`123T|12345`", the "T" represents the running of the ICT. If a "4" arrives before ICT expires, the digit map matching algorithm selects option two and continues the process.

5) The extra digit timer (EDT) runs after a match has been completed, even if the completion of the match required another timer to run (e.g, the ICT). Any digit that is detected while the EDT is running is returned in the observed events string, and the detection of that event will result in a OF response with RC=623 and the DC= parameter showing all digits detected prior to and during the EDT period. The EDT does not run if the digit map-matching algorithm determines an error condition exists (no match possible, no digits entered, etc.).

Some examples with commentary:

dm=123|1234

Option two (1234) cannot be matched - the algorithm will return immediately on detection of 123. If specified, EDT may run after the match, but the 4, if entered, will be ignored.

```
dm=123T|1234
```

The ICT will run after the 3 is entered. If the timer expires, the match (123T) is returned. If a 4 is detected before the expiration of ICT, the match (1234) is returned. If a different digit is detected, error processing (return, reprompt, as appropriate) is started.

6.3.11 Examples

This clause presents a number of syntax examples.

Play an announcement that consists of a single segment:

```
S: pa(an=file://12333)
```

Play an announcement that consists of multiple segments:

```
S: pa(an=file://ann798,file://ann300,file://ann4747)
```

Play an announcement that consists of a recording followed by three seconds of silence followed by a standalone voice variable:

```
S:pa(an=file://ann357,vb(sil,null,30),vb(my,usd,3999))
```

Play an announcement with an embedded variable. If the separate segments of the previous announcement were provisioned as a sequence with a segment id of ann43321, the following would be exactly equivalent to the previous example:

```
S: pa(an=file://ann43321<3999>)
```

Play an announcement with two embedded variables:

```
S: pa(an=http://jackstraw/audio/xyztel/hello

<3999,10151998>)
```

Play a prompt and collect a single digit. If need be, play a reprompt, a no digits prompt, and a success or failure announcement. Give the user three attempts to enter a digit:

```
S: pc(ip=file://ann27 rp=file://ann19 nd=file://ann102

fa=file://ann8 sa=file://ann777 na=file://ann31

dm=x)
```

Play a prompt and collect a single digit. If the user does not enter a digit replay the initial prompt. Give the user three attempts to enter a digit:

```
S: pc(ip=file://audio/ann77775 na=3 dm=x)
```

Play a prompt and record voice. If the user does not speak play a no speech prompt. Give the user two attempts to record:

```
S: pr(ip=http://brenda/audio/ann070500

ns=http:/althea/audio/no-speech na=2)
```

Play an announcement at ninety percent of its original speed and five decibels softer than its original volume. Play the announcement three times with two seconds of silence between plays.

```
S: pa(an=file://ann276 sp=90 vl=-5 it=3 iv=20)
```

Give the user two attempts to enter a three-digit pattern. Clear the digit buffer before playing the prompt.

```
S: pc(ip=file://438975 cb=true dm=xxx na=2)
```

Give the user three attempts to enter a three-digit pattern. If the user enters one digit or two digits on the first or second attempts a reprompt is played. If the user enters no digits on the first or second attempts a no digits reprompt is played. If all three attempts fail, a failure announcement is played. If one of the attempts is successful, a success announcement is played and the collected digits are returned to the call agent.

```
S: pc(ip=file://ann493 rp=5 nd=409 fa=file://ann923

sa=file://ann18337 dm=xxx)
```

Give the user three chances to enter an 11 digit number that begins with 0 or 1. If the user makes a mistake while entering digits, he can press the * key to discard any digits already collected, replay the prompt, and resume collection.

```
S: pc(ip=http://stella/blue/audio/ann5684

dm=0xxxxxxxxxx|1xxxxxxxxxx rsk=* na=3)
```

Give the user two chances to make a recording. After playing the prompt, wait 5 seconds for the user to speak, otherwise replay the initial prompt and try again. If the user does speak, wait for seven seconds after speech stops to make sure the user is finished. If the recording is successful, return a reference to the recording to the call agent.

```
S: pr(ip=file://ann432 prt=50 pst=70 na=2)
```

6.4 Advanced audio package

6.4.1 Abstract

The advanced audio package extends the base audio package by adding the set capability which the user can use to create an arbitrary number of user defined qualifiers to be used in resolving complex audio structures. For example, the user could define qualifiers for any or all of the following: language, accent, audio file format, gender, speaker, or customer.

Package Name: AAU.

6.4.2 Sets

A set is a provisioned collection of semantically related audio segments with an associated selector. Each set is assigned a unique URI. A set can contain physical segments, sequences, other sets, or variables. At runtime the value of the selector is used to determine which element of the set is played.

Individual selector types are not defined in the syntax (except for the pre-defined language selector, "lang") and are instead defined by the provisioner. A provisioner could define one or more of the following selector types: language, accent, gender, accent, customer, or day of the week. For each selector type, the provisioner shall define a range of valid values. The provisioner may also choose to define a default value. At runtime if a selector value is not supplied the default value is used.

6.4.3 Selectors

Selector types, except for the predefined "lang" (language) selector, are defined by the user. For each selector type, the user shall define a range of values that the selector can assume.

Selectors apply to individual audio segments. If an event specifies multiple segments, each segment may have its own set of selectors. If selectors are not specified for an audio segment, provisioned defaults are used.

For example, if the user defines a selector of type "phaseofthemoon", he might also define the legal values for that selector to be "new", "half", "full", "harvest", and "blue". For the selector to actually work at runtime, audio associated with each of the selector values shall be provisioned.

The three letter codes defined in ISO 639-2 [9], shall be used as values for user defined language selectors. For languages that have both a bibliographic and a terminology code, both codes should be supported. The following examples are taken from ISO 639-2 [9].

Table 10: Sample language codes

Code	Language
eng	English
fra	French
ger	German

Selectors are applied to variables only after the variable has been resolved. For instance if a date variable resolved to "October 15th, 1998" the voice with which the variable is spoken could resolve to either male or female if a gender selector had been defined.

Selectors are encoded as parameters to the URI segment id. If the URI refers to a physical segment on a node other than the Audio Server, to fetch the audio from the remote node the URI shall contain the information necessary for this node to resolve the URI to a specific physical segment. This does not imply that the remote node needs the same capability as the Audio Server to resolve complex audio references. The remote node could for example use a simple scheme such as encoding the hierarchical directory path to the physical in the URI.

6.4.4 Selector encoding

Provisioned segments and segments recorded at runtime are identified by URIs as defined in IETF 2396 [13], Uniform Resource Identifiers: Generic Syntax.

A URI can be a simple name or it can be a URL. If a URL refers to audio stored on a node other than the Audio Server it shall contain all the information necessary to resolve the URL to a physical segment. If the URL refers to a set, the selector types and values necessary to resolve the URL to a physical segment shall be encoded in the query field of the URL. URLs for audio local to the Audio Server should use the file: scheme. URLs for audio remote to the Audio Server should use the http: scheme. Table 11 shows some of the possibilities.

Table 11: Example URIs

Reference to local audio (set):
S: pa(an=http://localhost/audio/xyztel/welcome?lang=eng&gender=female)
Reference to remote audio (set):
S: pa(an=http://audio/xyztel/welcome?lang=eng&gender=female)

6.4.5 Variable order

When a provisioned segment containing more than one variable is referenced at runtime, the variable values shall be supplied in the order in which they occur in the provisioned segment. This principle extends to sets. If the elements of a set contain more than one variable then for all elements of the set the variables shall occur in the same order. Sets with elements containing variables that do not appear in the same order are not supported.

6.4.6 Overrides

A provisioned physical segment may be replaced (or overridden) by a persistent physical segment. The URI of the provisioned physical segment will then resolve to the persistent physical segment. The overriding persistent audio can subsequently be deleted and the original provisioned audio can be restored.

A provisioned physical segment may be overridden more than once. In this case, the URI of the provisioned physical segment refers to the latest overriding physical segment. When the overriding physical segment is deleted, the original provisioned physical segment is restored, even if the segment has been overridden multiple times.

Segment override could be used for a feature where a standard greeting is played to all customers calling a retail store. Occasionally the store manager may want to call a special number and record a temporary greeting that overrides the standard greeting, for instance a greeting that announces a sale or maybe a seasonal greeting of some kind. When the greeting is no longer wanted, the manager can call the special number, cancel the temporary greeting, and restore the standard greeting.

6.4.7 Parameters

Table 12: Parameters

Symbol	Definition	pa	pc	pr	ma
oa	override persistent audio	F	F	F	O
ra	restore persistent audio	F	F	F	O
NOTE: O = Optional M = Mandatory F = Forbidden					

Override Persistent Audio: The id of the segment to be overridden and the id of the overriding segment.

Restore Persistent Audio: The id of the segment to be restored.

6.4.8 Return codes

The following return codes are defined for the advanced audio package:

Table 13: Return codes

Return Code	Meaning
650	Bad selector type
651	Bad selector value
652	Missing selector
653	Missing selector value
654	Wrong number of selector
655	Remove override error
656	Override error
657	Unable to override non-existent segment id
658	Unable to remove override from non-existent segment id

6.4.9 Examples

This clause presents a number of examples of how sets and selectors are used.

Play an announcement in English.

S: pa(an=file://audio/xyztel/hello?lang=eng)

Play an announcement in a Danish, female voice with a Cajun accent.

S: pa(an=file://audio/xyztel/hello?lang=dan&

gender=female&accent=cajun)

Play the first part of an announcement in English, the second part in the default language, and the third part in French.

S: pa(an=file://ann1?lang=eng,file://ann2,

file://ann2?lang=fra)

Play an announcement with an embedded variable in English (the embedded variable is also played in English):

S: pa(an=file://ann4?lang=eng<101599>)

6.5 Formal syntax description

This description uses ABNF (IETF 2234 [14]) to formally describe the syntax of the Basic Audio Package and the Advanced Audio Package. The two packages have the same syntax except for the encoding of selector types and selector values in the query field of the URI and the persistent audio override capabilities. See IETF 2396 [13] for the syntax of encoding parameter value pairs in the query field of the URL.

```
AudPkgEvent = PlayAnnouncement / PlayCollect / PlayRecord / ManageAudio / OperationComplete / OperationFailed

PlayAnnouncement = [ AudioPkgToken SLASH ] PlayAnnToken

LPAREN PlayAnnParmList RPAREN

PlayCollect = [ AudioPkgToken SLASH ] PlayColToken

LPAREN [ PlayColParmList ] RPAREN

PlayRecord = [ AudioPkgToken SLASH ] PlayRecToken

LPAREN [ PlayRecParmList ] RPAREN

ManageAudio = [AudioPkgToken SLASH] ManageAudToken LPAREN ManageAudParmList RPAREN

OperationComplete = [ AudioPkgToken SLASH ] OpCompleteToken

LPAREN [OpCompleteParmList ] RPAREN

OperationFailed = [ AudioPkgToken SLASH ] OpFailedToken

LPAREN ReturnCodeParm RPAREN

PlayAnnParmList = PlayAnnParm *( WSP PlayAnnParm )

PlayColParmList = PlayColParm *( WSP PlayColParm )

PlayRecParmList = PlayRecParm *( WSP PlayRecParm )

ManageAudParmList = ManageAudParm *( WSP ManageAudParm )

OpCompleteParmList = OpCompleteParm *( WSP OpCompleteParm )

PlayAnnParm = ( AnnouncementParm / IterationsParm / IntervalParm /

DurationParm / SpeedParm / VolumeParm )

PlayColParm = ( InitPromptParm / RepromptParm / NoDigitsParm / FailAnnParm /

SuccessAnnParm / NoInterruptParm / SpeedParm / VolumeParm /

ClearBufferParm / DigitMapParm / FirstDigitParm / InterDigitParm /

ExtraDigitParm / RestartKeyParm / ReinputKeyParm /

ReturnKeyParm / NumAttemptsParm )

PlayRecParm = ( InitPromptParm / RepromptParm / NoSpeechParm / FailAnnParm /

SuccessAnnParm / NoInterruptParm / SpeedParm / VolumeParm /

ClearBufferParm / PreSpeechParm / PostSpeechParm /

RecordLenTimerParm / RestartKeyParm / ReinputKeyParm /

ReturnKeyParm / NumAttemptsParm )

ManageAudParm = (RecPersistParm / DeletePersistParm / OverrideAudioParm / RestoreAudioParm)

OpCompleteParm = ( NumAttemptsParm / AmtPlayedParm / DigitsColParm

RecordingIdParm / ReturnCodeParm / RecordLenParm)

AnnouncementParm = AnParmToken EQUALS Segmentlist

InitPromptParm = IpParmToken EQUALS Segmentlist

RepromptParm = RpParmToken EQUALS Segmentlist
```

NoDigitsParm = NdParmToken EQUALS Segmentlist

NoSpeechParm = NsParmToken EQUALS Segmentlist

FailAnnParm = FaParmToken EQUALS Segmentlist

SuccessAnnParm = SaParmToken EQUALS Segmentlist

OffsetParm = OffParmToken EQUALS OPTSIGNEDINT

DurationParm = DuParmToken EQUALS NUMBER

IterationsParm = ItParmToken EQUALS (NUMBER / MINUSONE)

IntervalParm = IvParmToken EQUALS NUMBER

SpeedParm = SpParmToken EQUALS SIGNEDINT

VolumeParm = VlParmToken EQUALS SIGNEDINT

NoInterruptParm = NiParmToken EQUALS BOOLSTR

ClearBufferParm = CbParmToken EQUALS BOOLSTR

DigitMapParm = DmParmToken EQUALS DigitMap

DigitMap = <defined in IETF 2705 [12]>

FirstDigitParm = FdtParmToken EQUALS NUMBER

InterDigitParm = IdtParmToken EQUALS NUMBER

ExtraDigitParm = EdtParmToken EQUALS NUMBER

PreSpeechParm = PrtParmToken EQUALS NUMBER

PostSpeechParm = PstParmToken EQUALS NUMBER

RecordLenParm = RlParmToken EQUALS NUMBER

RecordLenTimerParm = RltParmToken EQUALS NUMBER

RestartKeyParm = RskParmToken EQUALS DigitMap

ReinputKeyParm = RikParmToken EQUALS DigitMap

ReturnKeyParm = RtkParmToken EQUALS DigitMap

RecPersistParm = RpaParmToken EQUALS BOOLSTR

DeletePersistParm = DpaParmToken EQUALS SegmentId

OverrideAudioParm = OaParmToken EQUALS OverridenSegId OverridingSegId

OverridenSegId = SegmentId

OverridingSegId = SegmentId

RestoreAudioParm = RaParmToken EQUALS SegmentId

NumAttemptsParm = NaParmToken EQUALS NUMBER

AmtPlayedParm = ApParmToken EQUALS NUMBER

DigitsColParm = DcParmToken EQUALS KeySequence

RecordingIdParm = RidParmToken EQUALS UniversalResourceIdentifier

ReturnCodeParm = RcParmToken EQUALS 3*3(DIGIT)

```
KeyPadKey = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" / "*" / "#"

KeySequence = 1*64(KeyPadKey)

KeySet = 1*11(KeyPadKey)

Segmentlist = SegmentDescriptor *( COMMA SegmentDescriptor )

SegmentDescriptor = SegmentId [ EmbedVarList ] / VariableSeg

SegmentId = UniversalResourceIdentifier

UniversalResourceIdentifier = <defined in IETF 2396 [13]>

VariableSeg = VariableSegToken LPAREN FullSpecVar RPAREN

EmbedVarList = LANGLE NAME *( COMMA NAME ) RANGLE

FullSpecVar = ( DateVariable / DigitsVariable / DurationVariable /

MonthVariable / MoneyVariable / NumberVariable /

SilenceVariable / StringVariable / TextVariable /

TimeVariable / WeekdayVariable )

DateVariable = DateVarToken COMMA NullStrToken COMMA Date

Date = 8*8(DIGIT)

DigitsVariable = DigitsVarToken COMMA (EuropeanDnToken /

GenericDigitsToken) COMMA NUMBER

DurationVariable = DurationVarToken COMMA NullStrToken COMMA NUMBER

MoneyVariable = MoneyVarToken COMMA 3*3(ALPHA) COMMA OPTSIGNEDINT

MonthVariable = MonthVarToken COMMA NullStrToken COMMA Month

Month = "01" / "02" / "03" / "04" / "05" / "06" / "07" / "08" / "09" / "10" / "11" / "12"

NumberVariable =

(NumberVarToken COMMA CardinalNumberToken COMMA OPTSIGNEDINT) /

(NumberVarToken COMMA OrdinalNumberToken COMMA NUMBER)

SilenceVariable = SilenceVarToken COMMA NullStrToken COMMA NUMBER

StringVariable = StringVarToken COMMA NullStrToken COMMA *(KeyPadKey)

SilenceVariable = SilenceVarToken COMMA NullStrToken COMMA NUMBER

StringVariable = StringVarToken COMMA NullStrToken COMMA *(KeyPadKey)

TimeVariable = TimeVarToken COMMA (TwelveHourFormatToken /

TwentyFourHourFormatToken) COMMA 4*4(DIGIT)

WeekdayVariable = WeekdayVarToken COMMA NullStrToken COMMA NAME

AudioPkgToken = BaseAudPkgToken / AdvAudPkgToken

BaseAudPkgToken = "BAU"

AdvAudPkgToken = "AAU"

PlayAnnToken = "pa"
```

```
PlayColToken = "pc"

PlayRecToken = "pr"

ManageAudToken = "ma"

OpCompleteToken = "oc"

OpFailedToken = "of"

VariableSegToken = "vb"

AnParmToken = "an"

IpParmToken = "ip"

RpParmToken = "rp"

NdParmToken = "nd"

NsParmToken = "ns"

FaParmToken = "fa"

SaParmToken = "sa"

OffParmToken = "off"

NiParmToken = "ni"

ItParmToken = "it"

IvParmToken = "iv"

DuParmToken = "du"

SpParmToken = "sp"

VlParmToken = "vl"

CbParmToken = "cb"

DmParmToken = "dm"

FdtParmToken = "fdt"

IdtParmToken = "idt"

EdtParmToken = "edt"

PrtParmToken = "prt"

PstParmToken = "pst"

RltParmToken = "rlt"

RlParmToken = "rl"

RskParmToken = "rsk"

RikParmToken = "rik"

RtkParmToken = "rtk"

RpaParmToken = "rpa"

DpaParmToken = "dpa"

OaParmToken = "oa"
```

```
RaParmToken = "ra"

ApParmToken = "ap"

DcParmToken = "dc"

NaParmToken = "na"

RcParmToken = "rc"

RidParmToken = "rid"

DateVarToken = "dat"

DigitsVarToken = "dig"

DurationVarToken = "dur"

DayYrMonthToken = "dym"

MonthDayYrToken = "mdy"

MoneyVarToken = "mny"

MonthVarToken = "mth"

NumberVarToken = "num"

SilenceVarToken = "sil"

StringVarToken = "str"

TimeVarToken = "tme"

GenericDigitsToken = "gen"

EuropeanDnSToken = "edn"

CardinalNumberToken = "crd"

OrdinalNumberToken = "ord"

TwelveHourFormatToken = "t12"

TwentyFourHourFormatToken = "t24"

WeekdayVarToken = "wkd"

NullStrToken = "null"

BOOLSTR = "true" / "false"

NAMECHAR = ALPHA / DIGIT / "_" / "-"

NAME = 1*64(NAMECHAR)

NUMBER = DIGIT *31(DIGIT)

SIGNEDINT = ("+" / "-") DIGIT *31(DIGIT)

OPTSIGNEDINT = ["+" / "-"] DIGIT *31(DIGIT)

MINUSONE = "-1"

EQUALS = "="

COMMA = ","

LSQUARE = "["
```

RSQUARE = "]"

LANGLE = "<"

RANGLE = ">"

LPAREN = "("

RPAREN = ")"

SLASH = "/"

WSP = SP / HTAB

Annex A (informative): Call flow for network announcement

This clause provides an example call flow where a caller (MTA-o) invokes the "Last Number Redial" feature to determine the phone number of the dialing party (MTA-t). An Audio Server is used to play an announcement to the caller containing the previous caller's number and to present the option to the caller for completing a return call to MTA-t. It should be noted that this call flow, although a valid one, is merely an example that may or may not be used in practice.

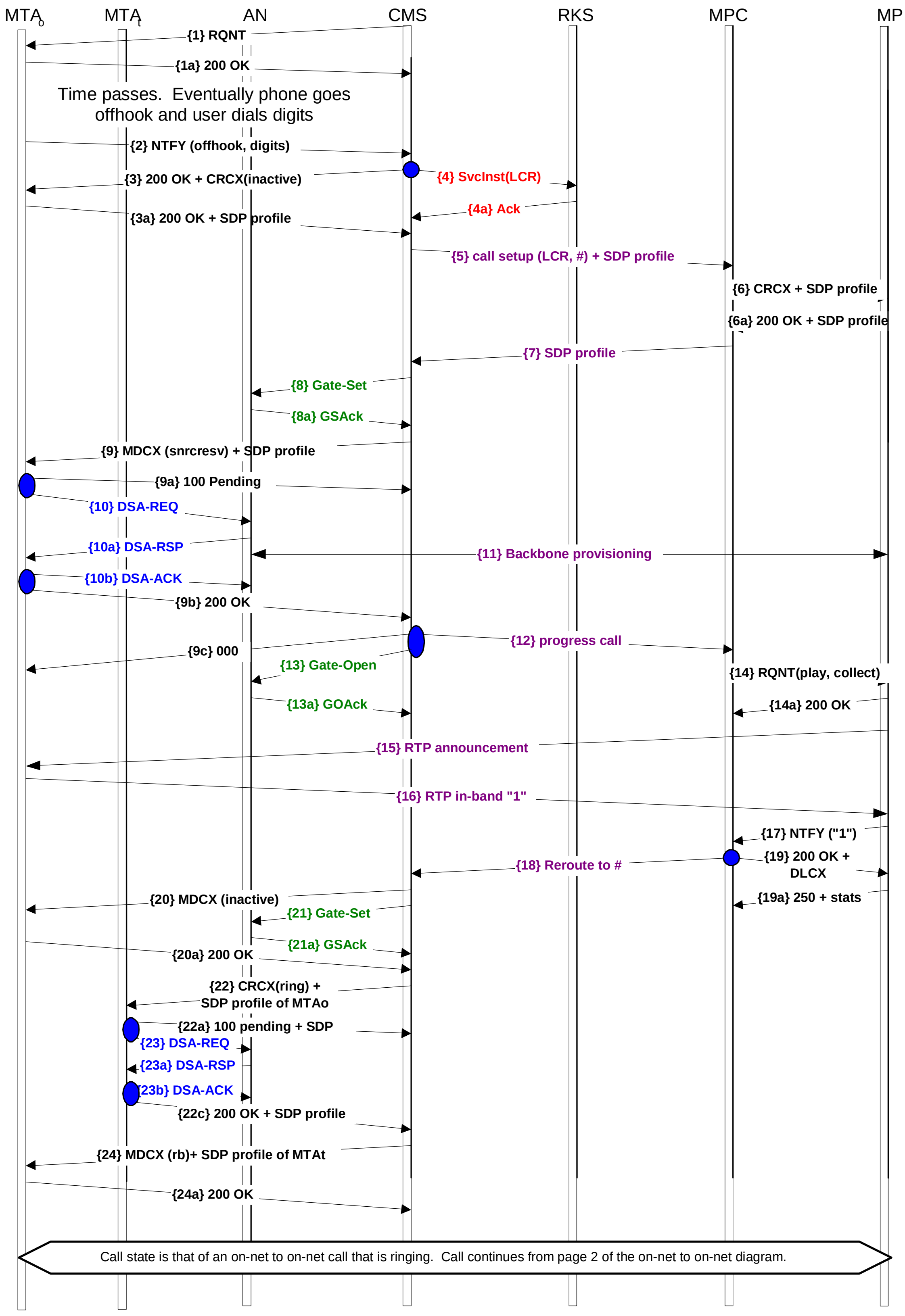


Figure A.1: Call flow for network announcement

Table A.1: Call flow details

Flow	Flow Description
1 <NCS>	CMS sends MTA-o a NotificationRequest instructing MTA to look for an off-hook event, and to report it. RQNT 1201 aaln/1@ec-1.mso.net MGCP 1.0 NCS 1.0 N: ca@ca1.mso.net:5678 X: 0123456789AB R: hd(A, E(R(hu, [0-9# *T](D)),S(dl))) D: (0T | 00T | 303[2-9]xxxxxx | 720[2-9]xxxxxx | 1[2-9]xxxxxxxxx | [3469]11 | 0[2-9]xxxxxxxxx | 01[2-9]xxxxxxxxxxxxxxT | 011xxxxxxxxxxxxxxxT)
1a <NCS>	MTA sends CMS an ACK in response to the command, repeating in the response the transaction id that the Call Agent attached to the query and providing a return code indicating success: 200 1201 OK
2 <NCS>	MTA sends CMS a Notification message indicating that an off-hook was observed and that the user requested the phone number of the Last Call Received (LCR). NTFY 2001 aaln/1@ec-1.mso.net MGCP 1.0 NCS 1.0 N: ca@ca1.mso.net:5678 X: 0123456789AB O: *,6,9
3 <NCS>	CMS sends MTA an acknowledgement of the notification. Piggybacked on the acknowledgment, the CMS sends MTA-o a Create connection message. The connection is created in inactive mode. Packetization parameters are passed in the CRCX message. 200 2001 OK . CRCX 1202 aaln/1@ec-1.mso.net MGCP 1.0 NCS 1.0 C: A3C47F21456789F0 L: p:10, a:PCMU, sc-rtp: 00/51; 62/51, sc-rtcp: 02/03; 01/03 sc-st: base64: pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQuueo== M: inactive N: ca@ca1.mso.net:5678 X: 0123456789AC R: hu
4 <Event Messages>	CMS creates the BillingCorrelationID for this transaction. CMS sends RKS a Svcinst(LCR) Message. RADIUS Message Header: <Code = Accounting-Request(1 octet, value = 4)> <Identifier (1 octet, value = 10)> <Length (2 octets, min value = 20, max value = 4096)> <Authenticator (16 octets, value = 0)> IPCablecom Event Message Header VSA: <Type = vendor specific (1 octet, value = 26)> <Length (1 octet, value = ???)> <vendor-ID = CableLabs (4 octets, value = 4491)> <Vendor Attribute Type = Event Message Header (1 octet, value = 1)> <Vendor Attribute Length (1 octet, value = 56)> <Vendor Attribute Value = <Version ID = IPCablecom 1.0 (2 octets, value = 1)> <Billing Correlation ID (16 octets, value = TTTTXXXXXCMSCCCC)> <Event Message Type = Call_Signaling_Start (2 octets, value = 1)> <Element Type = CMS (2 octets, value = 1)> <Element ID (8 octets, value = xxxxxCMS)> <Sequence ID (4 octets, value = AA05)> <Event Message Time and Date (17 octets, value = yyyymmddhhmmss.mm)> <Message Status = no known errors, message from trusted element (4 octets, value = ????)> <Message Priority = user-defined (1 octet, value = any)> <Attribute Count (2 octets, value = 4)> <Event Object = reserved (1 octet, value = 0)> >

Flow	Flow Description

3a
<NCS>
	MTA sends CMS an acknowledgement of the CRCX, adding its own SDP profile.
	
	200 1202 OK
	I: FDE234C8
	
	v=0
	o=- 25678 753849 IN IP4 128.96.41.1
	s=-
	c=IN IP4 128.96.41.1
	t=0 0
	m=audio 3456 RTP/AVP 0
	a=X-pc-csuites-rtp: 62/51
	a=X-pc-csuites-rtcp: 02/03 01/03
	a=X-pc-spi-rtcp: A7843B2
	a=X-pc-secret: base64:
	pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQueo==

4a <EM>	RKS sends CMS a RADIUS ACK in response to Service Instance message - Svcinst(LCR). RADIUS Message Header: <Code = Accounting-Response (1 octet, value = 5)> <Identifier (1 octet, value = 10)> <Length (2 octets, min value = 20, max value = 4096)> <Authenticator (16 octets, value = 0)>
5 <not specified>	CMS sends MPC all call setup information (LCR, #) including MTA-os SDP profile. [not specified]

6
<ASP>
	MPC sends MP a CreateConnection request in send-receive mode.
	
	CRCX 5050 ds/12/1@ec-2.mso.net MGCP 1.0 NCS 1.0
	N:ca@ca2.mso.net:5678
	C: A3C47F21456789F0
	L: p:10, a:PCMU, dg-gi: 1273 sc-rtp: 62/51, sc-rtcp: 02/03; 01/03 sc-st: base64:
	pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQueo==
	M: sendrcv
	X: 0123456789B0
	R: hd
	
	v=0
	o=- 25678 753849 IN IP4 128.96.41.1
	s=-
	c=IN IP4 128.96.41.1
	t=0 0
	m=audio 3456 RTP/AVP 0
	a=X-pc-csuites-rtp: 62/51
	a=X-pc-csuites-rtcp: 02/03 01/03
	a=X-pc-spi-rtcp: A7843B2
	a=X-pc-secret: base64:
	pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQueo==

6a
<ASP>
	MP sends MPC an acknowledgement of receipt of Create Connection message.
	
	200 5050 OK
	K:
	I: 32F345E2
	DQ-RI:D32B8593
	
	v=0
	o=- 25678 753849 IN IP4 128.96.41.1
	s=-
	c=IN IP4 128.96.63.25
	t=0 0
	m=audio 1296 RTP/AVP 0
	a=X-pc-csuites-rtp: 62/51
	a=X-pc-csuites-rtcp: 02/03
	a=X-pc-spi-rtcp: 453A78F1
	a=X-pc-secret: base64:
	pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQueo==

Flow	Flow Description
7<not specified>	MPC sends CMS MPs SDP profile. [not specified]
8 <DQoS [7] >	CMS sends CMTS a Gate-Set message including a local ID for use with gate coordination. Transaction ID - 3177 Subscriber - MTA Remote Gate Info - CMS address - 128.96.22.15 CMS Port - 2562 Authentication Algorithm=0x64 Security Key=FourScoreAndSevenYearsAgo Remote Gate ID - 8096 GateSpec Direction upstream Protocol UDP SourceAddress 129.96.41.1 (MTA-o) DestinationAddress ???.???.???.??? (MG) SourcePort 0 Destination Port 6540 b 120 r 12000 p 12000 m 120 M 120 R 12000 S 0 GateSpec Direction downstream Protocol UDP SourceAddress ???.???.???.??? (MG) DestinationAddress 129.96.41.1 (MTA-o) SourcePort 0 Destination Port 3456 b 120 r 12000 p 12000 m 120 M 120 R 12000 S 0 Flag = Auto commit Billing Info - Billing Correlation ID - TTTTXXXXXCMSCCCC RKS_Primary - 128.96.60.110, 5000 RKS_Secondary - 128.96.60.210, 5001 Real_time_Flag - 0 (false)
8a <DqoS [7]>	CMTS sends CMS an acknowledgment of the GateSet Transaction ID - 3177 Subscriber - MTA Gate ID - 37125 Activity Count - 2

Flow	Flow Description

9
<NCS>
	CMS sends MTA-o an MDCX message. This message indicates that the MTA should go into send-receive mode. This message also contains the session description of the Media Player.
	
	MDCX 1203 aaln/1@ec-1.mso.net MGCP 1.0 NCS 1.0
	N:ca@ca1.mso.net:5678
	C: A3C47F21456789F0
	I: FDE234C8
	M: sendrecv
	X: 0123456789AE
	R: hu
	L: dq-qi:37125
	
	v=0
	o=- 25678 753849 IN IP4 128.96.41.1
	s=-
	c=IN IP4 128.96.63.25
	t=0 0
	m=audio 1296 RTP/AVP 0
	a=X-pc-csuites-rtp: 62/51
	a=X-pc-csuites-rtcp: 02/03
	a=X-pc-spi-rtcp: 453A78F1
	a=X-pc-secret: base64:
	pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQuueo==

9a <NCS>	MTA-o sends CMS an acknowledgement of the MDCX message. 100 1203 PENDING
10 <J.112 [8]>	MTA-o sends CMTS a DSA request asking for bandwidth commitment in the access network. DSAREQ TransactionID 1 Upstream Service Flow Service Flow Reference 1 QoSParameterSetType Admitted(2) TimeoutAdmitted 200 ServiceFlowScheduling UGS(6) NominalGrantInterval 10ms ToleratedGrantJitter 2ms GrantsPerInterval 1 UnsolicitedGrantSize 111 AuthBlock 37125 DownStreamServiceFlow Service Flow Reference 2 QoSParameterSetType Admitted(2) TimeoutAdmitted 200 TrafficePriority 5 MaximumSustaninedRate 12,000 AuthBlock 37125 UpstreamPacketClassification ServiceFlowReference 1 PacketClassifierReference 1 ClassifierPriority 150 ClassifierActivationState Inactive (0) IPSourceAddress 128.96.41.1 (MTA) IPSourcePort 3456 IPDestinationAddress ???.???.???.??? (MG) IPDestinationPort 6540 IPProtocol UDP(17) DownstreamPacketClassification ServiceFlowReference 2 PacketClassifierReference 2 ClassifierPriority 150

Flow	Flow Description
	ClassifierActivationState Inactive (0) IPSourceAddress ???.???.???.??? (MG) IPDestinationAddress 128.96.41.1 (MTA) IPDestinationPort 3456 IPProtocol UDP(17)
10a <J.112 [8]>	CMTS sends MTA-o a DSA response indicating that the DSA request has been granted. DSARSP TransactionID 1 ConfirmationCode Success(0) Upstream Service Flow ServiceFlowReference 1 ServiceFlowID 1001 QoSParameterSetType Admitted(2) TimeoutAdmitted 200 ServiceFlowScheduling UGS(6) NominalGrantInterval 10ms ToleratedGrantJitter 2ms GrantsPerInterval 1 UnsolicitedGrantSize 111 AuthBlock 31001 DownStreamServiceFlow ServiceFlowReference 2 ServiceFlowID 2001 QoSParameterSetType Admitted+Active(6) TimeoutAdmitted 200 TrafficePriority 5 MaximumSustaninedRate 12,000 AuthBlock 32001 UpstreamPacketClassification ServiceFlowReference 1 PacketClassifierReference 1 ClassifierID 3001 ClassifierPriority 150 CalssifierActivationState Inactive (0) IPSourceAddress 128.96.41.1 (MTA) IPSourcePort 3456 IPDestinationAddress 128.96.63.25 (MG) IPDestinationPort 1296 IPProtocol UDP(17) DownstreamPacketClassification ServiceFlowReference 2 PacketClassifierReference 2 ClassifierID 3002 ClassifierPriority 150 ClassifierActivationState Active (1) IPSourceAddress 128.96.63.25 (MG) IPDestinationAddress 128.96.41.1 (MTA) IPDestinationPort 3456 IPProtocol UDP(17)
10b <J.112 [8]>	MTA-o sends CMTS an acknowledgement of the DSARSP. DSA-ACK TransactionID 1 ConfirmationCode Success(0)
11 <not specified>	Any backbone provisioning that is required is performed
9b <NCS>	MTA sends CMS a confirmation of transaction complete for MDCX. 200 1203 OK K:

Flow	Flow Description
9c <NCS>	CMS sends MTA an acknowledgement of the completion of the MDCX transaction. 000 1203
12 <not specified>	CMS notifies the MPC to progress the call [not specified]
13 <D-QoS>	CMS sends a GATE-OPEN message to CMTS GateOpen TransactionID - 81 Gate-ID - 37125
13a <D-QoS>	CMTS responds to GateOpen message GateOpenAck TransactionID - 81
14 <ASP>	MPC sends MP a RQNT message to play the appropriate announcement and prompt for digit collection. RQNT 5051 aaln/1@ec-1.mso.net MGCP 1.0 NCS 1.0 N: ca@ca1.mso.net:5678 X: 0123456789AB R: oc, of S: AAU/pc(ip=file://12345<5145551234>,file://34548 dm=x)
14a <ASP>	MP acknowledges receipt of RQNT from MPC 200 5051 OK
15 <ASP>	MP plays announcement to MTA-o via RTP media stream
16 <ASP>	In response to callers touch-tone, MTA-o sends MP a DTMF "1" via in-band signalling
17 <ASP>	MP sends MPC a Notification message indicating that a DTMF "1" was received. NTFY 7070 aaln/1@ec-1.mso.net MGCP 1.0 NCS 1.0 N: ca@ca1.mso.net:5678 X: 0123456789AB O: oc(dc=1 na=1)
18 <not specified>	MPC notifies CMS to re-route the call to the LCR #
19 <ASP>	MPC sends MP an acknowledgement of the NTFY and includes a piggybacked delete connection message. 200 7070 OK . DLCX 5052 aaln/1@ec-2.mso.net MGCP 1.0 NCS 1.0 C: A3C47F21456789F0 I: 32F345E2
19a <ASP>	MP sends MPC an acknowledgement of the DLCX and includes the call statistics collected by the MP. 250 5052 OK P: PS=1245, OS=62345, PR=780, OR=45123, PL=10, JI=27, LA=48
20 <NCS>	CMS sends MTA-o an MDCX message de-activating the connection. MDCX 1204 aaln/1@ec-1.mso.net MGCP 1.0 NCS 1.0 N:ca@ca1.mso.net:5678 C: A3C47F21456789F0 I: FDE234C8 M: inactive X: 0123456789AF R: hu
20a <NCS>	MTA-o sends CMS an acknowledgement of the MDCX message. 200 1204 OK
21 <DQoS>	CMS sends CMTS a Gate-Set message including the local ID for use with gate coordination. Transaction ID - 3177 Subscriber - MTA

Flow	Flow Description
	Remote Gate Info - CMS address - 128.96.22.15 CMS Port - 2562 Authentication Algorithm=0x64 Security Key=FourScoreAndSevenYearsAgo Remote Gate ID - 8096 GateSpec Direction upstream Protocol UDP SourceAddress 129.96.41.1 (MTA-o) DestinationAddress ???.???.???.??? (MG) SourcePort 0 Destination Port 6540 b 120 r 12000 p 12000 m 120 M 120 R 12000 S 0 GateSpec Direction downstream Protocol UDP SourceAddress ???.???.???.??? (MG) DestinationAddress 129.96.41.1 (MTA-o) SourcePort 0 Destination Port 3456 b 120 r 12000 p 12000 m 120 M 120 R 12000 S 0 Flag = Auto commit Billing Info - Billing Correlation ID - TTTTXXXXXCMSCCCC RKS_Primary - 128.96.60.110, 5000 RKS_Secondary - 128.96.60.210, 5001 Real_time_Flag - 0 (false)
21a <DQoS>	CMTS sends CMS an acknowledgment of the GateSet Transaction ID - 3177 Subscriber - MTA Gate ID - 37125 Activity Count - 2
22 <NCS>	CMS sends MTA-t a create connection message asking MTA-t to ring the phone. CRCX Includes the SDP profile of MTA-o. CRCX 1301 aaln/1@ec-2.mso.net MGCP 1.0 NCS 1.0 C: A3C47F21456789F0 L: p:10, a:PCMU, sc-rtp: 00/51; 62/51, sc-rtcp: 02/03; 01/03 sc-st: base64: pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQuueo== M: inactive N: ca@ca1.mso.net:5678 X: 0123456789AC R: hu S: rg v=0 o=- 25678 753849 IN IP4 128.96.41.1 s=- c=IN IP4 128.96.41.1

Flow	Flow Description

	t=0 0
	m=audio 3456 RTP/AVP 0
	a=X-pc-csuites-rtp: 62/51
	a=X-pc-csuites-rtcp: 02/03
	a=X-pc-spi-rtcp: A7843B2
	a=X-pc-secret: base64:
	pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQuueo==

22a
<NCS>
	MTA-t sends CMS a confirmation of transaction complete for CRCX and its SDP profile.
	
	100 1301 pending
	
	v=0
	o=- 25678 753849 IN IP4 128.96.41.1
	s=-
	c=IN IP4 128.96.10.10
	t=0 0
	m=audio 6789 RTP/AVP 0
	a=X-pc-csuites-rtp: 62/51
	a=X-pc-csuites-rtcp: 02/03 01/03
	a=X-pc-spi-rtcp: A7843B2
	a=X-pc-secret: base64:
	pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQuueo==

23 <J.112 [8]>	MTA-t sends CMTS a DSA request asking for bandwidth commitment in the access network. DSAREQ TransactionID 1 Upstream Service Flow Service Flow Reference 1 QoSParameterSetType Admitted(2) TimeoutAdmitted 200 ServiceFlowScheduling UGS(6) NominalGrantInterval 10ms ToleratedGrantJitter 2ms GrantsPerInterval 1 UnsolicitedGrantSize 111 AuthBlock 37125 DownStreamServiceFlow Service Flow Reference 2 QoSParameterSetType Admitted(2) TimeoutAdmitted 200 TrafficePriority 5 MaximumSustaninedRate 12,000 AuthBlock 37125 UpstreamPacketClassification ServiceFlowReference 1 PacketClassifierReference 1 ClassifierPriority 150 CalssifierActivationState Inactive (0) IPSourceAddress 128.96.41.1 (MTA) IPSourcePort 3456 IPDestinationAddress ???.???.???.??? (MG) IPDestinationPort 6540 IPProtocol UDP(17) DownstreamPacketClassification ServiceFlowReference 2 PacketClassifierReference 2 ClassifierPriority 150 ClassifierActivationState Inactive (0) IPSourceAddress ???.???.???.??? (MG) IPDestinationAddress 128.96.41.1 (MTA) IPDestinationPort 3456 IPProtocol UDP(17)

Flow	Flow Description
23a <J.112 [8]>	CMTS sends MTA-t a DSA response indicating that the DSA request has been granted. DSARSP TransactionID 1 ConfirmationCode Success(0) Upstream Service Flow ServiceFlowReference 1 ServiceFlowID 1001 QoSParameterSetType Admitted(2) TimeoutAdmitted 200 ServiceFlowScheduling UGS(6) NominalGrantInterval 10ms ToleratedGrantJitter 2ms GrantsPerInterval 1 UnsolicitedGrantSize 111 AuthBlock 31001 DownStreamServiceFlow ServiceFlowReference 2 ServiceFlowID 2001 QoSParameterSetType Admitted+Active(6) TimeoutAdmitted 200 TrafficePriority 5 MaximumSustaninedRate 12,000 AuthBlock 32001 UpstreamPacketClassification ServiceFlowReference 1 PacketClassifierReference 1 ClassifierID 3001 ClassifierPriority 150 CalssifierActivationState Inactive (0) IPSourceAddress 128.96.41.1 (MTA) IPSourcePort 3456 IPDestinationAddress 128.96.63.25 (MG) IPDestinationPort 1296 IPProtocol UDP(17) DownstreamPacketClassification ServiceFlowReference 2 PacketClassifierReference 2 ClassifierID 3002 ClassifierPriority 150 ClassifierActivationState Active (1) IPSourceAddress 128.96.63.25 (MG) IPDestinationAddress 128.96.41.1 (MTA) IPDestinationPort 3456 IPProtocol UDP(17)
23b <J.112 [8]>	MTA-t sends CMTS an acknowledgement of the DSARSP. DSA-ACK TransactionID 1 ConfirmationCode Success(0)

22c
<NCS>
	MTA-t sends CMS a 200 OK and its SDP profile.
	
	200 1301 OK
	
	v=0
	c=IN IP4 128.96.63.25
	m=audio 1296 RTP/AVP 0
	a=X-pc-csuites-rtp: 62/51
	a=X-pc-csuites-rtcp: 02/03

Flow	Flow Description

24
<NCS>
	CMS sends MTA-o an MDCX message indicating ringback and the SDP profile of MTA-t.
	
	MDCX 1205 aaln/1@ec-1.mso.net MGCP 1.0 NCS 1.0
	N:ca@ca1.mso.net:5678
	C: A3C47F21456789F0
	I: FDE234C8
	M: sendrecv
	X: 0123456789AF
	R: hu
	S: rb
	
	v=0
	o=- 25678 753849 IN IP4 128.96.41.1
	s=-
	c=IN IP4 128.96.10.10
	t=0 0
	m=audio 6789 RTP/AVP 0
	a=X-pc-csuites-rtp: 62/51
	a=X-pc-csuites-rtcp: 02/03 01/03
	a=X-pc-spi-rtcp: A7843B2
	a=X-pc-secret: base64:
	pV6BIIHWt+0gDkpgnuxgTfROxYAemhYJTHWgHNt1crTtEUKFatJfSdEFVQuueo==

24a <NCS>	MTA-o sends CMS an acknowledgement of the MDCX transaction. 200 1205 OK
NOTE: Call State is a ringing on-net to on-net call between MTA-o and MTA-t. The call proceeds as a standard On-net to On-net IPCablecom call.	

Annex B (informative): Call flow for MTA-stored announcement

This clause provides an example call flow where a User-1 attempts to call User-2. Due to facility problems on the terminating side the call can not be completed. The MTA associated with User-1 is instructed to play a local announcement. It should be noted, that this call flow, although a valid one, is merely an example that may or may not be used in practice.

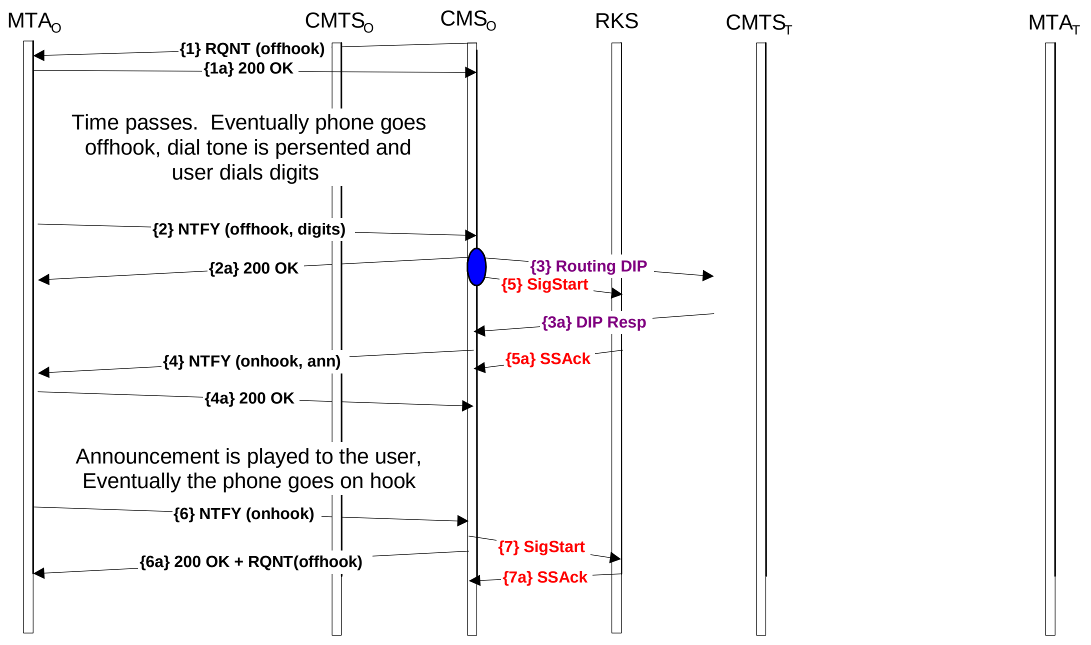

MTA_O
CMTS_O
CMS_O
RKS
CMTS_T
MTA_T
{1} RQNT (offhook)
{1a} 200 OK
Time passes. Eventually phone goes offhook, dial tone is persented and user dials digits
{2} NTFY (offhook, digits)
{2a} 200 OK
{3} Routing DIP
{5} SigStart
{3a} DIP Resp
{4} NTFY (onhook, ann)
{5a} SSAck
{4a} 200 OK
Announcement is played to the user, Eventually the phone goes on hook
{6} NTFY (onhook)
{7} SigStart
{6a} 200 OK + RQNT(offhook)
{7a} SSAck

Figure B.1: MTA call flow

B.1 Call flow details

Table B.1: call flow details

Flow	Flow Description	Depends upon these completion of these flows:	Triggers start of these flows:
	Initialization		
1 <NCS>	CMS sends MTAo a NotificationRequest instructing MTAo to look for an off-hook event, and to report it. RQNT 1201 aaln/1@ec-1.whatever.net MGCP 1.0 **NCS 1.X** N: ca@ca1.whatever.net:5678 X: 0123456789AB R: hd(E (R([0-9#*T](D), hu(N)), S(dl), ;)) D: (0T | 00T | [2-9]xxxxxx | 1[2-9]xxxxxxxxx | 011xx.T)		1a
1a <NCS>	MTAo sends CMS an ACK in response to the command, repeating in the response the transaction id that the Call Agent attached to the query and providing a return code indicating success: 200 1201 OK	1	
	Service Request		
2 <NCS>	MTAo sends CMS a Notification message indicating that an off-hook was observed. NTFY 2001 aaln/1@ec-1.whatever.net MGCP 1.0 **NCS 1.X** N: ca@ca1.whatever.net:5678 X: 0123456789AB O: hd, 3, 0, 3, 5, 5, 5, 1, 2, 1, 2	1, user stimulus	2a, 3, 4, 5
2a <NCS>	CMS sends MTAo an acknowledgement of the notification. 200 2001 OK	2	
3 <??>	CMS contacts the routing database requesting a mapping of the dialled number to a routable destination in the network.	2	3a
3a <??>	The routing database server responds to the CMS with the routing information.	3	4, 8
4 <NCS>	CMS sends MTAo a notification request message. The connection is created in inactive mode. Packetization parameters are passed in the CRCX message. RQNT 1202 aaln/1@ec-1.whatever.net MGCP 1.0 **NCS 1.X** N: ca@ca1.whatever.net:5678 X: 0123456789AC R: hu, oc, of S: A/ann(file://audio/23945)	2, 3a	4a, 5
4a <NCS>	MTAo sends CMS an acknowledgement of the RQNT, adding its own SDP profile. 200 1202 OK	4	6, 8

Flow	Flow Description	Depends upon these completion of these flows:	Triggers start of these flows:
Announcement is being played			
5 <EM>	CMS creates the BillingCorrelationID for this transaction. CMS sends RKS a Call_Signaling_Start Event Message. The message contents include: Event_Message_Header(Version_ID, BillingCorrelationID, "Call_Signaling_Start Event Message", Element_Type, Element_ID, Element_Seq_Num, Message_Timestamp, Message_Status, Message_Priority, Attribute_Count, Event_Object), Event_Time, MTA_Port_ID, Calling_Party_Number, Called_Party_Number The message format is: <insert example coded message>	2	5a
5a <EM>	RKS sends CMS a RADIUS ACK in response to Call_Signaling_Start **ACK** The message format is: <insert example coded message>	5	
User is listening to the announcement and hangs up			
11 <NCS>	MTAo sends CMS a notification that the attached device has gone on-hook. NTFY 2002 aaln/1@ec-2.whatever.net MGCP 1.0 **NCS 1.X** X: 0123456789AF O: hu		12, 13, 14
12 <NCS>	CMS sends MTAo an acknowledgement of the NTFY and includes a piggybacked delete connection message. 200 2002 OK . RQNT 1207 aaln/1@ec-2.whatever.net MGCP 1.0 **NCS 1.X** X: 0123456789B2 N: ca@ca1.whatever.net:5678 R: hd (E (dl:hu, D/[0-9# *T] (D) ;)) D: (0T | 00T | [2-9]xxxxxx | 1[2-9]xxxxxxxxx | 011xx.T)	11	12a, 15
12a <NCS>	MTAo sends CMS an acknowledgement of the DLCX and includes the call statistics collected by the MTA. 250 12?? OK	12	22, 25
14 <EM>	CMS sends RKS a Media_Connection_Stop Event Message. The message contents include: Event_Message_Header(Version_ID, BillingCorrelationID, "Media_Connection_Stop Event Message", Element_Type, Element_ID, Element_Seq_Num, Message_Timestamp, Message_Status, Message_Priority, Attribute_Count, Event_Object), Event_Time, Call_Termination_Cause The message format is: <insert example coded message>	11	14a

Flow	**Flow Description**	**Depends upon these completion of these flows:**	**Triggers start of these flows:**
14a <EM>	RKS sends CMS a RADIUS ACK in response to Media_Connection_Stop **ACK** The message format is: <insert example coded message>	14	

Annex C (informative): Bibliography

- ETSI TS 101 909-19-1: "Digital broadband Cable access to the public telecommunication network; IP Multimedia Time Critical Services; Part 19: IPCablecom Audio Server Protocol Specification; Sub-part 1: H.248 option".
- ETSI TS 101 909-16: "Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 16:Signalling for Call Management Server".
- ETSI TS 101 909-5: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 5: Dynamic Quality of Service for the Provision of Real Time Services over Cable Television Networks using Cable Modems".
- ITU-T Recommendation J.162: "Network call signalling protocol for the delivery of time critical services over cable television networks using cable modems".
- "Key words for use in IETFs to Indicate Requirement Levels", Bradner, S, BCP 14, IETF 2119, March 1997. www.ietf.org
- "Proposal for an MGCP Advanced Audio Package", Cromwell, D Version 0.0, June 2000, internet draft (this is a work in progress and currently has no status as a standard).
- "Requirements For Control Of A IVR Function", Cromwell, D, Durling, M Version 0.0, April, 1999, internet draft (this is a work in progress and currently has no status as a standard).

History

Document history		
V1.1.1	March 2002	Publication