IMTC Voice over IP Forum Service Interoperability Implementation Agreement

Draft 0.91

IMTC Voice over IP Forum Technical Committee

Jan. 13, 1997

Voice Over IP Implementation Agreement

IMTC Voice Over IP Technical Committee

File IA91.doc

Table of Contents

1. Introduction

1.1 Abstract

The VoIP Service Interoperability Implementation Agreement combines, clarifies and complements existing standards to provide a complete Internet telephony interoperability protocol. The foundation of the IA is the use of H.323 for call establishment and capability negotiation. Call signaling between service elements is handled via Q.931 as part of H.323, with each service element providing the necessary conversion from its local endpoint signaling to the H.323/Q.931 backbone protocol. Other telephony specific requirements such as the transfer and reproduction of DTMF data have been added to provide a high level of connectivity with the traditional telephone infrastructure. Directory services are also a critical element of an Internet telephony system, and is something currently missing from the base H.323 spec. The VoIP IA defines a comprehensive directory and call management service called the Call Management Agent System that itself integrates existing directory services such as LDAP with new dynamic IP address resolution mechanisms and provides an infrastructure for enhanced call management services.

To ensure interoperability, the VoIP IA specifies baseline requirements that all service elements must adhere to, and also allows optional extensions for some parameters.

1.2 Purpose

Client PC and workstation software has emerged in the marketplace from multiple vendors to support voice over the Internet. Existing software does not allow interworking between different vendor's products as voice coding, silence compression, addressing and related functions are not compatible. The purpose of this specification is to provide common functions that will allow products complying with this specification and its references to interoperate.

It is the goal of this specification to support two-party voice and similar audio communications over IP networks in a manner similar and compatible (via gateways) with existing SCN (Switched Circuit Network) telephone calls. Every attempt has been made to utilize existing IETF and ITU protocols and services. Interworking via H.323 gateways to SCN communications equipment is provided. While only two-party service is supported in this document it is anticipated that extensibility to multipoint communications may be implemented in the future in a manner backward compatible with this document and existing formal multi-point circuit mode conferencing standards.

1.3 Baseline Interoperability Requirement Summary

1. All connections are made using the H.323 / H.245 / H.225 session protocol suite (to be expounded upon)
2. Connections are made over TCP/UDP/IP protocol layers.
3. RTP is used to encapsulate real-time traffic.
4. The VoIP RTP profile RFC-xxxx is used to interpret the specific usage of RTP. This will include support for additional codec, silence and DTMF payloads.
5. VoIP clients and SCN gateways must support GSM 6.10 with RTP packaging.
6. SCN gateways must provide a signaling gateway between the local telephony signaling system and the Q.931 signaling protocol used by H.323.
7. Call address registration and resolution is made via the Call Management Agent protocol.

1.4 Terminology

Addressable: An entity on the Internet having a Transport Address. Not the same as being callable. A client or server is addressable and callable. A gatekeeper is addressable but not callable.

Call (noun): Point-to-point multimedia communication between two Internet endpoints. The call begins with the call setup procedure and ends with the call termination procedure. The call consists of the collection of reliable and unreliable channels between the endpoints. In case of interworking with some SCN endpoints via a gateway, all the channels terminate at the Gateway where they are converted to the appropriate representation for the SCN end system.

CMA Sys Entity: Any CMA Sys component, including client(s) and server(s).

CMAP: Call Management Agent Protocol. The protocol between a CMAC and a CMAS.

CMA Logic: The computation performed by the CMA for a specific request.

Callable: Capable of being called as described in Section ??. Clients, Servers and Gateways are callable, but Gatekeepers are not.

Caller: The entity initiating a call.

Called: The destination of a call.

E&M: Ear & Mouth signaling

Endpoint: An H.323 Gateway, CMA client, LDAP server, or CMA Server. An endpoint can call and be called. It generates and/or terminates information streams.

Gatekeeper: The Gatekeeper (GK) is an H.323 entity on the Internet that provides address translation and controls access to the network for H.323 Terminals and

Gateways. The Gatekeeper may also provide other services to the H.323 terminals and Gateways, such as bandwidth management and locating Gateways.

Gateway: An H.323 Gateway (GW) is an endpoint on the Internet which provides for real-time, two-way communications between H.323 Terminals on the Network and other ITU Terminals on a wide area network, or to VoIP Clients. Other ITU Terminals include those complying with Recommendations H.310 (H.320 on B-ISDN), H.320 (ISDN), H.321 (ATM), H.322 (GQOS-LAN), H.324 (GSTN), H.324M (Mobile), and V.70 (DSVD).

H.323 Entity: Any H.323 component, including H.323 Terminals, Gateways, Gatekeepers.

Information Stream: A flow of information of a specific media type (e.g. audio) from a single source to one or more destinations.

Internet address: The network layer address of a H.323 or CMA Sys entity as defined by the (inter)network layer protocol in use (e.g. an IP address). This address is mapped onto the layer one address of the respective system by some means defined in the (inter)networking protocol.

Internet: An inter-network of networks interconnected by bridges or routers. LANs described in H.323 may be considered part of such internetworks .

RAS Channel: Unreliable channel used to convey the registration, admissions, bandwidth change, and status messages (following H.225.0) between H.323 entities or CMA Sys entities.

Reliable Channel: A transport connection used for reliable transmission of an information stream from its source to one or more destinations.

Reliable Transmission: Transmission of messages from a sender to a receiver using connection-mode data transmission. The transmission service guarantees sequenced, error-free, flow-controlled transmission of messages to the receiver for the duration of the transport connection.

Soft Link: A referral from one CMA to another.

Subscriber: An “owner” of a CMA. A subscriber can be a person or an organization.

Switched Circuit Network (SCN): A public or private switched telecommunications network such as the GSTN, N-ISDN, or B-ISDN.

Transport Address: The transport layer address of an addressable H.323 entity as defined by the (inter)network protocol suite in use. The Transport Address of an H.323 entity is composed of the LAN address plus the TSAP identifier of the addressable H.323 entity.

Transport Connection: An association established by a transport layer between two H.323 entities for the transfer of data. In the context of H.323, a transport connection provides reliable transmission of information.

TSAP Identifier: The piece of information used to multiplex several transport connections of the same type on a single H.323 entity with all transport connections sharing the same LAN address, (e. g. the port number in a TCP/UDP/IP environment). TSAP identifiers may be (pre)assigned statically by some international authority or may be allocated dynamically during setup of a call. Dynamically assigned TSAP identifiers are of transient nature, i. e. their values are only valid for the duration of a single call.

Unicast: A process of transmitting messages from one source to one destination.

VoIP: Voice over Internet Protocol. The VoIP Forum is the developer of this specification

Well-known TSAP Identifier: A TSAP identifier that has been allocated by an (international) authority that is in charge for the assignment of TSAP identifiers for a particular (inter)networking protocol and the related transport protocols -- (e.g. the IANA for TCP and UDP port numbers). This identifier is guaranteed to be unique in the context of the respective protocol.

1.5 Acronyms

CMA: Call Management Agent

CMAA: Call Management Agent Address. An address through which a given CMA can be located and then accessed.

CMAS: Call Management Agent Server

CMA Sys: Call Management System

CMAC: Call Management Agent Client

CMAP: Call Management Agent Protocol

CS-ACELP: Conjugate Structure - Algebraic Code-Excited Linear Prediction

CT: Communication Terminal

CTT: Communication Terminal Type

CTA: Communication Terminal Address

CTAT: Communication Terminal Address Type

DTMF: Dual Tone Multiple Frequency

GSM: Global System for Mobile communications

IANA: Internet Assigned Numbers Authority

IETF: Internet Engineering Task Force

IP: Internet Protocol

ISDN: Integrated Services Digital Network

ITU: International Telecommunication Union

IVR: Interactive Voice Response.

LDAP: Lightweight Directory Access Protocol

LAN: Local Area Network

PBX: Private Branch eXchange

POTS: Plain Old Telephone Service

PRI: ISDN Primary Rate Interface

PSTN: Public Switched Telephone Network

QoS: Quality of Service

RAS: Registration, Admission and Status

RFC: Request For Comment

RTP: Real-Time Protocol

TCP: Transmission Control Protocol

TSAP: Transport Service Access Point

UDP: User Datagram Protocol

1.6 References

The following references contain provisions which are referenced in this text. At the time of publication, the editions indicated were valid.

[1] ITU-T Recommendation H.225.0 (1996): " Media Stream Packetization and Synchronization for Visual Telephone Systems on Non-Guaranteed Quality of Service LANs ".

[2] ITU-T Recommendation H.245 (1995): "Control of communications between Visual Telephone Systems and Terminal Equipment".

[3] ITU-T Recommendation T.120 (1994): "Transmission protocols for multimedia data"

[4] ITU-T Recommendation H.320 (1995): "Narrow-band ISDN visual telephone systems and terminal equipment".

[5] ITU-T Recommendation H.321 (1995): "Adaptation of H.320 Visual Telephone Terminals to B-ISDN Environments".

[6] ITU-T Recommendation H.322 (1995): "Visual Telephone Systems and Terminal Equipment for Local Area Networks which Provide a Guaranteed Quality of Service".

[7] ITU-T Recommendation H.324 (1995): Terminal for Low Bitrate Multimedia Communications".

[8] ITU-T Recommendation H.310 (1996): "Broadband audio-visual communications systems and terminal equipment".

[8] ITU-T Recommendation Q.931 (1993): "Digital Subscriber Signalling System No. 1 (DSS 1) - ISDN User-Network Interface Layer 3 Specification for Basic Call Control".

[10] ITU-T Recommendation Q.932 (1993): "Digital Subscriber Signalling System No. 1 (DSS 1) - Generic Procedures for the Control of ISDN Supplementary Services".

[11] ITU-T Recommendation Q.950 (1993): "Digital Subscriber Signalling System No. 1 (DSS 1) - Supplementary Services Protocols, Structure, and General Principles".

[12] ISO/IEC 10646-1 (1993): "Information Technology - Universal Multiple-Octet Coded Character Set (USC) -- Part I: Architecture and Basic Multilingual Plane".

[13] ITU-T Recommendation E.164 (1991) “Numbering Plan for the ISDN Era”.

[14] IETF RFC 822 <email address format>

[15] IETF <> <LDAP related documents>

[16] X.500 related documents

[17] X.509 certificate documents

[18] SSL/PCT documents

[19] H.323, RAS related documents

[20] GUID, Microsoft

[18] RFC 1889

[19] RFC 1890 "RTP Profile for Audio and Video Conferences with Minimal Control"

[20] ITU-T Recommendation H.323 "Visual Telephone Systems And Equipment For Local Area Networks Which Provide A Non-Guaranteed Quality Of Service"

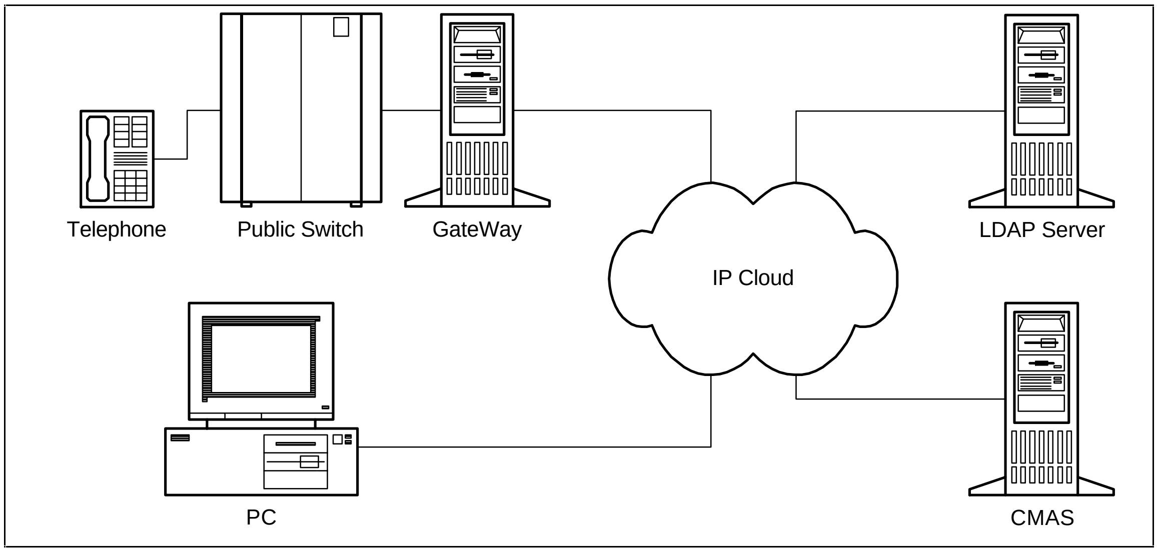
[21] GSM

[22] ITU-T Recommendation G.729 "Coding Of Speech At 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP)"

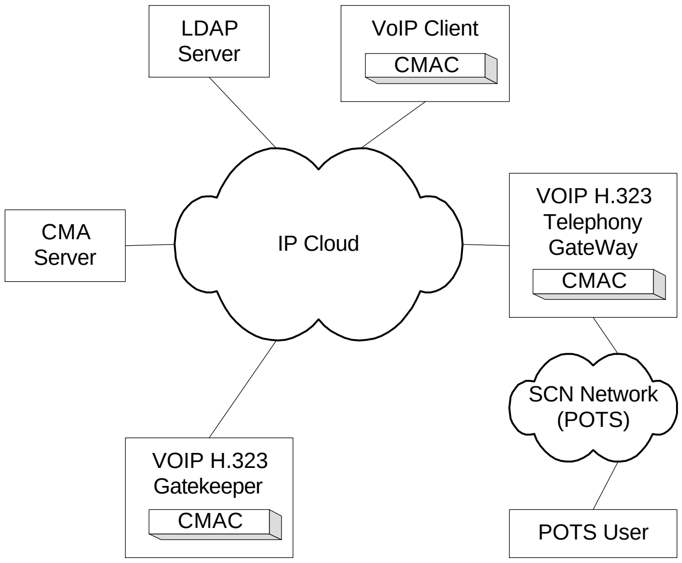
[23] ITU-T Recommendation G.764 "Voice Packetization - Packetized Voice Protocols"

2. VoIP Operational Environment

2.1 Physical VoIP network elements



2.2 Logical Voice Service Modules



2.3 Internet / LAN telephones

Internet telephones are terminals capable of using the entire Voice over IP stack. Each terminal may be separately addressed. Terminals will include personal computers, special-purpose telephone sets. Audio communications between the user and the terminal may occur via sound cards in personal computers or via special-purpose hardware adapting telephone sets to the VoIP operating environment.

2.4 Directory Servers

Directory Servers provide the function to map user names to IP addresses on a Voice over IP network, as well as provide functions for users to negotiate for call services.

A Directory Service takes as input a name or names for someone, and returns a set of attributes which have been previously stored in the directory for that person. The directory only returns what has been stored; it does no significant computation on the stored information, nor does it make any guarantees that information returned is current. The primary purpose of the directory is to ascertain whether a particular person is registered, and whether the stored information represents the terminal or person with whom communications is desired.

A Rendezvous Service takes as input an unambiguous identifier for a correspondent (i.e. it refers to exactly one registered user - possibly returned previously by a directory lookup), and returns current information about the user's current address(es), preferences about how to be reached (I-Phone, PSTN, cell phone, voice mail, etc.), and possibly other dynamically changing information computed by the location service. Note: unlike a directory, a location service may be modeled as an agent which sits actively on the network and acts on a user's behalf even when the user is unavailable.

2.5 PSTN gateways

Given the global deployment of POTS terminals, interaction with legacy networks will be an important element of a Voice over IP networks. Users on the Voice over IP network may address users on the PSTN or vice versa via VoIP-H.323 gateways. Gateways may connect directly to the PSTN using telephony protocols or may connect to private networks via PBX’s. Gateways may be embedded in PBX’s or be stand-alone equipment.

Gateways will also enable phone to phone conversation enabling a long distance conversation to be routed via an TCP/IP network.

Detailed information on the gateway reference model is available in **Annex A**

2.6 Sample Call Setup

A quick overview of how the different VoIP protocols work together is illustrated by following step-by-step a sample call setup between two Internet telephones:

<diagram and setup list to be included>

3. VoIP Stack Overview

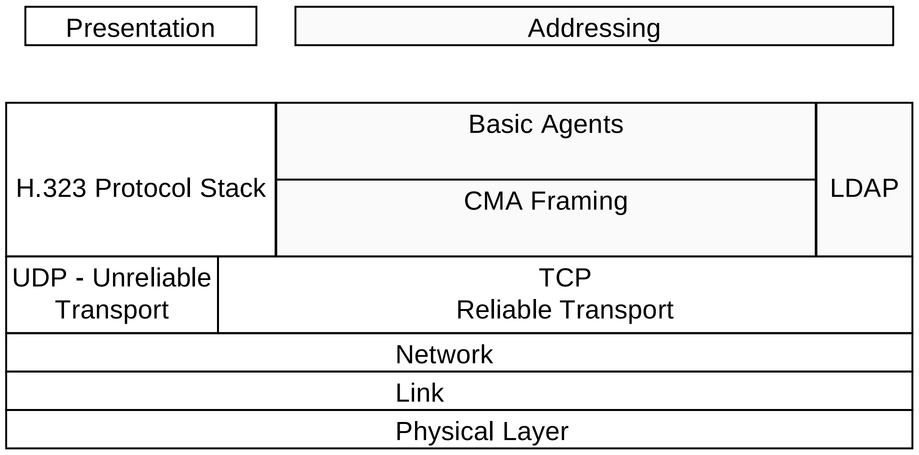


Figure 2 shows the overall stack used by Voice over IP network element

3.1 Presentation Layer

Within the context of VoIP, the presentation layer interprets the syntax of all transferred speech and 'in-band' control data. In this context, in-band refers to control mechanisms which are either traditionally carried within the voice band (i.e. DTMF digits) or are events which should be synchronized with transmitted speech packets (i.e. comfort noise parameters). Semantically, the presentation layer consists of two major components:

1. RTP payload type definition
2. Payload transfer syntax for each payload type

All data and in-band control packets are handled as RTP payload. Both statically and dynamically assigned payload types are used to differentiate between traffic types. VoIP uses current static definitions (as defined in RFC1890) whenever possible, and assigns dynamic payload types for payload specific to VoIP.

Annex B gives the mapping of RTP payload type indices to payload types.

3.1.1 Encoded Speech

Voice over IP elements involved in voice coding & decoding must include support for GSM 6.10 as the baseline coder. GSM 6.10 is assigned static payload type 3. . Defining a baseline coder does not imply that it is the best possible coder - it merely defines one which is widely available and implementable on a wide variety of platforms. Other coders may also be used - several G series coders have already been assigned static payload types as part of RFC1890, and others may be assigned dynamic payload types in future revisions of this IA.

Annex C describes the bit order and the way of packaging GSM 6.10 information in VoIP voice payload.

Annex D is a place holder for the transfer syntax of other codecs' voice payloads.

3.1.2 Voice Activity Detection

3.1.2.1 Introduction

Voice Activity Detection (aka Speech Activity Detection, Digital Speech Interpolation and Silence Deletion) is an effective tool for reducing the data throughput of a speech channel. This IA introduces the definition of a 'noise' coder RTP payload type that allows the use an alternate encoding method during detected periods of silence. This IA does not specify how to detect silence nor how to reconstruct suitable noise samples at the receiving end, but describes options for transmission of silence parameters.

3.1.2.2 VoIP Speech / Silence Indication Mechanisms

Two methods can be employed by VoIP to indicate speech/silence conditions:

3.1.2.2.1 Silence Indication Via Receive Underrun Conditions

The simplest silence indication method is simply to stop transmission of packets during silence and use the ensuing underrun condition at the receive end to indicate silence. Noise characteristics may then be derived from the previously received signal during probable periods of silence (i.e. a period of hangover at the end of speech bursts). Unfortunately, this method does not allow discrimination between silence and transmission error conditions that might also cause underruns. It also precludes any updates to noise playout parameters during a silent period.

3.1.2.2.2 Silence Indication Via Explicit Notification

Silence may also be indicated by sending one or more packets containing noise parameters during silence periods. This approach uses the RTP payload type transition from a speech coder to an alternate 'noise' coder to indicate the speech to silence transition, and allows multiple packets of the alternate 'noise' coder type to be sent during a given silent period.

This VoIP IA has defined only one of a number of possible noise information transmission schemes. This scheme, based on G.764 absolute noise level indexing, has been allocated the dynamic RTP payload type 120. This method transmits absolute level noise information in a single byte payload, which should be transmitted immediately after the last speech packet of a speech burst. Additional noise packets with updates to the noise level may be sent during long silence periods if desired.

Other noise information transmission schemes may be included in later versions of the VoIP IA and assigned to other dynamically allocated RTP payload types.

Redundancy of noise information packets is not considered important in the G.764 method, as the loss of noise information packets will, in the worst case, cause nothing more than a default to the 'Silence Indication Via Receive Underrun Conditions' case.

For baseline compatibility, receivers must at minimum throw away silence packets of payload type 120 without causing a functional disruption in other aspects of operation.

Annex E describes the bit order and packaging of G.764 silence information as VoIP payload.

3.1.3 DTMF Digit Carriage

Methods for DTMF digit carriage are still under investigation. The following is a proposed method for this carriage, but may be changed in later versions of this IA.

3.1.3.1 Introduction

Although call setup signaling will be carried using Q.931 as part of the H.323 framework, there is an additional requirement for the carriage of in-band signaling such as DTMF after call setup. With coders of sufficient quality, these signals can be carried as a voice signal and interpreted properly by far-end decoding equipment. Low bit-rate coders, however, may cause enough degradation in these signals to cause misinterpretation by decoding equipment. For these coders, an out-of-band signaling transfer syntax using a separate dynamically allocated RTP codec type can be used to indicate in-band signaling events to VoIP end-point equipment. Further study is required to determine which low bit-rate coders will explicitly require this method for reliable DTMF carriage.

Out-of-band DTMF packets are assigned dynamic payload type 121.

This IA does not specify any method for the initial detection of said events nor does it specify the manner in which the out-of-band information is reinserted at the VoIP endpoint.

3.1.3.2 Signaling Architecture

The general VoIP signaling architecture is based on a Q.931 backbone running between end-points, with conversion of local signaling semantics to Q.931 within each end-point. Physical interfaces and associated signaling schemes (e.g. pulse dialing, PRI) have local significance only - this preserves the any-to-any connectivity nature of VoIP. This concept is illustrated by example in Figure 1. Here, both the Internet telephone and gateway service elements provide termination of their local signaling format and semantic conversion to the Q.931 backbone. Providing any-to-any connectivity also precludes concepts such as CAS??telemetry.

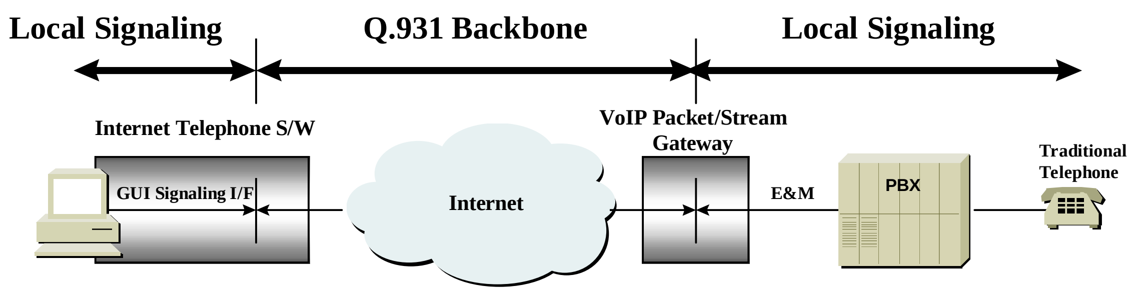


Figure 1 VoIP Signaling Connectivity

Similarly, once a connection is established via Q.931, in-band signaling events may occur that need to be handled in a similar any-to-any connectivity fashion.

3.1.3.3 Rationale for Out-of-band Carriage of 'In-band' Signaling

There are two principal reasons to carry certain signaling elements as a distinct out-of-band message type:

1. Signaling elements which have global utility and meaning but have no carriage mechanism within Q.931.

2. Signaling elements traditionally carried in-band as voice data but which may be adversely affected by low bit-rate coding.

3.1.3.4 Design Highlights

3.1.3.4.1 Encode the Edges of Digit On and Off with 20ms Window

The purpose of sending both the On and Off event is to support long digit duration and allow the duty cycle to be faithfully re-produced with 1ms time resolution. The 20ms Window constrains the maximum On/Off transition rate to be no faster than 20ms which should be quite adequate for both the minimal guard time and digit on time. This also eliminates the need to negotiate the digit On/Off time.

3.1.3.4.2 Overlapped Packet with Triple Redundancy

The DTMF packet payload carries the previous two 20 ms windows of DTMF activity as well as the current 20 ms window for a triple redundancy scheme.

3.1.3.4.3 Signal Level Encoding

This minimizes the amplitude hits if the first portion of the digit gets sent through the vocoder. Unless extra delay is applied to the voice versus digit detection logic end-to-end, its very likely that a portion of a digit will be encoded and synthesized by the vocoder before the digit relay scheme can kick in. If the vocoder has high fidelity with DTMF digits, then a great amplitude discontinuity between the vocoded versus relayed digit could be interpreted by some DTMF detector as a gap which will decode one digit as two. Therefore, transmitting the power measurement from the detector side to the re-generation side can allow the relayed digit to approximate the level as generated by the vocoder. The encoding of detected level also preserves the relative power between the DTMF level and the voice level. The receive side can always choose to ignore the level and use a locally configured level for the DTMF synthesizer.

This also eliminates the need to negotiate the generation level.

3.1.3.4.4 Digit Type Encoding

This encodes the On versus Off Edge for DTMF and also allows an extension to other digit type.

Annex F describes the bit order and packaging of DTMF digit information as VoIP payload.

3.2 User to User Session and Transport Protocol

The Session and Transport layer will provide the User to User and User to Gateway call setup and data carriage. The above is based on ITU-T H.323 and the relevant RTP transport protocol defined in H.225. Specific implementation details of the H.323 and the relevant RTP profile will be discussed in **Annex H**

3.3 Transport Layer

The Transport layers discussed in the scope of this document are the reliable transport, the TCP/IP, and the non-reliable transport the, UDP.

3.4 Network Layer

The Network layer discussed in the scope of this document is the TCP/IP suit of protocols as defined in......??.

3.5 Link Layer & Physical layer

These layer functions are out of the scope of this document However those layers must provide necessary bandwidth, error performance and latency for the proper operation of the VoIP endpoints.

Table 1 - Compressed/Uncompressed RTP bit rate (bps) and frame size (frame sz), vs. number of GSM frames per packet.

					Actual transmitted data over the modem			
		frames per packet	packets per sec	packet + Processing delay (ms)	Uncompressed		*Compressed*	
					frame sz	*actual bps*	*frame sz*	*actual bps*
GSM-RTP Packaging		1	50.00	50	80	*32000*	*43*	*17200*
		2	25.00	70	114	*22800*	*76*	*15200*
Coder params:		3	16.67	90	147	*19600*	*110*	*14667*
Frames/sec:	**50**	4	12.50	110	180	*18000*	*143*	*14300*
Bits/Sec	**13200**	5	10.00	130	214	*17120*	*176*	*14080*
Delay(ms)	**30**	10	5.00	230	380	*15200*	*343*	*13720*
		25	2.00	530	880	*14080*	*843*	*13488*
(Bits/frame)	264	50	1.00	1030	1713	*13704*	*1676*	*13408*
GSM - MS Packaging								
(Even frames per packet)		2	25.00	70	113	*22600*	*75*	*15000*
Coder params:								
Frames/sec:	**50**	4	12.50	110	178	*17800*	*141*	*14100*
Bits/Sec	**13000**							
Delay(ms)	**30**	10	5.00	230	375	*15000*	*338*	*13520*
(Bits/frame)	260	50	1.00	1030	1688	*13504*	*1651*	*13208*
G.723 - 5.3		1	33.33	75	67	*17867*	*30*	*8000*
		2	16.67	105	87	*11600*	*50*	*6667*
Coder params:		3	11.11	135	107	*9512*	*70*	*6223*
Frames/sec:	**33.3333333**	4	8.33	165	127	*8467*	*90*	*6000*
Bits/Sec	**5300**	5	6.67	195	147	*7840*	*110*	*5867*
Delay(ms)	**45**	10	3.33	345	248	*6614*	*210*	*5600*
		25	1.33	795	549	*5856*	*511*	*5451*
(Bits/frame)	159	50	0.67	1545	1051	*5606*	*1013*	*5403*
G.723 - 6.3		1	33.33	75	71	*18934*	*33*	*8800*
		2	16.67	105	95	*12667*	*57*	*7600*
Coder params:		3	11.11	135	119	*10578*	*81*	*7200*
Frames/sec:	**33.3333333**	4	8.33	165	142	*9467*	*105*	*7000*
Bits/Sec	**6300**	5	6.67	195	166	*8854*	*129*	*6880*
Delay(ms)	**45**	10	3.33	345	286	*7627*	*248*	*6614*
		25	1.33	795	643	*6859*	*606*	*6464*
(Bits/frame)	189	50	0.67	1545	1240	*6614*	*1203*	*6416*
G.729		1	100.00	30	57	*45600*	*20*	*16000*
		2	50.00	40	67	*26800*	*30*	*12000*
Coder params:		3	33.33	50	77	*20534*	*40*	*10667*
Frames/sec:	**100**	4	25.00	60	87	*17400*	*50*	*10000*
Bits/Sec	**8000**	5	20.00	70	97	*15520*	*60*	*9600*
Delay(ms)	**20**	10	10.00	120	148	*11840*	*111*	*8880*
		25	4.00	270	299	*9568*	*262*	*8384*
(Bits/frame)	80	50	2.00	520	552	*8832*	*515*	*8240*

PPP stuffing overhead
1.01

Packet Overhead:	Uncompressed	*Compressed*
PPP	6	*5*
IP	20	
UDP	8	
RTP	12	*2*
Header Refresh	0	*2*
Total	**46**	***9***

4. Call Management Agent

4.1 The CMA System

4.1.1 Overview

The call management agent system provides intelligent, communication terminal independent call management services. It is an essential link in providing the call setup information for Internet based telecom services including IP to IP, SCN to IP, IP to SCN, and SCN to SCN calls. This includes managing the various communication terminal addresses[1] of a given person or organization, the ability to provide the various dynamic mappings between addresses to allow all combinations of calls, and the ability to intelligently route calls according to agent based logic. The following scenario with the accompanying diagram portrays the underlying concept.

Suppose Jack has three communication terminals - a home phone, a business phone and an IP phone. During working hours, he wants all calls to be routed to his business phone. When he's at home, he wants all calls to be routed to his home phone, but calls from Joe to be routed to his laptop based IP phone. Alas, Jack has only a dialup account at an ISP. This means he doesn't have a fixed IP and it might be the case that he isn't online at a given time. In this case, he wants Joe's calls to be routed to his home phone number. Jack would "inject" this logic (0), along with the list of communication terminals he supports into his CMA, so the CMA will be able to perform the call routing accordingly. Now, Jack can be accessed by these various communication through a single logical address, his CMAA.

Now suppose Joe, an IP phone user, wants to call Jack. Suppose Joe doesn't have Jack's CMAA. He would thus first consult a white pages directory service, to get the CMAA (1). Now his IP phone would contact his own CMA and request it to contact Jack (2). Joe's CMA will now locate Jack's CMA, and contact it, identifying itself as Joe's CMA (3). Jack's CMA will now perform the computation according to the logic previously injected into it. If Jack's online, it will reply with the *current* IP address of Jack's IP phone. Otherwise, it will provide Joe's CMA with the E.164 telephone number which will be resolved to the appropriate IP address of the telephony gateway to call out from.

[1] *This includes the resolving **dynamic IP addresses** which are typically used by dialup Internet subscribers.*

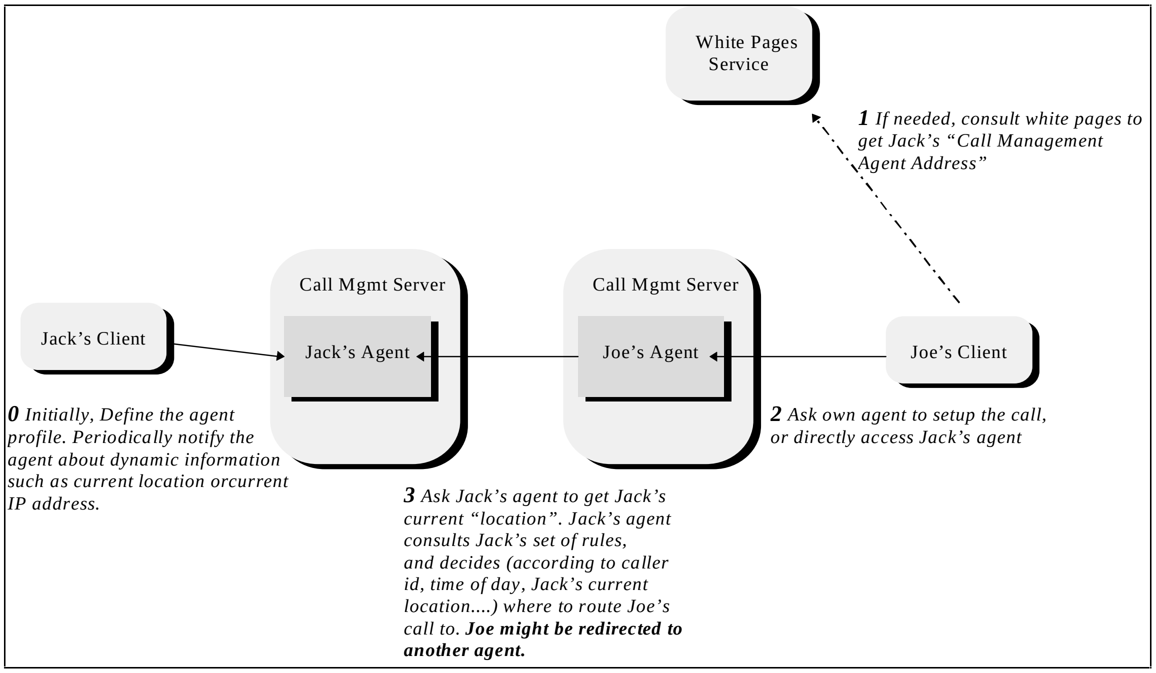

White Pages Service
1 If needed, consult white pages to get Jack's "Call Management Agent Address"
Call Mgmt Server
Call Mgmt Server
Jack's Client
Jack's Agent
Joe's Agent
Joe's Client
0 Initially, Define the agent profile. Periodically notify the agent about dynamic information such as current location orcurrent IP address.
2 Ask own agent to setup the call, or directly access Jack's agent
3 Ask Jack's agent to get Jack's current "location". Jack's agent consults Jack's set of rules, and decides (according to caller id, time of day, Jack's current location....) where to route Joe's call to. Joe might be redirected to another agent.

Figure 4-1 CMA System Sample Scenario

Given the needed call setup information, Joe's IP phone can now call the remote terminal (Be it a telephony gateway which in turn will call Jack's home telephone (e.g.,E.164) number, or Jack's actual IP phone address) directly.

4.1.2 System Operating Environment

The initial implementation of CMA system supports point-to-point communications. Interworking via H.323 gateways to SCN (Switched Circuit Network) communications equipment is provided. This operating environment is termed the VOIP Operating Environment.

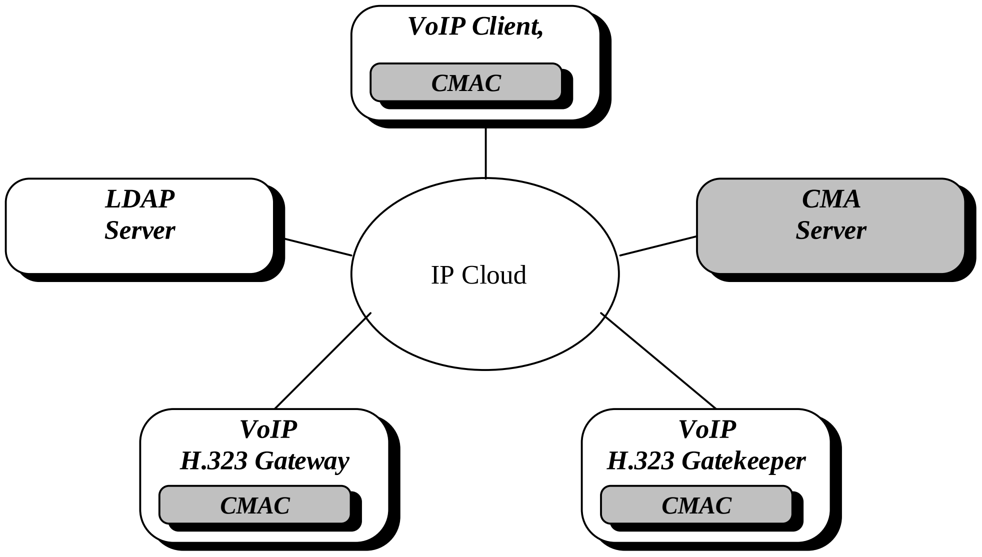


Figure 4-2 The Operating Environment

4.1.3 Terminology

Call (noun): Point-to-point multimedia communication between two Internet endpoints. The call begins with the call setup procedure and ends with the call termination procedure. The call consists of the collection of reliable and unreliable channels between the endpoints. In case of interworking with some SCN endpoints via a gateway, all the channels terminate at the Gateway where they are converted to the appropriate representation for the SCN end system.

CMA: Call Management Agent

CMAA: Call Management Agent Address. An address through which a given CMA can be located and then accessed.

CMAS: Call Management Agent Server

CMA Sys: Call Management Agent System

CMA Sys Entity: Any CMA Sys component, including client(s) and server(s).

CMAC: Call Management Agent Client

CMAP: Call Management Agent Protocol. The protocol between a CMAC and a CMAS.

CMA Logic: The computation performed by the CMA for a specific request.

Communication Terminal (CT): A terminal is an endpoint on the Internet or SCN which provides for real-time, two-way communications with another Terminal, or Gateway. This communication may consists of control, indications, audio, and/or data between the two terminals. A terminal may provide indications only, speech only, speech and data, or any combination.

CTT: Communication Terminal Type

CTA: Communication Terminal Address

CTAT: Communication Terminal Address Type

Endpoint: An H.323 Gateway, CMA client, LDAP server, or CMA Server. An endpoint can call and be called. It generates and/or terminates information streams.

Gatekeeper: The Gatekeeper (GK) is an H.323 entity on the Internet that provides address translation and controls access to the network for H.323 Terminals and Gateways. The Gatekeeper may also provide other services to the H.323 terminals and Gateways, such as bandwidth management and locating Gateways.

Gateway: An H.323 Gateway (GW) is an endpoint on the Internet which provides for real-time, two-way communications between H.323 Terminals on the Network and other ITU Terminals on a wide area network, or to VoIP Clients. Other ITU Terminals include those complying with Recommendations H.310 (H.320 on B-ISDN), H.320 (ISDN), H.321 (ATM), H.322 (GQOS-LAN), H.324 (GSTN), H.324M (Mobile), and V.70 (DSVD).

H.323 Entity: Any H.323 component, including H.323 Terminals, Gateways, Gatekeepers.

Internet address: The network layer address of a H.323 or CMA Sys entity as defined by the (inter)network layer protocol in use (e.g. an IP address). This address is mapped onto the layer one address of the respective system by some means defined in the (inter)networking protocol.

Internet: An inter-network of networks interconnected by bridges or routers. LANs described in H.323 may be considered part of such internetworks .

IVR: Interactive Voice Response.

RAS Channel: Unreliable channel used to convey the registration, admissions, bandwidth change, and status messages (following H.225.0) between H.323 entities or CMA Sys entities.

Reliable Channel: A transport connection used for reliable transmission of an information stream from its source to one or more destinations.

Reliable Transmission: Transmission of messages from a sender to a receiver using connection-mode data transmission. The transmission service guarantees sequenced, error-free, flow-controlled transmission of messages to the receiver for the duration of the transport connection.

Soft Link: A referral from one CMA to another.

Subscriber: An “owner” of a CMA. A subscriber can be a person or an organization.

Switched Circuit Network (SCN): A public or private switched telecommunications network such as the GSTN, N-ISDN, or B-ISDN.

VOIP Environment: The system of Communications Terminals, Servers, Gateways and Gatekeepers for communication over IP networks and interconnection to the SCN.

4.2 CMA System Architecture

4.2.1 Overview

The CMA system is based on having a CMA for each subscriber to the system. The CMA manages incoming and outgoing calls on behalf of the subscriber. For example, incoming calls can be automatically routed to a given communications terminal or rejected altogether according to caller id, time of day and the location of the callee.

The following diagram illustrates the relationship between the various components of the system: A Call Management Agent Server (CMAS) which manages a group of Call Management Agents (CMA's), and which can be accessed by Call Management Agent Clients (CMAC's) using the Call Management Agent Protocol (CMAP).

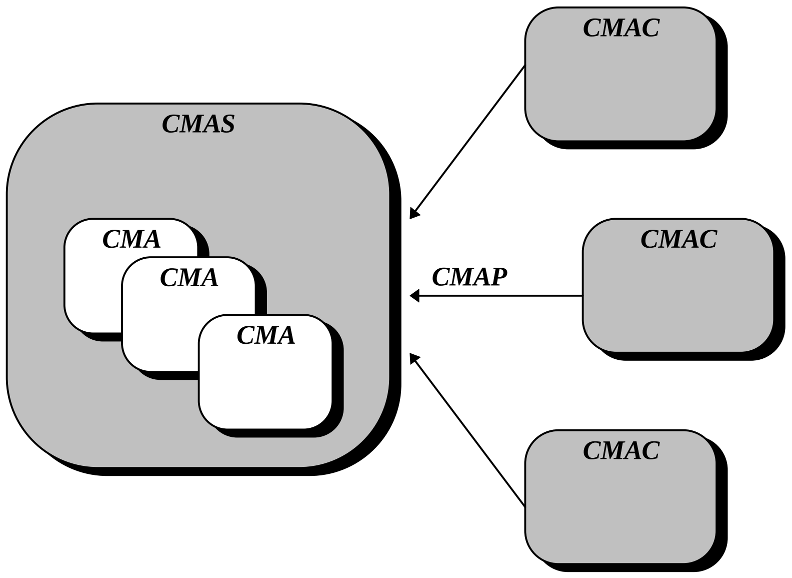


Figure 4-3 CMA System Components

The CMAC's access the CMAS in order to interact with the CMA's managed by the server. Interaction with the CMA's is for the purpose of configuring the CMA or for asking the CMA to perform some service. It is important to note that every CMAS also contains a CMAC component to be able to access other CMAS's.

The following sections define the various components of the system in detail.

4.2.2 CMA

A CMA is the agent which manages both *outgoing calls* from and *incoming calls* a subscriber's[2] set of communication terminals. The term "*manage*" in this context refers to knowing the list of the communication terminals through which this subscriber can be accessed, along with relevant attributes such as the communication addresses - And applying some logic to perform decisions as to which devices to route calls to given various input parameters such as the caller id, the time of day, the accessibility of the callee through a given communication terminal, and so forth. Typically, the communication terminal addresses would be IP addresses of IP phones, SCN #'s, Pager #'s, etc. A subscriber could have multiple communication terminals of the same type (Such as several phone numbers).

A given CMA is mainly associated with the following elements:

- A CMAS, where it resides.
- At least one *CMAA*, which is used to globaly uniquely identify it, and to locate the server where it resides.
- A *list of communication devices* which it manages, along with their various attributes which could be **dynamic**.
- *Call management logic* which is used by the agent to compute to which communication devices to route the caller to given various parameters such as the caller's id, the time of day, the current location of the callee, etc. ***The way the call management logic and its accompanying data is "injected" into a given agent is beyond the scope of this specification. It will be defined by a subsequent VoIP specification at some future timeframe.***

To contact a subscriber, one need's to contact the subscriber's CMA (Either directly or through the originator's own CMA), identify oneself (to a controled level, as one might like to be anonymous), and ask for the (set of) communication terminal address(es) one can use to contact the subscriber. The result might be one or more communication terminal addresses to call, or *a referral to another CMA* one should continue this call setup procedure with.

In some cases, one might ask to communicate with the remote subscriber on a specific type of communication terminal. This could be another (optional) input parameter to the callee's CMA that would be entered when the call is originated.

---

[2] Of course, a CMA can be associated with other entities, such as an *organization*. For sake of simplicity, we shall refer throughout the document to CMA's as belonging to a *person*.

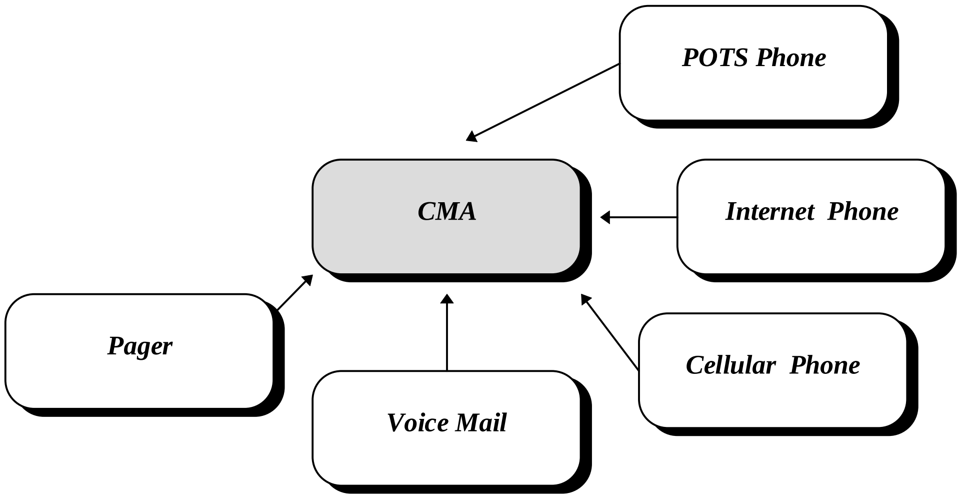


Figure 4-4 The CMA manages multiple communication terminals

The methods and information exposed by the CMA entity are specified in detail in section 4.5.

4.2.3 CMAA

A CMA is provisioned with *at least* one CMA address - Its CMAA. A CMAA is a globally unique address which is used to locate the CMA throughout the Internet (That is, locate the CMAS where the specific CMA resides at) to be able to interact with it. A subscriber to the CMA system can use his CMAA as a ***single, logical communication address,*** masking the details of the actual communication terminal used. The need for multiple CMAA addresses arises when confronted with the constraint to provide both Internet friendly and dialpad friendly addresses. Given these parameters, both RFC 822 e-mail and E.164 CMAA types are defined as part of the standard.

Please note the following regarding the E.164 CMAA:

- It could be a real SCN-allocated E.164 (A person's actual primary SCN phone number)
- It could be a E.164 # which has been allocated from a different country code (Such as an Internet country code).
- The E.164 sub-address might need to be used to provide some hints as to the location of the CMAS in charge of the CMA in question (TBD).

4.2.4 CMAS

The CMA Server (CMAS) is an entity which manages CMA's and is accessible through the CMAP. A given CMA System would typically encompass numerous CMAS's, each managing a given set of CMA's. CMAS's are accessible through the CMAP protocol which originates in CMAC's. As CMA's need to interact with other CMA's (For example, a caller's CMA must interact with the callee's CMA to resolve the needed call setup information), the CMAS includes a CMAC component through which it can interact with other CMAS's over the CMAP protocol.

A CMAS would typically provide a mechanism (which is beyond the scope of this specification) to allow one to "inject" the logic of a given CMA. This logic will control the way the CMA processes incoming and out going call requests.

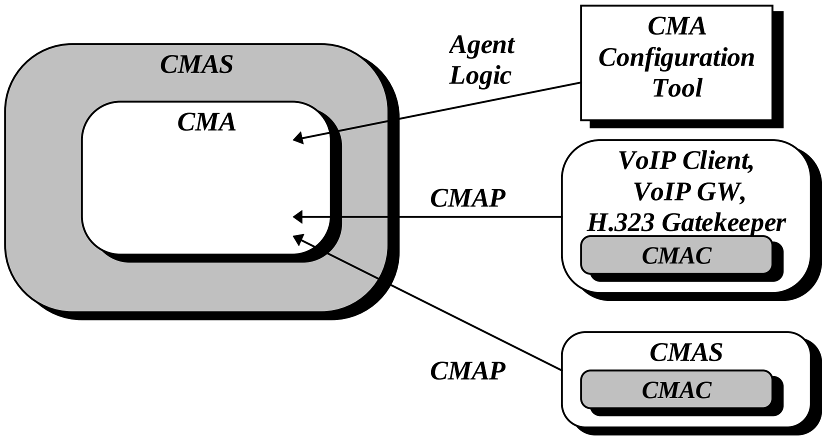


Figure 4-5 Typical CMAS interfaces

4.2.5 CMAC

A CMA Client (CMAC) is any application which interacts with the CMAS through the CMAP protocol. As illustrated in *Figure 4-5 Typical CMAS interfaces*, the following entities all fall under the CMAC category:

- VoIP client (terminal) applications
- VoIP H.323 SCN Gateways
- VoIP H.323 Gatekeepers
- CMAS servers

It is expected that the CMAC component will be embodied in the form of a library with a well defined API which host application can use to be able to interact with the CMA system. The definition of such an API is beyond the scope of this document although it might be within the scope of separate API definition documents from the VoIP forum. Such an API is expected to be derived from the method definitions of the CMA which are defined in chapter 4.5.

4.2.6 CMAP

The CMA Protocol (CMAP), is the protocol used for client to server and server to server interaction - That is, CMAC to CMAS and CMAS to CMAS[3]. The protocol syntax and transport are defined in chapter 4.6.

4.3 CMA Security

The following chapter deals with the various security issues relating to the CMA system. Please note that the basic design guideline was to reuse as much as possible from existing standards in this area.

4.3.1 Authentication

Authentication is a crucial security element of the CMA system, playing a role in:

- CMAS and CMA access control
- CMA call routing computations based on caller id

The authentication element is used to authenticate the identity of a CMAC contacting a CMAS - Either in the case of a client application contacting its CMA or a caller’s CMA contacting the callee ‘s CMA to gather the needed information to setup the call.

It is important to state that the following issues are *not* within the scope of the CMA system authentication element:

- Authenticating the identity of a client application to another client application (This is in the domain of the session control protocol)[4].

Authentication will be provided by using the SSL secure transport layer (*or some other equivalent such as PCT*). This will also provide a framework for encryption (TBD).

Please note that when a given CMAC contacts a CMAS, it must know beforehand if the transport channel over which it is going to contact the CMAS has to be secure or non secure. For this purpose, different well known ports will be used for secure and non secure access.

4.3.2 Digital Signatures

Information stored in the CMAS could be optionally digitally signatured using X.509 compliant certificates. This mechanism is not part of the VoIP baseline and will be defined in the .next major IA phase.

---

[3] *Due to the symmetric system architecture, the latter is actually the same as CMAC to CMAS.*

[4] *As both the caller’s CMA and callee’s CMA were involved in the call, it’s quite possible to be able to have the CMA system generate certificates or tokens for a given call so both caller and callee can authenticate the identity of the remote party.Tthis issue is extremely important for controlling access to TGWs.*

4.3.3 Token Based Access Control

A CMAS could optionally generate an *access token* to be used by the calling client when accessing a service such as a TGW. This token would then have to be passed on through the session control protocol between the two call endpoints.

The exact mechanism for the token generation is TBD.It could be defined in the form of an optional *Token Granting Agent* profile.

4.4 CMA & External Interfaces

The following chapter describes the mechanisms through which the CMA system interfaces with other related systems.

4.4.1 Directory Services: LDAP, X.500

Directory services are needed for the management of static, white-pages like information. The interface between the standard directory services systems such as LDAP and X.500 to the CMA system will be accomplished by extending the standard directory service "user" object schema to including the following additional fields:

- The person's CMAA.
- An optional pointer to the location of the CMAS in charge of the specific CMA.

Given these additional fields, one will be able to query a directory service and locate the CMAA of the callee, and then go through the chain of CMA related actions to setup the call.

The additional fields syntax is TBD.

4.4.2 Gatekeeper

The Gatekeeper, which is optional in an H.323 system, provides limited call control services to H.323 Entities, including the following primary services:

- Address Translation
- Admissions Control
- Bandwidth Control

In order to provide CMAC-less H.323 clients (client = terminal or gateway) with the basic needed call information, CMA compliant H.323 Gatekeepers will be needed. A CMA compliant H.323 Gatekeeper will have an H.323 Gatekeeper interface to interact with H.323 clients, and a CMAC to interact with CMAS's. It will basically translate H.323 RAS protocol requests to/from CMAP requests, allowing seamless integration of pure H.323 clients with the CMA system. The level of functionality which pure H.323 clients will receive from the CMA system is obviously limited to the H.323 Gatekeeper access protocol's syntax and semantics.

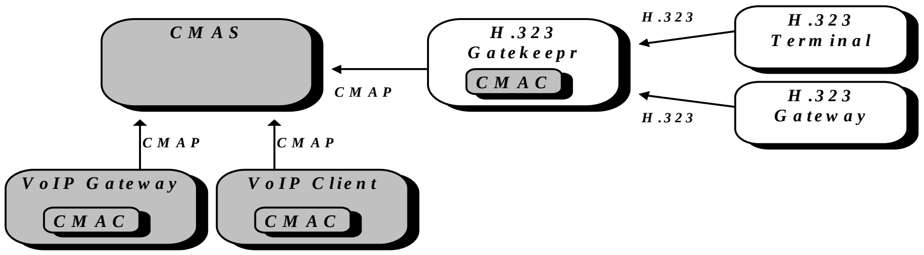


Figure 4-6 H.323 Gatekeeper Interface

The protocol conversion semantics are defined in section 4.7. The mappings will probably be based on the H.323 RAS **destinationInfo** field.

4.4.3 VoIP/H.323 Gateway

The H.323 Gateway provides the appropriate translation between transmission formats (for example H.225.0 to/from H.221) and between communications procedures (for example H.245 to/from H.242). The Gateway also performs call setup and clearing on both the Internet side and the SCN side. Translation or conversion between video, audio, and data formats may also be performed in the Gateway. In general, the purpose of the Gateway is to reflect the characteristics of a Internet endpoint to an SCN endpoint, and the reverse, in a transparent fashion.

In the VoIP context, the gateway would focus on translating Internet voice calls to SCN voice calls and vice versa (This would also enable SCN to SCN calls through the Internet). A VoIP gateway will basically be an H.323 based gateway with the addition of a CMAC component, enabling it to provide CMA services for calls originating in the SCN domain. Therefor, the gateway extends the types of communication terminals which can use the services of the CMA system to standard SCN terminals such as PSTN telephones.

The SCN user interface provided by the SCN side of the gateway is beyond the scope of this document. Typical user interfaces will be IVR based.

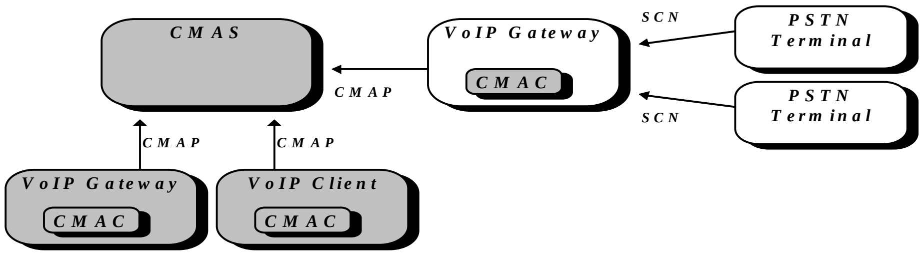


Figure 4-7 H.323/VoIP Gateway Interface

4.5 CMA Semantics

The following chapter defines the functional and data semantics of the Call Management Agent.

The data maintained by a Call Management Agent consists of the following:

- One or more *CMAAs* (Call Management Agent Addresses) which distinguish this agent from all other agents.
- If this Call Management Agent is simply a redirection pointer or *Soft Link* to another CMA, the CMA contains the CMAA of the target being redirected to. This allows a user to gracefully change his Call Management Agent. For example, this capability would be used to leave a forwarding pointer for a user who changes his E.164 telephone number, or his provider-based E-mail address.
- If this is a terminal Call Management Agent representing a callee, the CMA contains a set of *Communication Terminal Specifiers* (CTSPECs), one for each of the communication devices a VoIP client operating on behalf of this user's CMA has registered with the Call Management Agent Server.

These data types are defined in more detail in the following subsections. CMA Interface methods are defined beginning in section 4.5.2. The methods are first defined as an informal API-like specification, followed (in section ??) by the formal protocol definitions which constitute the message exchange used to perform the method.

4.5.1 CMA Object Instance Data

4.5.1.1 CMAA

A Call Management Agent Address is the fundamental data type which unambiguously identifies this agent. It is a "Name" for the CMA. A Call Management Agent may be identified by more than one address; all of the CMAAs associated with one CMA are considered synonymous. The architecture permits extensible forms of CMAA, however, initially we define two forms which must both be supported by all Call Management Agent Servers. These are:

1. An RFC822 compliant email address, or
2. A fully-qualified E.164 telephone number

Note that either of these is permitted in the **destinationInfo** field of an H.323 gatekeeper interaction.

4.5.1.2 Target CMA

If this Call Management Agent is a *SoftLink*, the Target CMA contains the CMAA which should be returned to a client as the Call Management Agent to contact instead of this CMA. If a given CMA contains both a Target CMA *and* one or more CTSPECs (see 4.5.1.3) , the target CMA should be ignored and this CMA be used in answering queries from CMA clients. This behavior allows relatively straightforward failover-based call forwarding in the absence of a more sophisticated agent which performs intelligent call forwarding.

4.5.1.3 CTSPEC

Communication Terminal Specifications are used by VoIP clients to inform their Call Management agents what their active communication devices are, and what address(es) are currently associated with each terminal. The architecture allows both multiple devices per call management agent and multiple equivalent addresses per terminal. The former capability allows a user to consistently manage multiple devices which may be controlled by different VoIP clients. The latter capability allows for improved availability and performance by permitting a given terminal to be reached multiple ways.

A CTSPEC contains the following information:

4.5.1.3.1 GUID

The Globally Unique Identifier (GUID) is an identifier which distinguishes this CTSPEC from all others both spatially and temporally. This identifier is generated by the VoIP client which controls or "owns" the communication terminal. Since a GUID generator is expected to be available on most if not all target VoIP platforms, this identifier is a GUID as specified in *(cite: Microsoft GUID Specification here??)*.

GUIDs are generated by clients. To prevent aliasing among identical communication devices, a client is expected to remember the GUID for its communication terminal for at least as long as the TTL for the CTSPEC (see section 4.5.1.3.2), even in the event of a crash.

4.5.1.3.2 Time-to-Live

Each CTSPEC contains a TTL which is specified by the client and maintained by the Call Management Agent. The client refreshes the TTL by performing a **KeepAlive** method (see section 4.5.4.2). If the TTL expires, the CTSPEC is silently deleted by the CMA. The TTL is a signed 32 bit integer, in units of seconds. A value of -1 indicates an *infinite* TTL. A CTSPEC with an infinite TTL will never be deleted by the CMA. Infinite TTL's are useful for communication terminals with static addresses such as POTS telephones and cellular telephones.

4.5.1.3.3 Terminal Category and Usage Qualifier

The terminal category is a string which describes the general *kind* of communication terminal the CTSPEC refers to. The string is meant to be descriptive and vendor extensible, but with some structure and conventions that enable parsing and interpretation by either VoIP clients or CMA methods that want to do clever things based on the kind of terminal the VoIP client would be communicating with. In order to enforce some basic functionality and consistency in the first deployment of VoIP systems, however, we define a few common terminal categories which all servers and most clients are expected to understand. These are specified below in Table 1:

Table 1: Terminal Categories

Terminal Category	Description
IP-Phone	A computer-based H.323 audio-only client
Telephone	A normal telephone handset, either POTS, Cellular or ISDN based using E.164 addressing
Fax	A G3 facsimile machine
Pager	A numeric or alphanumeric pager
Voice-Mail	A message-taking service such as a classic PBX-based voicemail system, an H.323 client which records audio messages, etc.
Attendant	A receptionist, secretary, or automated attendant.

These basic categories may be augmented by adding a usage qualifier to the string to further qualify the kind of terminal. As with basic terminal categories, usage qualifiers are vendor extensible, with a few qualifiers specified here for the most common situations. These are defined below in Table 2:

Table 2: Usage Qualifiers

Usage Qualifier	Description
Fixed	Wired, or stays at one location
Mobile	Wireless, or moves frequently from location to location
Business	Used for business communication
Personal	Used for personal communication

This combination of terminal category and terminal qualifier allows devices to be usefully categorized to distinguish, for example, *Personal-Voice-Mail*, from *Business-Voice-Mail*, or a *Business-Mobile-Telephone* from a *Business-Fixed-Telephone.*

4.5.1.3.4 Terminal Addresses

Associated with a CTSPEC is a set of terminal addresses (CTAddress) that can be used to communicate with that terminal. The addresses contain all of the information needed by the session protocol to initiate a session with the terminal, either directly as in the case of an IP phone application or H.323 conferencing application, or indirectly through a VoIP gateway to a PSTN or other telephony service. A CTAddress consists of a pair {address-type, value} as defined below. Where a CTSPEC contains more than one CTAddress, VoIP clients wishing to communicate with that terminal will interpret the set of addresses as being:

a) *equivalent* - all the addresses in fact reach the same terminal, and

b) *ordered* - the addresses are in decreasing order of preference from the point of view of the call management agent returning the information.

This allows a VoIP client, in cooperation with a Call Management Agent Server, to provide capabilities such as priority ordering of terminal addresses or load balancing among gateways (by explicitly randomizing the list order each time it is returned to a client).

Address types specify the syntax and usage for the CTAddress. An Address Type is a string, to allow for easy extensibility to new addressing forms. In order to enforce some basic functionality and consistency in the first deployment of VoIP systems, however, we define a few common address types which all servers and most clients are expected to understand. These are specified below in Table 3.

Table 3: CTSPEC Address types

Address Type	Value Syntax	Description
H.323	H.323 **transportAddress**	The destCallSignalAddress of an H.323 compliant VoIP client terminal (or VoIP gateway to the terminal)
E.164	digit string	A fully qualified E.164 telephone number
DNS	ASCII string	an IP host name or RFC822 email address. This can also be used as the destinationInfo for initiating an H.323 session.

?? In the DNS case, the address value needs to optionally convey the PORT and transport type - TCP, UDP, Secure TCP, etc.

4.5.1.3.5 Capability List

Associated with a CTSPEC is a capability list for the corresponding terminal. The capability list is intended to give the general capabilities of the terminal, so a potential caller can decide whether or not it is worth trying to communicate with the end user through this terminal. Therefore, it is used to indicate such things as "video capable", or "data conferencing capable" rather than to replace low level or detailed capabilities exchange such as is done via H.245 at session establishment time. A capability list is a string, to allow for easy extensibility to new addressing forms. The string syntax is a comma-separated list of tokens, where each token can be either:

- a keyword, such as "video", or
- a keyword=value

In order to enforce some basic functionality and consistency in the first deployment of VoIP systems, however, we define a few common keywords which all servers and most clients are expected to understand. These are specified below in Table 4.

Table 4: Capability list keywords

Keyword	Description

Video	The terminal can do real time video
Data	The terminal can do T.120 compliant data conferencing
ReceiveOnly	The terminal is only capable of receiving (e.g. a pager, radio, television, etc.)

4.5.2 Call Management Agent Method Summary

Table 5 below summarizes the methods which operate on a CMA and cross-references the section in which the method is defined.

Table 5: CMA Method Summary

Method	Description	**Reference**
CreateCMA	Creates a new Call Management Agent	4.5.3.1
RemoveCMA	Remove an existing Call Management Agent	4.5.3.2
AddSynonym	Adds a synonymous address to a Call Management Agent	4.5.3.3
RemoveSynonym	Removes an address from a Call Managment Agent and deletes the CMA if this is the last address associated with the CMA.	4.5.3.4
SetTargetCMAA	Associates a *SoftLink* target Call Management Agent with a CMA.	4.5.3.5
GetTargetCMAA	Returns the *SoftLink* target Call Management Agent Address associated with a CMA, if one exists	4.5.3.6
UnSetTargetCMAA	Disassociates a *SoftLink* target Call Management Agent Address, if one exists, from the CMA	4.5.3.7
SetCTSpecifier	Enters one or more new Communication Terminal Specifiers for a Call Management Agent, or replaces the data for an existing CTSPEC.	4.5.4.1
KeepAlive	Performs a refresh of the Time-To-Live of an existing Communication Terminal Specifier.	4.5.4.2
Resolve	Obtains the set of usable Communication Terminal Specifiers that the calling client can use to contact the VoIP client(s) associated with a given CMA.	4.5.5.1
GetSynonyms	Returns all of the Call Management Agent Addresses currently associated with the specified CMA.	4.5.5.2

4.5.2.1 Methods calling structure

All methods calls are asynchronous: an **Activate Method Request** is sent to the server, and then the server sends a **Reply** for that request.

The request always specifies the **Agent** to operate on, along with method-specific properties.

The reply return a **status**, which is either a success code, or a reason why the method was not activated. If that status is success, then a property list is returned (depending on the specific method).

4.5.2.1.1 Network Packing

CTSpec structures are packed into a **packedBSTR** property type when passed as a parameter:

- **guid** is copied as it is, as 16 bytes array
- **ttl** - 4-byte signed integer, in network byte ordering (MSB first)
- **category** - a zero-terminated string.
- **qualifiers, capabilities** - These are comma-seperated lists. They are packed each as a zero-terminated strings, just after the category string.

When packing CTSpecs into an array, a single **packedPROPLIST** is used. This Property List holds a list of **packedBSTR**, each representing a single CTSpec.

Similarly, arrays of other basic types (e.g. Strings) are a **packedPROPLIST** of elements of that particular type.

4.5.3 Methods for Creating and Maintaining CMAs

The following methods are used to create a CMA, remove a CMA, associate or disassociate a synonymous CMAA with a CMA, and create or remove a CMA *Softlink.*

4.5.3.1 CreateCMA

A new CMA is created using this method and providing a CMAA which identifies this particular call management agent. The CMA continues to exist until the last CMAA is disassociated with is (see section 4.5.3.2 for more information).

CreateCMA is implemented using **CreateInstance** PDU. The status of the creation is returned by the **CreateInstanceReply** PDU.

4.5.3.2 RemoveCMA

A CMA is removed using this method and providing a CMAA which identifies this particular call management agent.

DeleteCMA is implemented using **DeleteInstance** PDU. The status of the deletion is returned by the **DeleteInstanceReply** PDU.

4.5.3.3 AddSynonym

This method adds a CMAA to an existing CMA. The method is *idempotent* — it returns success whether or not the synonymous CMAA is already one of the CMAAs associated with this CMA.

4.5.3.3.1 AddSynonym Request:

The **dest** is the agent we want to add a synonym to

Properties	Value Type	Description
newCMAA	String	a synonymous CMAA to be associated with the agent

4.5.3.3.2 AddSynonym Reply:

Properties	Value Type	Description
Errcode	Integer	Possible values:
	(present only if call failed)	• **retConflictingCMAA** - A different agent with this CMAA already exists.

4.5.3.4 RemoveSynonym

This method removes a CMAA from an existing CMA. If this is the only CMAA currently associated with a CMA, the CMA object is deleted as a side-effect.

4.5.3.4.1 RemoveSynonym Request:

The **dest** is the agent we want to remove the synonym of.

Properties: none

4.5.3.4.2 RemoveSynonym Reply:

If the command was successful, the following properties are returned:

Properties	Value Type	Description
Deleted	Boolean	set to TRUE if last synonym of this CMA was removed

4.5.3.5 SetTargetCMAA

This method associates a *SoftLink* target CMAA with a CMA. If a target CMAA is present in a call management agent and the agent has no current CTSPECS associated with it, a query on this CMA will return the target CMAA in a *redirect* to the requesting client. Note that the target CMAA does not necessarily have to exist either at the time this call is made, nor later, since multiple servers may be involved and dangling pointers are always a hazard in such systems.

4.5.3.5.1 SetTargetCMAA Request:

The **dest** is the agent we want to redirect.

Property	Value Type	Description
targetCMA	String	The target CMAA this CMA is to be *SoftLinked* to.

4.5.3.5.2 SetTargetCMAA Reply:

The reply status informs whether the operation was successful or not.

Properties: none

4.5.3.6 GetTargetCMAA

This returns the *SoftLink* target Call Management Agent Address associated with a CMA, if one exists.

4.5.3.6.1 GetTargetCMAA Request:

The **dest** is the agent we want to query.

Properties: none

4.5.3.6.2 GetTargetCMAA Reply:

Properties	Value Type	Description
targetCMA	String	The target CMAA this CMA is to a *SoftLinked* to, if such softlink is set. No targetCMA is returned if no link exist.

4.5.3.7 UnsetTargetCMAA

This method disassociates a *SoftLink* target CMAA, if one exists, from the CMA.

4.5.3.7.1 UnsetTargetCMAA Request:

The **dest** is the agent we want to stop redirect.

Properties: none

4.5.3.7.2 UnsetTargetCMAA Reply:

The reply status informs whether the operation was successful or not.

Properties	Value Type	Description
targetCMA	String	The target CMAA this CMA was to a *SoftLinked* to, if such softlink was set. No targetCMA is returned if no link existed

4.5.4 Methods for maintaining CTSPECS

The following methods are used to maintain CTSPECS. We assume that each VoIP client has access to a local *GUID* generator that can be used to get that GUID for a new CTSPEC, and that the client has sufficient non-volatile storage to remember a GUID for at least the time period he specifies as the **TTL** of the CTSPEC.

4.5.4.1 SetCTSpecifier

This method enters one or more new CTSPECs for a CMA, or replaces the data for an existing CTSPEC. The CTSPEC to be created or modified is identified by the **GUID** field in each **CTSPEC** argument.

A CTSPEC can be deleted using this call by specifying a **TTL** of zero for the CTSPEC.

If the call fails, *none* of the requested CTSPECs have been either added, modified, or removed.

4.5.4.1.1 SetCTSpecifier Request:

The **dest** is the agent we want to set.

Properties	Value Type	Description
ctSpecs	Array of CTSpec	The list of CT specifiers which can be used to communicate with the callee.

4.5.4.1.2 SetCTSpecifier Reply:

Properties: none

4.5.4.2 KeepAlive

This method performs a refresh of the TTL of an existing CTSPEC. It performs two related functions for VoIP clients:

1. It provides a low-overhead method of refreshing the **TTL** of a CTSPEC, since it is intended to be more efficient than periodically doing a **SetCTSpecifier**.
2. If provides a VoIP client with some assurance that it has sufficient network connectivity to be able to receive calls.

Note that the server may reply with a different ttl value. The client is responsible to honor this server's request, and use that new (usually longer) **ttl** for future **KeepAlive** calls.

4.5.4.2.1 KeepAlive Request:

The **dest** is the agent we want to set.

Properties	Value Type	Description
guid	Binary	The GUID of the CTSPEC to be refreshed.

4.5.4.2.2 KeepAlive Reply:

Properties	Value Type	Description
ttl	Integer	a new TTL to used on future **KeepAlive** calls for this CTSPEC.
Errcode	Integer (present only if call failed)	Possible values: • **noSuchCTSPEC** - No CTSPEC associated with this CMA has a matching **guid**.

4.5.5 Methods for Obtaining information from a CMA

VoIP clients wishing to make calls use the following methods to obtain information about other VoIP endpoints, and especially to obtain the (often dynamic) addressing information needed by the session protocol to establish communication with another VoIP endpoint.

4.5.5.1 Resolve

This method obtains the set of usable CTSPECS that the calling client can use to contact the VoIP client(s) associated with a given CMA.

Note that this is a method of the **CALLER's** agent: This agent, which resides on the user's home-server performs the actual resolving of the remote agent's address.

The method returns the subset of the callee CMA's CTSPECS deemed usable by both the callee's and the caller's CMAs. As noted earlier, this list is to be treated as ordered in decreasing preference.

A CTSPEC returned by **Resolve** may be cached by a client no longer than the **TTL** in the CTSPEC.

4.5.5.1.1 Resolve Request:

The **dest** is the caller's agent.

Properties	Value Type	Description
calleeCMAA	String	The address of the requested callee's CMA. Note that a different value may be returned (see in the reply, below)
anonymousCall	Boolean	Set to TRUE to request that the caller's identity will not be revealed to the callee.

4.5.5.1.2 Resolve Reply:

Properties	Value Type	Description
calleeCMAA	String	the actual callee's address. The server might return a different CMAA from the original one's, if a *Softlink* is traversed

ctSpecs	Array of CTSPECS	an array of CTSpec which can be used to communicate with the callee identified by the **calleeCMAA**
errcode	Integer	Possible values:
	(present only if call failed)	• **calleeNotFound -** The calleeCMAA does not exist or cannot be located.

4.5.5.2 GetSynonyms

This method returns all of the CMAAs currently associated with the specified CMA.

4.5.5.2.1 GetSynonyms Request:

The **dest** is the agent to query

Properties: none.

4.5.5.2.2 GetSynonyms Reply:

Properties	Value Type	Description
synonymCMAAs	Array of Strings	The complete set of synonymous CMAAs currently associated with the agent

4.6 CMA Protocol Syntax

4.6.1 Introduction

The following chapter provides the ASN.1 syntax specification of the CMA Protocol used for client-server and server-server communication.

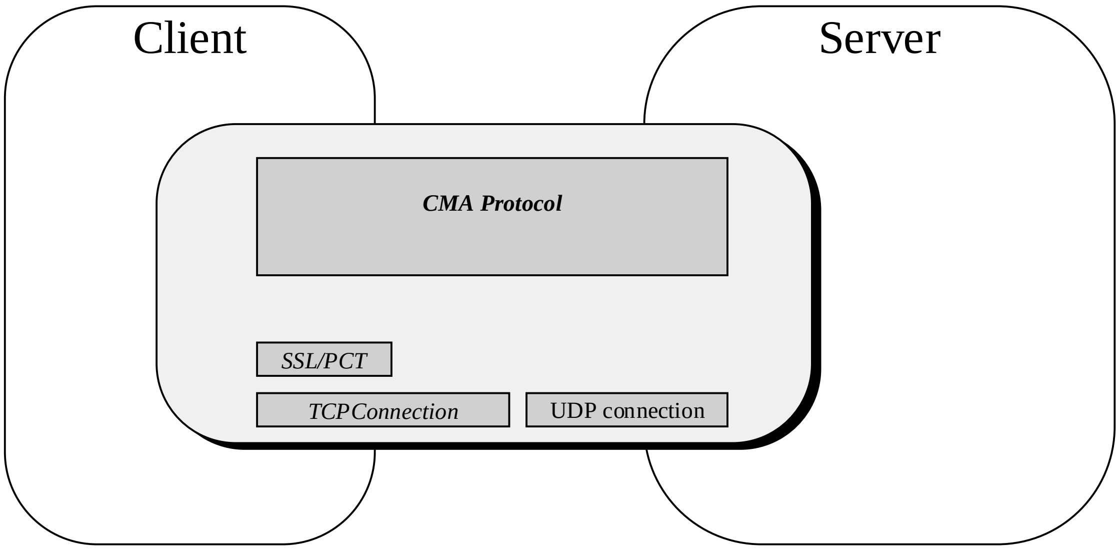


Figure 4-8 The CMAP stack layout

The *CMAP Protocol* is a simplified protocol for agent activation. This protocol can be considered the “carrier” protocol in the sense that it provides an extensible platform for accessing additional types of agents beyond the pure Call Management Agent.

4.6.2 Protocol Transport

Currently, the protocol is defined to work on TCP/IP connections to the well know TCP port (*TBD)*.

A UDP based protocol might also be used. The UDP packet would contain the list of commands to execute. Logically, the connection is opened and closed in a single request. In general, UDP should be limited to short query operations because of its unreliable nature.

A PCT/SSL-based transport will be used for authenticated and secured connections. Secure connections will work over the well known TCP port (*TBD)*.

After a connection is established, and the connect command is recognized, the application layer will check the authentication and encryption used to open the channel. If they are insufficient, the request will be rejected. In this case, the client will either fail, or re-open the connection with a higher level of authentication.

4.6.3 General Protocol Notes

Generally, the PDUs (Program Data Units) used in the protocol suite are divided into two types - requests and replies. All requests have a request ID and all replies carry within them the request ID so the reply could be associated with the request it answers.

All packets arriving at an CMAP server should be of a type called CMAPCOMMANDS. The packet is described in the ASN.1 notation below:

```
CMAPCOMMANDS ::= CHOICE
```

```
{
       -- command PDUs:
       getprop-pdu                     GETPROP-PDU,
       setprop-pdu                     SETPROP-PDU,
       activatemethod-pdu              ACTIVATEMETHOD-PDU,
       createinstance-pdu              CREATEINSTANCE-PDU,
       deleteinstance-pdu              DELETEINSTANCE-PDU,

       -- reply PDUs:
       redirect-pdu                    REDIRECT-PDU,
       getprop-reply-pdu               GETPROP-REPLY-PDU,
       setprop-reply-pdu               SETPROP-REPLY-PDU,
       activatemethod-reply-pdu        ACTIVATEMETHOD-REPLY-PDU,
       createinstance-reply-pdu        CREATEINSTANCE-REPLY-PDU,
       deleteinstance-reply-pdu        DELETEINSTANCE-REPLY-PDU,

}
```

The following chapters will describe the protocols by specifying in detail each of the ASN.1 PDUs used in each protocol.

4.6.4 Connect sequence

4.6.4.1 Overview

Before any CMAP operation could be performed, a CONNECT-PDU must be sent by the client.

It contains the protocol identifier used by the client. The server decides based on that information whether or not to accept the attempted connection, and which level of protocol to expose to the client.

The client does not wait for a reply for the Connect command - it immediately sends the first command to the server.

There is no “ACK” for the connect request. If the server replies to the command following the Connect, then the connect was accepted. If the first reply was CONNECT-NACK, then it means no further command following the Connect will get a reply.

4.6.4.2 PDU structure

4.6.4.2.1 CONNECT-PDU

```
CONNECT-PDU ::= SEQUENCE {
      protoIdent     OBJECT IDENTIFIER,
      application    OCTET STRING OPTIONAL,
      nonStandard SEQUENCE {
         id OBJECT IDENTIFIER,
         …
      } OPTIONAL
}
```

As explained above, the connect pdu should be the first pdu to arrive from a client on every attempted CMAP session (otherwise the server will automatically reject the calling client). The connect PDU contains the prodocol version used, and optionally client **application** information.

A non-standard structure allows for vendor-specific feature announcement.

The **protoIdent** protocol identifier should be **??voip.cma.protocol.major(1).minor(0)??.**

The protocol minor version will be incremented with each update of the protocol, while a compatibility is maintained with previous version. When the new version is incompatible with the previous one, the major version will be incremented. Thus, the general rule is that a server will accept a command with the same protocol major version, regardless of the minor version. It still might use the minor version code to "optimize" its replies to this level. If the server supports a higher major version, but still support previous one, (for backward compatibility), it will reply with a major version the same as the client's, possibly with a higher minor version.

Note that there is no **connect - ack** PDU. If the server answers a request, the the connect was successful.

If the connect protocol decides to reject the attempted connection (usually because of incompatible major versions) then a CONNECT-NAK-PDU is sent.

4.6.4.2.2 CONNECT-NAK-PDU

```
CONNECT-NAK-PDU ::= SEQUENCE {
      reason ENUM ( version-mismatch, busy, access-denied )
}
```

The CONNECT-NAK pdu is sent when the connect protocol decides to reject an attempted connection.

The reason of the rejection is sent in the **reason** field.

4.6.5 Basic CMAP operations

4.6.5.1 Overview

Generally, the CMAP protocol provides the application layer with the ability to perform the following operations on an agent residing on the server's agent directory:

- Create a new instance (of a given agent class)
- Delete an instance
- Get data properties of a given instance
- Set data properties of a given instance
- Activate a method of a given instance (with a set of given parameters)

Each of these operations is basically composed of a *request* sent to the server side, and a corresponding asynchronous *reply* sent from the server back to the client.

An additional important reply is the *redirect* reply, which is used by the server to refer requesting clients to other servers, or to another destination object.

All request packets have the following common fields:

- Each pdu starts with a **Protocol identifier**. This identifier should be the same as used by the Connect command (although it can use a sub-protocol of that major-version)
- The ***Agent Instance Id*** which identifies the agent instance on which the operation should be performed.
- The client ***Request Id*** which is used by the client to uniquely identify the outgoing request so it could handle multiple asynchronous requests on a single connection.
- An optional ***Session Id*** which is used for synchronization purposes. It is created and supplied by the server side, through application specific means (By using the *activate method* command). The need for such an id arises in situations where multiple copies of the **same** agent instance can exist in different, replicated servers. This allows the client and server to make sure they are speaking about the same physical agent instance. The client should save the last session-id returned by the server, and return it to the server on future invocations.
- A **destination**, which is the agent on which to operate, or to active the method of. The value of this parameter is carried on from previous command (on the same connection), so it need to be specified only once in a series of commands on the same object.

Please note that the basic CMAP protocol is generic and doesn't discuss any *agent class specific* definitions. Profiles of specific agent class definitions should define:

- Which methods are used (and their corresponding parameters).
- Which result codes are used.
- Which attributes are used.

4.6.5.2 PDU Structure

4.6.5.2.1 CMAP Agent Identifier

A CMAP agent identifier is built out of 3 fields:

- **class** - the actual class of this agent, whether it is a call management agent, a token generation agent, etc. The actual classes are defined by a given, specific agent profile. In this context, we'll be defining the call management agent profile in chapter ??.
- **namespace** - an indentifier for the namespace used by the actual instance address.
- **instance** - the actual instance address, within the current namespace.

Both class, subclass and namespace are OBJECT IDENTIFIERS, defined under the VoIP OID hierarchy.

The instance address is a string (e.g: E.164 phone number, H.323-id).

4.6.5.2.2 General ASN.1 definitions

4.6.5.2.2.1 General

```
-- a basic definition of a string in the CMAP
CMAPString ::= IA5String,


-- CMA address: either H.323-id or E.164 number.
CMAA ::= CMAPString,
```

4.6.5.2.2.2 Agent identifier

An Agent Identifier is used to identify the agent that will be used to carry on the given command. The agent is identified by its **class** and **instance**. The instance id is the unique instance identifer of this agent within this class. The basic agent class defined in the context of this document is of class **call manager agent**. The instance id of the call management agent is its CMAA.

An optional **subclass** vendor-specific identifier may be used to request specific behaviour from this class.

```
rfc822Namespace OBJECT IDENTIFIER ::= { ?? voip . cma . namespace . 1 },
e164Namescape OBJECT IDENTIFIER ::= { ?? voip . cma . namespace . 2 },
classCallMgmtAgent OBJECT IDENTIFER ::= { ?? voip . cma . class . callmgmt(1)
},

-- CMAP agent identifier
CMAPAgent ::= {
       class      OBJECT IDENTIFIER,
       subclass   OBJECT IDENTIFIER OPTIONAL,
       namespace  OBJECT IDENTIFER,
       instance   CMAPString,
}
```

4.6.5.2.2.3 Properties

Each command caries a list of properties. Each property has its id (**PID)** and its **Value**.

PID is a 32-bit integer code.

A "request" command carries a **PIDList**, which is an array of such ids.

A "reply" command carries a **PackedPropertyList**, which holds the Ids with their values.

The following ASN.1 definitions define the various support Property value types:

```
-- property identifier: 32bit integer
PID ::= INTEGER,

-- supported value types for properties:
PackedValue ::=
CHOICE {
      -- value type is unknown
       packedUNKNOWN  [0]  IMPLICIT INTEGER,

      -- this indicates that the type is valid, but the value isn't
       packedInvalid  [1]  IMPLICIT NULL,

       packedLPSTR         [2]  IMPLICIT OCTET STRING,
       packedBYTE          [3]  IMPLICIT INTEGER,
       packedWORD          [4]  IMPLICIT INTEGER,
       packedDWORD         [5]  IMPLICIT INTEGER,
     packedBSTR          [6]  IMPLICIT OCTET STRING,
     packedBOOL          [7]  IMPLICIT INTEGER,

     -- for a nested PROPLIST inside a PROPLIST
     packedPROPLIST [12] PackedPROPLIST
}


PackedProperty ::= SEQUENCE {
     pid   PID,
     value PackedValue
}

PIDList ::= SEQUENCE OF PID,
PackedPropertyList ::= SEQUENCE OF PackedProperty,

-- various error status codes
PropertyStatus ::= ENUM (
             ok(0),
             not-found,          -- requested agent does not exist
             server-down,        -- unable to connect to target server
             no-authentication   -- the server cannot authenticate the request
             no-access,          -- no premission for that request
             unkown-error )
```

4.6.5.2.3 GETPROP-PDU

```
GETPROP-PDU ::= SEQUENCE
{
     protoIdent    OBJECT IDENTIFIER,
     dest     CMAPAgent,
     req-id   INTEGER,
     pidlist  PIDList,
      session-id     SessionID OPTIONAL
}
```

The getprop pdu is used to get a list of attributes associated at a specific time with a specific instance of a CMA agent.

The information describing the requested CMA agent instance is passed as the ***dest*** (destination) field. The request is also assigned a request number which is passed as the ***req-id*** (as described in the overview section above). The **pidlist** field contains a list of the ids of the agent's properties which we want to know.

The Server replies to the getprop query with a getprop-reply pdu as described below:

4.6.5.2.4 GETPROP-REPLY-PDU

```
GETPROP-REPLY-PDU ::= SEQUENCE
{
      protoIdent     OBJECT IDENTIFIER,
      dest     CMAPAgent OPTIONAL,
      reply-id INTEGER,
      status   PropertyStatus,
     props    PackedPropertyList,
     session-id     SessionID OPTIONAL
}
```

As before the getprop-reply-pdu contains the **dest** fields which contains the information about the CMA agent instance whose properties are returned.

The **reply-id** is exactly the same as the requst-id (req-id field) of the getprop-pdu which we are answering.

The **status** field indicates whether or not the query operation was successful. The status field indicates either success, failure or partial-success (not all requested properties could be retrieved).

The **props** field contain a list of all the requested properties (with their values) which could be retrieved from the server.

As explained in the overview, if this operation is related to a specific session (i.e. to a specific "login" operation) it may contain a **session-id** field, indicating to which session this pdu is related.

4.6.5.2.5 SETPROP-PDU

```
SETPROP-PDU ::= SEQUENCE
```

```
{
     protoIdent    OBJECT IDENTIFIER,
     dest    CMAPAgent OPTIONAL,
     req-id  INTEGER,
    props   PackedPropertyList,
    session-id      SessionID OPTIONAL
}
```

The setprop-pdu is used to set a set of properties of an CMAP agent instance to a set of given values. The pdu is very similar to the getprop-pdu and actually differs only in one field. The getprop pdu contains an PIDList which is a list of properties to be retrieved from the server while the setprop pdu contains PackedPropertyList field which contains a property list which will be associated with an agent instance and later retrieved using the getprop-pdu.

The SETPROP-PDU is answered by a SETPRPOP-REPLY-PRU described below.

4.6.5.2.6 SETPROP-REPLY-PDU

```
SETPROP-REPLY-PDU ::= SEQUENCE
{
     protoIdent         OBJECT IDENTIFIER,
     dest     CMAA OPTIONAL,
     reply-id INTEGER,
     status   PropertyStatus,
     failed-props  PIDList,
     session-id         SessionID OPTIONAL
}
```

This pdu is a reply to the setprop-pdu described above. This PDU contains the usual **dest**, **reply-id** and **session-id** fields.

The setprop reply also contains a **status** field. This field contains the status of the set operation. If not all properties could be set, then a non-zero status is returned.

When an error status is returned, the field a **failed-props** (which may be empty) is also returned - containing the property IDs of the properties that could not be set.

4.6.5.2.7 ACTIVATEMETHOD-PDU

```
ACTIVATEMETHOD-PDU ::= SEQUENCE
{
     dest      CMAPAgent OPTIONAL,
     req-id    INTEGER,
     method    CMAPString,
     params    PackedPropertyList,
     session-id          SessionID OPTIONAL
}
```

The ACTIVATEMETHOD-PDU is used to invoke a specific method on a given agent instance on the server. The PDU contains the usual **dest**, **req-id** and **session-id** fields (as described in the overview and in the getprop-pdu).

The activatemethod-pdu also contains a **method** field which is the name of the specific agent method[5] to invoke on the given agent instance. This method can be either a *class* method or an *instance* method. There's no distinction between these two method categories on the protocol level.

The pdu also contains a **params** field which contains a list of parameters to be passed to the method to be activated.

The activatemethod-pdu is answered by an ACTIVATEMETHOD-REPLY-PDU as described below.

4.6.5.2.8 ACTIVATEMETHOD-REPLY-PDU

```
ACTIVATEMETHOD-REPLY-PDU ::= SEQUENCE
{
     protoIdent          OBJECT IDENTIFIER,
     dest      CMAPAgent OPTIONAL,
     reply-id  INTEGER,
     status    PropertyStatus,
     params    PackedPropertyList,
     session-id          SessionID OPTIONAL
}
```

The ACTIVATEMETHOD-REPLY-PDU contains the usual **dest**, **reply-id** and **session-id** fields.

The **status** field indicates whether or not the method activation on the server was successful or not. Any value other than “ok” means the method was **not** invoked.

Note that the status is not the “return value” from the method.

Any value the method returns is passed back as a parameter in the **params** list. The list can also be emtpy, if no return value is required.

4.6.5.2.9 CREATEINSTANCE-PDU

```
CREATEINSTANCE-PDU ::= SEQUENCE
{
     protoIdent     OBJECT IDENTIFIER,
     dest     CMAPAgent OPTIONAL,
     req-id   INTEGER,
     session-id     SessionID OPTIONAL
}
```

The CREATEINSTANCE-PDU is used to create an instance of a specific CMA agent on a specific server.

The only fields in the createinstance pdu are the **dest** field which indicates the agent which we want to create an instance of and the server we want to create it on. The **req-id** and **session-id** are the same as in the other CMAP PDUs.

This pdu is answered by the CREATEINSTANCE-REPLY-PDU described below

4.6.5.2.10 CREATEINSTANCE-REPLY-PDU

```
CREATEINSTANCE-REPLY-PDU ::= SEQUENCE
{
     protoIdent     OBJECT IDENTIFIER,
     dest     CMAPAgent OPTIONAL,
     reply-id INTEGER,
     status   PropertyStatus DEFAULT ( ok ),
     session-id     SessionID OPTIONAL
}
```

The CREATEINSTANCE-REPLY-PDU contains the usual dest, reply-id and session-id fields, and in addition a status field containing a status code for the attempted agent creation.

4.6.5.2.11 DELETEINSTANCE-PDU

```
DELETEINSTANCE-PDU ::= SEQUENCE
{
      dest     CMAPAgent OPTIONAL,
      req-id   INTEGER,
      session-id     SessionID OPTIONAL
}
```

The DELETEINSTANCE-PDU is used to delete a specific instance of a CMA agent from a specific CMAP server. The only parameters needed by the server for this operation are the **dest**, **req-id** and **session-id**.

This pdu is answered by the DELETEINSTANCE-REPLY-PDU described below.

4.6.5.2.12 DELETEINSTANCE-REPLY-PDU

```
DELETEINSTANCE-REPLY-PDU ::= SEQUENCE
{
      dest     CMAPAgent OPTIONAL,
      reply-id INTEGER,
      status   PropertyStatus DEFAULT ( ok ),
      session-id     SessionID OPTIONAL
}
```

Just like in the CREATEINSTANCE-REPLY-PDU this pdu contains the usual **dest**, **reply-id** and **session-id** fields, and a status field indicating the status of the attempted deleteinstance operation.

4.6.5.2.13 REDIRECT-PDU

```
REDIRECT-PDU ::= SEQUENCE
{
      protoIdent     OBJECT IDENTIFIER,
      reply-id INTEGER,
      status   ENUM ( permanent, temporary),

      --either new-server or new-dest (or both) MUST appear.
      new-server     OCTET STRING OPTIONAL,
      new-dest CMAPAgent OPTIONAL,
}
```

The REDIRECT-PDU is a reply which could be initiated by any CMAP server as either a single or an extra reply to any pdu it receives. The redirect pdu is intended to cause the client contacting a specific server to redirect its operation to a different CMAP server.

This pdu contains a **reply-id** to indicate which pdu caused this redirect pdu to be sent. The **status** indicates which kind of a redirect this is (e.g. **permanent** - redirect all similar requests to the new server, **temporary** - just for the context of the given request). The **new-server** field contains the address of the new server to contact - either the DNS name or the dotted-decimal notation of its address, followed by colon and the port to use (e.g: “127.0.0.1:123” or “cma.domain.com:3221”).

The **new-dest** holds a new CMAP agent instance, to be used.

4.7 RAS Mappings

The following section defines the mapping from H.323 RAS to CMAP. This mapping is optional and pertains to vendors wishing to develop a CMA enabled Gatekeeper as described in section 4.4.2.

RAS verb	***CMAP verb***	***Comments***
RRQ	SetCTSpecifier Request	Will use the *IP-Phone* terminal category, and the *H.323* address type.
RCF/RRJ	SetCTSpecifier Reply	Errors replies are sent as RRJ messages.
URQ	SetCTSpecifier Request	Will be set as an H.323 terminal, with TTL 0.
UCF/URJ	SetCTSpecifier Reply	Errors replies are sent as URJ messages.
LRQ	Resolve Request	
LCF/LRJ	Resolve Reply	Error replies are sent as LRJ messages.

Please note that a general translation rule would be that the H.323 aliases specified in the RAS protocol need to be CMAA’s.

The exact parameter translations need yet to be defined.

5. Annexes (Normative)

Annex A: CMA System Specification

Annex B: RTP Payload Types

The RTP protocol reserves 7 bits for the definition of payload. The following table gives the payload type mapping. Current RTP and RFC 1890 definitions payload type mappings have been preserved. Additional static mappings for G.723.1 and G.729 will be included when available.

0	G.711 μ-law PCM audio	24	HDCC video
1	1016 audio	25	CelB video
2	G.726 32 kbit/s ADPCM audio	26	JPEG video
3	GSM 6.10 audio	27	CUSM video
4	unassigned	28	nv video
5	DVI4 audio (8 kHz)	29	PicW video
6	DVI4 audio (16 kHz)	30	CPV video
7	LPC audio	31	H261 video
8	G.711 A-law PCM audio	32	MPV video
9	G.722 audio	33	MP2T video
10	L16 audio (stereo)	34-71	unassigned
11	L16 audio (mono)	72-76	reserved
12	TPS0 audio	77-95	unassigned
13	VSC audio	96-119	dynamically assigned audio
14	MPA audio	120	G.764 noise parameter (prop.)
15	G.728 audio	121	interpreted DTMF data (prop.)
16-22	unassigned	122-127	control (prop.)
23	RGBB video		

Annex C: GSM 6.10 Transfer Syntax

Introduction

The GSM audio compression, utilizes a frame of 160 samples (each sample is a 16 bits signed word). And compresses it into a frame of 260 bits. Those 260 bits are made up of different variables, each variable takes up a different number of bits.

The VoIP forum selected the use of the RTP basic packaging type which enables the packaging of even or odd number of GSM frames within a single packet of RTP.

General Packaging Issues

The GSM audio encoding used in the RTP protocol differs from that used by the ACM?? Codec supplied by Microsoft, not by the calculation of the different fields, and not by the order of the fields in the frame, but by the packing of those fields in every GSM frame buffer.

In the ACM Codec, every two frames (320 samples) are packed into a buffer of 65 bytes (520 bits). The packing begins from the least significant bit of every byte, and the least significant bits of the field are packed first. For instance, if the first field is F1 and contains 6 bits, and the second field is F2 which also contains 6 bits, they are packed in the following way: the first byte contains F1 in bits 0-5, and the lower two bits of F2 in bits 6-7. The second byte contains the high 4 bits of F2 in bits 0-3, and the 4 least significant bits of F3 in bits 4-7, and so on. The 33rd byte contains the last 4 bits of the first GSM frame in its lower 4 bits (0-3) and the first 4 bits of the next frame (the lower 4 bits of F1) in bits 4-7.

In the GSM encoding used by RTP, the bits are packed beginning from the most significant bit. Every 160 sample GSM frame is coded into one 33 byte (264 bit) buffer. Every such buffer begins with a 4 bit signature (0xD), followed by the MSB encoding of the fields of the frame. The first byte thus contains 1101 in the 4 most significant bits (4-7) and the 4 most significant bits of F1 (2-5) in the 4 least significant bits (0-3). The second byte contains the 2 least bits of F1 in bits 6-7, and F2 in bits 0-5, and so on. The order of the fields in the frame is as follows:

GSM variable names & numbers

#	Field Name	# of bits	#	Field Name	# of bits
1	LARc[0]	6	22	xmc[9]	3
2	LARc[1]	6	23	xmc[10]	3
3	LARc[2]	5	24	xmc[11]	3
4	LARc[3]	5	25	xmc[12]	3
5	LARc[4]	4	26	Nc[1]	7
6	LARc[5]	4	27	bc[1]	2

7	LARc[6]	3	28	Mc[1]	2
8	LARc[7]	3	29	xmaxc[1]	6
9	Nc[0]	7	30	xmc[13]	3
10	bc[0]	2	31	xmc[14]	3
11	Mc[0]	2	32	xmc[15]	3
12	xmaxc[0]	6	33	xmc[16]	3
13	xmc[0]	3	34	xmc[17]	3
14	xmc[1]	3	35	xmc[18]	3
15	xmc[2]	3	36	xmc[19]	3
16	xmc[3]	3	37	xmc[20]	3
17	xmc[4]	3	38	xmc[21]	3
18	xmc[5]	3	39	xmc[22]	3
19	xmc[6]	3	40	xmc[23]	3
20	xmc[7]	3	41	xmc[24]	3
21	xmc[8]	3	42	xmc[25]	3

#	Field Name	# of bits	#	Field Name	# of bits
43	Nc[2]	7	60	Nc[3]	7
44	bc[2]	2	61	bc[3]	2
45	Mc[2]	2	62	Mc[3]	2
46	xmaxc[2]	6	63	xmaxc[3]	6
47	xmc[26]	3	64	xmc[39]	3
48	xmc[27]	3	65	xmc[40]	3
49	xmc[28]	3	66	xmc[41]	3
50	xmc[29]	3	67	xmc[42]	3
51	xmc[30]	3	68	xmc[43]	3
52	xmc[31]	3	69	xmc[44]	3
53	xmc[32]	3	70	xmc[45]	3
54	xmc[33]	3	71	xmc[46]	3
55	xmc[34]	3	72	xmc[47]	3
56	xmc[35]	3	73	xmc[48]	3
57	xmc[36]	3	74	xmc[49]	3
58	xmc[37]	3	75	xmc[50]	3
59	xmc[38]	3	76	xmc[51]	3

The RTP Packaging

So if F.i signifies the ith bit of the field F, and bit 0 is the least significant bit, and the bits of every byte are numbered from 7 to 0 from left to right, then in the RTP encoding we have:

Byte #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	1	0	1	LARc[0].5	LARc[0].4	LARc[0].3	LARc[0].2
1	LARc[0].1	LARc[0].0	LARc[1].5	LARc[1].4	LARc[1].3	LARc[1].2	LARc[1].1	LARc[1].0
2	LARc[2].4	LARc[2].3	LARc[2].2	LARc[2].1	LARc[2].0	LARc[3].4	LARc[3].3	LARc[3].2

Packaging [n] GSM frames within one RTP package Payload.

Basically the RTP payload used defines the way of marking the number of frames per RTP package. More detailed information can be found in the RTP spec (ref.??).

1101	GSM 6.1[n]	1101	GSM 6.1[n+1]	1101	GSM 6.1[n+2]

The MS Packaging

And in the MS codec encoding we have:

Byte #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	LARc[1].1	LARc[1].0	LARc[0].5	LARc[0].4	LARc[0].3	LARc[0].2	LARc[0].1	LARc[0].0
1	LARc[2].3	LARc[2].2	LARc[2].1	LARc[2].0	LARc[1].5	LARc[1].4	LARc[1].3	LARc[1].2
2	LARc[4].1	LARc[4].0	LARc[3].4	LARc[3].3	LARc[3].2	LARc[3].1	LARc[3].0	LARc[2].4

The 33rd byte which is the connecting byte of two frames will be:

Byte #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
32	LARc[0].3	LARc[0].2	LARc[0].1	LARc[0].0	xmc[51].2	xmc[51].1	xmc[51].0	xmc[50].2
	First Variable of second Frame				Last Bits of First Frame			

Transfer Characteristics

Unit Frame Size Delay: 20 ms

Delay for N Packed Frames: N*20 ms

Annex D: Other Codec Transfer Syntax

Placeholder for other codec transfer syntaxes (i.e. G.729, G.723.1), if not defined elsewhere in existing standards.

Annex E: G.764-Based Noise Level Transfer Syntax

The G.764-based VAD noise level packet contains a single byte message to the receiver to play comfort noise at the absolute dBmO level specified by the G.764 level index. This message would normally be sent once at the beginning of a silence period (which also indicates the transition from speech to silence), but rate of noise level updates is implementation specific. The mapping of the index to absolute noise levels measured on the transmit side is given in Table 'x'.

MSbit LSbit

0*	0*	0*	0*	G.764 Level Index

* - set to zero

Table 'x': G.764 Level Index Mapping

Index	Noise Level (dBrncO)
0	Idle Code
1	16.6
2	19.7
3	22.6
4	24.9
5	26.9
6	29.0
7	31.0
8	32.8
9	34.6
10	36.2
11	37.9
12	39.7
13	41.6
14	43.8
15	46.6

Annex F: DTMF Digit Transfer Syntax

Out-of-band 'in-band' signaling will be carried via RTP/UDP as dynamically allocated coder payload 121.

The Digits Transfer Syntax comprises the Digits Packet Format and the Digits Transfer Procedure.

Digits Packet Format

At the originating VoIP client the detected digits are inserted into a Digits Packet. Packets carrying digits will be identified as a specific RTP dynamically allocated codec type.

Each Digit Packet contains 3 windows of digit transition. The first windows represents the Current 20ms period, the 2nd is the Recent and the 3rd is the Previous.

	Bit-7	**Bit-6**	**Bit-5**	**Bit-4**	**Bit-3**	**Bit-2**	**Bit-1**	**Bit-0**
Octet 1	reserved			Signal Level				
Octet 2	Digit-Type[0]			Edge-Location[0]				
Octet 3	reserved			Digit-Code[0]				
Octet 4	Digit-Type[-1]			Edge-Location[-1]				
Octet 5	reserved			Digit-Code[-1]				
Octet 6	Digit-Type[-2]			Edge-Location[-2]				
Octet 7	reserved			Digit-Code[-2]				

Sequence Number (8 bits)

Each increment of the sequence represents a period of 20ms.

Signal Level (5 bits)

The power level of each frequency is in -dBm0 (a range of 0 to -31 dBmO).

Digit Type (3 bits)

Code	Digit Type
000	Digit Off
001	Digit On
010	reserved
011	reserved

Edge-Location (5 bits)

A 20ms windows is used to encode the edge when a digit is turned on and off. This is the delta time, 0 to 19ms, from the beginning of the current frame in ms. If there is no transition, the edge location will be set to 0 and the Digit Type of the previous windows will be repeated.

Digit-Code (5 bits)

Digit Code	DTMF Digits
00000	0
00001	1
00010	2
00011	3
00100	4
00101	5
00110	6
00111	7
01000	8
01001	9
01010	*
01011	#
01100	A
01101	B
01110	C

01111	D

Digits Transfer Procedures

Procedure for Transmission of Digits Packets

When the transmitter detects a validated digit, it will start sending a Digits Packet every 20ms. Since each packet covers 60ms of Digit On/Off Edge information, there is redundancy of the edge information. The RTP sequence number is incremented by one in each transmitted packet.

When the digit activity is off, the transmitter should continue to send 3 more Digits Packet for 60ms.

Internet Telephone client applications will often be the source of digit activity (e.g. GUI dialpads) without the need for actual detection of in-band signals. In this case, digit timing information would be derived from the relative timing of GUI events and the active outbound voice stream, with level information would be fixed to a default value. An acceptable default value is -10 dBmO.

Procedure for Interpreting Received Digits Packets

When the receiver gets a Digits Packet, it will generate digits according to the location of the On and Off edges. Silence will be applied to the duration after an Off edge and before an On edge. Digits will be generated after and On edge and before an Off edge.

If the sequence number is one greater than the last received sequence number, the receiver appends the Current edge information to the previously received information.

If the sequence number is two greater than the last received sequence number, the receiver appends the Recent and Current edge information to the previously received information.

If the sequence number is three greater than the last received sequence number, the receiver appends the Previous, Recent and Current edge information to the previously received information.

If the sequence number is more than three greater than the last received sequence number, the receiver appends the Recent and Current edge information to the previously received information. It fills in the gap with the static values based on the previously received packet.

If a Voice Packet is received at anytime, an Off edge should be appended to the previously received Digits On/Off Edge information.

Annex G Network Management

MIB specifics

The VoIP Forum plans to develop a set of MIBs for SNMP management of VoIP network

Annex H H.323 Implementation Specific Details.

Important - needs to be added.

6. Appendixes (Informative)

Quality of Service (QoS)

Given the nature of packet networks, establishing a given Quality of Service (QoS) is a key component of Voice over IP terminals. Maintaining QoS for both VoIP traffic and other traffic is a key consideration for IP networks serving VoIP terminals.

Delay

One of the key elements of end user perceived quality is end to end delay. Delay will be affected by:

- Framing delay: the amount of time represented in the voice packet.
- Coder Delay: voice coders have certain inherent delays.
- Packetization delay: a terminal or gateway will have delays passing the voice packet through its IP stack and injecting into the IP network.
- Transit Delay: voice packets transported through IP networks will experience delays related to the transmission time of the packet across each link and queuing and processing delays in routers inside the network.

Delay Variance

Packets transmitted over IP networks will arrive with variable delays. The variance in inter packet arrival times is called jitter. Accommodation for this variable delay must be made at the terminating endpoint. Initially this can be achieved by the addition of a fixed delay FIFO on the receiver. Alternatively, a receiver may attempt to measure jitter and adaptively size the jitter buffer. The jitter buffer is sized such that for the distribution of arrival times, an acceptable number of packets are not played due to the fact that they have been delayed by an interval exceeding the jitter buffer size.

Delay sensitive packets can be adversely affected by the traffic mix when passing over slow links (typically the last link at the user's access to a WAN. Over a 56K link, it takes over 200ms to transmit a 1500 byte frame (as for file transfer or e-mail). See Carsten Bormann[6] for an excellent treatment for this problem. Voice packets that are queued behind this will be subjected to increases in jitter of up to the transit time of the largest frame size for the link. Terminals should require stacks and access routers that include procedures that give enhanced treatment for real-time traffic.

Packet Loss

Packet loss can be a result of CRC errors on voice packets or loss due to congestion. In addition, from the point of view of the coder, a packet delayed enough to exceed the jitter

[6] http://

buffer size must be dealt with as a lost packet. In any case, the coder must deal with the lost packet by some method. These may include simply not playing a packet, repeating the last packet or some implementation-specific lost packet replacement technique.

Packets can also be lost due to congestion and congestion management techniques. In a best-effort delivery service, packets are queued in buffers of finite sizes in routers. When those buffers overflow, packets are generally discarded.

Congestion Management

Traditionally, UDP traffic does not have a congestion management feedback mechanism such as Van Jacobson. Therefore, high-volume UDP traffic tends to displace TCP traffic, which will back off in the face of congestion. Without other congestion control mechanisms, UDP traffic will not even be aware of congestion, while the TCP traffic is starved.

A technique for managing congestion due to UDP traffic is called Random Early Drop, or RED[7]. RED operates by subjecting traffic contributing to congestion to random packet loss. The presumption is that the application will adapt to packet loss in a useful way, probably by reducing its offered load.

Admission Control

The Resource ReServation Protocol (RSVP) is another technique for congestion management. Its function is to attempt to secure reserved bandwidths for particular flow specifications (T-specs??). In a properly operating network, flows operating within their T-specs will not be subject to congestion, eliminating packet loss due to congestion. In addition, RSVP may cause the packet scheduler within a router to alter the delivery order of packets in such a way that the delay experienced by any given packet falls within the specification described by the T-spec, which will tend to contributed to lower jitter.

It is beyond the scope of this specification to define methods in which RSVP can be used to improve VoIP performance, although future revisions of the IA may address specific RSVP implementation details

Echo Cancellation

All telephony voice services currently in use today reflect some level of echo back to a user. This echo can be caused in several ways - two to four-wire hybrid mismatches and acoustic coupling are two examples. The term echo is used broadly here as simply the return of a signal's reflection back to the originator. However, users typically associate echo with some time delay - this is because a user will only begin to differentiate the returned signal from the original if the delay between the two is greater than 20-30 ms. In fact, some low-delay echo is required for handsets to provide sound reinforcement of the original signal - this is typically called sidetone. Besides giving necessary reinforcement, sidetone also gives the user the feeling that their telephone set is 'working'.

[7] Sally Floyd, TBD.

Packet voice networks, however, are much more likely to introduce sufficient latency to cause what a user would consider an audible echo. The echo path is round-trip, and thus any speech coding, packetization and buffering delays are accumulated in both directions of transmission, increasing the likelihood of audibility.

Echo suppressors and echo cancellers are devices which attempt to remove a user's reflected signal while preserving as much as possible the far-end talker's signal transmitted to the user. Echo suppressors are devices which use some sort of half-duplex switching technology to block.

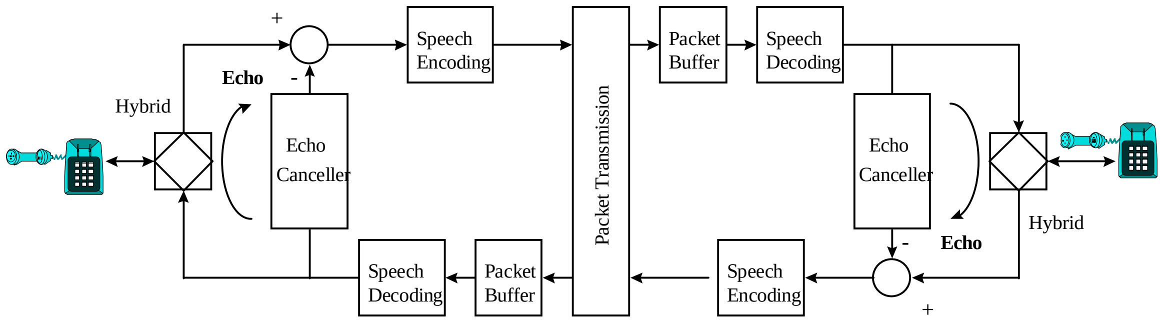
Service Models and Requirements

Half-Duplex H/W

Client S/W built on a half-duplex hardware base (i.e. half-duplex PC sound card) is not required to provide an echo control device.

Full-Duplex H/W

Client or gateway S/W built on a full-duplex hardware base (i.e. full-duplex PC sound card) is required to provide some means of echo control. This can range from a push-to-talk mechanism to voice-activated switches to full-duplex echo cancellation systems. It is recommended that where possible, a full-duplex echo cancellation system be implemented, meeting either G.165 Network Echo Cancellation or G.167 Acoustic Echo Cancellation requirements where appropriate.



Dial-up (Is this necessary Annex A covers this well??)

It may be noted that in order to initiate a call between two VoIP endpoints, the calling party needs to know the IP address of the called party. In the existing Internet infrastructure, dialup users are being assigned a different IP address whenever they dial into the Internet. The result is that a caller has no way of determining the called IP address without using an external mechanism to resolve the called IP address.

The above led the VoIP forum to add the CMA spec to the VoIP implementation agreement.