*Contact Bryan Hill
VideoServer Inc.
E-mail
Tel: +1 781 505 2159
Fax: +1 781 505 2101
bhill@videoserver.com

Brian Rosen
FORE Systems, Inc.
E-mail
Tel +1 724 742 6826
Fax +1 724 742 6744
brosen@fore.com

Internet Engineering Task Force
INTERNET DRAFT
March 25, 1999
Expires September 25, 1999
<draft-ietf-megaco-protocol-00.txt>

Fernando Cuervo
Nortel Networks
Christian Huitema
Telcordia Technologies
Keith Kelly
NetSpeak
Brian Rosen
FORE Systems
Paul Sijben
Lucent Technologies
Eric Zimmerer
Level 3 Communications

Status of this document

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

TABLE OF CONTENTS

TABLE OF FIGURES

1. SCOPE

MEGACOP/Recommendation H.GCP defines the protocols used between elements of a physically decomposed multimedia gateway. There are no functional differences from a system view between a decomposed gateway, with distributed sub-components potentially on more than one physical devices, and a monolithic gateway.
This RFC/recommendation does not define how gateways, multipoint control units or integrated voice response units (IVRs) work. Instead it creates a general framework that is suitable for these applications.

Packet network interfaces may include IP, ATM or possibly others. The interfaces will support a variety of SCN signalling systems, including tone signalling, ISDN, ISUP, QSIG, and GSM. National variants of these signaling systems will be supported where applicable.

MEGACO/H.GCP might also apply to the MC-MP interface.

2. REFERENCES

1. ITU-T Recommendation H.225.0 (1998): "Call Signaling Protocols and Media Stream Packetization for Packet Based Multimedia Communications Systems ".
2. ITU-T Recommendation H.245 (1998): "Control Protocol for Multimedia Communication"
3. ITU-T Recommendation H.323 (1998): "Packet Based Multimedia Communication Systems"
4. 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"
5. ITU-T Draft Recommendation H.246 (1998), "Interworking of H-series multimedia terminals with H-series multimedia terminals and voice/voiceband terminals on GSTN and ISDN"

{Editors note Add IETF references back}

3. DEFINITIONS

Access-Gateway: A type of gateway that provides a User to Network (UNI) network interface such as ISDN.
Back-haul: The transport of signaling information from a media termination gateway like a MG to a signaling gateway such as MGC. For example, a layer 3 protocol such as Q.931 might be transported between MG and MGC such that the MGC terminates layer 3, although the MG terminates layers 1 and 2.
Gatekeeper (GK): A functional entity serving a gateway providing services such as authentication, authorization, alias resolution and call routing.
H.323 Signaling: This function in the decomposed gateway supports normal H.323 signaling, such as H.225.0, H.245, or H.450.x as described in H.323.
Media Gateway (MG): The media gateway converts media provided in one type of network to the format required in another type of network. For example, an MG could terminate bearer channels from a switched circuit network (i.e., DSOs) and media streams from a packet network (e.g., RTP streams in an IP network). This gateway may be capable of processing audio, video and T.120

alone or in any combination, and will be capable of full duplex media translations. The MG may also play audio/video messages and perform other IVR functions, or may perform media conferencing.

Media Gateway Controller (MGC): Controls the parts of the call state that pertains to connection control for media channels in a MG.

Multipoint Control Unit (MCU): A gateway that controls the setup and coordination of a mulit-user conference that typically includes processing of audio, video and data.

Network Access Servers: A gateway function in a MG that converts modem signals from an SCN network and provides data access to the Internet.

Residential gateway: A gateway that interworks an analog line to the packet network.

SCN FAS Signaling Gateway: This function contains the SCN Signaling Interface that terminates SS7, ISDN and other signaling links where the call control channel and bearer channels are collocated in the same physical span.

SCN NFAS Signaling Gateway: This function contains the SCN Signaling Interface that terminates SS7, ISDN and other signaling links where the call control channels are separated from bearer channels. The may be a one to many relationship where many MGCs are deployed to leverage the capabilities of the powerful SS7 interface.

Trunk: A communication channel between two switching systems such as a DS0 on a T1 or E1 line.

Trunking Gateways: A gateway between SCN network and packet network that typically terminates a large number of digital circuits.

4. ABBREVIATIONS

This recommendation defines the following terms.

ATM	Asynchronous Transfer Mode
BRI	Basic Rate Interface
CAS	Channel Associated Signaling
DTMF	Dual Tone Multi Frequency
FAS	Facility Associated Signalling
GK	GateKeeper
GW	GateWay
IP	Internet Protocol
ISUP	ISDN User Part
MG	Media Gateway
MGC	Media Gateway Controller
NAS	Network Access Server
NFAS	Non Facility Associated Signalling
PRI	Primary Rate Interface
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SCN	Switched Circuit Network
SG	Signalling Gateway
SS7	Signalling System N°7

UNI User to Network Interface

5. CONVENTIONS

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

6. ARCHITECTURE

Editor's Note:

Section 6 is highly ITU specific. It will be either substantially edited to remove reliance on H.323 concepts, or will be removed entirely from the IETF version.

6.1 Reference Model

The section identifies a group of interfaces and functions to be used to decompose gateways. This RFC/recommendation will address each interface and its resulting protocol but certain gateway implementations may choose to group two or more functional components into a single physical device. For this reason interfaces may provide a capability to transparently backhaul other protocols.

Decomposed gateways using MEGACO/H.GCP may employ a wide variety of interfaces, which leads to considerable variation in network models. A commonly envisioned gateway would interface the SCN network to a packet or cell data network. In the figure below the packet/circuit media component terminates SCN media channels and converts the media streams to packet based media on the packet network interface. The protocol for interface A is used to create, modify and delete gateway media connections. The control logic component accomplishes signalling interworking between the SCN and packet sides of the gateway.

Interfaces B1 and B2 form the packet signalling transport/interworking interface between the gateway entities on the IP network and the decomposed gateway controller function. Interface C describes the ISDN type call control function between the FAS SCN services and the gateway control logic. NFAS SCN signalling is conveyed between the SCN signalling transport termination and the SCN signalling termination on interface D. The B1 and B2 interfaces represent the H.245 and H.225 and similar signaling interfaces, respectively.

The Resource control elements differentiate between a high level understanding resources in the gateway controller and a lower level under standing of resources in a gateway device.

The SCN interfaces are described as a low level interface that transports signaling and a high level SCN signaling termination that interfaces with the controller of this gateway. This can be FAS signaling such as ISDN PRI or NFAS signaling such as SS7.

This figure does not represent a physical decomposition at this point. The challenge for gateway vendors is to group these components into physical devices and implement the associated interfaces

in order to produce highly scaleable, multi-vendor gateways. MEGACO/H.GCP will define these interfaces to facilitate implementation of a wide range of decomposed gateways. The X indicates the external network gateway or terminal interface, the Y the external packet interface and the Z is the external SCN interface

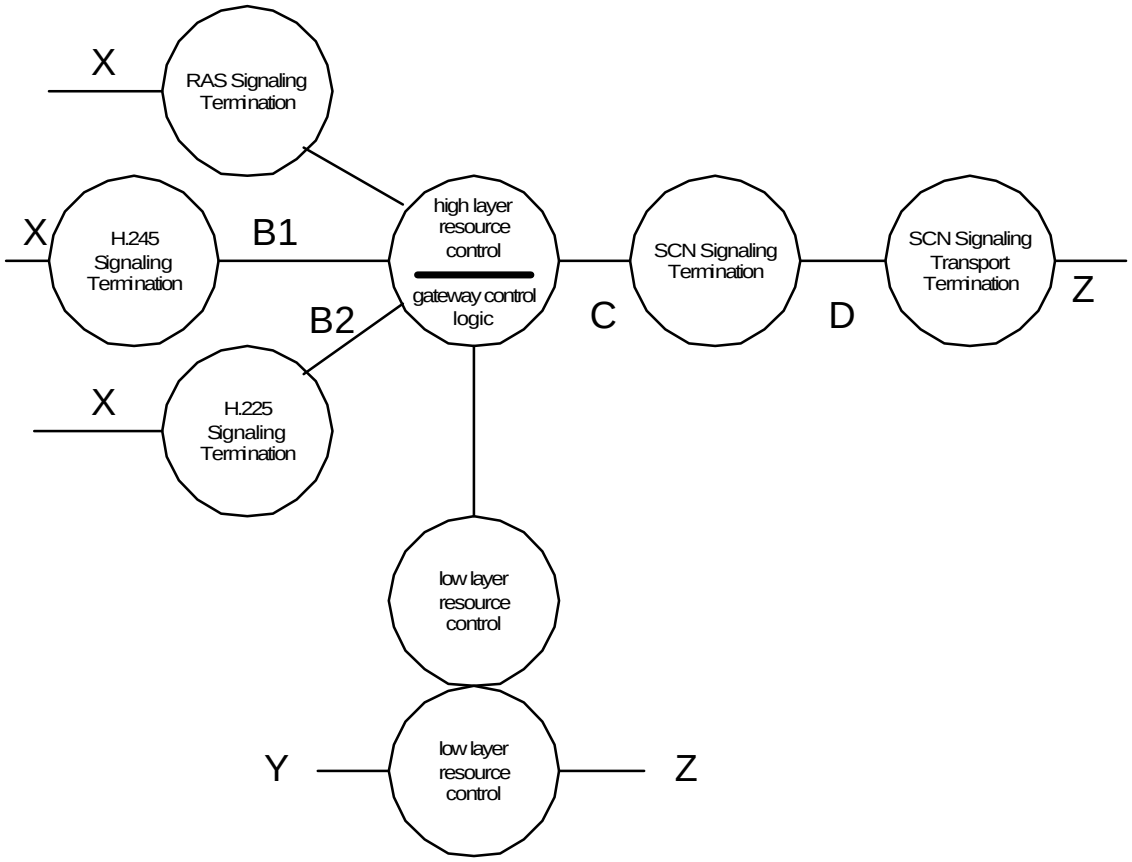


Figure 1 MEGACO/H.GCP Functional Architecture

6.2 Physical Decompositions

This section describes examples of possible gateway decompositions and the internal interfaces that are required. In all cases the external interfaces such as H.323/SIP and SCN remain unchanged. The controller portion of the physical gateway is called the Media Gateway Controller (MGC). The MGC's functions are to:

- handle H.225 RAS messaging with an external gatekeeper.
- optionally handle the SS7 signaling interface
- optionally handle the H.323 signaling interface

The Media Gateway (MG) component:

- terminates the IP network interface
- terminates the SCN network span
- may handle H.323 signaling in some physical decompositions
- may handle FAS SCN signaling in some physical decompositions.

Decomposed gateways need not realize all interfaces but the MGC/MG split exposing interface A is a Mandatory part of all decompositions. This will allow an MGC to control different types of MGs that may be optimized for certain applications (e.g. voice versus multimedia H.320/H.323 gateways). Decompositions exposing interfaces B2 and C between the MG and MGC may require a protocol to backhaul signaling from the MG tp the MGC. The functionality to achieve this decomposition is for further study.

The MG terminates the IP or ATM media on the packet network side and bearer channels on the SCN network interfaces. The packet side may be IP, native ATM, or an ATM network interface where audio and video packets traverse native ATM connections according to ITU H.323 Annex C. The MGC and MG differentiate between high-level and low-level resource management elements. The MGC is responsible for high-level resource management where it understands the availability of resources, such as echo cancellers, but does not assigned specific resources to specific gateway sessions. The MG is responsible for low-level resource allocation and management in addition to the hardware manipulations required to switch and process media streams within the media gateway.

6.2.1 Separate SS7 Gateway

The figure below represents one possible gateway decomposition that is a typical North American arrangement for an ISUP-to-H.323 Gateway, where the SG, MGC, and MG functions decomposed into separate physical devices. This arrangement exposes an ISUP signaling transport interface D and the device control interface A.

To facilitate interoperability MEGACO/H.GCP gateways must at a minimum support interface A and contain internal H.323 and SCN signaling in the MGC.

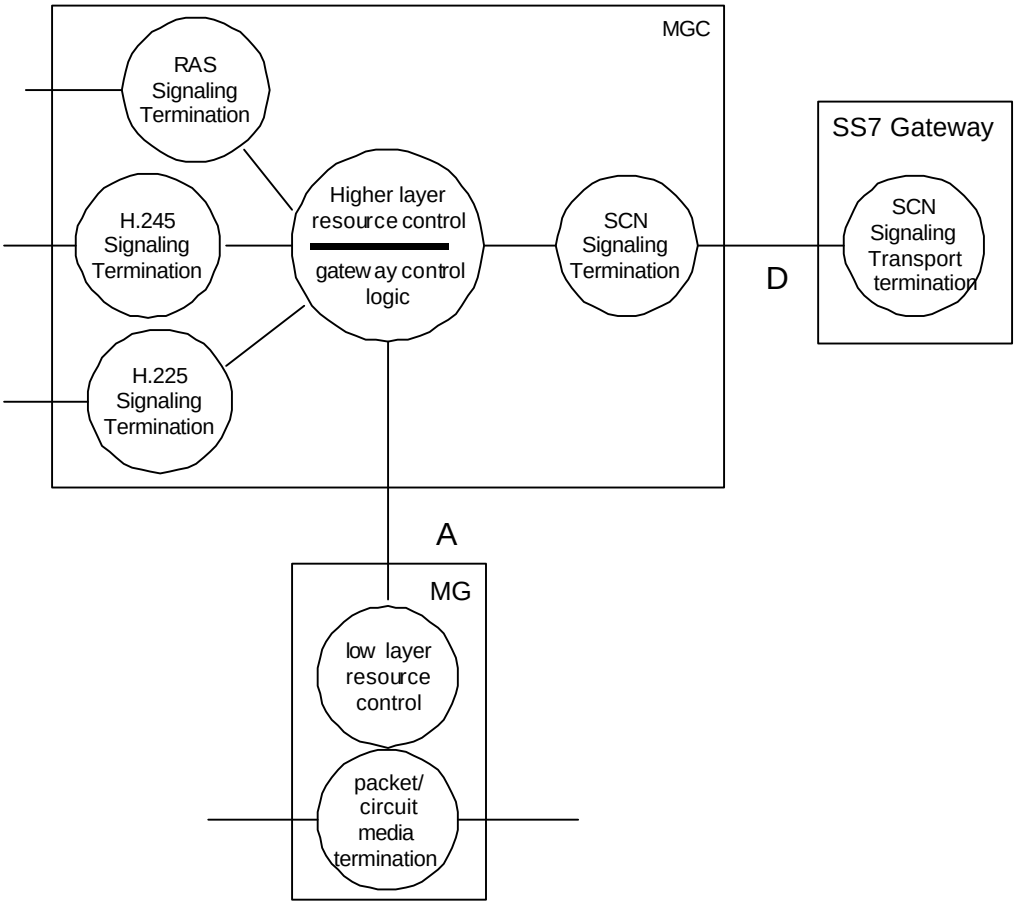


Figure 2 SS7 gateway Decomposition

6.2.2 FAS Gateway Decomposition

This gateway decomposition isolates the FAS SCN services such as ISDN PRI on the MG and retains the H.323 signaling on the MGC. This exposes the C and A interface between the MG and MGC.

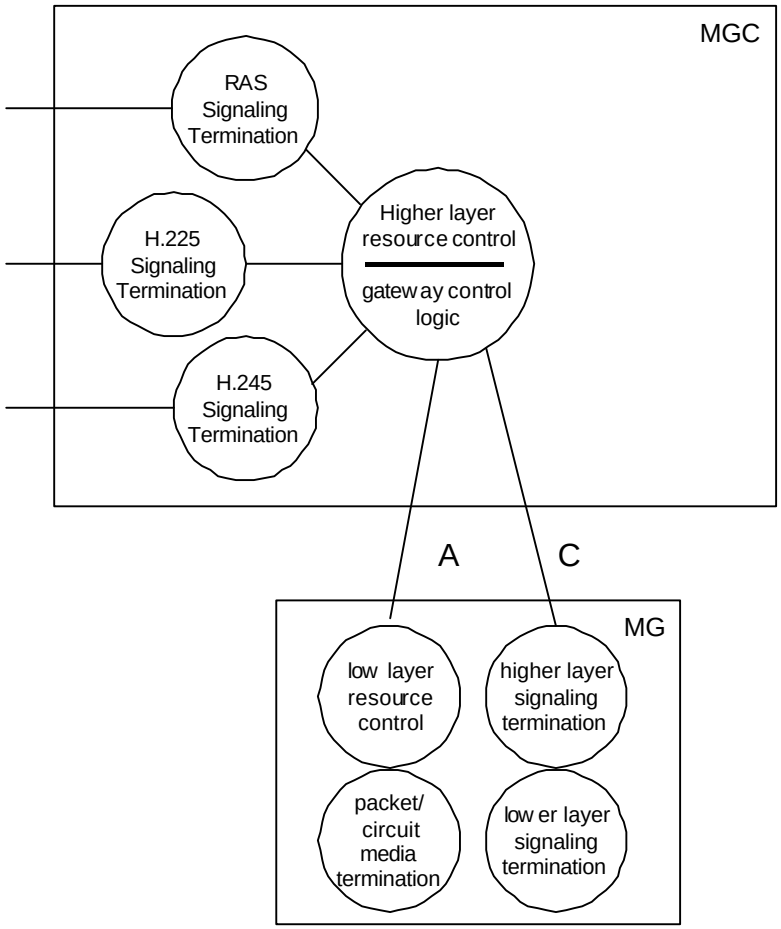


figure 3 FAS Gateway with H.323 Signaling in MG

6.2.3 SS7 Gateway with H.323 Signaling in the MG

This decomposition leverages the SS7 interface of the MGC and deploys the H.323 signaling on the MG exposing interfaces D, A and B.

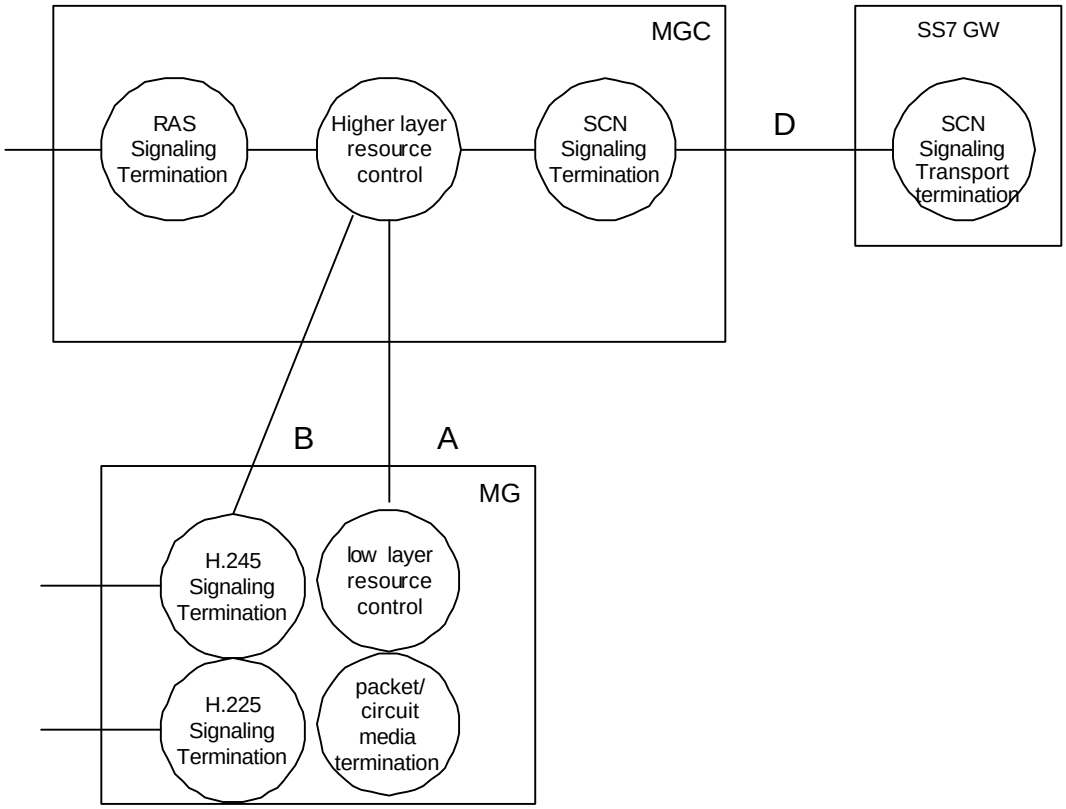


Figure 4 SS7 gateway with H.323 in MGC

6.2.4 FAS and H.323 Signaling in the Media Gateway

Customers require H.320 gateways that are decomposed such that H.323 and SCN signaling are both present on the MG along with the packet and circuit terminations. In this decomposition signaling is handled locally by the MG and event notifications are reported to the MGC.

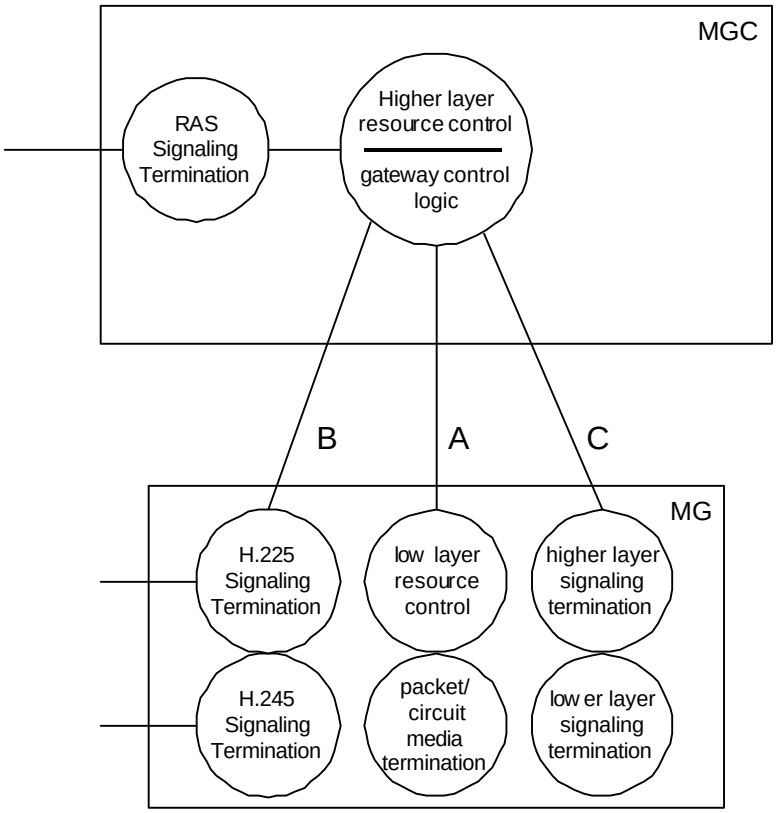


Figure 5 H.323 and FAS in MG

6.2.5 SS7 in the Media Gateway

This decomposition terminates the SS7 network in the MG and exposes the D interface between the MGC and MG.

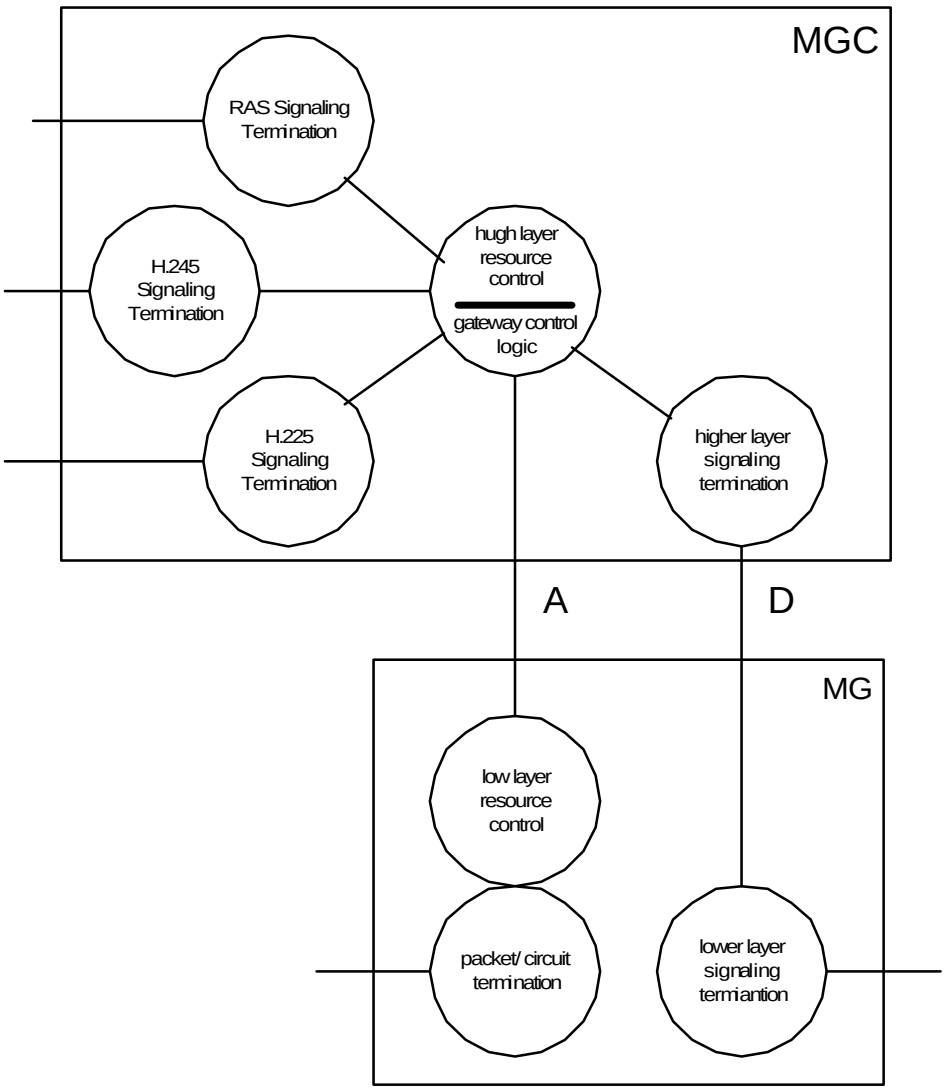


Figure 6 SS7 Terminated in the Media Gateway

7. REQUIREMENTS

Requirements for MEGACO/H.GCP are found in:
draft-ietf-megaco-reqs-03.txt

8. INTERFACE A PROTOCOL

8.1 Overview

The protocol used to control Media Gateways from Media Gateway Controllers will be described in this section. First to be introduced will be the connection model and terminology used to

describe the controllable logical elements of the Media Gateway. Following that will be specification of the Commands provided to manipulate the logical elements of the connection model. The Commands will be described using an application programming interface and a generic Command syntax. Command encoding specifications will follow. The grouping and processing of Commands within Transactions will then be discussed. Examples will be given to illustrate the use of the protocol.

8.2 Connection Model

8.2.1 Basic Concepts

The connection model for the interface A protocol describes the logical entities, or objects, within the Media Gateway that can be controlled by the Media Gateway Controller. The main abstractions used in the connection model are Terminations and Contexts.

A *Termination* is that is capable of sourcing and/or sinking one or more media. In a multimedia conference, a Termination can be multimedia, and sources or sinks multiple media streams. The media stream parameters, as well as modem, and bearer parameters are encapsulated within the Termination. A *Context* is an association between a collection of Terminations that make up a single conference. This can be a point-to-point call or a multipoint conference of a single medium type (e.g. voice) or multiple media types (e.g. voice, video and data). A Context containing more than two Terminations describes the conference bridging properties (e.g., audio mixing, video switching, video mixing, who sees/hears whom etc). This is depicted graphically by means of a star configuration, where the bridging functionality is in the center and Terminations are on the edges. There is a special type of Context, the *null* Context, that contains all Terminations that are not part of a conference. For instance, in a decomposed access gateway, all idle lines are represented by Terminations in the null Context.

Following is a graphical depiction of these concepts. The diagram gives several examples and is not meant to be an all-inclusive illustration. The empty box in each of the Contexts represents a logical association of Terminations implied by the Context. If more than two Terminations are connected to this box, it can be seen as the logical conference bridge.

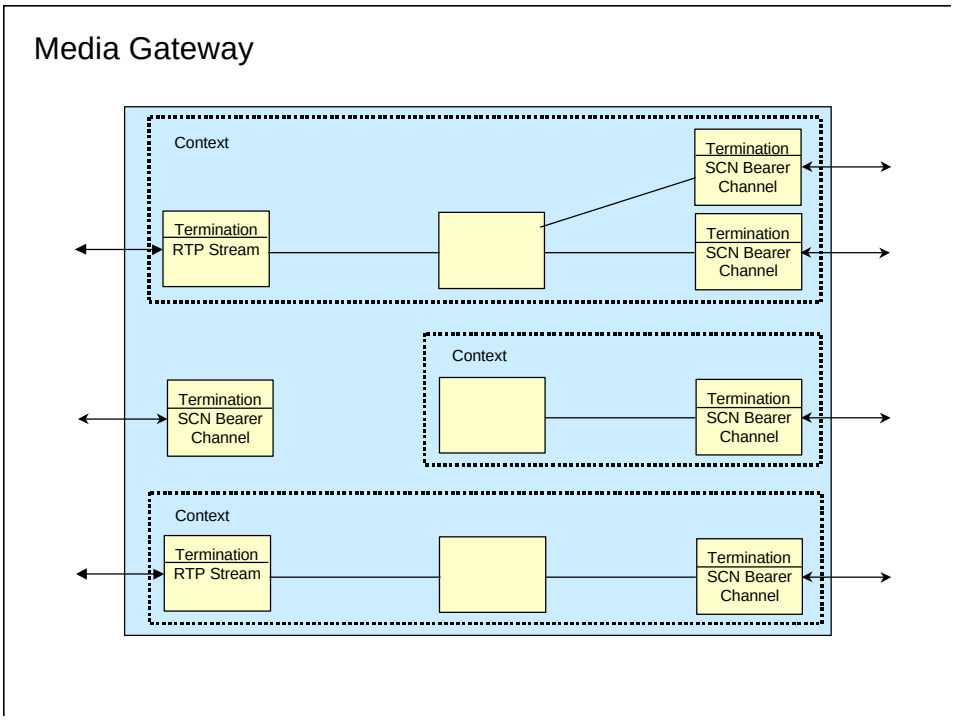


Figure 8: Example MEGACO/H.GCP Connection Model

The example below shows a call waiting scenario in a decomposed access gateway, illustrating the relocation of a Termination between Contexts. Terminations T1 and T2 belong to Context C1 in a two-way audio call. A second audio call is waiting for T1 from Termination T3. T3 is alone in Context C2. T1 accepts the call from T3, placing T2 on hold. This action results in the moving of T1 into Context C2, as shown below.

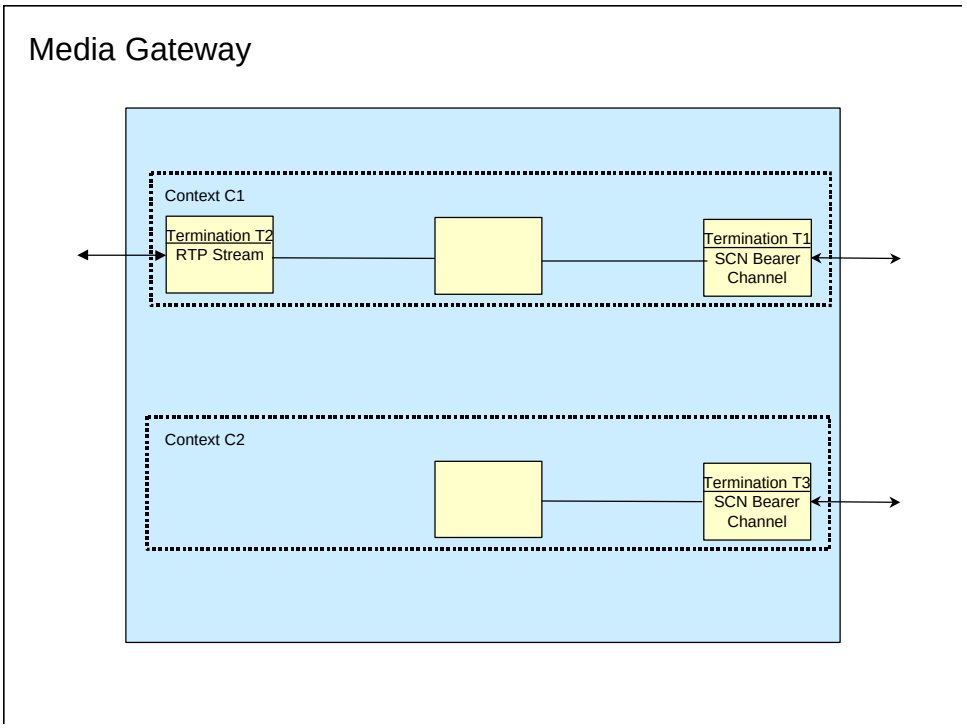


Figure 9 Call Waiting Scenario / Alerting Applied to T1

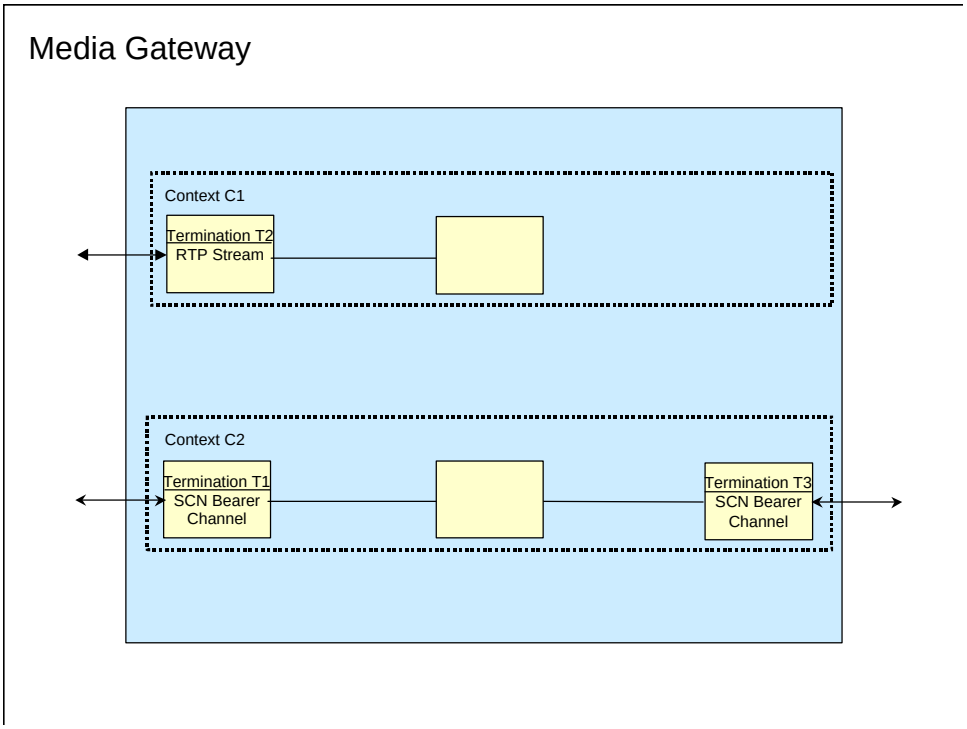


Figure 10Call Waiting Scenario / Answer by T1

8.2.2 Contexts

8.2.2.1 General

A Context is an assocation between a number of Terminations that describes a conference. The Context describes the topology (who hears/sees whom) and the media mixing and/or switching parameters if more than two Terminations are involved.

There is a special Context, the *null* Context, which does not describe a conference. Instead it contains Terminations that are not in any conference. Terminations in the null Context can have their parameters examined or modified, and may have events detected on them.

In general, an Add command is used to add Terminations to Contexts. If the MGC does not specify the Context to which the Termination is to be added, the MG creates a new Context. A Termination may be removed from a Context with a Subtract command, and a Termination may be moved from one Context to another with a Move command.

The maximum number of Terminations in a Context is an MG property. Media gateways that offer only point-to-point connectivity allow at most two Terminations per Context. Media gateways that support conference calls may allow three or more terminations per Context.

8.2.2.2 Context Parameters

The parameters of Contexts include

- ContextID,

8.2.2.3 Context Dynamics

The interface A protocol can be used to (implicitly) create Contexts and modify the parameter values of existing Contexts. The protocol has commands to add Terminations to Contexts subtract them from Contexts, and to move Terminations between Contexts. Contexts are deleted implicitly when the last remaining Termination is subtracted from it.

8.2.3 Terminations

8.2.2.4 Overview

A Termination is a logical entity on a MG that sources and/or sinks media and/or control streams. A Termination is described by a number of characterizing parameters. Terminations have unique identities, assigned by the MG at the time of their creation. Terminations representing physical interfaces to the MG will be created at boot time and are permanent. Terminations representing stream sources/sinks that are instantiated by higher layer protocols (such as RTP streams) exist only for the duration of the stream. Generally, such a Termination is created by the MG on an Add command, and destroyed by the corresponding Subtract command. In contrast, when a physical Termination is Add'ed to or Subtract'ed from a Context, it is taken from or to the null Context, respectively.

Multimedia conferencing may make use of multiplexing. For example, Recommendation H.221 describes such a function. Multiplexes are represented in Megaco/H.GCP by a Multimedia Context, a Muxing Termination and input Terminations. The muxing Termination may have a

parameter ("Bearer") that indicate, for instance, the order in which input streams are to be multiplexed. All input streams to the muxing Termination must be represented in the context by their corresponding input Terminations. Terminations are added and subtracted in the usual manner.

For convenience, if the Bearer parameter lists any Terminations that are not currently in the Context, such Terminations are added to the context as if individual Add commands listing the Terminations were invoked. The Media Type of such Terminations is initialized to a Multiplex-Type specific value. Also as a convenience, if a Bearer parameter lists a Termination already in the Context, the Media Type of such a Termination is changed to that Multiplex-Type specific value.

8.2.2.5 Termination Dynamics

The interface A protocol can be used to create Terminations and to modify parameter values of existing Terminations. These modifications include the possibility of adding or removing events and/or signals. The Termination parameters, and events and signals are described in the ensuing sections.

8.2.2.6 Termination Parameters

The tables in this section define all parameters used to describe Terminations. There are a number of common parameters, and parameters specific to media streams. For a media stream, there is a set of common parameters, and there sets of parameters for the received and transmitted flows.

The parameters listed below can be used to completely describe any Termination, whether a simple voice Termination, a fax Termination, a NAS Termination, an RTP termination or a multiplex Termination.

Most parameters have default values. If a Termination is created with the default value for a parameter, this does not have to be set explicitly. In other words, absence of a value for a mandatory parameter in a command shall be interpreted as if the default value had been used. The default values are chosen in such a way that creating voice Terminations can be done most efficiently.

{ Editor's note: Default values still have to be defined. }

Common Termination Parameters

Parameter	**Description**	**Support**
Name	MG assigned, unique identifier	M
Termination State	OutOfService, …	M
Network type	Identifies the network type (.e.g. IP, ATM, FR)	CM
Bearer Descriptor	Ordered list of bearers for multiplex terminations	CM
Modem Descriptor	Identifies modem type and parameters when applicable	CM
MUX Descriptor	Describes multiplex type for multimedia terminations (e.g. H.221, H.223, H.225.0)	CM
Media Descriptor	A list of media stream specifications (see below)	O
Eventbuffer Descriptor	Describes how events are buffered and/or processed	M
Event Descriptor	Describes events to be listened for by the MG and what to	O

	do when an event is detected	
Signal/action Descriptor	Describes signals and/or actions to be applied (e.g. ringback)	O
Scripting Descriptor	Optional parameter that identifies a script (e.g. dial plan) used on the Termination	O
Media Stream Sync	Set of streamIDs of media streams that require synchronization (ie audio/video lip sync)	CM
Non-standard Descriptor	The non-standard field allows inclusion of vendor-specific extensions	O

(M = mandatory, CM = Conditional Mandatory, O = Optional)

Events can be detected at all levels (the bearer, modem, multiplex and media levels. Similarly, signals and actions can be applied at all levels. For instance, ringback can be applied on a DS0, a mode change can be applied to an H.242 control channel.

Events as well as signals/actions are grouped in packages. There is a DTMF event package, for example, that describes the DTMF events. See below for more information.

There are three types of signals:

- on/off – the signal lasts until it is turned off,
- timeout – the signal lasts until it is turned off or a specific period of time elapses,
- brief – the signal duration is so short that it will stop on its own unless a new signal is applied that causes it to stop.

Each media stream in the Termination is represented by a group of Media Stream Parameters. A stream is identified by the stream ID. The streamID is used to link the streams in a Context that belong together. If there is only one media stream, the streamID may be left out.

Media Stream Parameters

Parameter	**Description**	**Support**
StreamID	Identifies the context stream to be associated with this termination media flow (e.g., 1, 2, 3, …	CM
Medium	Describes media type (audio, video, data, control, …)	M
Transport	Transport that carries the media for packet networks (e.g., RTP/UDP/IP, UDP/IP, TCP/IP)	CM
Transport QoS	Network specific quality of service	O
Max Jitter Buffer	Max jitter buffer size (in milliseconds)	CM

The table below describes each media flow parameter. In an audio only gateway session an instance of the following parameters will be represented for the transmit and a second instance for the receive media flow.

RX and TX Media Specific Parameters

Parameter	**Description**	**Support**
Activity	Boolean indicating whether media is flowing	M
Bearer descriptor	Address in use for packet network	CM
Packetization	Depends on encoding (e.g. audio samples per packet etc.)	CM

Encoding descriptor	Media encoding description (e.g., G.7xx, H.26x, MPEG)	CM
Encryption spec	Specific media encryption specified for this medium	O
Audio gain	Gain applied to audio in decibels	CM
Echo cancellation	Boolean indicating whether echo cancellation is applied (for audio streams)	CM

It is clear that not all combinations of values make sense. The MG is assumed to know which combinations of values it supports. If a MGC requests a combination that is not supported, the MG shall return an error message.

8.2.2.7 Templates

Setting all values of all parameters for every Termination created leads to large messages from MGC to MG. The *Template* mechanism can be used to allow a reduction in message size. A Template is a (partially) specified list of Termination parameters and values referred to by a TemplateID. Once Templates have been set up, a MGC can include a TemplateID in the command, and optionally give parameter-value pairs for parameters not specified in the Template or provide parameter-value pairs overriding values specified in the Template.

8.3 Commands

8.3.1 General Usage

The protocol provides Commands for manipulating the logical entities of the protocol connection model, Contexts and Terminations. Commands provide control at the finest level of granularity supported by the protocol. For example, Commands exist to add Terminations to a Context, modify Terminations, subtract Terminations from a Context, and audit properties of Contexts or Terminations. Commands provide for complete control of the properties of Contexts and Terminations. This includes things such as specifying which events a Termination is to report and which signals/actions are to be applied to a Termination.

Most Commands are for the specific use of the Media Gateway Controller as command initiator in controlling Media Gateways as command responders. However, there are several Commands for the Media Gateway to use as command initiator in reporting events that have occurred to the controller as command responder.

8.3.2 Command Entity Names and Common Parameters

Many Commands share common parameters. This subsection enumerates these common parameters. Parameters and parameter usage specific to a given Command type are described in the subsection that describes the Command.

8.3.2.1 Overview of Commands

The protocol has 9 commands. The commands are sent to the MG by the MGC, except Notify and ServiceChange. These latter two are sent to the MGC by the MG.

1. **Add**. The Add command adds a termination to a context. The Add command on the first Termination in a Context is used to create a Context.

2. **Modify**. The Modify command modifies the properties of a termination.
3. **Subtract**. The Subtract command disconnects a Termination from its Context and returns statistics on the Termination's participation in the Context. The Subtract command on the last Termination in a Context deletes the Context.
4. **Move**. The Move command atomically moves a Termination to another context.
5. **Audit**. The Audit command returns the values of properties of Terminations.
6. **Notify**. The Notify command allows the Media Gateway to inform the Media Gateway Controller of the occurrence of events in the Media Gateway.
7. **ServiceChange**. The ServiceChange Command allows the Media Gateway to notify the Media Gateway Controller that a Termination or group of Terminations is about to be taken out of service or has just been returned to service. ServiceChange is also used by the MG to announce its availability to an MGC (registration), and to notify the MGC of impending or completed restart of the MG.
8. **Put**. The Put command allows a MGC to load various types of data to MGs, these include Templates and Scripts.
9. **Delete**. The Delete command allows a MGC to delete data previously loaded in the MG by means of the Put command.

These commands are detailed in Sections **Error! Reference source not found.** through **Error! Reference source not found.**.

8.3.2.2 Wildcarding Parameter Values in Commands

Some parameters may be wildcarded in commands. Two wildcard constructs are provided: "all" and "choose". The "all" construct allows a Command to specify all possible values of a name component. For example, all Terminations can be subtracted from a Context by means of this construct. The "choose" construct allows a command initiator to specify that it would like the command responder to select and return a possible value for a parameter. This mechanism, for example, allows the MGC to have the MG select a DS0 within a DS1.

8.3.2.3 Specifying Parameters

Termination parameters are structured into a number of descriptors.

In any descriptor, each of the parameter may be fully specified, under-specified, or unspecified.

1) Fully specified parameters - have a single, unambiguous value that the command initiator is instructing the command responder to use for the specified parameter.
2) Under-specified parameters - have a list of potential values. The list order specifies the command initiator's order of preference of selection. The command responder chooses one value from the offered list and returns that value to the command initiator.
3) Unspecified parameters - (i.e.-mandatory parameters not specified in the descriptor) result in the command responder retaining the previous value (in the case of a parameter to a Modify command), or default value (in the case of an Add command) for that property.

8.3.2.4 Bearer Descriptor

For a Multiplex Termination, the BearerDescriptor specifies the Terminations to be used for the inputs of the mux. If the Terminations mentioned in the Bearer Descriptor are not in the Context, they are added to the Context. There is no default value.

8.3.2.5 Modem Descriptor

The Modem descriptor specifies the modem type and parameters, if any.

By default, no modem is present in a Termination.

8.3.2.6 Multiplex Descriptor

In multimedia calls, a number of media streams are carried on a (possibly different) number of bearers. The multiplex descriptor associates the media and the bearers. The way this is done depends on the multiplex type:

- H.221
- H.223,
- H.224,
- H.225.0.

8.3.2.7 Media Descriptor

A Media descriptor is a, possibly empty, list of Media Stream descriptors. If there are multiple media, the individual Media Stream descriptors shall include StreamIDs. Presence of multiple media implies the presence of a non-null Multiplex descriptor.

8.3.2.8 Media Stream Descriptor

A Media Stream descriptor specifies the parameters of a bidirectional media stream. These parameters are structured into common parameters and descriptors for the sent and received flows. The common parameters include the MediaID, which is mandatory for multimedia Terminations, and the mandatory Media Type (default value: audio). For packet networks, the transport type is mandatory as well. Possible values are RTP/UDP/IP, UDP/IP, TCP/IP, … The default is RTP/UDP/IP. The optional Transport QoS parameter to specify desired Quality of Service is for further study. The maximum jitter buffer size is mandatory for audio and video. Its default value is 10 ms.

8.3.2.8.1 Send and Receive Stream Descriptors

The Send and Receive Stream descriptor parameters describe the processing of media flow to and from a Termination. The permitted properties of each of these descriptors depend upon the type of transport and the media type.

1) Receive Stream descriptor – describes media processing properties of the incoming stream; for a TDM channel this would include things like echo cancellation options and gain

characteristics; for an RTP stream it would include things like the local IP address, receive RTP UDP port, encoding method and packetization interval.

2) Send Stream descriptor – describes media processing properties of the outgoing stream. This only makes sense if information about the other end of the media stream is needed, such as in the case of an RTP stream needing to know the receive RTP UDP port of the far end, or if the media parameters for the send and receive streams may be different.

8.3.2.9 EventBuffer Descriptor

The EventBuffer descriptor consists of two parameters:

1) EventBufferProcessingMode – specifies whether buffered events should be processed or discarded;

2) EventBufferNotificationMode – specifies whether the Media Gateway is expected to generate at most one notification (step by step) or multiple notifications (loop) in response to the current Command's request.

8.3.2.10 EventsDescriptor Properties

The EventsDescriptor parameter contains a RequestIdentifier and a list of events that the Media Gateway is requested to detect and report. The RequestIdentifier is used to correlate the request with the notifications that it may trigger. Requested events include, for example, fax tones, continuity tones, and on-hook and off-hook transitions.

With each event is associated a notification action, an optional embedded EventsDescriptor and SignalsDescriptor, and an optional Termination management action. There are two possible notification actions:

1) report the event immediately;

2) ignore the event.

The embedded Signal descriptor, if present, is used as a replacement for the current Signal descriptor. It is possible, for example, to specify that the dial-tone Signal be generated when an off-hook Event is detected, or that the dial-tone Signal be stopped when a digit is detected. If no embedded Signal descriptor is specified, the production of Signals continues as specified in the command.

8.3.2.11 SignalsDescriptor Properties

A SignalsDescriptor is a parameter that contains the set of signals/actions that the Media Gateway is asked to apply to a Termination. Signals apply to the bearer level, whereas actions apply to the higher levels (modem, multiplex, and media). For instance, ringing tones or DTMF tones may be applied to a DS0 bearer, while a mode change action may be applied to the H.242 control channel in an H.320 session). The signals and actions specified in a SignalsDescriptor must be supported by the Termination being addressed by the Command using this parameter.

Signals are mutually exclusive. That is, if a signal is being applied to a Termination when another signal is requested, then the previous signal is discontinued and the newly requested signal is applied.

Signals and actions are defined in packages. The initial packages are described later in this document.

8.3.2.12 DigitMapDescriptor

A DigitMap is a dialing plan resident in the Media Gateway used for detecting and reporting digit events received on a Termination. The DigitMapDescriptor parameter contains a set of DigitMap names and values to be assigned. The DigitMap name is visible within a scope, which can be the Media Gateway itself, a hierarchical group of Terminations, or a specific Termination. DigitMaps are assigned through the standard Termination manipulation Commands of the protocol. The scope of the DigitMap becomes that which is specified by the scope of the Termination name used in the Command.

1) If the Command is applied to the "all" wildcarded Termination, the DigitMap is visible within the entire scope of the Media Gateway.
2) If the Command is applied to a hierarchical Termination name, the DigitMap is visible to all Terminations whose name begins with the specified prefix.

The DigitMapDescriptor contains a set of DigitMap names and values to be assigned:

1) A new DigitMap is created by specifying a name that is not yet defined at this level of the naming hierarchy. The value must be present.
2) A DigitMap value is updated by supplying a new value for a name that is already defined at this level of the naming hierarchy.
3) A DigitMap is deleted by supplying an empty value for a name that is already defined at this level of the naming hierarchy. A wildcard naming convention can be used to delete all the DigitMaps associated with a specific Termination.

The collection of digits according to a DigitMap may be protected by an interdigit timer, which can assume two values:

1) If the Media Gateway can determine that at least one more digit is needed for a digit string to match any of the allowed patterns in the digit map, then the interdigit timer value should be set to a long duration (e.g.-16 seconds).
2) If the DigitMap specifies that a variable number of additional digits may be needed then the interdigit timer value should be set to a medium duration (e.g.-8 seconds).

A "long interdigit" timer and a "medium interdigit timer" are parameters associated with a DigitMap.

8.3.2.13 Statistics

The Statistics parameter provides information describing the status and usage of a Termination during its existence within a specific Context. The particular statistical properties that are reported for a given Termination are fixed based upon the definition of the Termination class.

8.3.3 Command Application Programming Interface

Following is an Application Programming Interface (API) describing the Commands of the protocol. This API is shown to illustrate the Commands and their parameters and is not intended

to specify implementation (e.g.-via use of blocking function calls). It will describe the input parameters and return values expected to be used by the various Commands of the protocol from a very high level. Command syntax and encoding are specified in later subsections. All parameters enclosed by square brackets ([. . .]) are considered optional.

8.3.3.1 Add

The Add Command adds a Termination to a Context.

[TerminationID]
[, ReceiveMediaStreamDescriptor]
[, SendMediaStreamDescriptor]
Add(TerminationID,
[TemplateID,]
[TerminationState]
NetworkType
[, BearerDescriptor]
[, ModemDescriptor]
[, MuxDescriptor]
[, MediaDescriptor]
[, EventBufferDescriptor]
[, EventsDescriptor]
[, SignalsDescriptor]
[, ScriptingDescriptor])

The TemplateID parameter identifies a Template to be used for creating the Termination. If this parameter is present, all parameters that follow either specify values of Termination parameters not fixed in the Template, or they override values of Termination parameters that were fixed in the Template.

{Editor's note: Now that the activity of media streams has been moved to the MediaStreamDescriptor, I'm not sure that the TerminationState parameter is still needed. }

The TerminationState parameter is optional. If it is omitted, default TerminationState properties will be assumed from the Termination class.

The mandatory NetworkType parameter indicates what type of bearer network will be used.

If the NetworkType specifies a switched network, the optional BearerDescriptor may be used to identify the particular bearer. It is also possible to leave the bearer unspecified or partially specified. In this case, the MG shall select a bearer and return the BearerDescriptor.

The optional ModemDescriptor and MuxDescriptor specify a modem and multiplexer if applicable.

The MediaDescriptor is a (possibly empty) list of MediaStreamDescriptors.

The EventsDescriptor parameter is optional. If present, it provides the list of events that should be detected on the Termination.

The SignalsDescriptor parameter is optional. If present, it provides the list of signals that should be applied to the Termination.

The ScriptingDescriptor parameter is optional. If present, it describes the name and the type of a script previously loaded into the MG.

8.3.3.2 Modify

The Modify Command modifies the properties of a Termination.

[ModemDescriptor,]
[MuxDescriptor,]
[MediaDescriptor]
Modify(TerminationID,
[TemplateID],
[TerminationState,]
[BearerDescriptor,]
[ModemDescriptor,]
[MuxDescriptor,]
[MediaDescriptor
[EventsDescriptor,]
[SignalsDescriptor,]
[ScriptingDescriptor])

{Editor's note: Would the MG ever fill in parameters for Modem or Mux by itself? If not, the optional return parameters of the Modify command should disappear. }

TerminationID in the input parameters represents the Termination that is being modified.

The TerminationState parameter is optional. If it is omitted, the existing TerminationState properties will not be modified. If present, only the TerminationState properties specified will be altered.

{Editor's note: Is the TerminationState parameter is still useful in Modify? }

The MediaDescriptor parameter is optional. If it is omitted, the existing MediaDescriptor properties will not be modified. If it is present and under-specified, then the return MediaDescriptor will give the fully specified values assigned by the Media Gateway.

The EventsDescriptor parameter is optional. If present, it provides the list of events tobe detected on the Termination. The SignalsDescriptor parameter is optional too. If present, it provides the list of signals to be applied to the Termination. The ScriptingDescriptor parameter is optional. If present, it describes the type and name of a script previously loaded in the MG.

8.3.3.3 Subtract

The Subtract Command disconnects a Termination from its Context and returns statistics on the Termination's participation in the Context.

{ Editor's note: I deleted a number of parameters. The reason is that Subtract should remove a Termination from a Context and *destroy the Termination. If the Termination is still used after being removed from its current Context, a Move command should be used (possibly with the null Context as destination). }*

Statistics |

*[TerminationID,
Statistics]
Subtract(TerminationID)

TerminationID in the input parameters represents the Termination that is being subtracted. The TerminationID may be fully specified or may be the 'ALL' wildcard value indicating that all Terminations in the Context of the Subtract Command are to be subtracted.

The Statistics parameter is returned to report information collected on the Termination or Terminations specified in the Command. The information reported applies to the Termination's or Terminations' existence in the Context from which it or they are being subtracted.

8.3.3.4 Move

The Move Command moves a Termination to another Context from its current Context in one atomic operation.

[ModemDescriptor,]
[MuxDescriptor,]
[MediaDescriptor]
Move(TerminationID,
[TemplateID,]
[TerminationState,]
[ModemDescriptor,]
[MuxDescriptor,]
[MediaDescriptor,]

[EventBuffer Descriptor,]
[EventsDescriptor,]
[SignalsDescriptor,]
[ScriptingDescriptor])

The TerminationID specifies the Termination to be moved. The Move command is part of a Transaction (see Section 8.4). The ContextID of the Transaction specifies to which Context the Termination is moved. By convention the Termination is subtracted from its previous Context.

The TemplateID, TerminationState, ModemDescriptor, MuxDescriptor, MediaDescriptor, EventsDescriptor, SignalsDescriptor and ScriptingDescriptor parameters are processed as in the Modify Command.

8.3.3.5 Audit

{ Editor's note: This section on Audit requires more work. One thing I noticed is that it is the first section in the document that refers to root Terminations/root names.

As it stands, Audit has two functions. It allows a MGC to find out about the active calls on an MG, and it allows an MG to find out about MG capabilities and circuit status. For the former

function, the Audit command operates on Terminations/Contexts, for the latter function, it operates on bearer circuits. The semantics would improve if these two functions are separated }

The Audit Command returns properties associated with Terminations and or physical bearers.

[TerminationID]
[TerminationState,]
[LocalTerminationDescriptor,]
[RemoteTerminationDescriptor,]
[EventsDescriptor,]
[SignalsDescriptor,]
[DigitMapDescriptor,]
[Capabilities,]
[Statistics]
Audit([TerminationID |
NetworkType, BearerDescriptor],
RequestInfo)

The TerminationID or BearerDescriptor in the input parameters specifies the Termination or bearer circuit that is being audited. In case a Termination is specified, the audit Command shall be applied to the specific Context of the requested Termination(s), which can be the null Context. Audit can also be used by the MGC to gather information about physical bearer circuits currenlty not in use in any Termination. The RequestedInfo parameter describes the information being requested for the specified Termination(s). The following information can be requested with this Command (note that an empty RequestedInfo parameter returns only the TerminationID):

1) TerminationState – current TerminationState values
2) BearerDescriptor – current BearerDescriptor values
3) ModemDescriptor – current ModemDescriptor values
4) MediaDescriptor – current MediaDescriptor values
5) EventBufferDescriptor – current EventBufferDescriptor values
6) EventsDescriptor – current EventsDescriptor values
7) SignalsDescriptor – current SignalsDescriptor values
8) ScriptingDescriptor – current ScriptingDescriptor values
9) Statistics – current Statistics (i.e.-mid call values)
10) Capabilities – properties that the Media Gateway is prepared to support for the specified Termination or bearer circuit

When auditing a bearer circuit, only the Capabilities of the MG w.r.t. to that circuit are reported.

{Editor's note: The table below I copied from the MEGACO draft. I should have looked at it more carefully. For instance, it contains "root name", and it is not possible to audit all Terminations in the null Context. }

The following illustrates other information that can be obtained with the Audit Command:**ContextID** **TerminationID** **Information Obtained**

Specific	all	List of Terminations in a Context
Specific	wildcard	List of matching Terminations in a Context
Null	root name	Audit of Media Gateway state and events
Null	all	List of all Terminations in the Media Gateway
Null	all/	List of all "top level" Terminations
Null	wildcard	List of all matching Terminations
Unspecified	root name	Audit of Media Gateway null Context
Unspecified	all	List of all Terminations in the Media Gateway and the Context(s) to which they are associated
Unspecified	all/	List of all "top level" Terminations in the Context which they are associated
Unspecified	wildcard	List of all matching Terminations in the Context which they are associated

8.3.3.6 Notify

The Notify Command allows the Media Gateway to notify the Media Gateway Controller of events occurring within the Media Gateway.

```
Notify(TerminationID,
    ObserverdEvents)
```

The TerminationID parameter specifies the Termination issuing the Notify Command. The TerminationID must be a fully qualified name.

The ObservedEvents parameter contains the RequestID and a list of events that the Media Gateway detected in the order that they were detected. The RequestID returns the RequestID parameter of the EventsDescriptor that triggered the Notify Command. It is used to correlate the notification with the request that triggered it. The events in the list must have been requested via the RequestedEvents parameter of the triggering EventsDescriptor. The list must contain the events that were either accumulated (but not notified) or treated according to digit map (but no match found yet) and well as the final event that triggered the detection or provided a final match in the digit map. Each event in the list is accompanied by properties associated with the event and an indication of the time that the event was detected. Unsolicited Notify Commands are not possible.

8.3.3.7 ServiceChange

The ServiceChange Command allows the Media Gateway to notify the Media Gateway Controller that a Termination or group of Terminations is about to be taken out of service or has just been returned to service.

```
[MGCIdentity]
    ServiceChange(TerminationID,
                  ServiceChangeMethod,
                  ServiceChangeReason,
                  [ServiceChangeDelay])
```

The TerminationID parameter specifies the Termination(s) that are taken out of or returned to service. Wildcarding of Termination names is quite useful here, with the exception that the

"choose" mechanism shall not be used. Use of the "root" keyword indicates a ServiceChange affecting the entire Media Gateway.

The ServiceChangeMethod parameter specifies the type of ServiceChange that will or has occurred:

1) Graceful – indicates that the specified Terminations will be taken out of service after the specified ServiceChangeDelay; established connections are not yet affected, but the Media Gateway Controller should refrain from establishing new connections and should attempt to gracefully tear down existing connections
2) Forced – indicates that the specified Terminations were taken abrubtly out of service and any established connections associated with them were lost
3) Restart – indicates that service will be restored on the specified Terminations after expiration of the ServiceChangeDelay; the Terminations are assumed to now not be associated with any Context

The ServiceChangeReason parameter specifies the reason why the ServiceChange has or will occur.

The optional ServiceChangeDelay parameter is expressed in seconds. If the delay is absent or set to zero the delay value should be considered to be null. In the case of a "graceful" ServiceChangeMethod, a null delay indicates that the Media Gateway Controller should wait for the natural removal of existing connections and should not establish new connections. The ServiceChangeDelay is always considered null in the case of the "forced" method.

The Media Gateway Controller may return an MGCIdentity parameter that describes the Media Gateway Controller that should preferably be contacted for further service by the Media Gateway. In this case the Media Gateway must reissue the ServiceChange command to the new Media Gateway Controller.

A ServiceChange Command specifying the "root" keyword for the TerminationID is a registration command by which a Media Gateway announces its existence to the Media Gateway Controller. The Media Gateway is expected to be provisioned with the name of one primary and some number of alternate Media Gateway Controllers. The ServiceChangeMethod shall be "forced" for this usage. Acknowledgement of the ServiceChange Command completes the registration process.

8.3.3.8 Put

The Put command allows a MGC to load various types of data to MGs. These include Templates and Scripts.

Put (ID, type, data)

The ID parameter sets the name of the template or script. The type parameter indicates what kind of data follows (Template, Script, …), and the data parameter contains the actual data.

8.3.3.9 Delete

The Delete command allows a MGC to delete data previously loaded in the MG by means of the Put command.

Delete (ID)

The parameter of the command specifies the data (Template, Script, …) to be deleted. There is no return parameter.

8.3.4 Generic Command Syntax

{editors note: need to describe entire command syntax in BNF, or something else.}

8.3.5 Command Error Codes

All commands are acknowledged which carries a return code that is characterized as successful completion, transient error or permanent error.

8.3.6 Command Parameter Syntax

{*editors note: parameter syntax can be expressed in four possible ways as described below*

1) *Expressed via SDP*

   This option requires an extension to SDP to allow it to express mutually exclusive capabilities on a MG. This is illustrated by MG implementations that use general purpose DSP for thing like audio processing. There is also value in having parameters refer to an instance of a cap set rather than repeating the parameter each time a message is sent.

2) *H.245 capability expression*

   H.245 style parameters handle the mutually exclusive capabilities and has been design to pass references to the established capabilities when expressing parameters in messages. This option does not imply a particular message encoding.

3) *Combination*

   Combining H.245 and SDP style parameter expression to create a new style has been discussed as an option.

4) *MGC understands both*

   The MGC could be required to understand both types of parameter syntax leaving it up to the MG to choose either method. Detractors of this option point out that this places excessive burden on the MGC implementation.

}

8.3.7 Command Encoding

{Editors note Command encoding options are listed blow

1) *ASN.1, basic encoding rules (BER) or packed encoding rules (PER)*

   ASN.1 can be used to create relatively small messages but it places the burden of decoding and encoding on the MG and this is considered excessive for small limited function MG implementations.

2) *fixed point binary*

This option also creates small messages and does not add excessive burden to simple MG implementations.

3) Text

Text encoding is probably the simplest to understand when looking at raw messages during debug but will likely lead to larger messages than fixed point binary encoding. This can become excessive in deployments where many, high port count MGs, are deployed along with a single MGC.

}

8.4 Transactions

8.4.1 General Usage

Commands from the Media Gateway Controller to the Media Gateway are grouped into Transactions, each of which is identified by a TransactionID. Transactions consist of one or more Actions. An Action consists of a series of Commands that are limited to operating within a single Context. Consequently each Action typically must specify a ContextID. However, there are two circumstances where a specific ContextID is not provided with an Action. One is the case of modification of a Termination outside of a Context. The other is where the controller requests the gateway to create a new Context. Following is a graphic representation of the Transaction, Action and Command relationships.

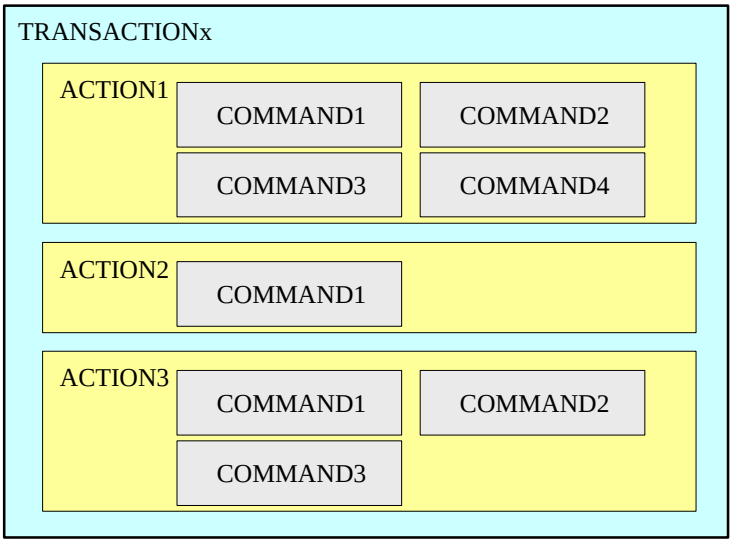


Figure 11 Transactions, Actions and Commands

Transactions are presented to the Media Gateway as TransactionRequests. Corresponding responses to a TransactionRequest are received in a single reply. There are two types of replies, a TransactionAccept and a TransactionReject.

Transactions allow Commands within them to share fate and guarantee ordered Command processing. That is, Commands within a Transaction are executed sequentially according to an

"all or nothing" rule. A TransactionAccept includes successful Responses for all of the Commands in the corresponding TransactionRequest. A TransactionReject is sent when one of the Commands included in the Transaction fails

8.4.2 Common Parameters

8.4.2.1 Transaction Identifiers

Transactions are identified by a TransactionID, which is assigned by the Media Gateway Controller and is unique within the scope of the controller.

8.4.2.2 Context Identifiers

Contexts are identified by a ContextID, which is assigned by the Media Gateway and is unique within the scope of the Media Gateway. The Media Gateway Controller must use the ContextID supplied by the Media Gateway in all subsequent Transactions relating to that Context. The protocol makes reference to two distinguished values that may be used by the Media Gateway Controller when it has no ContextID to use in a Transaction:

1) The "null" Context, which is used to refer to a Termination that is currently not associated with a Context.
2) The "unspecified" Context, which is used to request that the Media Gateway create a new Context.

8.4.3 Transaction Application Programming Interface

Following is an Application Programming Interface (API) describing the Transactions of the protocol. This API is shown to illustrate the Transactions and their parameters and is not intended to specify implementation (e.g.-via use of blocking function calls). It will describe the input parameters and return values expected to be used by the various Transactions of the protocol from a very high level. Transaction syntax and encodings are specified in later subsections.

8.4.3.1 TransactionRequest

The TransactionRequest is invoked by the Media Gateway Controller. There is one Transaction per request invocation. A request contains one or more Actions, each of which must specify its target Context and one or more Commands per Context.

TransactionRequest(TransactionID=<TransactionIDvalue>,
ContextID=<ContextIDvalue> Command [Command] … [Command]
. . .
ContextID=<ContextIDvalue> Command [Command] … [Command])

The TransactionID parameter must specify a keyword TransactionIDvalue for later correlation with the TransactionAccept or TransactionReject response from the Media Gateway.

The ContextID parameter must specify a keyword ContextIDvalue to pertain to all Commands that follow up to either the next specification of a ContextID parameter or the end of the

TransactionRequest, whichever comes first. The ContextIDvalue may be fully specified, unspecified, or null.

The Command parameter represents one of the Commands mentioned in the "Command Details" subsection titled "Application Programming Interface".

8.4.3.2 TransactionAccept

The TransactionAccept is invoked by the Media Gateway. There is one accept invocation per Transaction. An accept contains one or more Actions, each of which must specify its target Context and one or more Responses per Context. The invocation of a TransactionAccept implies successful execution of all Actions and Commands from the corresponding TransactionRequest.

TransactionAccept(TransactionID=<TransactionIDvalue>,
ContextID=<ContextIDvalue> Response [Response] … [Response]
. . .
ContextID=<ContextIDvalue> Response [Response] … [Response])

The TransactionID parameter must specify a keyword TransactionIDvalue for correlation with the corresponding TransactionRequest from the Media Gateway Controller.

The ContextID parameter must specify a keyword ContextIDvalue to pertain to all Responses that follow up to either the next specification of a ContextID parameter or the end of the TransactionRequest, whichever comes first. The ContextIDvalue may be fully specified or null.

The Response parameters each represent one of the Responses mentioned in the "Command Details" subsection titled "Application Programming Interface".

8.4.3.3 TransactionReject

The TransactionReject is invoked by the Media Gateway. There is one reject invocation per Transaction. A reject contains one or more Actions, each of which must specify its target Context and one or more Responses per Context. Responses for the Commands in a TransactionReject are issued as follows:

1) All Commands before the point of failure in the Command sequence should have a Response equal to "non-executed".

2) The failed Command should have a Response with a reason code specified.

Commands after the point of failure are not processed and, therefore, Responses are not issued for them.

TransactionReject(TransactionID=<TransactionIDvalue>,
ContextID=<ContextIDvalue> Response [Response] … [Response]
. . .
ContextID=<ContextIDvalue> Response [Response] … FailedResponse])

The TransactionID parameter must specify a keyword TransactionIDvalue for correlation with the corresponding TransactionRequest from the Media Gateway Controller.

The ContextID parameter must specify a keyword ContextIDvalue to pertain to all Responses that follow up to either the next specification of a ContextID parameter or the end of the TransactionRequest, whichever comes first. The ContextIDvalue may be fully specified, unspecified, or null.

The Response and FailedResponse parameters each represent one of the Responses mentioned in the "Command Details" subsection titled "Application Programming Interface". The specific Response in a TransactionReject must be "unexecuted". The FailedResponse represents the response to the Command that failed.

8.4.4 Transaction Syntax

{editors note: need to express transaction syntax in BNF or ASN.1 or something else}

8.4.5 Transaction Encoding

{editors note: need to express transaction encoding in binary encoding, text encoding or something else}

8.5 Example Use Cases

8.5.1 Residential Gateway to Residential Gateway Call

{editors note: this section is a candidate for an informative appendix}

This example scenario illustrates the use of the elements of the protocol to setup a Residential Gateway to Residential Gateway call over an IP-based network. For simplicity, this example will assume that both Residential Gateways involved in the call are controlled by the same Media Gateway Controller.

Programming Residential GW Analog Line Terminations for Idle Behavior

{editors note: I have not yet completely modified this section to reflect the agreed compromise between the "Minneapolis Model" and the "Santiago" model }

The following illustrates the API invocations from the Media Gateway Controller and Media Gateways to get the Terminations in this scenario programmed for idle behavior. Both the originating and terminating Media Gateways have idle AnalogLine Terminations programmed to look for call initiation events (i.e.-offhook) by using the Modify Command with the appropriate parameters. The null Context is used to indicate that the Terminations are not yet involved in a Context.

1. The MGC programs a Termination in the NULL context.

```
<SID> Trans <TRID0>
        Ctxt NULL
             Add T1D1
                  NetworkType=analogue
```

```
                    MediaDescr ([(StreamID=<MID1>,
                    Media=audio, ReceiveMediaDescr
                    (Activity=off, packetization=10ms,
                    Encoding=G.711 μ-law, Gain=2dB,
                    VoiceActDet=y), SendMediaDescr
                    (Activity=on) /* no other parameters */
                    )])
                    EventBufferDescr (ProcessingMode=Process,
                    NotificationMode=StepByStep)
                    SignalsDescr (none)
                    EventsDescr (offhook)
                    ScriptingDescr (Dialplan0)
```

The dialplan script has been loaded into the MG previously. Its function is to wait for the OffHook, turn on dialtone and start collecting DTMF digits.

2. The MG1 accepts the Context creation:

```
<SID> TransAccept <TRID0>
   Ctxt <CID1>
```

3. A similar exchange happens between MG2 and the MGC resulting in an idle Termination called <TID5>.

Collecting Originator Digits and Initiating Termination

The following builds upon the previously shown conditions. It illustrates the API invocations from the Media Gateway Controller and originating Media Gateway (MG1) to get the originating Termination (TID1) through the stages of digit collection required to initiate a connection to the terminator's Media Gateway (MG2).

4. MG1 detects an offhook event from User 1 and reports it to the Media Gateway Controller via the Notify Command.

```
<SID> Trans <TRID1>
   Ctxt NULL
        Term <TID1>
             Notify (Event=OffHook)
```

5. The MGC creates a new context and adds the Termination to it

```
<SID> Trans <TRID2>
     Ctxt ANY
          ADD <TID1>
               EventsDescr (OnHook)
```

6. And the move is acknowledged

```
<SID> TransAccept <TRID2>
     Ctxt <CID1>
```

7. Next, digits are accumulated by MG1 as they are dialed by User 1. When an appropriate match is made of collected digits against the currently programmed Dialplan for TID1, another Notify is sent to the Media Gateway Controller.

```
<SID> Trans <TRID3>
```

```
        Ctxt <CID1>
             Notify <TID1>
                  Event=digits,
                  value=155512345
```

8. The controller then analyzes the digits and determines that a connection needs to be made from MG1 to MG2. An RTP termination is added to the context in MG1.

```
<SID> Trans <TRID4>
     Ctxt <CID1>
          Add ANY
               NetworkType=IP4
               MediaDescr ([(StreamID=<MID1>,
               Medium=audio, Transport=RTP/UDP/IP,
               MaxJitterBuffer=40ms, ReceiveMediaDescr
               (Activity=on, BearerDescr (IPaddr=ANY,
               UDPport=ANY), Packetization=20ms,
               Encoding=G.723, Gain=0dB), SendMediaDescr
               (Activity=off, BearerDescr (IPaddr=ANY,
               UDPport=ANY)) )])
               EventBufferDescr (ProcessingMode=Process,
               NotificationMode=StepByStep)
```

9. MG1 acknowledges the new Termination and fills in the IP address and the UDP port.

```
<SID> TransAccept <TRID4>
   Ctxt <CID1>
        Term <TID2>
             MediaDescr ([(StreamID=<MID1>,
               ReceiveMediaDescr (
              BearerDescr (IPAddr=<IP1>, UDPPort=<P1>)) )])
```

10. The MGC will now associate <TID5> with a new Context, and establish an RTP Stream (i.e.-<TID6>) receiveOnly connection through to the originating user, User 1.

```
<SID> Trans <TRID5>
     Ctxt ANY
          Move <TID5>
               SignalsDescriptor (Ring)
               ScriptingDescriptor())
               // this clears the previous dialplan
          Add ANY
               NetworkType=IP4
               MediaDescr ([(StreamID=<MID1>,
               Medium=audio, Transport=RTP/UDP/IP,
               MaxJitterBuffer=40ms, ReceiveMediaDescr
               (Activity=on, BearerDescr (IPaddr=ANY,
               UDPPort=ANY), Packetization=20ms,
               Encoding=G.723, Gain=0dB), SendMediaDescr
               (Activity=off, BearerDescr (IPaddr=<IP1>,
               UDPport=<P1>)) )])
               EventBufferDescr (ProcessingMode=Process,
               NotificationMode=StepByStep)
               SignalDescriptor (RingTone)
```

11. This is acknowledged

```
<SID> TransAccept <TRID5>
   Ctxt <CID2>
        Term <TID5>
        Term <TID6>
             MediaDescr ([(StreamID=<MID1>,
               ReceiveMediaDescr (
                 BearerDescr (IPAddr=<IP2>, UDPport=<P2>) )])
```

12. The two gateways are now connected and the user hears the RingBack. The MG now waits until User2 picks up the receiver and then the two way call is established.

```
   <SID> Trans <TRID6>
        Ctxt <CID2>
             Notify <TID5>
                  OffHook

   <SID> Trans <TRID7>
        Ctxt <CID2>
             Modify <TID5>
                  EventsDescr(onHook)
             Modify <TID6>
                  SignalDescr() // clear the ringback
   <SID> Trans <TRID8>
        Ctxt <CID1>
             Modify <TID2>
             MediaDescr([(StreamID=<MID1>,
             SendMediaDescr (
               BearerDescr (IPAddr=<IP2>, UDPport=<P2>)) )])
   <SID> TransAccept <TRID7>
        Ctxt <CID2>
             Term <TID5>
             Term <TID6>
   <SID> TransAccept <TRID8>
        Ctxt <CID2>
             Term <TID2>
```

8.5.2 Multimedia Gateway Examples

Multimedia sessions using this protocol will use multimedia Terminations and Contexts for example for H.320 ISDN connections and for IP based multimedia connections. The MGC determines the need for multimedia Contexts from the SCN or IP side call signaling. Once multimedia is detected the MGC will create the Context and appropriate Terminations.

In general, a Termination will associate all media of an individual user and handles network jitter Streams sourced/sinked by a Termination are identified by a StreamId to instruct the MG how to connect them. Media from terminations with identical streamIDs are connected. The MGC can instruct the MG to synchronize streams by setting Context properties. One of the properties of a Context is the mixing properties. In the figure below these are represented by the black dots in the context.

The concept of connecting streams makes for a straight forward implementation of functionality such as speech-to-text transmediation. If a MG supports this type of operation, a MGC can assign identical StreamIDs to a speech stream and a text stream to indicate that incoming speech should be transformed into text.

8.5.2.1 H.320 Gateway

The Context for a point-to-point multimedia call in an H.320-H.323 gateway contains two Terminations. It contains a Termination that sources and sinks the H.221 frames on DS0s. This Termination references a number of DS0s, six in the example for a call with a total bandwidth of 384 kbit/s. The mux/demux descriptor of the Termination describes how the audio, video and data streams are transported over the six 64 kbit/s bearer channels.

The second Termination sources and sinks the media flows on the packet network side. Assuming that there are audio, video and data streams, the Termination contains three bearer descriptors. No multiplexing is involved: every media flow maps to one stream on the network.

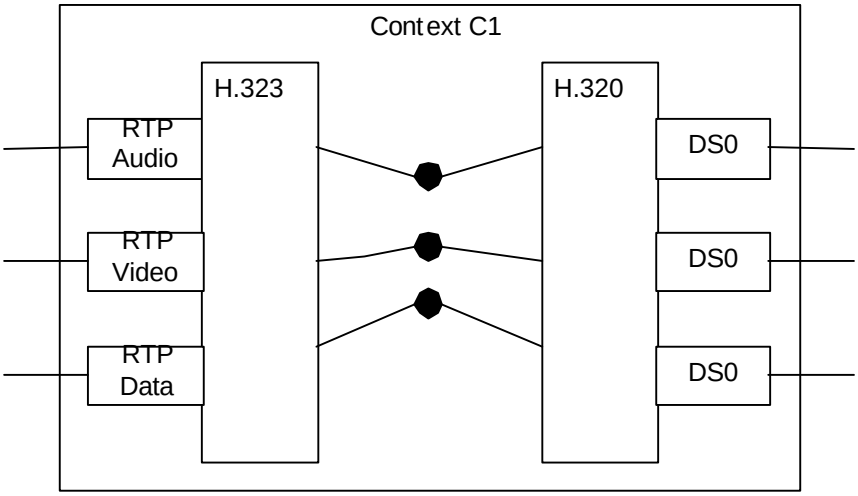


Figure 12 H.320 Gateway Context

The following is a call flow for a point-to-point H.320 to H.323 call initiated from the WAN side. The call flow shows that either the H.323 or H.320 side can initiate opening (or closing) an audio or video channel through the gateway. In H.320, there is the requirement that such mode changes take at most 20 milliseconds. In the call flow we see that messages are exchanged between MG and MGC to inform the MGC of a request for a mode change from the H.320 side. The MGC will then send an OLC to the H.323 terminal.

1. MGC gets an incoming call with (Q.931) call type of data and sends an Add command to MG, to create a Context with one Termination, indicating in the parameters for the termination that Bonding will be used and that the multiplex type is H.221.

```
<SID> Transaction TR1
   ContextID=ANY
         Add (TermID=ANY
           NetworkType=ckt,
           BearerDescr ([(Type=DS0, Facility=x1, Trunk=y1)])
           MuxDescr(Type=H.221, …)
           MediaDescr ([(StreamID=<MID1>, Medium=audio,
             ReceiveMediaDescr (Activity=off, Encoding=G.711
         A-law, …),
             SendMediaDescr (Activity=on) )])
           EventBufferDescr (EventProcessingMode=Process,
             EventNotificationMode=StepByStep)
           SignalsDescr (<list of packages>)
           EventsDescr (ReqID=<ReqID1>, [(Bonding/CallID,
         Notify), …]) )
           ScriptingDescr (ID=<SCRID1>, Type=basic)
```

The BearerDescr field contains an ordered list of bearer channels used in the call (here the length is one). At this stage in the session there is only an audio channel, using 56 kbit/s G.711 coder (corresponding to H.221 mode 0F). A script is used to monitor the line for Bonding and H.221 in-band H.221 framing.

2. The MG acknowledges the Context creation, informing the MGC of the ContextID C1 and TerminationID T1 it assigned.

```
<SID> TransAccept TR1
   ContextID <CID1> Add (TermID=<TID1>)
```

3. When the termination finds Bonding, it assigns a Bonding call ID x and accepts the proposed call transfer rate requested by the calling H.320 endpoint. The MG then sends a Notify message to the MGC informing it of the Bonding call identifier ‘x’ and the transfer rate:

```
<SID> Transaction <TRID2>
   Context <CID1>
         Notify (TermID=<TID1>, ReqID=<ReqID1>,
         [(Bonding/CallId, x), (Bonding/TransfRate, 384)])
```

4. The MGC allocates additional phone numbers for the call and sends requests the MG to send these back to the calling side by changing the signals descriptor:

```
<SID> Transaction <TRID3>
   Context <CID1>
         Modify (TermID=<TID1>, SignalsDescr
         (Bonding/AddPhoneNrs, [N1, N2, N3, N4, N5]))
```

5. The MG accepts the MGC’s message and sends the additional phone numbers back to the calling side via Bonding.
6. When the SGW notifies the MGC of incoming calls for the phone numbers associated with Bonding call x, the MGC sends Modify commands to the MG to add the appropriate DS0 bearer channels to the Termination created previously; the MG acknowledges these commands. For instance, for the first additional DS0 that is added:

```
<SID> Transaction <TRID4>
   ContextID=<CID1>
        Modify (TermID=T1, BearerDescr([(Type=DS0,
        Facility=x1, Circuit=y1), (Type=DS0, Facility=x1,
        Circuit=y2)]))
```

With the final Modify in which bearer channels are added, the MGC requests the MG to notify it when H.221 frames are detected.

7. Once H.221 framing is found a Notify is sent to the MGC.
8. The MGC instructs the Termination to listen for DTMF tones in the audio stream, and possibly to play an announcement to the calling user.
9. The audio announcement will be played and then the Termination will listen for DTMF tones in the audio portion of the mux and it commences a TCS4/IIS signaling exchange in the BAS channel. The information received is considered the destination alias z for this call.
10. The MG Notifies the MGC of at least three pieces of information: 1) frame alignment found, 2) H.320 capabilities, and 3) destination alias z. The MGC sends ARQ to resolve IP address for alias z
11. Once the address is resolved, the MGC does H.225 call setup. . . gets caps from H.323 etc. Note that we assume that the MGC sets up the H.245 connection with the called party.
12. The MGC sends a Modify to the H.320 termination causing a new capability set to be sent from the GW to the H.320 terminal, based on the received capabilities the MGC got from the H.323 endpoint.
13. MGC may get an OLC from H.323 side for audio, the MGC will then Add a packet Termination to the Context.

```
<SID> Transaction <TRID5>
   ContextID=<CID1>
        Add (TermID=ANY
          NetworkType=IP4
          MuxDescr (H.225.0)
          MediaDescr([(StreamID=<MID1>, Medium=Audio,
            Transport=RTP/UDP/IP, …,
            ReceiveMediaDescr (Activity=on, Encoding=…,
              BearerDescr (IPaddr=ANY, UDPport=ANY),
            SendMediaDescr (Activity=on, Encoding=…,
              BearerDescr (Ipaddr=<IP1>, UDPport=<P1>) )])
          EventBufferDescr (EventProcessingMode=Process,
            EventNotificationMode=StepbyStep)
          SignalsDescr (…)
          EventDescr (…)
```

The MGC assigns a StreamID to the media stream to allow the MG to identify the streams that have to be connected within the Context. The MG acknowledges the command and reports the assigned TerminationId to the MGC, as well as the IP address and UDP port it selected for the ReceiveMediaDescr.

14. The MGC sends a Modify to the H.320 termination sending a MediaDescr to the MG:

```
<SID> Transaction <TRID6>
   ContextID <CID1>
```

```
        Modify (TermID=<TID1>
          MediaDescr([(StreamID=1, Type=Audio, …)])
```

Of course the MG acknowledges the Modify command. At this point the MG knows that the audio streams from the packet and circuit sides have to be connected because they have the same StreamID.

15. The H.320 side may do a mode switch to H.263 video for example. The H.320 termination will then send an event to the MGC requesting H.263 video:

```
<SID> Transaction <TRID7>
   ContextID <CID1>
         Notify (TermID=<TID1>
            ReqID=<ReqID2>
              [(H.242/ModeChange, AddH263, …)])
```

16. The MGC must send and OLC to the H.323 side.
17. The MGC modifies the packet termination, by adding another RTP flow and changing the media descriptor to include the video. In the same Transaction, the H.320 termination is modified to include the video:

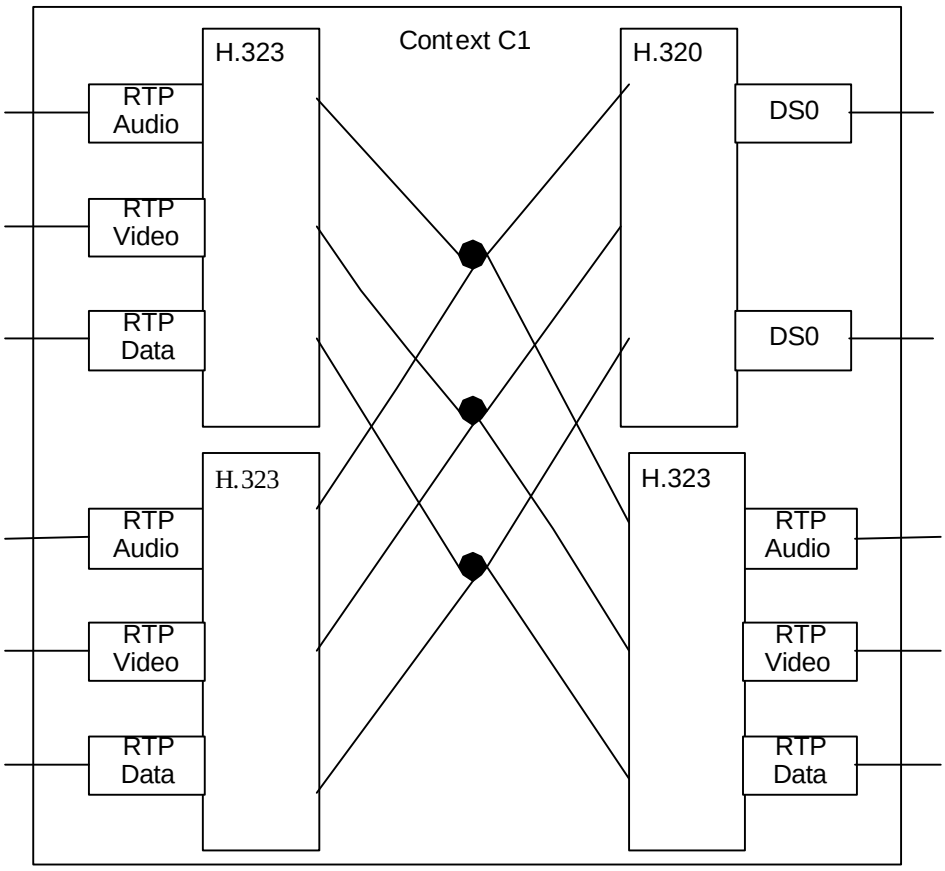
```
<SID> Transaction <TR8>
   ContextID C1
         Modify (TermID=<TID2>
           MediaDescr ([(StreamID=<MID1>), /* unchanged */
             (StreamID=<MID2>, Medium=video,
         Transport=RTP/UDP/IP,
               ReceiveMediaDescr (Activity=on, Encoding
         (H.263, …),
                 BearerDescr (IPaddr=ANY, UDPport=ANY),
               SendMediaDescr (Activity=on, Encoding (H.263,
         …),
                 BearerDescr (IPaddr=<IP3>, UDPport=<P3>))
             ) ])

         Modify (TermID=<TID1>
           MediaDescr ([(StreamID=<MID1>), /* unchanged */
            (StreamID=<MID2>, Media=video,
               ReceiveMediaDescr (Activity=on),
               SendMediaDescr (Activity=on)
            ) ])
```

The MG already knows the parameters of the video stream on the H.320 side. The only thing that the controller has to do is to set the StreamID of the video stream.

8.5.2.2 Multipoint Context Example

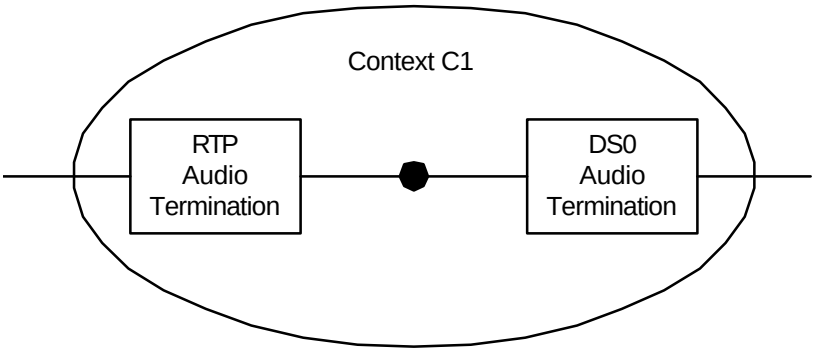
This example shows how a multimedia context can be used to bridge with an H.320 user s and three H.323 users into a single multipoint conference. In the picture the types of media flowing

over the links between the terminations are shown for clarity. The bridging functionality is a context property, there is no separate bridge entity in the connection model.



8.5.2.3 Single Media Call

The single media the call flow example describes a call that originates in the SCN and is terminated in the packet network. The packet network signaling in this example is H.323 but other signaling protocols such as SIP can be used, the purpose of the example is to describe MG/MGC interactions.



The assumption is made that the signalling between the signalling gateway (SGW) and MGC is based on Q.931. This does not indicate that no other signalling can be used on this interface.

1. The SGW sends a *Setup* message to the MGC after receiving an IAM from a SCN switch.
2. From the IAM message, the MGC may infer which circuit on which MG is involved and where to terminate the call. How the MGC does this, is outside the scope of this document.
3. The MGC creates a Context for the call. The Context contains two terminations: one for the SCN side and one for the packet side:

```
<SID> Trans <TRID1>
      Ctxt ANY
            Add ANY
                  NetworkType=ckt
                  BearerDescr ([(Type=DS0, Circuit=13,
                  Facility=2)])
                  ModemDescr (None) /* May be omitted */
                  MuxDescr (None)   /* May be omitted */
                  MediaDescr ([(StreamID=<MID1>, Medium=audio,
                  ReceiveMediaDescr (Activity=off,
                  Encoding=G.711 µ-law, Gain=2dB,
                  VoiceActDet=y, EchoCancelation=G.165),
                  SendMediaDescr (Activity=on)
                    /* No other parameters needed */ )])
                  EventBufferDescriptor
                  (ProcessingMode=Process,
                  NotificationMode=StepByStep)
                  SignalsDescr (<list of packages>)
                  EventsDescr (<list of packages>)
                  ScriptingDescr (none)
            Add ANY
                  NetworkType=IP4
                  MuxDescr (H.225.0)
                  MediaDescriptor ([(StreamID=<MID1>,
                  Medium=audio,
                  Transport=RTP/UDP/IP, MaxJitterBuffer=40ms,
                  ReceiveMediaDescr (Activity=on, BearerDescr
```

```
                    (Ipadd=ANY, UDPPort=ANY, Packetization=20ms,
                    Encoding=[G.711 μ-law, G.723], Gain=0dB),
                    SendMediaDescr (Activity=off, BearerDescr
                    (IPaddr=ANY, UDPport=ANY)) )])
                    EventBUfferDescr (ProcessingMode=Process,
                    NotificationMode=StepByStep)
                    SignalsDescr (<list of packages>)
                    EventsDescr (<list of packages>)
                    ScriptingDescr (none)
```

In general the Media descriptor is a list. The different media flows are identified by a `StreamID`. In the case that there is only one medium, this `StreamID` can be omitted.

We see in the command syntax how the MGC leave the assignment of names for both the Context itself and the Terminations in it to the MG.

4. The MG accepts the Context creation:

```
<SID> TransAccept <TRID1>
   Ctxt <CID1>
         Term <TID1>
         Term <TID2>
               MediaDescr ([(StreamID=<MID1>,
                  ReceiveMediaDescr (BearerDescr (IPaddr=<IP1>,
                    UDPport=<P1>)))])
```

This shows how the MG reports to the MGC what parameters it filled in for the IP address and UDP port to which media should be addressed.

5. The MGC sends a *Setup* message to the destination endpoint, here assumed to be a H.323 endpoint (terminal, GW, …). It indicates in the fastStart element that either G.711 or G.723 may be used for the voice stream.

6. The H.323 endpoint sends an *Alerting* message back to the MGC, informing it of the codec to be used (assume G.723 for both directions) and the transport address.

7. The MGC sends a `Modify` command to the MG to set the Activity and BearerDescriptor for the packet side SendMediaDescriptor:

```
<SID> Trans <TRID2>
      Ctxt <CID1>
              Modify <TID2>
                 MediaDescr ([(StreamID=<MID1>,
                   SendMediaDescr (Activity=on,
                     BearerDescr (IPaddr=<IP2>, UDPport=<P2>),
                             Encoding=G.723))])
```

8. The MG accepts the `Modify` commands:

```
<SID> TransAccept <TRID2>
      Ctxt <CID1>
      Term <TID2> …
```

9. The MGC sends an *Alerting* message to the SGW.

10. The called endpoint at some instance sends a *Connect* message to the MGC.

11. In response to the *Connect*, the MGC sends a `Modify` command to the MG to change the Activity of the ReceiveMediaDescr on the SCN side:

```
<SID> Trans <TRID3>
      Ctxt <CID1>
           Modify <TID1>
              MediaDescr ([(StreamID=<MID1>,
              ReceiveMediaDescr (Activity=on))])
```

12. The MGC sends a *Connect* message to the SGW

13. The MG accepts the Modify command:

```
<SID> TransAccept <TRID3>
   Ctxt <CID1>
         Term <TID1>
```

8.5.2.4 Single Media Call using Templates

The single media the call flow example describes a call that originates in the SCN and is terminated in the packet network. The packet network signaling in this example is H.323 but other signaling protocols such as SIP can be used, the purpose of the example is to describe MG/MGC interactions.

The assumption is made that the signalling between the signalling gateway (SGW) and MGC is based on Q.931. This does not indicate that no other signalling can be used on this interface.

1. After the registration phase, the MGC loads Templates to the MG for voice on DS0 and voice on IP:

```
<SID> Trans <TRID0>
        Ctxt ALL
             Put (<TemplID1>, type=Templ, data = (
                  TemplateID = ANY
                  NetworkType=ckt,
                  BearerDescr ([(Type=DS0, Circuit=ANY,
                  Facility=ANY)])
                  ModemDescr (None) /* May be omitted */
                  MuxDescr (None)   /* May be omitted */
                  MediaDescr ([(StreamID=<MID1>,
                  Medium=audio, ReceiveMediaDescr
                  (Activity=off, packetization=10ms,
                  Encoding=G.711 μ-law, Gain=2dB,
                  VoiceActDet=y, EchoCancelation=G.165),
                  SendMediaDescr (Activity=on) /* no other
                  parameters */ )])
                  EventBufferDescr (ProcessingMode=Process,
                  NotificationMode=StepByStep)
                  SignalsDescr (<list of packages>)
                  EventsDescr (<list of packages>)
                  ScriptingDescr (None) )
```

```
                )
          Put (<TemplID2>, type=Templ, data = (
                NetworkType=IP4
                MuxDescr (H.225.0)
                MediaDescr ([(StreamID=<MID1>,
                Medium=audio, Transport=RTP/UDP/IP,
                MaxJitterBuffer=40ms, ReceiveMediaDescr
                (Activity=on, BearerDescr (IPaddr=ANY,
                UDPport=ANY), Packetization=20ms,
                Encoding=G.723, Gain=0dB), SendMediaDescr
                (Activity=off, BearerDescr (IPaddr=ANY,
                UDPport=ANY)) )])
                EventBufferDescr (ProcessingMode=Process,
                NotificationMode=StepByStep)
                SignalsDescr (<list of packages>)
                EventsDescr (<list of packages>)
                ScriptingDescr (None) )
                )
```

2. The SGW sends a *Setup* message to the MGC after receiving an IAM from a SCN switch.
3. From the IAM message, the MGC may infer which circuit on which MG is involved and where to terminate the call.
4. The MGC creates a Context for the call. The Context contains two terminations: one for the SCN side and one for the packet side. It uses the Templates it loaded at registration time:

```
<SID> Trans <TRID1>
      Ctxt ANY
            Add ANY
                  TemplateID=<TemplID1>
                  BearerDescr ([(Type=DS0, Circuit=13,
                  Facility=2)])
            Add ANY
                  TemplateID=<TemplID2>
                  MediaDescr ([(MediaID=<MID1>,
                  ReceiveMediaDescr (BearerDescr (IPAddr=<IP1>,
                  UDPport=ANY)) )])
```

We see here how the use of Templates leads to smaller messages for setting up calls.

5. The MG accepts the Context creation:

```
<SID> TransAccept <TRID1>
   Ctxt <CID1>
         Term <TID1>
         Term <TID2>
               MediaDescr ([(StreamID=<MID1>,
                 ReceiveMediaDescr (BearerDescr (IPAddr=<IP1>,
                   UDPPort=<P1>)) )])
```

This shows how the MG reports to the MGC what parameters it filled in, in this case the ContextID, TerminationIDs and the local UDP port selected to receive the packet media.

6. The MGC sends a *Setup* message to the destination endpoint, here assumed to be a H.323 endpoint (terminal, GW, …). It indicates in the fastStart element that either G.711 or G.723 may be used for the voice stream.
7. The H.323 endpoint sends an *Alerting* message back to the MGC, informing it of the codec to be used (assume G.723 for both directions) and the transport address.
8. The MGC sends a `Modify` command to the MG to set the Activity and BearerDescriptor for the packet side SendMediaDescriptor:

```
<SID> Trans <TRID2>
      Ctxt <CID1>
          Modify <TID2>
              MediaDescr ([(StreamID=<MID1>,
                  SendMediaDescr (Activity=on
                    BearerDescr (IPAddr=<IP2>, Port=<P2>)) )])
```

   If the called endpoint had indicated it could only do G.711 audio, the `Modify` command would also have had to override the default codec of the Template.
9. The MG accepts the `Modify` commands:

```
<SID> TransAccept <TRID2>
      Ctxt <CID1>
         Term <TID2> …
```

10. The MGC sends an *Alerting* message to the SGW.
11. The called endpoint at some instance sends a *Connect* message to the MGC.
12. In response to the *Connect*, the MGC sends a `Modify` command to the MG to change the activity of the SendMediaDescriptor on the SCN side to SendRecv:

```
<SID> Trans <TRID3>
      Ctxt <CID1>
           Modify <TID1>
              MediaDescriptor ([(StreamID=<MID1>,
              ReceiveMediaDescr (Activity=on) )])
```

13. The MGC sends a *Connect* message to the SGW
14. The MG accepts the Modify command:

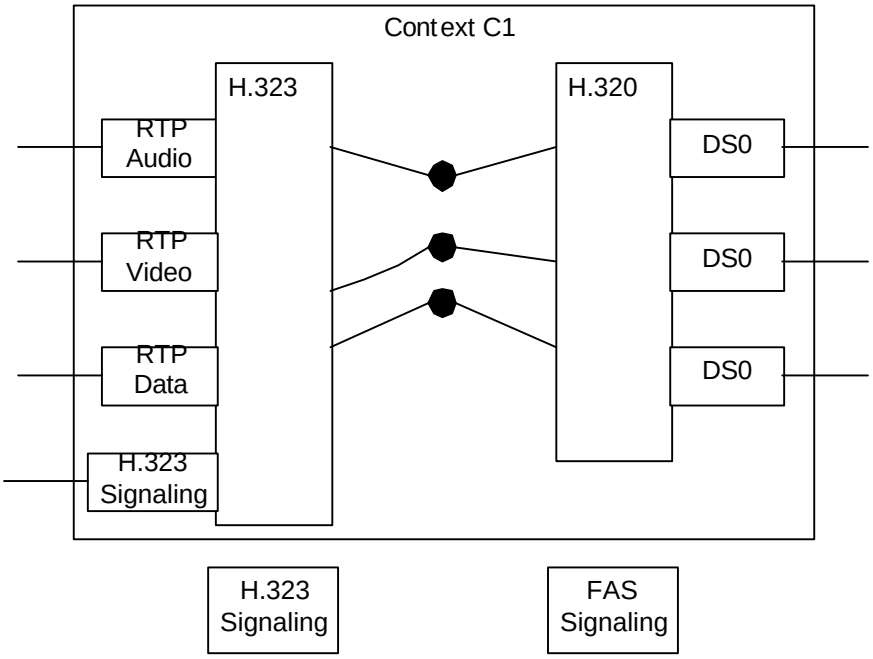
```
<SID> TransAccept <TRID3>
   Ctxt <CID1>
         Term <TID1>
```

After this call is terminated, no new Templates have to be loaded to the MG to set up new calls. In other words, the first Step of the call flow does not have to be repeated.

8.5.2.5 H.323 and FAS Signaling in MG

{Editor's note: This section has not been updated to reflect the changes of the Santiago meeting. }

The following flow describes an H.320 to H.323 gateway call where the signaling and media terminations for the SCN and packet network both reside on the MG. In this case the MG terminates an ISDN PRI interface and the packet interface is H.323 IP also resides on the MG. The signaling capabilities are represented as physical H.323 and FAS terminations that exchange events with the MGC to indicate changes in call state. This gateway session uses Bonding channel aggregation making the example a super-set of gateway call that uses H.221 channel aggregation.



1. The MG creates a FAS termination and an H.323 termination on power-on and reports these events to the MGC in a Notify.
2. MGs FAS termination gets an incoming call with (Q.931) call type of data, the MG sends a Notify to the MGC.
3. The MGC creates a context and termination, containing H.320 and Bonding packages, to connect B channel.
4. When the termination finds Bonding, it assigns a Bonding call ID x and then allocates additional phone numbers for the call and sends these back to the calling side via Bonding.
5. The MG sends a Notify event to the MGC.
6. The MGC sends an Add to the MG to create a multimedia context and sends a Modify to move the DS0 termination in the new multimedia context .
7. The FAS termination detect signaling for 5 additional B channels, sends Notify to MGC, the MGC creates a termination in the MG
8. The MG detect Bonding and sends a Notify to the MGC
9. The MGC sends a modify to move the termination into the multimedia context
10. The remaining B channels connect in the same manner
11. After all five additional DS0s have been added the H.320 termination can complete Bonding and start looking for H.221 framing.
12. Once H.221 framing is found a Notify is sent to the MGC.

13. The H.221 Termination will play the audio announcement and listen for DTMF tones in the audio portion of the mux and it commences a TCS4/IIS signaling exchange in the BAS channel. The information received is considered the destination z alias for this call.
14. The MG Notifies the MGC of the destination alias z.
15. The MGC sends ARQ to resolve IP address for alias z
16. Once the address is resolved, the MGC Adds an H.323 signaling to the multimedia context
17. The H.323 termination in the context starts H.225/h.245 call setup and media negotiation sequence.
18. The MG sends a Notify to the MGC announcing the completion of call signaling.
19. In the MG capabilities received from the H.323 terminations are forward to the H.320 termination and vice versa. The GW sends a Notify to the MGC whenever an H.242 or H.245 media capability exchange occurs.
20. MG may get an OLC from H.323 side for audio, the MG will send a Notify to the MGC
21. The MGC will Add an Audio/RTP termination to the context.
22. The MGC will send a Modify to the H.221 termination causing the H.221 mux to change and the selected audio channel (G.711, G.723 etc.) to be opened. The H.320 side may do a mode switch to H.263 video for example. The H.221 termination will then send a Notify to the MGC requesting H.263 video.
23. The MGC will send an Add Video/RTP termination to the context and a Modify H.320/video to the context.

8.6 Transport

The transport mechanism for MGC/MG communication has not been chosen. TCP and possibly TUDP, UDP with Timeouts or SIGTRAN are all options for this transport implementation. It may be necessary to specify a transport protocol in this specification.

8.6.1 Transport capabilities, and relationship to Transport Layer

MEGACO/H.GCP transport needs are the same in all application states and independent of what particular commands are being sent, it is logical to think of the reliable transport as a separate layer, as shown below. However, there is no accepted ITU/IETF protocol that focuses on the needs of real time control as is needed by this protocol. Therefore, the mechanisms to provide the required characteristics of the reliable transport should be directly included in the MEGACO/H.GCP protocol layer.

{Editors Note, ran out of time to define the interface between MEGACO/H.GCP and Transport}

8.7 Security

If unauthorized entities use the protocol, they would be able to set-up unauthorized calls, or to interfere with authorized calls. The primary security mechanism employed by the protocol is IPSEC [RFC2401]. Support of the AH header [RFC2402] affords authentication and integrity protection on messages passed between the MG and the MGC. Support of the ESP header [RFC2406] can provide confidentiality of messages if desired.

Implementation of IPSEC requires that the AH and ESP headers be inserted between the IP and UDP headers. This presents an implementation problem for MEGACO/H.GCP protocol

implementations where the underlying network implementation does not support IPSEC. As an interim solution, the MEGACO/H.GCP protocol defines an optional AH header within the protocol header. The header fields are exactly those of the AH header as defined in [RFC2402]. The semantics of the header fields are the same as the "transport mode" of [RFC2402], except for the calculation of the Integrity Check Value (ICV). In IPSEC, the ICV is calculated over the entire IP packet including the IP header. This prevents spoofing of the IP addresses. To retain the same functionality, the ICV calculation should be performed across the entire transaction prepended by a synthesized IP header consisting of a 32 bit source IP address, a 32 bit destination address and an 16 bit UDP port in the MSBs of a 32 bit word. The Authentication Data is assumed to be zero as in [RFC2402]. The "Next Header" and "RESERVED" fields MUST be set to "zero".

The ICV calculation is thus performed over a structure that would look like:

When the AH-within-MEGACO/H.GCP mechanism is employed when TCP is the transport Layer, the UDP Port above becomes the TCP port, and all other operations are the same.

Implementations of this protocol using IPv4 MUST support the interim AH-within-MEGACO/H.GCP scheme. Implementations SHOULD implement IPSEC AH header if the underlying network system supports it. Implementations may support the ESP header. IPSEC and AH-within-MEGACO/H.GCP must not be used at the same time. IPv6 Implementations are assumed to have IPSEC implementations and must not use the AH-within-MEGACO/H.GCP scheme.

When employing the AH header, either in IPSEC or AH-within-MEGACO/H.GCP, all implementations of the protocol MUST implement section 5 of [RFC2402] which defines a minimum set of algorithms for integrity checking using manual keys. MEGACO/H.GCP implementations SHOULD implement IKE [RFC2409] to permit more robust keying options. MEGACO/H.GCP implementations employing IKE SHOULD implement RSA signatures and authentication with RSA public key encryption.

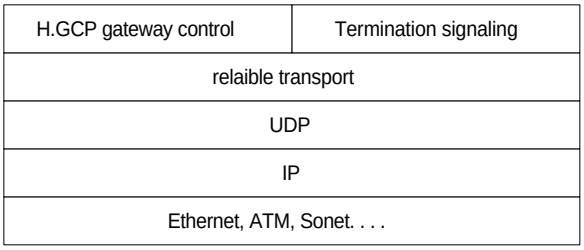


Figure 13 Security Structrure

MEGACO/H.GCP implementations employing the ESP header [RFC2406] MUST implement section 6 of [RFC2406] which defines a minimum set of algorithms for integrity checking and encryption.

NOTE: The AH-within-MEGACO/H.GCP scheme is defined as interim.

Adequate protection of the connections must be achieved if the MG and the MGC only accept messages for which authentication services of the AH header have been configured. Employing the ESP header for encryption service must provide additional protection against eavesdropping, thus forbidding third parties from monitoring the connections set up by a given termination

The encryption service should also be requested if the session descriptions are used to carry session keys, as defined in SDP.

These procedures do not necessarily protect against denial of service attacks by misbehaving MGs or misbehaving MGCs. However, they will provide an identification of these misbehaving entities, which should then be deprived of their authorization through maintenance procedures.

8.7.1 Protection of Media Connections

The protocol allows the MGC to provide MGs with "session keys" that can be used to encrypt the audio messages, protecting against eavesdropping.

A specific problem of packet networks is "uncontrolled barge-in." This attack can be performed by directing media packets to the IP address and UDP port used by a connection. If no protection is implemented, the packets must be decompressed and the signals must be played on the "line side".

A basic protection against this attack is to only accept packets from known sources, checking for example that the IP source address and UDP source port match the values announced in the RemoteTerminationDescriptor. This has two inconveniences: it slows down connection establishment and it can be fooled by source spoofing:

- To enable the address-based protection, the MGC must obtain the remote session description of the egress MG and pass it to the ingress MG. This requires at least one network roundtrip, and leaves us with a dilemma: either allow the call to proceed without waiting for the round trip to complete, and risk for example, "clipping" a remote announcement, or wait for the full roundtrip and settle for slower call-set-up procedures.
- Source spoofing is only effective if the attacker can obtain valid pairs of source destination addresses and ports, for example by listening to a fraction of the traffic. To fight source spoofing, one could try to control all access points to the network. But this is in practice very hard to achieve.

An alternative to checking the source address is to encrypt and authenticate the packets, using a secret key that is conveyed during the call set-up procedure. This will not slow down the call set-up, and provides strong protection against address spoofing.

8.8 MG-MGC Control Interface

The control association between MG and MGC is initiated at MG cold start, and announced by a ServiceChange message, but can be changed by subsequent events, such as failures or manual service events. While the protocol does not have an explicit mechanism to support multiple MGCs controlling a physical MG, it has been designed to support the multiple logical MG (within a single physical MG) that can be associated with different MGCs.

8.8.1 Multiple Virtual MGs

A virtual MG consists of a set of statically partitioned Terminations. The model does not require that other resources be statically allocated, just Terminations. The mechanism for allocating Terminations to virtual MGs is a management method outside the scope of the protocol. Each of the virtual MGs appears to the MGC as a complete MG client.

In many cases, a physical MG may have only one network interface, which must be shared across virtual MGs. In such a case, the packet/cell side Termination is shared. It should be noted however, that in use, such interfaces require an ephemeral instance of the Termination to be created per flow, and thus sharing the Termination is straightforward. This mechanism does lead to a complication, namely that the MG must always know which of its controlling MGCs should be notified if an event occurs on the interface.

In normal operation, the MG will be instructed by the MGC to create network flows (if it is the originating side), or to expect flow requests (if it is the terminating side), and no confusion will arise. However, if an unexpected event occurs, the MG must know what to do.

If recovering from the event requires manipulation of the interface state, there can be only one MGC who can do so. These issues are resolved by allowing any of the MGCs to create EventDescriptors to be notified of such events, but only one MGC can have read/write access to the physical interface properties; all other MGCs have read-only access. The management mechanism is used to designate which MGC has read/write capability, and is designated the Master MGC.

Each virtual MG has it's own Root Termination. In most cases the values for the properties of the Root Termination are independently settable by each MGC. Where there can only be one value, the parameter is read-only to all but the Master MGC.

8.8.2 Cold Start

An MG is pre-provisioned by a management mechanism outside the scope of this protocol with a Primary and (optionally) an ordered list of Secondary MGCs. Upon a cold start of the MG, it will issue a ServiceChange command with a "Restart" method, on the Root Termination to it's primary MGC. If the MGC accepts the MG, it will send a Transaction Accept, with the MGCIdenty set to itself. If the MG recieves an MGCIdentity not equal to the MGC it contacted, it sends a ServiceChange to the MGC specified in the MGCIdentity. It continues this process until it gets a controlling MGC to accept its registration, or it fails to get a reply. Upon failure to obtain a reply, either from the Primary MGC, or a designated successor, the MG tries it's pre-provisioned Secondary MGCs, in order.

8.8.3 Failure of an MG

If an MG fails, but is capable of sending a message to the MGC, it sends a ServiceChange with an appropriate method (graceful or forced) and specifies the root TerminationID. When it returns to service, it sends a ServiceChange with a "Restart" method.

Pairs of MGs that are capable of redundant failover of one of the MGs are accommodated by allowing the MGC to send duplicate messages to both MGs. Only the Working MG must accept or reject transactions. Upon failover, the Primary MG sends a ServiceChange command with a "Failover" method and a "Failed MG" reason. The MGC then uses the primary MG as the active MG. When the error condition is repaired, the Working MG can send a "ServiceChange" with a "Restart" method.

8.8.4 Failure of an MGC

If the MG detects a failure of it's controlling MGC, it attempts to contact the next MGC on its pre-provisioned list. It starts it's attempts at the beginning (Primary MGC), unless that was the MGC that failed, in which case it starts at it's first Secondary MGC. It sends a ServiceChange message with a "Failover" method and a "Failed MGC" reason.

In partial failure, or manual maintenance reasons, an MGC may wish to Direct its controlled MGs to use a different MGC. To do so, it sets the "ControllingMGC" property of the root Termination to its new MGC. As a side effect, the MG should send a ServiceChange message with a "Forced" method and a "MGC directed change" reason to the designated MGC. If it fails to get a reply, or fails to see an Audit command subsequently, it should behave as if it's MGC failed, and start contacting secondary MGCs.

When the MGC initiates a failover, the handover should be transparent to Operations on the gateway. Commands in progress continue, transaction replies are sent to the new MGC, and the MG should expect outstanding transaction replies from the new MGC. All connections should stay up.

It is possible that the MGC could be implemented in such a way that a failed MGC is replaced by a working MGC where the identity of the new MGC is the same as the failed one. In such a case, "ControllingMGC" would be replaced with the previous value. In such a case, the MG shall behave as if the value was changed, and send a ServiceChange message, as above.

Pairs of MGCs that are capable of redundant failover can notify the controlled MGs of the failover by the above mechanism.

9. SIGNALING BACKHAUL

Signaling backhaul refers to transparent transportation of signaling for the B, C or D interface from the MG to the MGC. This function is For Further Study.