INTERNATIONAL
TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 1997 –2000

APC-xxxx

May 1999

Original: English

Question 14/16

STUDY GROUP 16 - CONTRIBUTION

SOURCE[*] : Bryan Hill
TITLE: Proposed Recommendation Gateway Control Protocol

Summary

This contribution provides the text for draft Recommendation "Gateway Control Protocol" . It is proposed that this draft be decided at the May 19xx meeting of SG16.

[*] Contact: Bryan Hill
VideoServer, Incorporated
Burlington, MA. USA
E-mail: bhill@videoserver.com
Tel.: +1 781-505-2159
Fax: +1 781-505-2101

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.GCP
(May 1999)

Gateway Control Protocol

ITU-T Recommendation H.GCP

FOREWORD

The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the International Telecommunication Union. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, established the topics for study by the ITU-T Study Groups, which, in their turn, produce Recommendations on these topics.

ITU-T Recommendation H.GCP was prepared by the ITU-T Study Group 16 (1998-1999) and was approved by the WTSC (Place, Month xx-xx, 199x).

Summary

To achieve greater scalability, this recommendation decomposes the H.323 Gateway function defined in H.246 into functional sub-components and specifies protocols these components use to communicate. This allows implementations of H.323 gateways to be highly scalable and encourages leverage of widely deployed SCN network capabilities such as SS7 switches. This also enables H.323 gateways to be composed of components from multiple vendors distributed across multiple physical platforms. The purpose of the this recommendation is to add capabilities currently defined for H.323 systems and is intended to provide new ways of performing operations already supported in H.323.

Table of Contents

1. SCOPE

Recommendation H.GCP breaks up the H.323 multimedia gateway model defined in H.246 into functional sub-components and defines protocols over which the components interact. There are no functional differences between H.GCP gateways, with distributed sub-components potentially on more than one physical device and H.246 gateway. The protocols required to achieve this decomposition will reuse existing H.323 standards and propose extensions as needed. In the event the existing protocols are not sufficient a new protocol will be designed.

This recommendation does not define how H.323 gateways, multipoint control units or integrated voice response units (IVRs) work. Instead it creates a general framework that is suitable for these applications.

H.GCP gateways provide protocol interworking between H-series multimedia terminals and other H-series multimedia terminals, voice/voiceband terminals on GSTN or ISDN, and multi-call applications on the GSTN. H.323 gateways provide the required translation of control and media streams to allow interworking between terminals running different protocols.

Packet network interfaces may include UDP/IP or ATM interfaces. H.323 gateways must support G.711 audio and H.261 video and may support other media algorithms such as G.728, G.729, G.722 and H.263.

The SCN side of the gateway 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.

This recommendation will support and improve interworking between H.323 and other H-series protocols where this interworking falls within the scope defined as

- packet network to switched circuit network,
- switched circuit network to packet network,
- switched circuit network to switched circuit network via a packet network, and
- packet network to packet network via a switched circuit network.

This recommendation shall support the interworking of supplementary services between H.323/H.450.x and other signalling systems.

The ITU protocols for H.323 Multipoint Control (MC) to Multipoint Processor (MP) as well as H.320 MCUs are also in scope. However, the new MC-MP protocol work shall avoid duplication of H-series multipoint control functions.

2. REFERENCES

1. ITU-T SG16 Monterey TD89 (1999)

2. ESTI TIPHON DTS 02003 v.0.0.1 (1999-03)
3. IETF <draft-ietf-megaco-protocol-00.txt>: “MEGACO Protocol Proposal Working Draft”.
4. ITU-T Recommendation H.225.0 (1998): " Media Stream Packetization and Synchronization for Visual Telephone Systems on Non-Guaranteed Quality of Service LANs ".
5. ITU-T Recommendation H.245 (1998): "Control of communications between Visual ???????
6. ITU-T Recommendation H.323 (1998): “Visual Telephony Systems and Equipment for Local Area Networks which provide a non-guaranteed Quality of Service”
7. 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".

3. TERMS AND DEFINITIONS

Access-Gateway: A type of gateway that provides the User to Network (UNI) network interface such as ISDN and may be supported by a Media Gateway.
Back-haul: This refers to the transport of signaling information from a media termination gateway like a MG to a signaling gateway such as MGC.
Gatekeeper (GK): This H.323 entity function performs man functions including authentication, authorization, alias resolution and call routing for H.323 entities.
H.323 Signaling: This function in the decomposed gateway interfaces to the H.323 gatekeeper for IP side connections using the GK routed call model and/or directly to other H.323 entities like other H.323 gateways and multimedia codec’s that make connections using the direct call model.
Media Gateway (MG): The media gateway terminates and processes SCN media channels, and IP media channels. This gateway will be capable of processing audio, video and T.120 and will be capable of full duplex media translations. An H.323 gateway is a type of media gateway that may perform audio transcoding. The MG may also play audio/video message and perform other IVR functions or may perform media conferencing.
Media Gateway Controller (MGC): Controls the parts of the call state that pertain to connection control for media channels in a MG. It maps SCN signaling and call control information into the packet network call state and control information. This gateway includes an H.323 interface communicating with H.323 endpoints, gatekeepers and MCUs.
Multipoint Processors (MP) An H.323 function that may provide centralized processing of audio, video and or T.120 data streams in a H.323 multipoint conference.
Multipoint Controller (MC): An H.323 function that provides setup and coordination of multi-media conferences.
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. An H.323 MCU consists of MC functionality and possible one or more MC functions.
Network Access Servers: A gateway function in a MG that converts modem signals from an SCN network and provides data access to the Internet.
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 networks and packet network that typically terminates a large number of digital circuits.
Voice over ATM gateways: A gateway between an SCN network and ATM network that typically terminates a large number of digital circuits.

4. ABBREVIATIONS AND ACRONYMS

This recommendation defines the following terms.

ATM	Asynchronous Transfer Mode
BRI	Basic Rate Interface
DTMF	Dual Tone Multi Frequency
FAS	Facility Associated Signalling
GK	GateKeeper
GW	GateWay
IP	Internet Protocol
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. ARCHITECTURE

5.1 Reference Model

The terms of reference [1] set forth a group of interfaces and functions to be used to decompose H.323 gateways. This 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.

In the figure below the packet/circuit media component terminates SCN media channel and converts these streams to packet based media on the packet network interface. The controller will use the interface A protocol to create, modify and delete gateway media connections. The control logic component will accomplish signalling interworking between the SCN and H.323 sides of the gateway.

Interface B is a protocol that will describe the packet signalling transport/interworking interface between the H.323 entities on the IP network and the decomposed gateway controller function. Interface C will describe the ISDN type call control function between the FAS SCN services and the gateway control logic. Interface D is a protocol that conveys the NFAS SCN signalling function and the controller. This decomposition provides the flexibility to conserve ss7 code points and allows the SS7 switch to serve multiple decomposed gateway controllers.

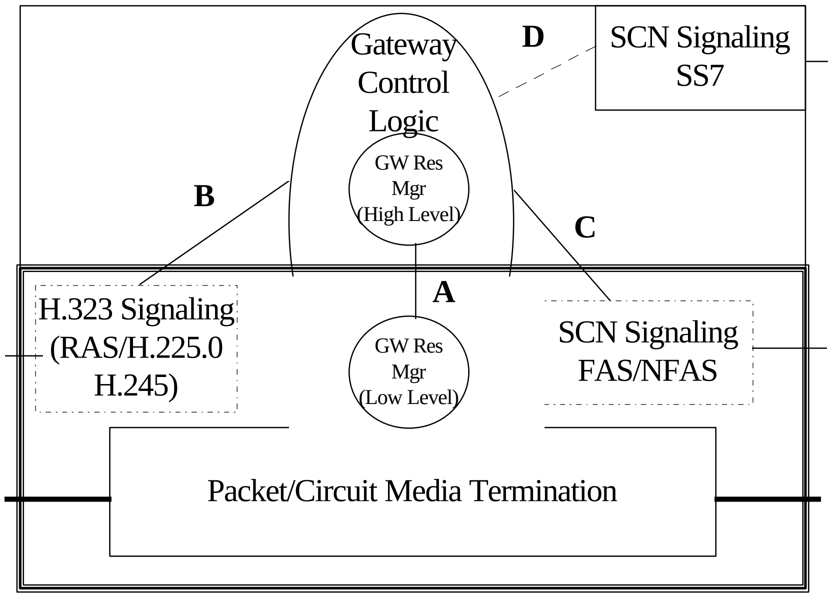


Figure 1: H.323 Gateway Decomposition (TD89)

The ESTI TIPHON reference architecture [2] represents another effort to address the same gateway decomposition problem. The figure below in the ETSI TIPHON functional architecture for creating highly scalable SCN/IP voice gateways. A complete description of this model is available in reference [2]. The functionality encompassed by the Media GW Controller (MGC), The Signaling Gateway (SG) and Media GW (MG) address the same functionality as the SG16 terms of reference gateway decomposition model above.

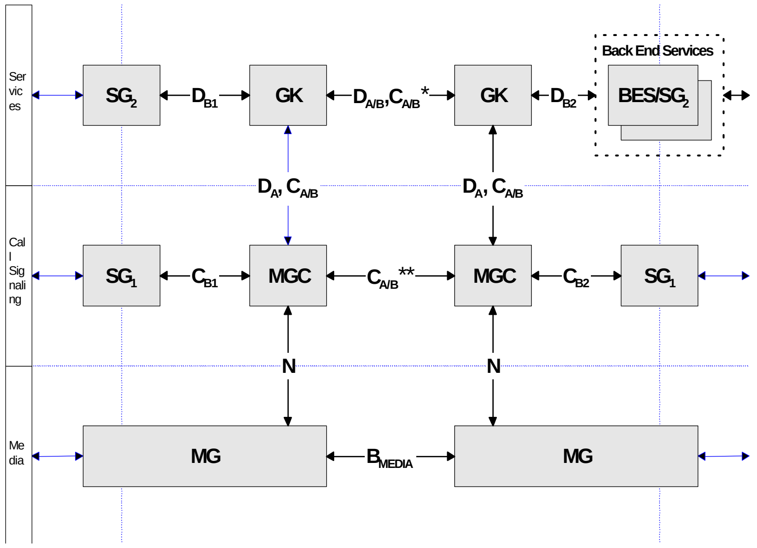


Figure 2: ETSI TIPHON Functional Architecture

* exists for a Gatekeeper routed calls
** exists for a Direct routed calls
D = Services, C = Call signalling, B = Media
Suffix A is used to identify information flows that relates to the IP part of the TIPHON network.
Suffix B is used to identify information flow that relates to the SCN network.

The architecture model for H.GCP merges these models and further abstracts the functional components (see figure 3). The notation in the figure uses a colon (:) to separate the ETSI TIPHON and SG16 ToR name for each interface. Where interfaces were not explicitly defined by TIPHON or SG16 are denoted by a question mark (?).

The TIPHON C (sub A/B) interface represents the H.323 signaling on the IP side and in this figure it has been split into C1, C2 and C3 representing the H.225 RAS, H.225 and H.245 protocol components respectively. The functionality In the H.323 RAS, H.225 and H.245 bubbles from [1] are broken into three separate bubbles in Figure 3.

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

The SCN interfaces describes a low level interface that terminates the protocol on the network interface and a high level signaling termination that interface these components with the controller of this gateway. The FAS signaling interface is unique to the of SG16 terms of reference and the NFAS (ss7) interface that was present in both models.

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 H.323 gateways. The H.GCP recommendation will define these interfaces such that a wide range of decomposed H.323 gateway implementations are possible.

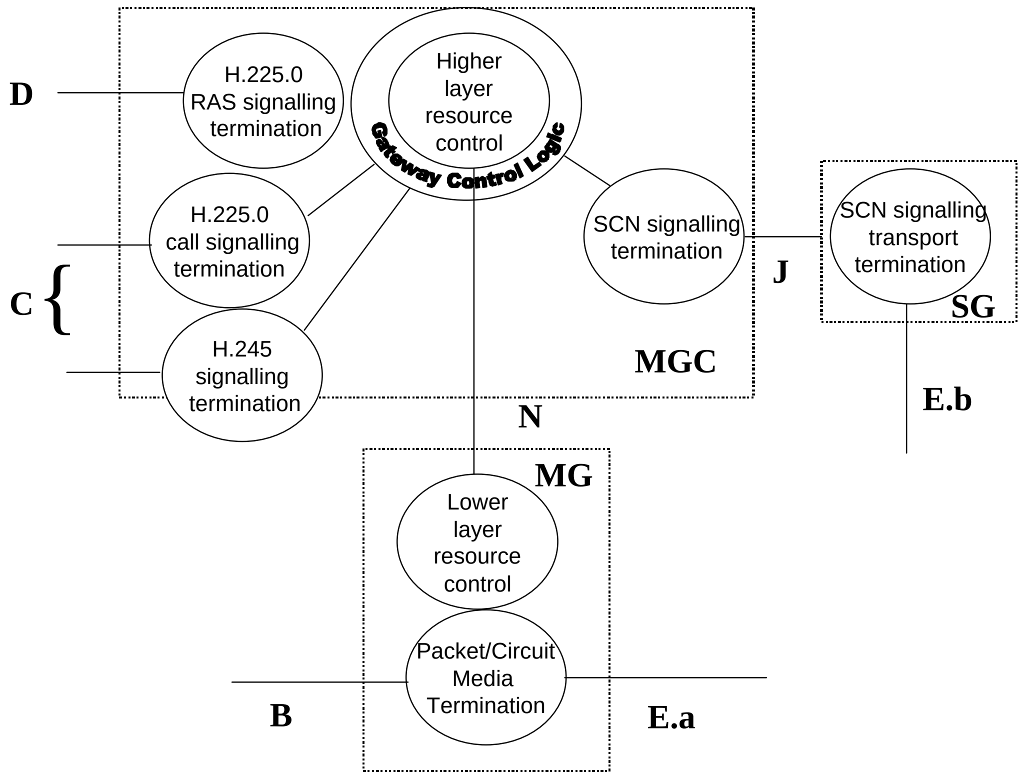

D
H.225.0 RAS signalling termination
Higher layer resource control
Gateway Control Logic
H.225.0 call signalling termination
C
H.245 signalling termination
SCN signalling termination
J
SCN signalling transport termination
SG
MGC
N
E.b
MG
Lower layer resource control
Packet/Circuit Media Termination
B
E.a

Figure 3: H.GCP Functional Architecture

5.2 Physical Decompositions

This section begins to refer to the gateway functions and interfaces present above as part of section 5.1 into physical devices. The controller portion of the physical gateway is called the Media Gateway Controller (MGC) and it may handle the SS7 and H.323 signaling interfaces and it will handle H.225 RAS messaging with an external gatekeeper. The Media Gateway (MG) component will terminate the IP network interface and the SCN network span and may handle H.323 signaling and/or FAS SCN signaling in some physical decompositions. Decomposed gateways need not realize all interfaces but the MGC/MG split exposing interface A is a fundamental part of all decompositions. This will allow an MGC to control different types of MGs that may be optimized for certain application (e.g. voice versus multimedia H.320/H.323 gateways).

The MG terminates the RTP/RTCP on the packet network side and bearer channels on the SCN network interfaces. The IP side may be an ATM network interface where audio and video packets traverse native ATM SVC connections according to ITU H.323 Annex C.

The MGC and MG differentiate between high-level and low-level resource management elements. 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. In contrast the MGC is responsible for high-level resource management where it understands the availability of resources, such as echo canceller, but does not assigned specific resources to specific gateway sessions.

The binding between MGC and MG will be a many MG to single MGC arrangement where the MG also is aware of a standby MGC. The switchover of a MG to another MGC can occur due to a graceful switchover or after a timeout indicating loss of a Primary MGC for any number of reasons.

5.2.1 Separate SS7 Gateway

The figure below represents 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 J:d and a device control interface N:a.

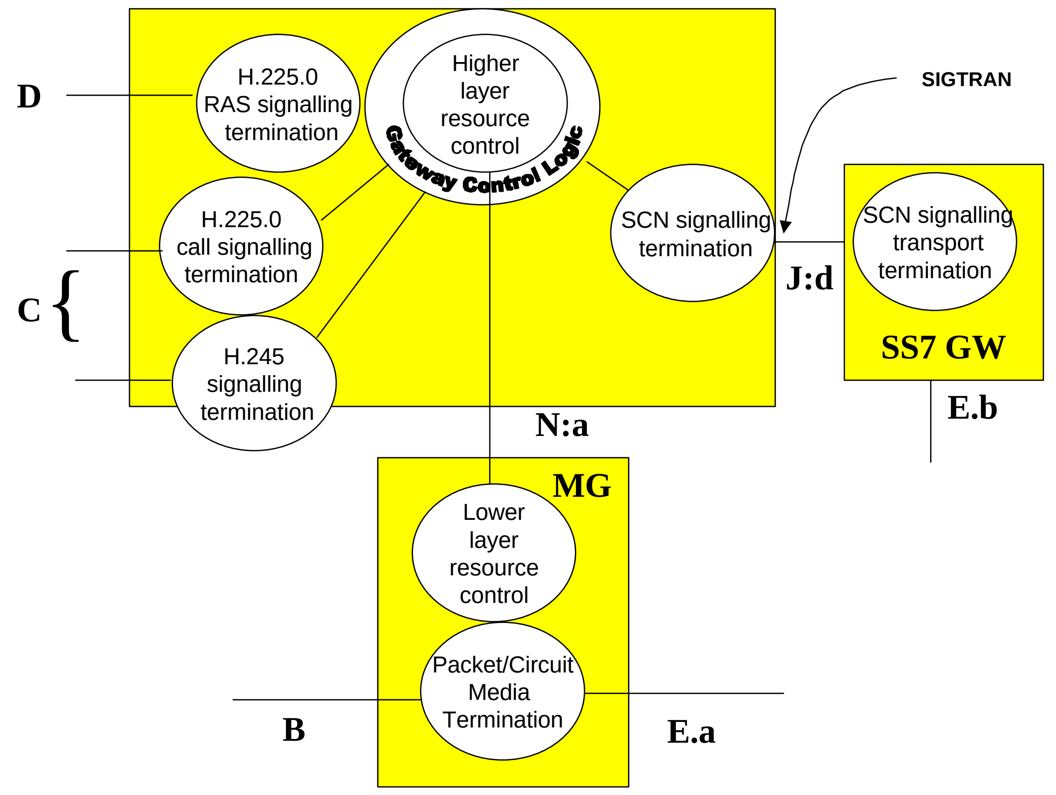

D
C
H.225.0 RAS signalling termination
H.225.0 call signalling termination
H.245 signalling termination
Higher layer resource control
Gateway Control Logic
SCN signalling termination
SIGTRAN
SCN signalling transport termination
SS7 GW
J:d
E.b
N:a
MG
Lower layer resource control
Packet/Circuit Media Termination
B
E.a

Figure 4, SS7 Gateway Decomposition

5.2.2 FAS Gateway Decomposition

5.2.3 SS7 Gateway with H.323 Signaling in the MG

5.2.4 FAS and H.323 Signaling in the Media Gateway

6. REQUIREMENTS

6.1 General Requirements

The following are general requirements

1. Interfaces A, B and C will be considered by this recommendation
2. Audio only, H.320 and other H series multimedia gateway connections will be considered
3. Multimedia ATM sessions will be supported according to H.323 Annex C
4. Voice ATM sessions will be supported over AAL1, AAL2 and AAL5
5. MCUs will be supported
6. The MGC will register with the GK once on behalf of all MGs it controls.

6.2 Interface A Requirements

The MGC/MG media control interface consists of 4 components to setup and control media in a MG. These are

1. Connection Control - Creation, modification, and deletion of media stream connections within the Media Gateway.
2. Media Attributes - Specification of the transformations to be applied to media streams as they pass through the Media Gateway, both initially as connections are created and subsequently during the life of the connection.
3. Content Insertion - Requests to the Media Gateway to insert content (tones and announcements) into the media streams, either on explicit request from the Media Gateway Controller or beginning and ending with the detection of specified events within entity itself.
4. Event Handling - Requests to the Media Gateway to report and possibly take actions upon detection of well-defined events within the media streams.

A second group of requirements are necessary to construct a robust capability between an MGC and MG, these are

1. Modularity and Extensibility – The protocol must allow MGC to determine specific capabilities of each MG and allows future improvements to products to be added
2. Resource Management – The MG must have the ability to report resource availability to MGC
3. Control Session Management – these requirements define the way MGC and MG announce their existence to each other. This association will maintain a high availability and reliability.
4. Control Session Security – The MGC/GC connection must provide security

In general the Media Gateway shall have the capability to indicate to the controller whenever some or all of a request cannot be executed. It shall be possible for the Media Gateway to indicate the general nature of the problem and to provide further details, possibly including vendor-specific content.

The general design goals are scalability built around a low-level control interface (interface A). Other design goals are,

1. to minimize encoding and decoding complexity.
2. Either the MGC or the MG can initiate an association across the interface or this can occur automatically.
3. Use appropriate methods to minimize traffic and delay
4. Support optional vendor specific extensions
5. A mechanism that supports communication confirmation and associated success and failure responses.
6. support for redundant controllers
7. Support a mechanism to negotiate versions and extensions of the protocol.
8. An MGC may query an MG for version information.

6.2.1 Connection Control

Connection Control shall support at least the following types of linkage:

1. circuit to packet for IP
2. circuit to packet for ATM (e.g. H.323 Annex C operation)
3. circuit to circuit (e. g. circuit side fallback or circuit ATM)

4. packet to packet (IP or ATM)

The connection mode may be unicast or multicast, or inactive and it may be uni-directional or bi-directional. On Internet connections, IPv4 is mandatory and IPv6 may be supported.

On the circuit side, SS7 requires the ability to perform a loopback for continuity testing. This requirement applies especially to loopback through a circuit at the edge of the Media Gateway but different types of loopbacks are possible and should be supported by the media gateway.

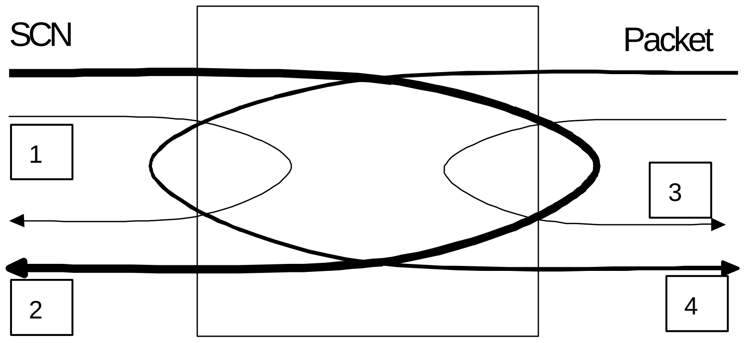


Figure 5, Loopbacks

The Media Gateway control interface shall support the following capabilities for connection control.

1. Ability to identify the IP, ATM, and/or circuit local network interface points, and local connections within the Media Gateway between two or more of these interfaces. These interfaces may be designated using a “hierarchical identifier” for both physical (carrier, card, circuit) and/or logical (trunk group/trunk).
2. It must be possible to wildcard the low-order portion of the network interface point allowing the Media Gateway to select it based on availability and return its full attribute list to the controller.
3. Wildcarding shall be provided to allow a simultaneous reference to multiple endpoints.
4. Ability for controller to select RTP port or the MG may specify the port to use. RTCP Port selection is optional.
5. Ability to convey the requested QoS parameters applicable on the packet side of a media stream connection.
6. Ability to convey the requested bearer capabilities applicable on the SCN side of a media stream connection.
7. Ability to convey QoS statistics for an established connection at any time during the call and at teardown.
8. Support for loopbacks as defined in ITU-T H.320. {ed. where does this belong?}

6.2.2 Media Attributes

The protocol shall be able to convey the following media attributes:
1. Media protocol used (RTP, fax-protocol, ...)
2. Payload type (e.g. codec),
3. Codec-related attributes like packetization interval, jitter buffer size and silence suppression where appropriate

4. Generation of comfort noise during silent periods.
5. Application of encryption/decryption and identification of the encryption schemes.
6. Echo cancellation
7. Lawful interception of the content of a specified media stream.
8. It should be possible to support additional endpoint attributes as they are defined without modifying the existing semantics.
9. Asymmetric media connections (i.e. where different audio or video algorithms are used in each direction of a full duplex gateway media flow).

The protocol shall allow
1. the specification of the endpoint attributes when the connection is created;
2. Modification of endpoint attribute values for an already-existing connection (e.g. in response to a H.245 FlowControlCommand or ReplacementFor OLC).

6.2.3 Content Insertion

The protocol shall support
1. the ability to request the sending of a specified tone or announcement, at any time, in a specified direction
2. the specification of the conditions under which the sending should stop
3. The ability to request muting of a media stream.
4. The ability for the controller to request the Media Gateway to insert and detect tones as required for SS7 continuity testing and other forms of testing.

6.2.4 Events

1. The MGC shall support the ability to instruct the Media Gateway to detect, notify and possibly act upon specified events (e.g. DTMF tones).

6.2.5 Modularity and Extensibility

1. It is essential that the protocol be both modular and extensible.
2. The protocol shall provide the means whereby an MGC can determine the capabilities supported by a particular Media Gateway.
3. The protocol shall support backward compatibility as new versions are released.
4. The protocol shall allow the possibility of vendor-specific extensions.

6.2.6 Resource Management

1. The protocol shall provide the means for the controller to determine resource availability within the associated media gateway. The protocol shall allow for unsolicited messages between the Media Gateway and Media Gateway Controller.
2. It shall be possible for the Media Gateway to indicate to the controller that it lacks sufficient resources to carry out a given command.
3. It shall be possible for the Media Gateway Controller to audit the commitment of resources to connections. It shall further be possible for the controller to order that specific resource assignments be cleared if it finds that they are invalid.
4. It shall be possible for the Media Gateway Controller to audit the Connection State of connections in Media Gateways with which it is associated.

5. It shall be possible for the Media Gateway to report changes in operational status of significant resources from in-service to out-of-service and vice versa.

6.2.7 MG-MGC Association Management

1. The protocol shall provide the means to establish and remove an MGC-MG association between a specific controller and a specific Media Gateway.
2. It shall be possible for a Media Gateway to establish an MGC-MG association with an alternate Media Gateway Controller if its currently associated controller becomes unavailable.
3. It shall be possible for either the Media Gateway or the Media Gateway Controller to detect loss of the control association.

6.2.8 Scripting Capability

The protocol shall provide the means to download scripts to be executed autonomously by the Media Gateway.

6.2.9 Security

Provide defense against the following threats:

1. denial of service attacks
2. unauthorized use of resources
3. modification of message content
4. loss of confidentiality of user service
5. repudiation of commands

6.2.10 Transport

1. Reliable delivery of messages
2. Ordered delivery of commands/responses to a particular "control stream", this implies ordered delivery of commands to/from a particular Termination or Context, but not necessarily ordered across Terminations or Contexts.
3. Limited maximum time to deliver commands.
4. Rapid detection of failure in a control stream.
5. Ability to achieve very high fanout from MGC to MGs.
6. Ability to handle multiple MGCs controlling individual MGs in a distributed system and vice versa. However this must be optional so that smaller/simpler systems can be efficiently implemented.
7. Ability for the application to initiate flushing of messages successfully sent through the transport, for example to back off for failover handling.

7. PROTOCOL

7.1 Overview

The H.GCP 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.

<EDITOR: More will be added here as the protocol is worked out… >

7.2 Connection Model

7.2.1 Basic Concepts

The connection model for the H.GCP protocol describes the logical entities within the Media Gateway that can be controlled by the Media Gateway Controller using the protocol. The main abstractions used in the connection model are Terminations and Contexts.

Terminations are logical entities representing physical endpoints, such as analog loops or timeslots on a Time Division Multiplexed (TDM) channel, as well as more ephemeral representations of flow terminations, such as RTP streams

Contexts represent a star configuration (i.e.-a variable number of interconnected nodes) of Terminations of a single media type. The Context configuration illustrates a logical association between the Terminations belonging to the Context. For example, a Context containing a TDM audio Termination and an RTP stream audio Termination would describe an audio SCN to IP call. A Context will always contain one or more Terminations. It is possible for a Termination to exist outside of a Context. An example of this is a TDM channel not being used in an active call. 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.

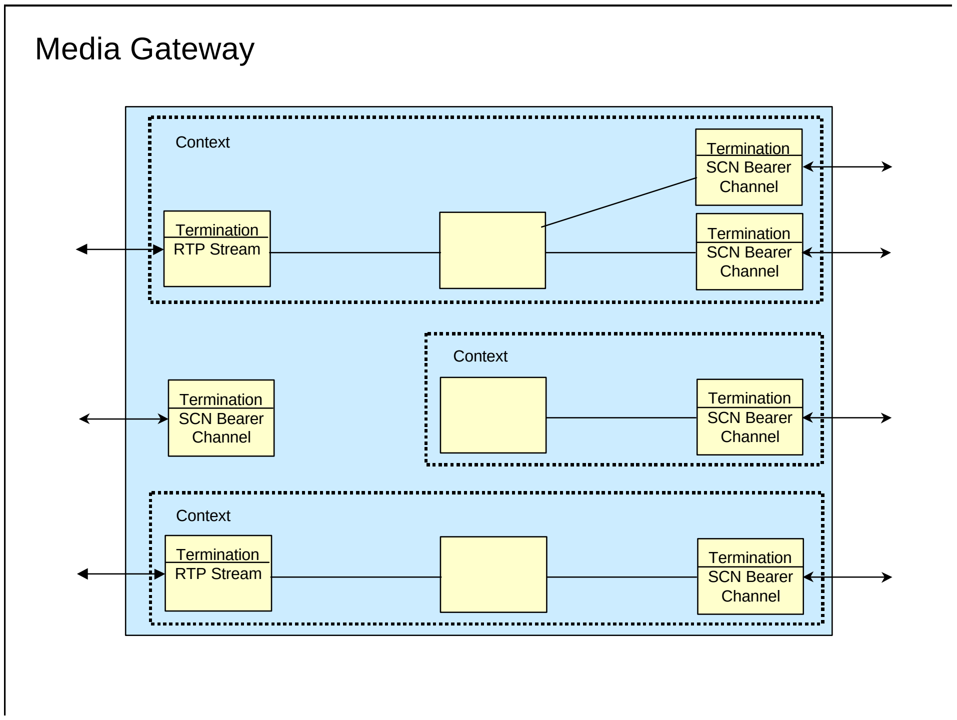


Figure 6: Example H.GCP Connection Model

This example call waiting scenario illustrates the relocation of a Termination between Contexts. Terminations T1 and T2 belong to Context C1 in 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.

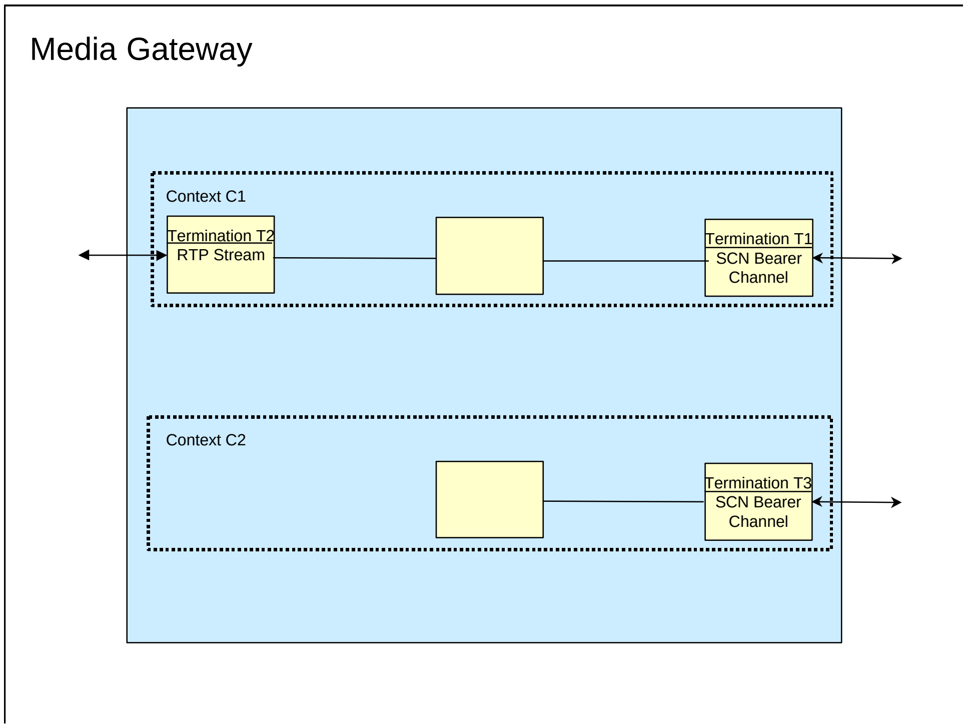


Figure 7: Call Waiting Scenario / Alerting Applied to T1

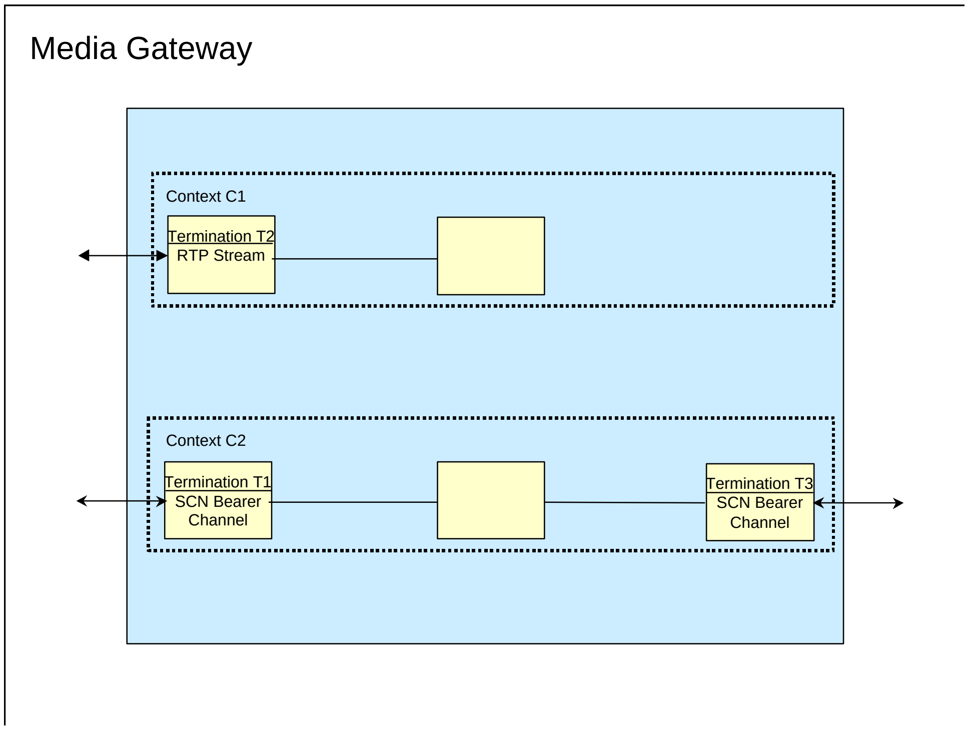


Figure 8: Call Waiting Scenario / Answer by T1

7.2.2 Contexts

7.2.2.1 Overview

Contexts are logical associations of Terminations of a single media type. Contexts may be linked together to provide logical associations between Contexts, as necessary.

<Editor an example of Context linking may be the association of audio and video Contexts of a multimedia call (e.g.-for lip synchronization). . . see open issues 1,2,3>

7.2.2.2 Context Properties

Membership in a Context class determines the properties supported by the instantiated Context object. The Context's class provides the default values and allowable ranges for its properties. Examples of Context properties are the links it may have to other Contexts, its Termination connection/mixing parameters (e.g.-who hears whom or who sees whom), and its Termination multiplexing and inverse multiplexing support properties. It should be noted that Context classes cannot be created or modified using the commands of the H.GCP protocol. They are static concepts used to model the logical entities within the Media Gateway.

A Media Gateway may limit the number of Terminations that it supports in a given Context. For example, a limit of two might exist for very simple gateways. A limit of three might be common for gateways that support up to three-way calling but not multi-way calling of any larger proportions

7.2.2.3 Context Dynamics

The H.GCP protocol can be used to instantiate Context objects from their classes and can then modify the values of the default properties given to the Contexts during their creation. The protocol has commands to add Terminations to a Context, subtract Terminations from a Context, and move Terminations between Contexts. Note that a Context is created by the addition of its first Termination and is destroyed by the subtraction of its last remaining Termination. The H.GCP protocol also supports linking of Contexts together and unlinking of Contexts from one another.

<Editor last sentence pending resolution of H.320 lip sync issue >

7.2.2.4 Context Classes

<EDITOR: Issue 4, need to identify BaseContextClass, AudioContext, VideoContext, DataContext classes>

7.2.3 Terminations

7.2.3.1 Overview

Terminations are logical representations of physical devices, such as TDM channels, and ephemeral information flows, such as IP packet streams. Typically, Terminations representing physical entities will have a semi-permanent existence. For example, a Termination representing a TDM channel might exist for as long as it is provisioned in the gateway. Terminations representing ephemeral information flows would usually exist only for the duration of their use in a Context. All Terminations are instantiated members of their specific Termination class.

All Terminations are named entities having hierarchically defined names. They are given a unique name (i.e.-the last component of the hierarchical name will be unique) by the Media Gateway. Wildcarding may be used in the specification of a name within H.GCP protocol commands.

7.2.3.2 Termination Properties

Membership in a Termination class determines the properties supported by the instantiated Termination object. The Termination's class provides the default values and allowable ranges for its properties. Examples of Termination properties are the Termination's address, media parameters, security properties, the events that can be generated by the Termination, and signals that can be applied to it. It is possible for a Termination class to coalesce its properties via Termination class inheritance as well as through containment of generic groupings of events and signals called packages. It should be noted that Termination classes and generic packages cannot be created or modified using the commands of the H.GCP protocol. These are static concepts used to model the logical entities within the Media Gateway.

7.2.3.3 Termination Dynamics

The H.GCP protocol can be used to instantiate Termination objects and can then modify the values of the default properties given to the Terminations at the time of their instantiation. As Terminations are added to or moved between Contexts it may be necessary to specify state and media parameters for, events to be detected on, and signals to be applied to such Terminations within the Context in which they exist.

For Terminations representing physical entities it might also be desirable to modify their properties as they are subtracted from a Context. This may be the case if a configuration different to the one provided by the Termination class' defaults is needed. Such semi-permanent Terminations may also have their properties modified outside of their existence within a Context.

7.2.3.4 Termination Property Packages

The Media Gateway Controller may ask to be notified of certain events occurring on a Termination (e.g.-off hook, DTMF tone detected) and may also request certain signals to be applied to a Termination (e.g.-play dial tone, send DTMF tones). Events and signals are properties of Terminations and may be grouped into packages when such events and signals are related and apply to more than one Termination type. In this case it is possible to describe the properties of a given Termination class by listing its supported properties along with the event and signal packages that apply to it. The common packages supported by the H.GCP protocol will be enumerated below. It should be noted that signals are divided into different types depending upon their behavior:

1) On/off – the signal lasts until it is turned off
2) Timeout – the signal lasts until it is turned off or a specific period of time elapses
3) Brief – the signal duration is so short that it will stop on its own unless a new signal is applied which causes the signal to stop

<EDITOR: Issue 5>

<EDITOR: Issue 6>

< need to define termination packages and/or point to an annex >

7.3 Commands

7.3.1 General Usage

The H.GCP 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 within or outside of a Context, 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 are to be applied to a Termination.

Most Commands are for the specific use of the Media Gateway Controller in controlling Media Gateways. However, there are several Commands for the Media Gateway to use to report events that have occurred back to the controller.

7.3.2 Command Entity Names and Common Parameters

Naming of logical entities within Commands is important and will be described here. Also, many Commands share common parameters. This subsection will enumerate these common parameters. Parameters and parameter usage specific to a given Command type will be described in the subsection that describes the Command.

<EDITOR: Issues 4 and 6>

7.3.2.1 Naming Conventions within Commands

Names for Terminations are hierarchical. A separation character delimits the components of the name from one another. An example name of a Termination object representing a channelized T3 might be “com1/3/17”. This name refers to a T3 called “com1”. The “3” specifies the third DS1 in the T3, and the “17” specifies the seventeenth DS0 (timeslot) within the DS1.

Names in Commands may use wildcards. 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 DS1s of the T3 specified in the previous example can be referred to with “com1/all”. The “choose” construct allows a Command sender to specify that it would like the recipient to select and return a possible value of a name component. Using the previous example again, “com1/3/choose” could be used in a Command to request that a DS0 on the third DS1 of T3 named “com1” should be selected and returned.

7.3.2.2 Specifying Properties

Commands specify their parameters through the use of descriptors. Each type of descriptor supported by the H.GCP protocol identifies a set of properties that may be modified for the Context or Termination affected by the Command. The legal properties that may be modified through use of a descriptor are those that are legal for the Context or Termination class in question.

<EDITOR: Issue 7>

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

1) Fully specified properties - have a single, unambiguous value that the Media Gateway Controller is instructing the Media Gateway to use for the specified property.
2) Under-specified properties - have a list of potential values. The list order specifies the Media Gateway Controller’s order of preference of selection. The Media Gateway chooses one value from the offered list and returns that value to the Media Gateway Controller.
3) Unspecified properties - (i.e.-legal properties not specified in the descriptor) result in the Media Gateway retaining the previous value for that property.

<EDITOR: Issue 8>

7.3.2.3 TerminationState Properties

The TerminationState parameter groups a list of properties that define the state of the Termination, but which are not directly linked to the media flow, event or signal properties of the Termination. The properties associated with the TerminationState are:

1) TerminationMode – the mode of the Termination’s information flow with respect to the outside of the Media Gateway; values are
    a) sendOnly
    b) receiveOnly
    c) sendAndReceive
    d) inActive
    e) outOfService
    f) loopBack
    g) continuityTest

<EDITOR: Issue 9>

2) EventBufferProcessingMode – specifies whether buffered events should be processed or discarded
3) 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

<EDITOR: Issue 10>

7.3.2.4 TerminationDescriptor Properties

The LocalTerminationDescriptor and RemoteTerminationDescriptor parameters describe the processing of media flow to and from a Termination. The legal properties of each of these descriptors depends upon the type of Termination being addressed by the Command.

1) LocalTerminationDescriptor – provides media processing properties of a local Termination; 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 IP address, receive RTP UDP port, encoding method and packetization interval
2) RemoteTerminationDescriptor – provides media processing properties of a remote Termination that is associated with this local Termination; this only makes sense if the local Termination needs information about the other end of its media stream, such as in the case of an RTP stream needing to know the receive RTP UDP port of the far end

These descriptors are described in the form of an SDP descriptor, which contains properties that are part of the definition of the specific Termination class. These sets of properties can be enhanced by vendor-specific optional or mandatory extensions. Properties in these descriptors may be fully specified, under-specified, or (by omission) unspecified.

<EDITOR: Issue 8>

7.3.2.5 SignalsDescriptor Properties

A SignalsDescriptor is a parameter that contains the set of signals that the Media Gateway is asked to apply to a Termination (e.g.-apply ringing or play DTMF tones). The signals specified in a SignalsDescriptor must be legal properties of the Termination class being targeted 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 one applied

<EDITOR: Issue 11>

7.3.2.6 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 EventsDescriptors and SignalsDescriptors, and an optional Termination management action. The notification actions are:

1) Report the event immediately along with the accumulated list of observed events
2) Accumulate the event in an event buffer, but withhold notification
3) Accumulate the event according to a specified digit map
4) Treat the event according to a specific script

5) Ignore the event

The embedded SignalsDescriptor, if present, is used as a replacement for the current SignalsDescriptor. 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 event is detected. If no embedded SignalsDescriptor is specified, the production of signals continues as specified in the Command.

Implementations should be able to support at minimum one level of embedding of SignalsDescriptors and EventsDescriptors.

Only one Termination management action is defined, the Swap Termination action. It can be used when a Context contains more than one active Termination in addition to the Termination on which the triggering event is detected. The Swap Termination action assumes that the Terminations are assigned in order in the Context. The action modifies the TerminationMode parameter of the Terminations in the following fashion:

1) The TerminationMode of the Termination on which the event is detected remains unchanged.
2) The TerminationMode of the other currently active Termination is set to “inActive”.
3) The TerminationMode of the previously inactive Termination is set to “sendReceive”.

This action can be used to implement call waiting. The EventsDescriptor can map an event (typically a hook-flash) to a local Swap Termination function.

<EDITOR: Issue 12>

7.3.2.7 DigitMapDescriptor

The DigitMapDescriptor parameter contains a set of DigitMap names and values to be assigned. A DigitMap is a Media Gateway resident dialing plan for detecting and reporting of digit events received on a Termination. It is defined to have a name and a value. 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 H.GCP 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 interdigital 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 interdigital 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 interdigital timer value should be set to a medium duration (e.g.-8 seconds).

A “long interdigital” timer and a “short interdigital timer” are parameters associated with a DigitMap.

7.3.2.8 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.

7.3.3 Command Application Programming Interface

Following is an Application Programming Interface (API) describing the Commands of the H.GCP 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.

7.3.3.1 Add

The Add Command adds a Termination to a Context.

```
[TerminationID,]
[LocalTerminationDescriptor,]
[RemoteTerminationDescriptor]
←        Add(TerminationID,
              [TerminationState,]
              [LocalTerminationDescriptor,]
              [RemoteTerminationDescriptor,]
              [EventsDescriptor,]
              [SignalsDescriptor,]
              [DigitMapDescriptor])
```

TerminationID in the input parameters represents the Termination that is being added to the Context. If it is under-specified, then the return TerminationID will give the value selected by the Media Gateway for this Termination.

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

The LocalTerminationDescriptor parameter is optional. If it is omitted, default LocalTerminationDescriptor properties will be assumed from the Termination class. If it is present and under-specified, then the return LocalTerminationDescriptor will give the fully specified values returned by the Media Gateway.

The RemoteTerminationDescriptor parameter is optional. If it is not applicable to the Termination class or if it is applicable to the Termination class but not yet known, it may be omitted and provided via later use of the Modify Command. If it is present and some properties are under-specified, then the return RemoteTerminationDescriptor will give the fully specified values for these properties returned by the Media Gateway. In this case, unspecified properties will receive the default values assumed from the Termination class. Note that if the descriptor is applicable to the Termination class but omitted, the TerminationState parameter specifies the behavior for the media stream received from the remote end. The only TerminationState values that make sense in this case are receiveOnly, inActive, loopBack, and continuityTest. In the receiveOnly case the Media Gateway would be expected to process received media. In the other valid cases it would ignore them. All invalid specifications of TerminationState should cause the Add command to be responded to with and error return.

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 DigitMapDescriptor parameter is optional. If present, it describes the name and the value of a digit map in the Context of the Termination.

7.3.3.2 Modify

The Modify Command modifies the properties of a Termination.

```
[LocalTerminationDescriptor,]
[RemoteTerminationDescriptor]
←       Modify(TerminationID,
               [TerminationState,]
               [LocalTerminationDescriptor,]
               [RemoteTerminationDescriptor,]
               [EventsDescriptor,]
               [SignalsDescriptor,]
               [DigitMapDescriptor])
```

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.

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

Media Gateway. If present, only the LocalTerminationDescriptor properties specified will be altered.

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

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 DigitMapDescriptor parameter is optional. If present, it describes the name and the value of a digit map in the Context of the Termination.

Modify may be used for Terminations not associated with a Context. In this case it changes only the specified properties of the specified Termination. If the TerminationID is wildcarded, then the changes indicated in the Modify Command apply to all Terminations whose names are matched by the wildcarding specification.

7.3.3.3 Subtract

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

```
Statistics |
*[TerminationID,
  Statistics]
←        Subtract(TerminationID,
                   [TerminationState,]
                   [EventsDescriptor,]
                   [SignalsDescriptor,]
                   [DigitMapDescriptor])
```

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

The LocalTerminationDescriptor and RemoteTerminationDescriptor of a semi-permanent Termination are reset to their default values.

When applied to a semi-permanent Termination, the Subtract Comand may optionally include the TerminationState, EventsDescriptor, SignalsDescriptor and DigitMapDescriptor parameters. In this case the parameters are processed as specified in the Modify Command.

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.

7.3.3.4 Move

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

```
[LocalTerminationDescriptor,]
[RemoteTerminationDescriptor]
←       Move(TerminationID,
               [TerminationState,]
               [LocalTerminationDescriptor,]
               [RemoteTerminationDescriptor,]
               [EventsDescriptor,]
               [SignalsDescriptor,]
               [DigitMapDescriptor])
```

The Audit Command returns properties associated with Terminations.

```
[TerminationID,]
[TerminationState,]
[LocalTerminationDescriptor,]
[RemoteTerminationDescriptor,]
[EventsDescriptor,]
[SignalsDescriptor,]
[DigitMapDescriptor,]
[Capabilities,]
[Statistics]
←       Audit(TerminationID,
               RequestedInfo)
```

The TerminationID in the input parameters represents the Termination that is being audited. The audit Command shall be applied either to the “null” Context or to the specific Context of the requested Termination(s). 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) LocalTerminationDescriptor – current LocalTerminationDescriptor values
3) RemoteTerminationDescriptor – current RemoteTerminationDescriptor values (if applicable)
4) EventsDescriptor – current EventDescriptor values
5) SignalsDescriptor – current SignalsDescriptor values
6) DigitMapDescriptor – current DigitMapDescriptor values
7) Statistics – current Statistics (i.e.-mid call values)
8) Capabilities – properties that the Media Gateway is prepared to support for that Termination

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 Conext(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

7.3.3.6 Notify

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

```
        Notify(TerminationID,
               ObservedEvents)
```

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.

7.3.3.7 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 abruptly out of service and any established connections associated with them were lost

<EDITOR: Issue 13>

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.

<EDITOR: Issue 14>

7.3.3.9 Response

<EDITOR: Issue 15>

7.3.4 Generic Command Syntax

7.3.5 Command Error Codes

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

7.3.6 Command Encoding

7.4 Transactions

7.4.1 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.

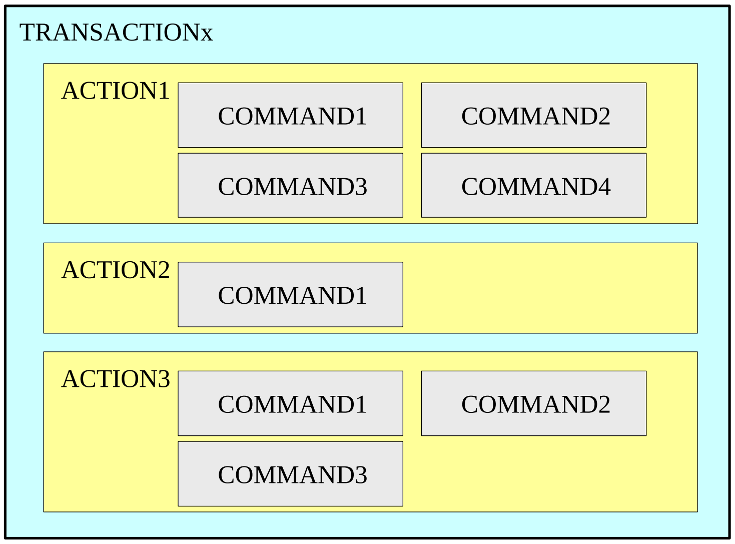


Figure 9: Transactions, Actions and Commands

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

7.4.2 Common Parameters

7.4.2.1 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.

7.4.2.2 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 H.GCP 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.

7.4.3 Transaction Application Programming Interface

Following is an Application Programming Interface (API) describing the Transactions of the H.GCP 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.

7.4.3.1 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".

7.4.3.2 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".

7.4.3.3 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.

<EDITOR: Issue 16>

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.

7.4.4 Transaction Syntax

7.4.5 Transaction Encoding

7.5 Example Use Cases

7.5.1 Residential Gateway to Residential Gateway Call

This example scenario illustrates the use of the elements of the H.GCP 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.

7.5.1.1 Programming Residential GW Analog Line Terminations for Idle Behavior

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.

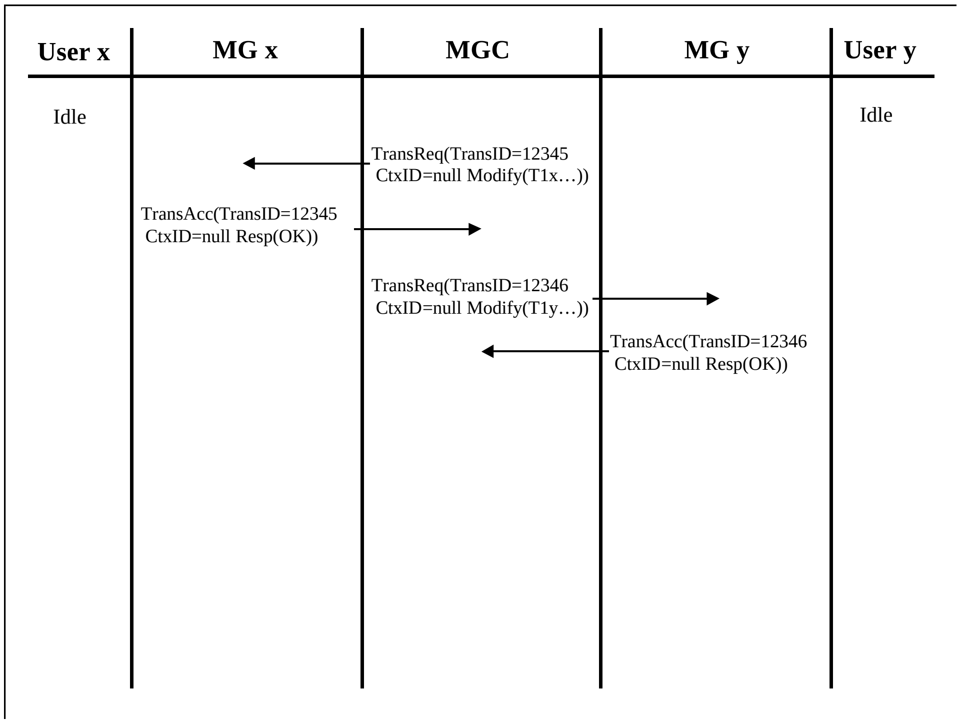

User x
MG x
MGC
MG y
User y
Idle
Idle
TransReq(TransID=12345 CtxID=null Modify(T1x…))
TransAcc(TransID=12345 CtxID=null Resp(OK))
TransReq(TransID=12346 CtxID=null Modify(T1y…))
TransAcc(TransID=12346 CtxID=null Resp(OK))

Figure 10: Programming Idle AnalogLine Terminations

The connection model that follows illustrates the stable idle states of the AnalogLine Terminations. In both of the Residential Media Gateways the respective Terminations are in existence outside of any Context (i.e.-they are semi-permanent in nature).

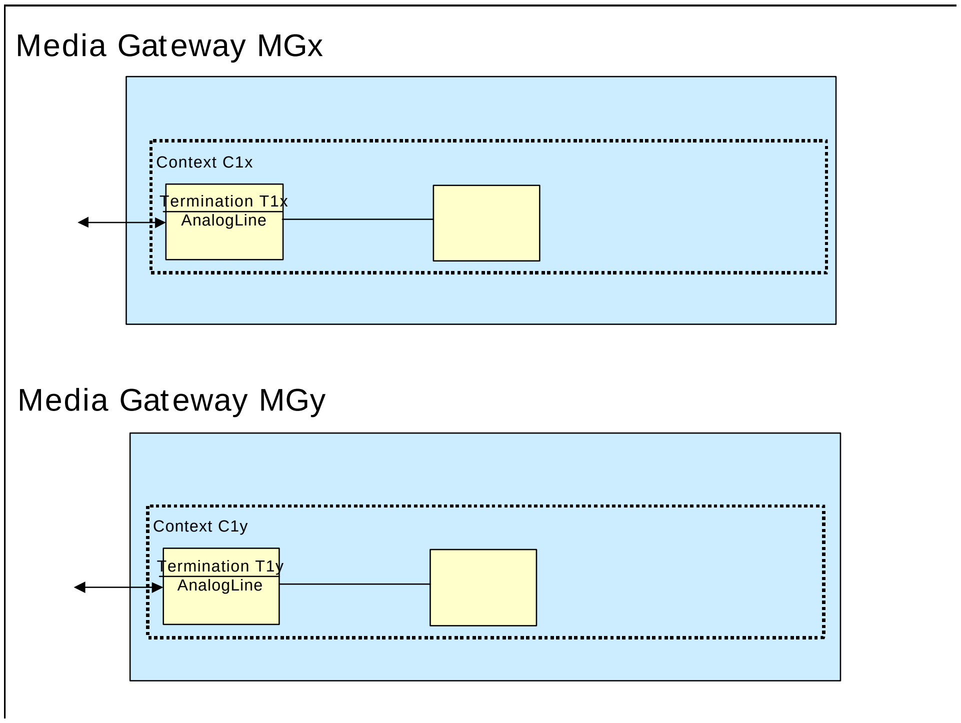


Figure 11: Idle AnalogLine Terminations

7.5.1.2 Collecting Originator Digits and Initiating Termination

The following builds upon the previously shown stable state conditions. It illustrates the API invocations from the Media Gateway Controller and originating Media Gateway (MGx) to get the originating Termination (T1x) through the stages of digit collection required to initiate a connection to the terminator's Media Gateway (MGy).

First, MGx detects an offhook event from User x and reports it to the Media Gateway Controller via the Notify Command. The Media Gateway Controller responds and uses an Action on the null Context to send a Modify Command to MGx to get T1x programmed with the appropriate DigitMap and to get dial tone applied.

Next, digits are accumulated by MGx as they are dialed by User x. When an appropriate match is made of collected digits against the currently programmed DigitMap for T1x, another Notify is sent to the Media Gateway Controller. The controller then analyzes the digits and determines that a connection needs to be made from MGx to MGy.

The controller then uses an Action on the unspecified Context to send two Add Commands to MGx. The first adds T1x and the second adds an RTP Stream Termination, expecting MGx to return a TerminationID for it. A new ContextID is returned by MGx (C1x) to contain these two Terminations and a TerminationID (T2x) is returned for the RTP Stream Termination. At this point MGx has associated T1x and T2x, and has established an RTP Stream (i.e.-T2x) receiveOnly connection through to the originating user, User x.

<EDITOR: Issue 17>

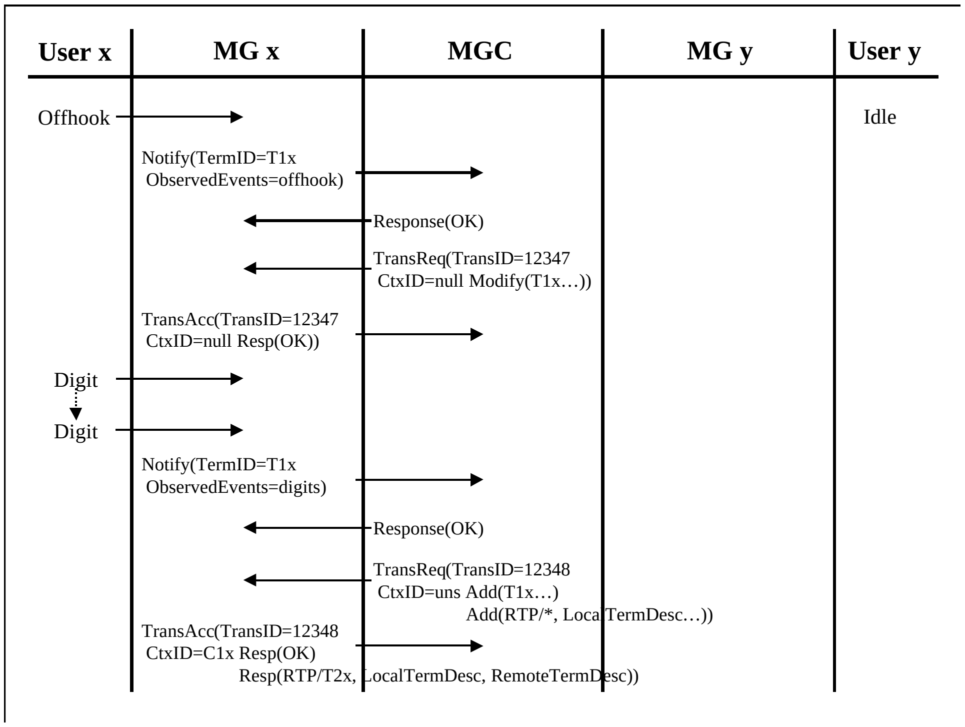

User x
MG x
MGC
MG y
User y
Offhook
Idle
Notify(TermID=T1x ObservedEvents=offhook)
Response(OK)
TransReq(TransID=12347 CtxID=null Modify(T1x…))
TransAcc(TransID=12347 CtxID=null Resp(OK))
Digit
Digit
Notify(TermID=T1x ObservedEvents=digits)
Response(OK)
TransReq(TransID=12348 CtxID=uns Add(T1x…) Add(RTP/*, LocalTermDesc…))
TransAcc(TransID=12348 CtxID=C1x Resp(OK) Resp(RTP/T2x, LocalTermDesc, RemoteTermDesc))

Figure 12: Collecting Digits and Initiating Connection

The connection model that follows illustrates the condition that the two Media Gateways are in before the Media Gateway Controller continues by establishing a connection to the terminator's

Media Gateway (MGy). If there were more than one Media Gateway Controller, this is the point in the call where signaling between controllers would be occurring.

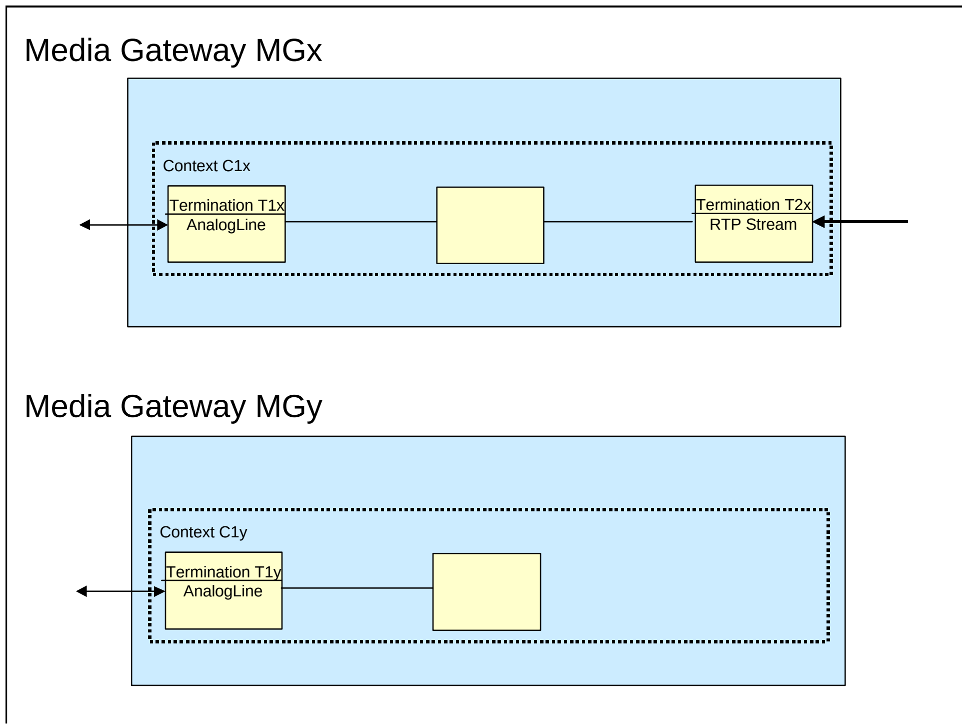


Figure 13: Digit Collection Complete/Connection Initiated

7.5.1.3 Connecting to Terminator and Alerting

The following continues from the previously shown examples. It illustrates the API invocations from the Media Gateway Controller, MGx and MGy to get the originating Termination (T1x) connected in receiveOnly mode to terminating Termination (T1y), and to get alerting applied to T1y and ringback tone applied to T1x.

The controller continues its work by using an Action on the unspecified Context to send two Add Commands to MGy. The first adds T1y and the second adds an RTP Stream Termination, expecting Mgy to return a TerminationID for it. A new ContextID is returned by MGy (C1y) to contain these two Terminations and a TerminationID (T2y) is returned for the RTP Stream Termination. At this point MGy has associated T1y and T2y, and has established an RTP stream sendAndReceive connection to the terminating user, User y.

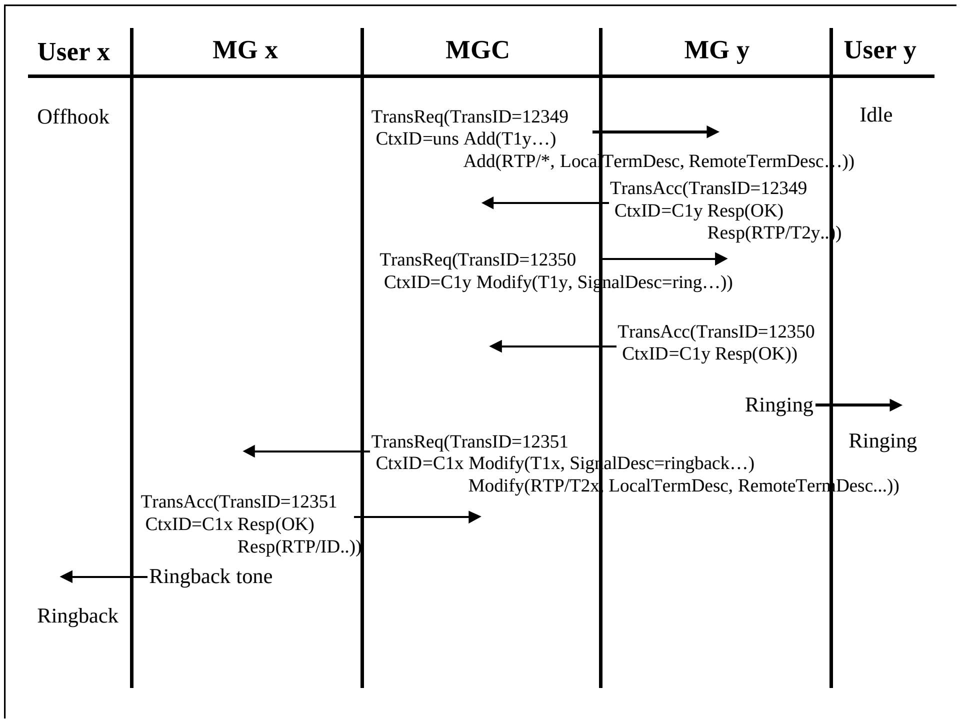


Figure 14: Connecting Terminator and Alerting

The connection model that follows illustrates the condition that the two Media Gateways are in after alerting has been applied to the terminator and ringback tone given to the originator.

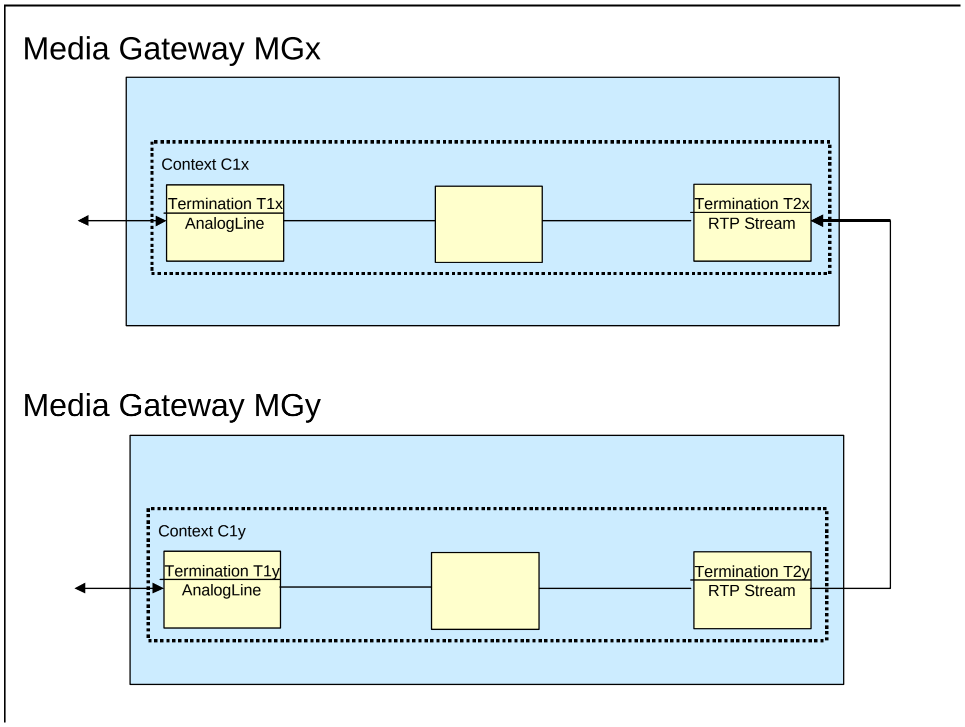


Figure 15: Connection Complete/Alerting Initiated

7.5.1.4 Processing Answer and Connecting Two Way Talk Path

The following finishes the use case by showing the effect of answer by the terminating user, User y. It illustrates the API invocations from the Media Gateway Controller, MGx and MGy to get the originating Termination (T1x) connected in sendAndReceive mode to terminating Termination (T1y).

User y goes offhook, which results in a Notify Command being sent to the Media Gateway Controller from MGy. The controller responds and uses an Action on the C1x Context to send two Modify Commands to MGx to terminate ringback tone and cut the path through in both directions.

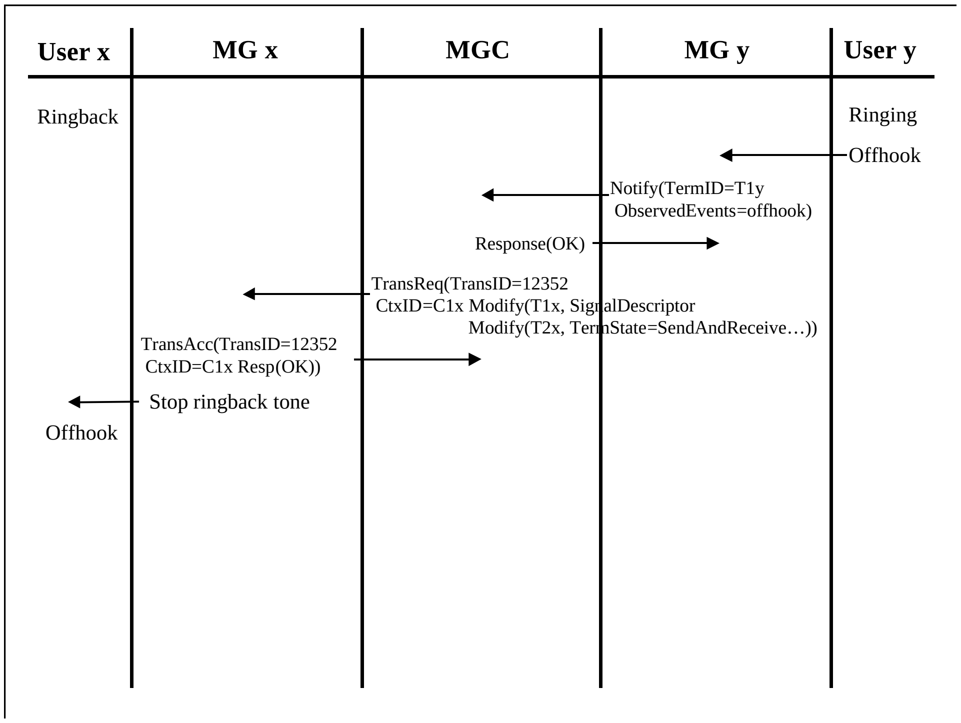

User x
MG x
MGC
MG y
User y
Ringback
Ringing
Offhook
Notify(TermID=T1y ObservedEvents=offhook)
Response(OK)
TransReq(TransID=12352 CtxID=C1x Modify(T1x, SignalDescriptor Modify(T2x, TermState=SendAndReceive…))
TransAcc(TransID=12352 CtxID=C1x Resp(OK))
Stop ringback tone
Offhook

Figure 16: Processing Terminator Answer

The connection model that follows illustrates the condition that the two Media Gateways are in after the two way connection has been established as a result of answer by the terminator, User y.

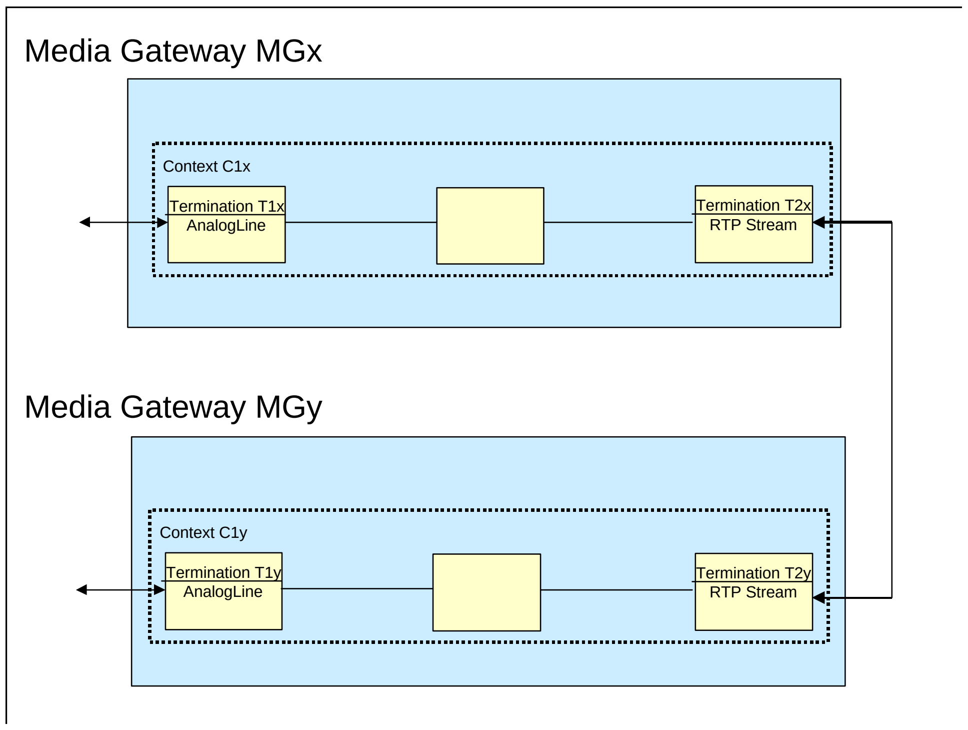


Figure 17: Call Answered/Two Way Talk

7.6 Transport

The transport mechanism for MGC/MG communication has not been chosen. It is likely that TCP will be an option. If the SIGTRAN transport mechanism is suitable for H.GCP, that may be specified. It may be necessary to specify a transport protocol in this specification. Either the SIGTRAN mechanism or an H.GCP specified mechanism will be optional on the MG and required on the MGC.

7.6.1 Transport capabilities, and relationship to Transport Layer

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 which 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 H.GCP protocol layer.

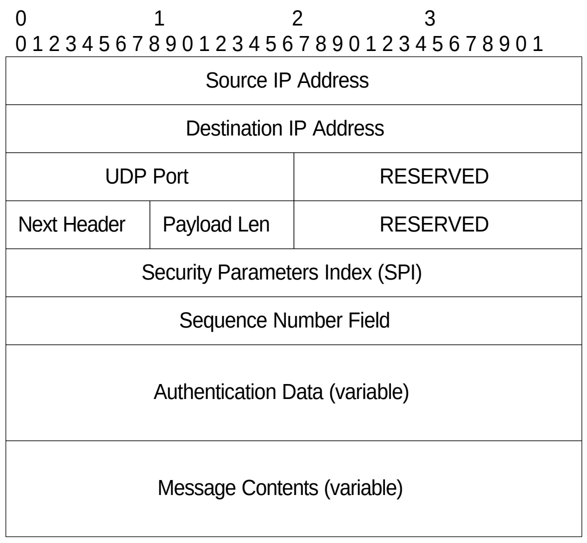
H.GCP gateway control	Termination signaling
relaible transport	
UDP	
IP	
Ethernet, ATM, Sonet. . . .	

7.7 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 H.GCP protocol implementations where the underlying network implementation does not support IPSEC. As an interim solution, the 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-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-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-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-H.GCP scheme.

When employing the AH header, either in IPSEC or AH-within-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. H.GCP implementations SHOULD implement IKE [RFC2409] to permit more robust keying options. H.GCP implementations employing IKE SHOULD implement RSA signatures and authentication with RSA public key encryption. 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-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.

7.8 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.

ANNEX A: Open Issues

8. OPEN ISSUES

1. Physical Termination Stream Multiplexing
   Physical endpoints may multiplex more than one media stream at a given time. For example, in H.320 gateways a given B-channel can contain audio, video and signaling information. Using the rule that a Context contains a single media type, this implies that a physical endpoint may be referenced simultaneously by Terminations in separate Contexts. This then implies that physical endpoints need to be a separate concept from Terminations or that Terminations need to be allowed in more than one Context.

2. Physical Termination Stream Inverse Multiplexing
   A given media stream may span multiple physical endpoints. For example, an H.320 call may split a video stream between multiple B-channels (there are other good examples in the H.221 realm too). A Context needs a way to deal with such inverse multiplexing and this could possibly be specified as a Context property.

3. Context Connection Properties
   Contexts may need to specify the type of connections that need to be supported between Terminations. For example, an operator break-in situation where Termination1 can hear Termination3 (the operator), but Termination2 cannot hear Termination3.

4. Context Classes and Properties Enumerated
   The valid Context classes and properties must be enumerated. The Command parameters probably also need to add the ContextDescriptor to support this.

5. Brief Signal Interruption
   There is an open point of debate as to whether Brief signals should be allowed to be interrupted. The example is whether you would want to interrupt a DTMF tone.

6. Termination Classes, Properties and Packages Enumerated
   The valid Termination classes, properties and property packages must be enumerated. Similarly, the Command parameters need to be synchronized with these properties.

7. Specification of Illegal Properties
   We need to decide how to deal with Command specification of properties not legal for a given Context or Termination class. What is the response given? Error? Ignore?

8. Under-Specification of Mutually Exclusive Property Sets
   For sets of under-specified properties a descriptor may need to indicate (through the syntax of the protocol) mutually exclusive sets of options. For example, the Media Gateway Controller may specify the use of certain combinations of video codecs with audio codecs and expect the Media Gateway to choose from a set of mutually exlusive options. Support for this gets interesting if we stick with the idea of one media type per Context.

   Also, using SDP supposes resolution of a few questions. We have to make sure that we can associate a unique SDP profile for each Termination class. We also have to register additional properties to make sure that we can support all the current variations of H.245. The simultaneous and mutually exclusive capabilities in H.245 have no way of being described in SDP, as it is linear in nature. Support of H.245 is mandatory.

9. Testing Support
   There is debate over whether the protocol can or should specify all forms of testing in the TerminationMode. If it does, then a more exhaustive list must be provided. If not, then it may be better to define only one mode, "test", and then define events and signals for specific tests. Another idea is to provide specific Termination classes to provide testing functionality. In this case a Context would contain the Termination being tested and another Termination with the test functionality.

10. TerminationState Needed
    The properties in the TerminationState parameter seem like a conglomeration of unrelated things that could be better organized. TerminationMode and Event buffering aren't related. Also, TerminationMode IS media-flow related information. Shouldn't TerminationMode be part of the LocalMediaDescriptor and the Event buffering stuff be part of the EventsDescriptor?

11. SignalDescriptor Processing in Media Gateway
    Text from MEGACO (hard to figure out): pertaining to receipt of a signal to apply to a Termination…

    If a signal has already been attached to a Termination, the previous signal is removed and the specified one is attached. No events which would have occurred on the previous signal will be generated subsequent to a command that modifies the signal descriptor, although the MGC may not have received an event from the previous signal prior to sending the command, and in fact the MG may have detected such an event, but may not have notified the MGC of it when it receives the new command. The behavior of the MG in such a circumstance is not defined. It may send the notification , or it may flush it.

    <EDITOR: We should think this through. What if you expected an event after a brief signal, but never get it? Is that always OK? Should we define behavior (notify or flush?)? Note that the MGC always has to handle either case. Possibly we define that a notify is always sent, but with a "superceded" qualifier to let the MGC know that it was replaced. Also, what about the case of Terminations that can handle playback of more than one signal at a time? >

12. Embedded Events versus Scripting Support
    Embedded events and the Swap Termination Termination management action are defined in lieu of a more robust scripting mechanism. If and when such a mechanism is defined, this mechanism may be deprecated or removed.

13. Statistics on Terminations Taken Out of Service
    We need to decide whether the MGC must Subtract Terminations being taken out of service from their Contexts to gather statistics, the ServiceChange Command automatically sends statistics, or the Audit Command must be used.

14. Reporting Terminations That Stay Out of Service
    What if the Termination(s) have just gone down and are still out of service? Don't we need to indicate that via ServiceChange? Probably then the Media Gateway would issue a new ServiceChange when they are back in service.

15. Responses API Added
    We need to add the API for Responses to the Commands section.

16. TransactionReject ContextID Specification

Should we allow TransactionReject ContextID specifications to be un-specified or should we require the Media Gateway to formulate a ContextID for Actions that have an error response.

17. Returning RTP Port in Add Command
Is the RTP port returned from the Media Gateway part of the TerminationID parameter or the LocalTerminationDescriptor?