Internet Engineering Task Force P. Sijben
Internet Draft Mike Buckley
Tom Fritz
John Wachter
John Seegers
Chia Li
Lucent Technologies
Expires in Six Months November 1998

Toward the PSTN/Internet Inter-Networking MEDIA DEVICE CONTROL PROTOCOL

<draft-sijben-megaco-mdcp-00.txt>

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

Status of this Memo

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

1. Abstract

This draft defines a protocol for use between media processors (commonly known as media gateways) and their controllers. The scope includes voice, video and data applications. A model is introduced to describe the functionality of the media gateway and operations on that model are described. Profiles are described that allow efficient operation.

2. Scope

This draft defines a protocol for use between media processors (commonly known as media gateways) and their controllers. The scope includes voice, video and data applications.

2.1 Media Gateway Applications

The scope of media gateways in this Recommendation includes the following applications:

- **Trunking gateways** that interface between SCN networks and Voice over IP network. Such gateways typically interface to SS7 signalling on the SCN and manage a large number of digital circuits.
- **Voice over ATM gateways** which operate much the same way as voice over IP trunking gateways, except that they interface to an ATM network.
- **Access gateways** that interface ISDN (PRI and BRI) and traditional analogue voice terminal interfaces to a Voice over IP network.
- **Residential gateways** are access gateways for a small number of voice terminals that can be colocated with a cable modem or set top box.
- **Network Access Servers**, that convert modem signals from an SCN network and provide data access to the Internet.
- **Multipoint Processors** that provide multipoint connectivity between terminals on SCN and packet networks.
- **Interactive Voice Response systems** that provide automatic voice response and switching services in response to DTMF signals from the SCN.

2.2 Interfaces

The arrangement of functional elements within an H.323 SCN to IP Gateway and the grouping of these into Media Gateway (MG) and Media Gateway Controller (MGC) elements is shown in Figure 1.

Interface A is the primary control interface between the MG and MGC. However, in some applications it may be desirable to do some of the signalling from the MG rather than from the MGC. Examples are SS7 and Q.931 backhaul and H.245 backhaul as well as autonomous H.245 negotiation. In these circumstances additional interfaces B and C exist between the MG and MGC.

The media device control protocol provides a common signalling and control framework for Interfaces A, B and C.

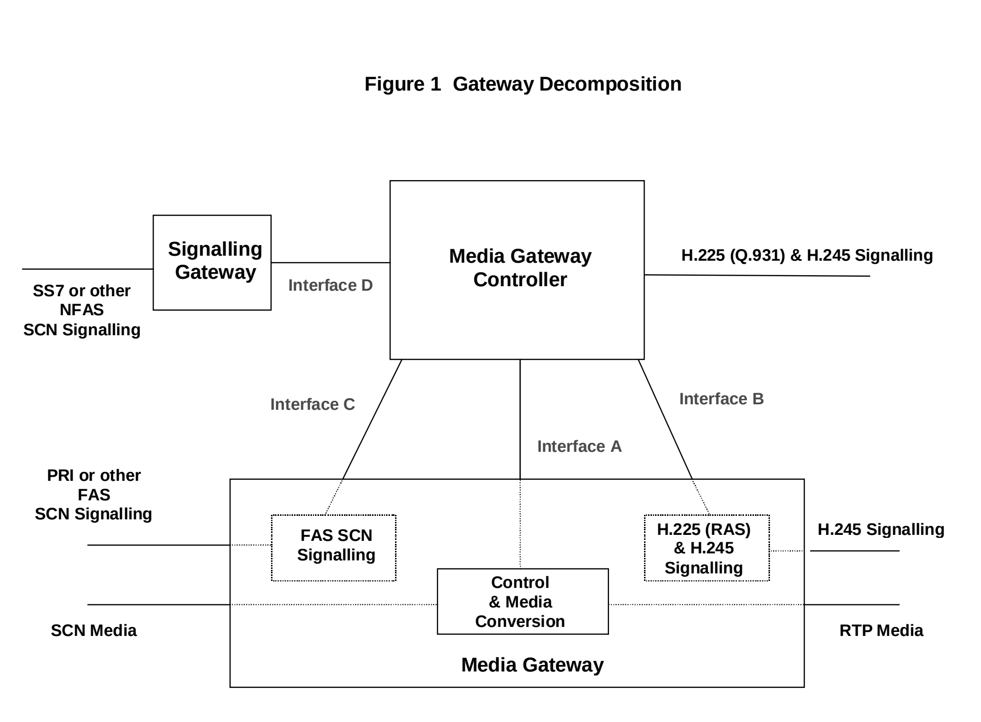


Figure 1. Gateway decomposition

2.3 Architecture

The generic architecture is given in Figure 2. One media gateway is controlled by one (primary) controller but backup (secondary) controllers may be standing by.

The controller maintains the call state and maps media streams and required

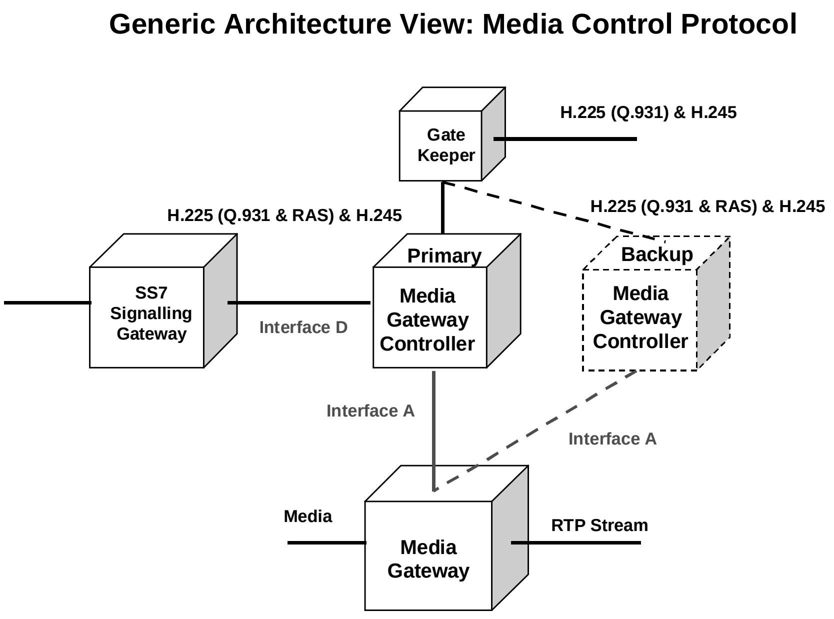


Figure 2 Generic architecture

operations on the streams to one or more media gateways to perform the actual processing. More architecture examples are given in Annex 2.

2.4 Approach

An object oriented control model is used to represent the elements within the MG and the way in which these are combined together across a wide range of applications and gateway types. The protocol performs operations on this model

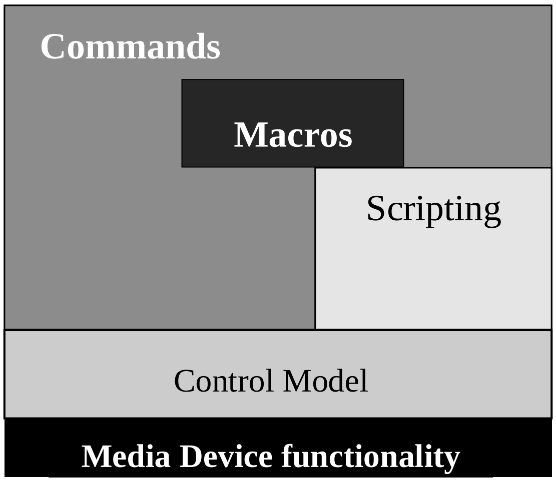


Figure 3. Structure

Figure 3 gives an overview of the approach that is the basis of this protocol. The processing functionality (comprised of DSPs and generic CPUs) of the MG is represented within the control model as abstract resources. The model is shared between the MG and MGC and Control Functions operate on the model to achieve the desired degree of control.

Beyond the basic control function operations scripting can be used to relegate certain common operations to the MG and allow Macros to present a simpler control interface in these circumstances.. (A simple implementation of the protocol can be achieved by the use of appropriate macros only, see section 8).

Section 4 describes the control model upon which the protocol is based. Section 5 describes the abstract resources which make up the media gateway.
Section 6 describes the format of the messages exchanged between the MG and MGC.
Section 7 describes the control functions which make up the messages via which control of the MG takes place..
Section 8 presents profiles created for specialized applications.

Section 9 describes the binary scheme used to encode the information content of the messages.
Section 10 describes the transactions and some call flows (more call flows are described in Annex 3)
Annex 1 gives an overview of the requirements upon which the protocols based

3. Definitions

Signalling Gateway (SG): A functional entity connected to the SCN that terminates SS7 or other signalling links. In general, the Signalling Gateway function maintains only enough information about the call state to manage the protocol interfaces. After processing the incoming SCN signalling data, the Signalling Gateway will deliver this information to the appropriate Media Gateway Controller. The gateway could pass the signalling information to several separate Media Gateway Controllers to make economic use of the powerful SS7 interface.

Media Gateway Controller (MGC): Controls the overall call state for a number of separate voice or other multimedia channels. It maps SCN signalling and call control information into the packet network call state and control information. This device includes an H.323 Signalling Interface for communicating with other H.323 signalling elements (i.e, Gatekeepers, MCUs). The Controller contains all connection control logic.

Media Gateway (MG): A device that provides the media mapping and/or transcoding functions. It maps or transcodes the media stream into the Packet Network that could be ATM or IP.

Gatekeeper (GK): This function performs authentication, authorization and call routing for the packet network.

Session: A Session is a generic term for a call or conference etc. Thus a phone call is a Session and a conference call similarly is one Session. The Session (call) state is maintained in the MGC. Multiple media gateways may be involved in one Session. Each media stream is assocated with a Session.

Connection: a tree of Resources and Inter-Connections within a MG which operate *unidirectionally* on one media type and which connect one or more Edge Points (e.g. UDP ports) to other Edge Points (e.g. trunks or DS0's) or End Processors (e.g. DTMF detectors). Multiple Connections make up a Session.

Resource: an objects that models functionality in the MG.

Edge Point: an internal representation of an external connection from the MG to the outside world. They are modelled as Resource objects with a number of characterizing Properties.

End Processor: A Resource object which terminates a media stream within the MG. They may be a source or sink to the media stream. (e.g. DTMF detectors, message players, etc)

Pipe Processor: A Resource object within the MG through which media streams pass and are processed. (e.g. a transcoder, an echo-canceller).

Meta-Resource: A Resource object that transcends a single Session and is media stream independent.

Inter-Connection: An object in the MG that connects two (or more) Resources.

Message: the unit of communication between the MG and MGC.

Control Function: a command, indication or response between the MG and MGC and contained within a message.

4. Control Model

The Media Gateway is considered to be made up of a number of Resources that must be connected together and controlled to make up media connections through the gateway. The resources are modeled as objects that have certain properties and are connected together to achieve the desired functionality via Inter-Connection objects. Trees of Resource objects and Inter-Connection objects make up a media Connection. The media Connections are unidirectional so it is possible to support different capabilities in each direction, e.g. encoding and encryption keys. Connections are combined into Sessions.

4.1 Sessions

A Session is a generic term for a call or conference etc. Thus a phone call is a Session and a conference call similarly is one Session. The Session (call) state is maintained in the MGC. Multiple media gateways may be involved in one Session. A Session may encompass media streams of different types (audio, video and data) In which case each media stream is associated with the same Session.

This Session may encompass multiple MGs as multiple MGs may be involved in a Connection.

Each session is identified by a *SessionID*. When an MGC controls resources on multiple MGs which form part of a single Session the *SessionID* will be the same for all MGs. [1]

A Session object contains the *SessionID* and a list of its Connections.

4.2 Connections

A Connection is a tree of Resources and Inter-Connections which operate unidirectionally on one media type and which connects one or more Edge Points (e.g. UDP ports) to other Edge Points (e.g trunks or DS0's) or End Processors (e.g. DTMF detectors).

Multiple connections make up a session. For example:

- a point to point voice Session is be made up of two unidirectional voice Connections,
- a multicast Session is made up of a number of unidirectional Connections that connect to multicast streams (e.g. multicast IP addresses or multicast ATM VCs),
- a multipoint conference Session is made up of a number of unidirectional Connections,
- a point to point multimedia Session is made up a series of voice, video and data Connections, etc.

Each Connection is identified by a *ConnectionID* which is unique within a session.

A Connection object contains a reference to the source of the connection.

4.3 Resources

Resources are objects that model functionality in the media gateway.

There are three types of Resources associated with media channels;

- Edge Points,
- End Processors, and.
- Pipe Processors.

[1] Even though Sessions (calls) that involve multiple controller are for further study, a Session that is managed by multiple MGCs has the same *SessionID* in each MGC.

Another type of Resource, Meta-Resources, are generally independent of media streams and are Session independent. Examples of the first are echo cancellers and DS0 interfaces; examples of Meta-Resources are the channel associated SS7 interface and control connections (Interfaces B and C in Figure 1 respectively).

Resources are identified by a unique *ResourceID* which also conveys the Resource Type.

A listing of Resource types and permitted operations on the Resources is given in Section 5.

4.3.1 Edge Points

Edge Points are internal representations of external connections from the MG to the outside world. They are modelled as Resource objects with a number of characterizing Properties.

Edge Points are the only mandatory supported class of Resources in the model , all other Resources are optional.

4.3.2 Pipe Processors

Pipe Processors are Resource objects with an input and an output which process a media stream (for example, transcoders, echo cancellers).

4.3.3 End Processors

End Processors are resource objects which terminate a media stream within the MG. They may be a source or sink to the media stream. (for example DTMF detectors, message players, etc).

4.3.4 Meta Resources

Meta Resources are Resources that transcend a single Session and are media stream independent. The primary Meta-Resources (which must always be supported) Control Interface A. (Interfaces B and C in Figure 1 are also Meta-Resources). Scripting Modules are also Meta-Resources. Section 5.5 describes this in more detail.

4.4 Events

Resources may generate Events. For example, an Edge Point may generate an event in response to a remote connection being lost or the QoS value on a RTCP connection falling below a minimum threshold, or a DTMF detector may generate an event in response to a DTMF tone detected.

The events defined for each type of resource are for further study.

4.5 Inter-Connections

Inter-Connections are objects which provide connections between resources. The use of multiple ResourceIDs in an Inter-Connection allows media streams to be branched within the MG. e.g. a DTMF detector may be connected in parallel with a voice transcoder or a legal intercept connection established (see Figure 4).

Each Inter-Connection is identified by an *InterConnectionID*.

A listing permitted operations on Inter-Connections is given in Section 5.2.

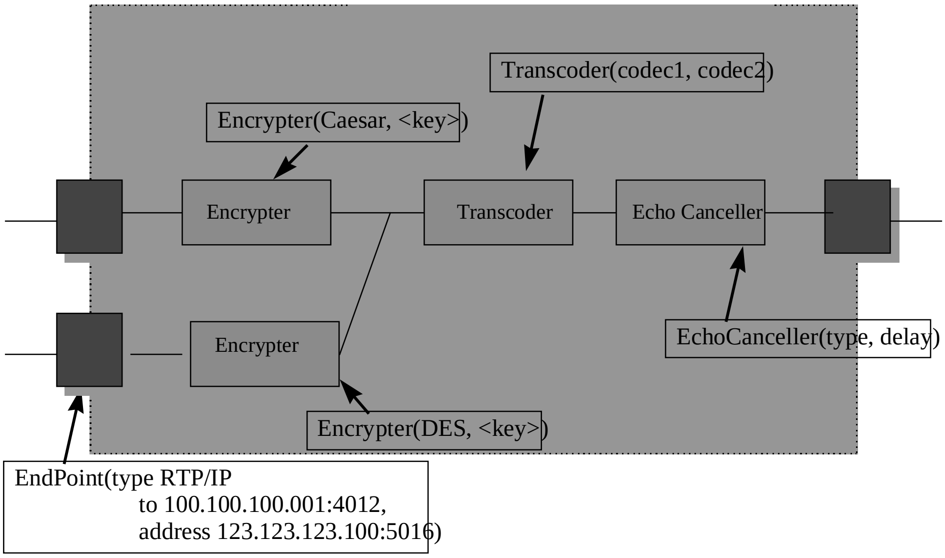


Figure 4 Example of Interconnection

5. Resources

Resources are divided in Modules. Except for the edge point module, all modules are optional. Some resources within a Module may be optional to that Module.

Resources have the following base properties:

- *ModuleID*
- *ResourceTypeID*, (unique within a module)
- *TypeCode*,
- *SubtypeCode*.

Additional parameters are defined for individual Modules. These parameters are always optional and may be specified by the controller or left in default values or filled in implicitly by the MG. Specification for default values at startup is for further study.

First the operations on the resources are defined next the properties of the resources are defined. The properties mentioned here are a first draft, the exact definition is for further study.

5.1 Operations on the resources

The major objects in the model are Resources and Interconnections. The model is modified by operations on these objects.

Five methods (operations) are defined on resources. Depending on the type of the resource some may not be appropriate:

Method	Description	Parameters
New ()	Creates a new resource object.: This method will return the *ResourceID* of the newly created	ResourceType, List of Parameter and value pairs

	resource instance or a suitable error message (for further study).	
Modify ()	Modifies the parameters of a resource.	List of Parameter and value pairs
Query ()	Queries the parameters of a resource	List of Parameter numbers
Delete ()	Deletes a resource	*none*
ReConnect ()	Connects the resource to a new or extra inter-connect	List of Interconnects
UnConnect()	To remove the resource from a connection.	List of InterConnects

5.2 Operations on interconnections

Since interconnections are a kind of resource they have similar methods:

Operation	Description	Parameters
New()	To create a new Interconnection. This method will return the *InterconnectionID* of the newly created resource instance or a suitable error message (for further study).	List of ResourceIDs to connect to
Query()	To query the resources it connects to. This method will return a list of ResourceIDs.	*none*
ReConnect()	To add resources to this Interconnection.	List of ResourceIDs to connect to
UnConnect()	To remove inter connections.	List of ResourceIDs
Delete()	To delete an Interconnect	*none*

5.3 Wildcarding

By not specifying parameters in a New() or Modify() method, the parameters may be wildcarded, this allows the MG to make sensible choices on its own. In this way a resource may be created with no or very little parameters and the details implied by the rest of the connection and are taken care of by the MG

5.4 Edge Resource Module

Edge resources are mandatory in the MG. The types that are supported naturally depend on the available interfaces.

5.4.1 Base types

The Edge Point types and their characterizing properties are defined in Table 1:

Table 1 Edge Point Types and Base Properties

Edge Point Type	Base Properties
RTP/UDP/IP	direction (in/out), own IP address & port; remote IP address & port, jitter buffer size, packetization, media payload type, media frames per packet. silence suppression
Trunk Line	direction (in/out), module, frame, carrier, slot, circuit.
ATM VC	direction (in/out), address & VC number.
Modem	own IP address, protocol (PPP/SLIP) connect speed

5.4.2 IP specific options

Additionally, IP based Edge Points may have the ability to request QoS using RSVP, in which case the additional properties in Table 2 will apply:

Table 2 Additional Properties for IP Edge Point Types

Edge Point Type	Properties
RTP/UDP/IP	RTCP port qosMode (guaranteedQOS or controlledLoad) tokenRate (rate in bytes/sec) bucketSize (size in bytes) peakRate (peak bandwidth bytes/sec) minPoliced (maxPktSize size in bytes)

5.4.3 ATM specific options

ATM based Edge Points may have the following additional QoS related properties:

Table 2 Additional Properties for ATM Edge Point Types

Edge Point Type	Properties
ATM VC	maxNTUSize (units in octets) Atm mode: atmUBR; atmrtVBR; atmnrtVBR; atmABR or atmCBR

5.4.4 Modem specific options

For further study.

5.4.5 Events

All Edge Points have the following events:

Name	Meaning
Connection Lost	

5.4.5.1 IP based events

For RTCP capable Edge points the “QoSlow” indication is defined that signals that the QoS has dropped below a predefined value.

5.4.5.2 ATM based events

For further study

5.4.5.3 Modem events

For modem Edge Points a “connected“ event is defined that has the speed as a parameter.

5.5 Meta resources

Meta resources are not directly connected to one session. There are currently three kinds of meta resources defined:

- The Control channel
- Signaling backhaul
- Scripting

5.5.1 Control channel

The control channel resource is always supported. it allows the MG to be controlled. It is possible to have multiple objects of this type, for primary and secondary control connections. The properties of this resource are:

ControlleeAddress	Address of the interface on the MG(e.g. IP addr+UDP port)
ControllerAddress	Address of the controller (e.g. IP addr+UDP port)
Relationship	value: primary/secondary/peer
protoversion	Protocol version ID
BufferInController	
BufferInControlle	
HeartBeatInterval	
ackTimer	
resultTimer	
SessionKey	Encryption key of the control session

Currently the only event that is defined in the *Heartbeat* which takes no parameters. For the sematics of the Heartbeat see Section 10.8.
Note: the Peer operation is for further study but is intended for communication between controllers so that one could build hierarchies of controllers to increase resource utilization.

5.5.2 Signaling backhaul

For some applications it may be desirable or necessary to do some of the signaling from the media gateway box rather than from the controller. A separate resource exists to pass on channel associated signalling (from MG to MGC in *Indication* control functions, from MGC to MG in *command* control functions).
Several types of signaling resources can be defined e.g., ISUP/SS7, Q.931, H.225, H.245.

In the case of the connection setup protocols like ISUP/SS7, Q.931 and H.225 one signaling resource instance per signaling connection would be involved for multiple sessions. The definition of these is for further study.

5.5.3 Scripting processor

A scripting processor can catch events sent from certain resource objects and act on them by issuing commands described in Section 7 or methods on the resources directly and issue new *events* to the controller.

The scripting resource is a very important part of the working of a MG. The scripting processor will allow the low level functionality of the resources to be aggregated to more complex activities.

An example would be to play a message, collect digits, play another message, collect more digits. (like the requirements described in draft-cromwell-navdec-media-req-00) On the low level that resources operate on, this would include a lot of commands or method calls on resources and a lot of events being sent. The scripting processor should be able to do all this. In this way the scripting processor can be used to reduce the amount of communication between the controller and the MG and will reduce the amount of attention the controller must give to each session. The scripting processor has its own language that is for further study, the requirements for the language differ for the applications and the size of the MG. Possible candidates are. Java, Perl, and Python but also a new scripting language could be defined.

5.6 Media Resources

5.6.1 Trunk module

5.6.1.1 echo canceller

Type:	PipeProcessor
Use:	To cancel echoes. The canceller is created with one of its two media streams as first pipe, a ReConnect() method is used to attach the reverse path to the canceller.
Type Number:	TBD
SubType:	TBD

Extra parameter: none
Indications: none

5.6.2 DTMF module

5.6.2.1 DTMF detector

Type: EndProcessor
Use: to detect DTMF in a media stream
TypeNumber: TBD
SubType: TBD
Extra parameters: number of digits to collect before indication is sent, and timeout in ms.
Indications: sends indications enumerating the received DTMF digits

5.6.2.2 DTMF generator (optional)

Type: EndProcessor
Use: to generate DTMF into a media stream
TypeNumber: TBD
SubType:
Extra parameters: DTMF digit to be sent , length in ms (digit length, interdigit interval)
Indications: indicates ready with playing.

5.6.2.3 DTMF filter (optional)

Type: PipeProcessor
Use: To filter DTMF signals
Type Number: TBD
SubType: TBD
Extra parameter: Codec
Indications: none

Note: To support out of band DTMF signaling with erasure of DTMF from the media stream, one would use a branch where media terminates at a DTMF generator for one branch, and the DTMF filter is the other branch

Note 2: It may be desirable to create combined resources that perform the function of multiple DTMF resources in one.

5.6.3 IVR module

5.6.3.1 tone player

Type: EndProcessor
Use: to generate tones
TypeNumber: TBD
SubType: Some tones are ennumerated in subtype
01 dialtone
02 2^{nd} dialtone
03 busy tone

04 ringing tone
05 error tone
06 ….

Extra parameters: length in ms
Indications indicates ready with playing.

Note that the tone may be localized, dialtone is not the same everywhere.

5.6.3.2 message player (optional)

Type: EndProcessor
Use: to play prerecorded messages
TypeNumber: TBD
SubType: stored message number
Extra parameters: length in ms
optional: number of repetitions
Indications: indicates ready with playing.

Note: a message playing is stopped by deleting the appropriate *ResourceID* or modify its length to 0.

Note: One could extend the message player to support parameters into a message (e.g., "You have x minutes remaining in your account", where x is the parameter).

5.6.3.3 message recorder (optional)

Type: EndProcessor
Use: to recorded messages
TypeNumber: TBD
SubType: TBD
Extra parameters: maximum length in ms
Indications: Optional: silence detected.

5.6.4 Audio module

5.6.4.1 Transcoder (optional)

Type: PipeProcessor
Use: To transcode between several types of codecs
Type Number: TBD
SubType: TBD
Extra parameters: Codec In Codec Out[2]
Indications: TBD

5.6.4.2 Comfort Noise generator

Type: PipeProcessor
Use: To generate comfort noise
Type Number: TBD
SubType: TBD
Extra parameters: Codec
Indications: TBD

[2] The codecs are enumerated according to H.245 or RTP. TBD

5.6.4.3 Gain (optional)

Type: PipeProcessor
Use: adjust the volume of the audio stream
Type Number: TBD
SubType: TBD
Extra parameters: Codec, gain
Indications:

5.6.5 Tone Module

This module will house all kinds of tone detectors to support MF, SIT tones, other call progress tones, etc.

5.6.5.1 Modem detect (optional)

Type: EndProcessor
Use: to detect the presence of a modem on the other end.
TypeNumber: TBD
SubType: TBD
Extra parameters: Boolean whether to send a tone first or just wait
Indications: Modem detected.

The indication can be used to dynamically attach a modem Edge Point when a modem is detected. This is a typical activity for a scripting engine.

5.6.6 A/V module

5.6.6.1 Synchronizer

Type: PipeProcessor
Use: synchronize two media streams (typically audio and video) The stream that will be the time base will be connected first, the stream that is to be synchronized is ReConnected.
Type Number: TBD
SubType: TBD
Extra parameters: max positive delay. Max negative delay
Indications:

Note: this would assume that RTP timing information is kept along with the media stream. This is for further study.

5.6.7 Encryption module

5.6.7.1 encrypt

Type: PipeProcessor
Use: To encrypt media stream
Type Number: TBD
SubType: algorithm used
Extra parameter: key
Indications: None

5.6.7.2 Decrypt

Type: PipeProcessor
Use: To decrypt media stream
Type Number: TBD
SubType: algorithm used
Extra parameter: key
Indications: None

5.6.8 Bridge module

Type: PipeProcessor
Use: Mix several incoming connections. A bridge resource is allocated and connected as a pipe processor in a media stream, thus binding the correct input and output. Further additions to the bridge are performed using the ReConnect method (invoked through a Connect command). See Figure 5 for an example. The UnConnect() method (invoked from the Delete command) is used to remove participants from the bridge.
TypeNumber: TBD
SubType: algorithm encoding
Extra parameters: codec (??)
Indications:

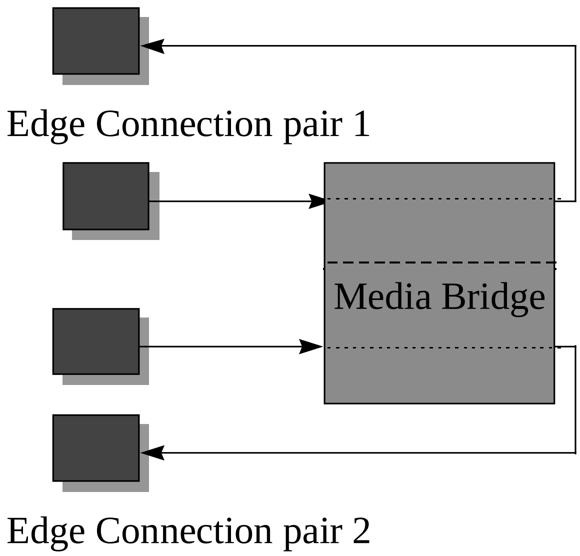


Figure 5. media bridge example

The example in Figure 5 shows one bridge and two bidirectional channels each made up of a connection pair with in and output edge points. For voice applications; the inputs are mixed into the bridge. The output for each connection is the mixed signal minus the corresponding input signal.

5.6.9 Access module

Note: these may have to be properties of a PSTN access Edge point, this is for further study.

5.6.9.1 In band signaling detect

Type: EndProcessor
Use: to catch inband signaling
TypeNumber: PSTN (others may be possible as well)

SubType: TBD Enumerate country specifics here?
Extra parameters: none
Indications: 01 on-hook
02 off hook
03 hookflash detected
....

5.6.9.2 In band signaling generate

Type: EndProcessor
Use: to generate inband signaling
TypeNumber: PSTN (others may be possible as well)
SubType: TBD Enumerate country specifics here?
Extra parameters: action: 01 swivel polarity, 02 send metering pulse...
Indications: none

5.6.10 H.245 support module

This module allows H.245 connections to be setup from the MG. An H.245 connection is closely linked to as session, hence the H.245 signaling resource is connected to both edge points of the applicable connection and can subsequently be used to arrange part of the connection parameters. This is not a media connection but indicates to the MG how the H.245 connections are to be linked to the media connections.

The media box can send up an **event** when a H.245 message is received and H.245 messages are generated by sending the resource a **Modify** command. The exact information is these messages are for future study but will probably carry the H.245 information rather then the full ASN.1 encoding.

The definition then becomes:

Type: EndProcessor
Use: to perform H.245 session associated signaling
TypeNumber: TBD
SubType: TBD?
Extra parameters: remote H.245 address
Indications: for further study

The exact properties of this resource are for further study.

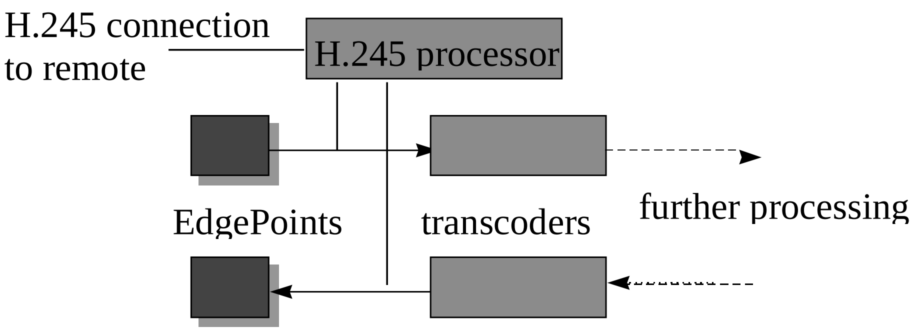


Figure 6. H.245 processor example

One can create the model as shown in Figure 6 by connecting to one media stream on creation of the H.245 resource and using the ReConnect() method for the reverse path. By connecting the H.245 resource to the edge points like this, one can signal to the media box that a H.245 channel is needed for that connection.

A simple MG can simply relay all the information. A more intelligent box can filter out the H.245 information it can handle itself and only pass on the more complex issues. A clever design of indications and messages could deliver a number of message sets for each type of operation.

A heavyweight media box could be told to setup a connection automatically by creating a pair of edge points with the RTP addresses and ports wildcarded and connect those to a H.245 resource while giving it the appropriate H.323 endpoint address for its connection. After the H.245 communication has completed the MG can fill in the IP address and port and other parameters like CODEC for itself.

Other issues with H.245 signaling
In the case of H.245 signaling one can have a connection going through the media box that requires a H.245 interface on both ends or a situation as shown in Figure 7. The figure shows an asymmetrical setup than can be found when:

- Terminal 1 speaks H.323 V1 (or declines Fast Start) and Terminal 2 wants FastStart
- Terminal 2 is in a PSTN domain

Such a setup will present some interesting issues for the information flow between the controller and the media gateway.

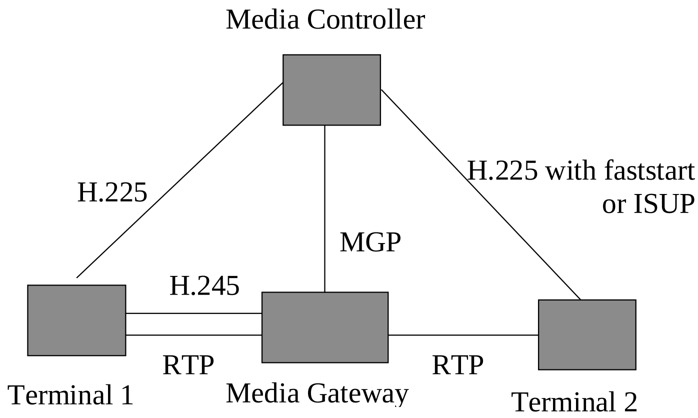


Figure 7. Asymmetrical setup. (Gatekeepers are omitted for clarity.)

5.7 Vendor Specific Modules

Vendor specific modules look extend the concept of standardized modules. The moduleID of a vendor specific module is extended with a vendor ID. See the Section 9 on Encoding on the details.

Vendor specific extensions of other modules are not allowed, if a vendor wishes to extend a standard module he has to create a new module with the resources of the standard module in it.

6. Messages

Control of the MG by the MGC takes place through the exchange of messages between the MGC and MG.

6.1 Message Structure

Each message is made up of a number of control functions (See Section 7). A sequence of related control functions makes up a transaction (See Section 10). Messages may contain information relevant to several concurrent transactions and concurrent sessions.

The message must contain an integral number of control functions. The maximum number of control functions that may be included in a message varies as the maximum length of the messages may vary. The maximum length of the message is negotiated during the registration procedure (See Section 10.7).

6.2 Transport

Messages may be carried over a reliable or unreliable underlying network. A mechanism of sequencing and acknowledging control functions is provided within the protocol.

6.3 Security

Each control session may be encrypted. A *CryptoToken* field is included in the message header to provide the security mechanism and is based on the procedures defined in the ITU-T Recommendation H.235. The use of encryption is determined during the registration procedure (See Section 10.7). Initial encryption keys and algorithms will typically be entered in the MGC and MG manually. Updates to the encryption keys may be done a Modify command on the SessionKey parameter of the control session resource. An alternative is to use the Registration control function, in that case the command may be issued by both MGC and MG. Integration with the framework in H.235 is for further study.

7. Control Functions

Control of the MG by the MGC is performed using a set of control functions. Control functions are the means whereby specific commands, indications and responses are conveyed between the MGC and MG.

7.1 Notation

The control functions are denoted in EBNF syntax, A short recap follows below:

```
STRING                  a token
<identifier>            an identifier
<def> ::= ………           a definition
<clause1> | <clause2>   clause 1 or clause 2
{ <clause> }            clause is optional
```

7.2 Control Function Definitions

7.2.1 Connect

Connect is the command from the MGC to the MG to establish a *Connection* or *Inter-Connection* within the MG.

7.2.2 ConnectConfirm

ConnectConfirm is the response from the MG to a Connect command.

7.2.3 Modify

Modify is the command from the MGC to the MG to change a resource or the Parameters of a resource within the MG.

7.2.4 ModifyConfirm

ModifyConfirm is the response from the MG to a *Modify* command.

7.2.5 Delete

Delete is the command from the MGC to the MG to delete a Session, Connection or resource within the MG. (Inter-Connects are deleted implicitly when the resource objects to which they are connected are deleted).

7.2.6 DeleteConfirm

DeleteConfirm is the response from the MG to a *Delete* command.

7.2.7 Query

Query is the command from the MGC to the MG requesting information. The *Query* command can be used to get all kinds of information from the MG.

7.2.8 QueryResponse

QueryResponse is the response from the MG to the *Query* command.

7.2.9 Registration

Registration is a control function from either the MGC or the MG requesting initiation of a Registration transaction which is necessary for a control session.

7.2.10 RegistrationResponse

RegistrationResponse is the response from either the MGC or the MG to a *Registration* control function.

7.2.11 Indication

Indication is a control function sent from the MG to the MGC in response to an event occurring within the MG. E.g. a connection being lost or a DSP board breaking down. Events may be caught in a scripting processor in that case no indication follows upon the event. All events that are not caught internally are sent to the controller with mention if the resource object that caused them.

7.2.12 Acknowledgment

Acknowledgment is a control function sent to acknowledge the correct receipt of a specific control function. See also Section 9 for implicit acknowledgments.

7.2.13 Termination

Termination is a control function sent by either the MG or MGC to terminate a control session or registration transaction.

7.2.14 Macro

Macro is a control function sent by the MG to call a macro that is either standardized or specified in a scripting processor. Note: A macro can initiate a script or just functionality that can be encoded using a script.

7.2.15 MacroResponse

MacroResponse is a control function sent to respond to a *macro* command.

7.3 Control Function Formats

7.3.1 General

Each control function is identified by a unique *ControlFunctionID*, and a *SessionID* indicating the Session to which the control function applies. A number of other parameters, specific to each control function type are associated with the control function.

7.3.2 Connect

Using the *Connect* command new connections are created or additions are made to existing connections. *Connect* can be used to create a connection with multiple branches in a single command.

The connection created with this command is unidirectional. The first End Resource in the command is the source of the connection and the rest of the command describes the media path downstream. Note: see below how a bidirectional connection can be created by using multiple connect commands in one message.

The command takes the following form:

```
<connect> ::= CONNECT <SessionID> {<ConnectionID>} (<NewEdgeResource>
|
                         <interconnectID>) <branchlist>

<branchlist> ::= {<branchlist>} <branch>

<branch> ::= <StartBranchToken> {<branchlist>} {(( <NewPipeResource> |
             <piperesourceID> )
               <intraconnectID>{<branchlist>})}
              (<NewEndResource> |  <endresourceID>)
             <EndBranchToken>

<NewEndResource> ::= <EdgePointType> { <props> }
                   | <EndProcessorType> { <props>}

<NewPipeResource> ::= <PipeProcessorType> { <props> }

<props> ::= {<props>} <property>

<property>:== <type><value>
```

The direction of the connection is given by the sequence in the Connect. The source of the connection is the first clause of the CONNECT (so the new edge resource or the interconnect), the direction of the rest of the resources is implicit.

Note that a resource may be created by specifying the values of a few or no parameters. Most of these parameters may be filled in by the MG because they are implicit (e.g. the encoding and the relevant interconnectIDs). A resource that has insufficient parameters will be idle until (yet allocated) enough parameters are filled in.

The *Connect* command will cause new resource instances to be created and interconnections between them. The mapping between the command and the operations on the model is as follows:

> The connect command creates a session object if one does not exist. If the source of the connection is an Edge resource a new EdgeResource object is created, a new connection instance is created and linked to this edge resource, subsequently the connection is which is linked to the session.
>
> Interconnection instances are created implicitly and do not need to be signaled. The Branch construct allows the signaling of interconnects connecting with multiple resources.
>
> New resources that are signaled using the NewEndResource and NewPipeResource clauses cause New() methods on the appropriate resource types with the named properties filled in. The appropriate inter connects are also implicit.

For some applications it might be desirable to implicitly denote a ReConnect to a previously created resource. For instance to allow bidirectional streams to be created with echo cancellers and other resources that use the ReConnect() method. In that case the resource is given as a parameter in which Connection the resource has been mentioned earlier.

7.3.2.1 Example

The structure of the CONNECT message is made to be capable of creating very complex structures. Yet it also can make simple connections. The figure below shows a pair of simple connect messages setting up a session in a trunk gateway:

```
CONNECT SessionID=42
        ConnectionID=1
        NewDS0Trunk (circuit=123)
        <startbranchToken>
            NewEchocanceller ()
            NewTranscoder (to=G.723.1)
            NewRTP (to=120.120.120.2:4040)
       <EndBranchToken>
CONNECT SessionID=42
        NewRTP (port=4050, codec=G729))
        <StartBranchToken>
            NewTranscoder ()
            NewEchocanceller (ConnectionID=1)
            NewDS0Trunk (circuit=124)
        <EndBranchToken>
```

The controller fills in the sessionID (42) so new Session object is created (SessionType.New(ID=42)).
A new Connection object is created (ConnectionType.New(ID=1)) resulting in a connection ID (<Connection,1>) which can be linked to the session (<Session,42>.ReConnect(<Connection,1>)
The first connection is created from a circuit on a trunk (named 123). (this implies the encoding), a new EdgePoint object is created with the appropriate parameters (EdgePointType.New(circuit=123)) This will result in a new Resource object for instance with ID <EdgePoint,10>. This is the source of this connection so it will be linked to the connection (<Connection,1>).ReConnect(<EdgePoint,10>))

At the StartBranch token the MG can create a new InterConnect Object. (InterConnectType,New(<EdgePoint,10>). Resulting in a new Interconnect ID (<InterConnect,100>) So it can be linked to the resource (<EdgePoint,10>.ReConnect(<InterConnect,100>)).

The NewEchoCanceller is created without parameters and linked to the Interconnect. A new interconnect is created between the EchoCanceller orject and the Transcoder. (the input CODEC can be inherited from the output CODEC from the Canceller which can be taken from the CODEC of the trunk.

Etc.

The second CONNECT command uses the same session ID as the first so only a new connection object is created and which is ReConnected to <Session,42>
The EchoCanceller called for in the second CONNECT can be reconnected to the one created in the first CONNECT command, which is signaled by supplying the clause <ConnectionID=1>.

7.3.3 ConnectConfirm

The connect confirm is sent as a response to the Connect command. It gives information about the connection that has been established. This information can be used later to change the connection or parameters in its resources. The command takes the following form:

```
<connresp> ::= <ConnectionID> <errorMessage> |( {<EndResourceID>}
                <InterconnectID> <branchinfolist>)

<branchinfolist>::= {<branchinfolist>}<branchinfo>

<branchinfo> ::= <StartBranchToken> {<branchinfolist>}
                {(<PipeResourceID> <interconnectID>
                 {<branchinfolist>})} <EndResourceID> |
                  NullID) <EndBranchToken>

<endresourceid> ::= <EdgePointID> | <EndProcessorID>

<errorMessage> ::= //For further study
```

The *InterConnectIDs* allow a subsequent *Connect* command to be issued to create a new branch.

The returned *ResourceIDs* allow subsequent Modify operations to change parameters in the resource.

7.3.4 Modify

Modify is the command from the MGC to the MG to change a resource or the Parameters of a resource within the MG.

The command takes the following form:

```
<mod> ::= Modify <resourceID> <props>


<resourceID> ::= <EndResourceID> | <PipeResourceID>
```

7.3.5 ModifyConfirm

The command takes the following form:

```
<modconf> :== <OK> | <errorMessage>
```

7.3.6 Delete

Delete is the command from the MGC to the MG to delete a Session, Connection or resource within the MG. Deleting a resource object will free it up for use in other Connections. The delete command takes the following form:

```
<del>::= Delete ( <SessionID> {<connectionID>} ) | <resourceID>
```

Note: A Delete on a resource object will also implicitly remove the rest of the flow downstream until the first Inter-Connect which is fed by another source. (So a Delete on an source Edge Point can be equivalent to a Delete on the Connection.)

7.3.7 DeleteConfirm

Since a delete will always succeed, the *DeleteConfirm* response is empty and indicates a positive confirmation of a *Delete* command.

7.3.8 Query

The query command can be used to query the state of the MG as a whole or of individual objects in the model. The command contains one or more of the following parameters:

Parameter	Purpose
Info	returns general information on the MG, vendor and type
AllSessions	returns a list of all *SessionIDs*
<SessionID>	returns a list of all *ConnectionID*s
FreeResources	returns a list of all free resources with a full description
<ConnectionID>	returns a list like that went into connect
<ResourceID>	returns resource properties
<InterConnectID>	returns a list of connected *ResourceID*s

An example use of a *Query* command is to get parameters from an Edge Point that were wildcarded. (E.g. the UDP port used). A *Query* on an RTCP enabled IP connection could give the QoS statistics.

7.3.9 QueryResponse

The command takes the following form:

```
<QueryResponse>::= <string> // for Info
                 | <sessionIDlist> // for AllSessions
                 | <connectionIDlist> //for <SessionID>
                 | <resourcetypelist> //for FreeResources
                 | <connect> // for <ConnectionID>
                 | <props> // for <resourceID>
                 | <resourceIDlist> //for <InterconnectID>

<sessionIDlist> ::= {<sessionIDlist>} <SessionID>

<connectionIDlist> ::= {<connectionIDlist>} <connectionID>

<resourcetypelist> :== {<resourcetypelist>} <resourcetype>

<resourcetype> ::= <EdgePointType> | <PipeProcessorType> |
                   <EndProcessorType>

<resourceIDlist> ::= {<resourceIDlist>} <resourceID>
```

The case when a query response becomes too long for the message size (see below) is for further study.

7.3.10 Registration

The command has the following form:

```
<reg> ::= Register (Primary|Secondary|Peer) <regparams>


<regparams> ::= {<regparams>} <regparam>


<regparam>::= ControllerAddress <value>
             | ControlleeAddress <value>
             | ProtocolVersionID <value>
             | BufferInController <value >
             | BufferInControllee <value >
             | hartbeatInterval <value >
             | ackTimer <value >
             | resultTimer <value >
             | KeyUpdate <value>
```

The value <ackTimer> is used to indicate the amount of time in which a command must be acknowledged. The value is defined in milliseconds a typical value would be less than 500 milliseconds.

The value of <resultTimer> gives the maximum time within which a command must return a result. The value is defined in millseconds, a typical value would be less than a second.

The media gateway will send a heartbeat *Indication* every <heartbeatInterval> seconds. The reception of this *Indication* will be acknowledged by the controller. The value is defined in seconds, a typical value might be 10 seconds.

7.3.11 RegistrationResponse

The command takes the following form:

```
<regresp> ::= <reg> | <ok>
```

7.3.12 Indication

The command takes the following form:

```
<ind> ::= <sessionID> <resourceID> <EventID> {<value>}
```

7.3.13 Acknowledgement

The command takes no parameters.

7.3.14 Termination

The command has the following form:

```
<term> ::= Terminate (<Primary|Secondary>) <ControllerAddress>
           <ControlleeAddress>
```

7.3.15 Macro

Two kinds of macros can be called:

1. macros that have been predefined in profiles (see Section 8) in that case the resource ID is that of the resource maintaining the Control Session
2. macros that have been downloaded in a scripting processor.

In both cases a number of parameters can be given to the macro.
The macro command takes the following form:

```
<macro> ::= (<resourceID> <macroID> ) <props>
```

7.3.16 MacroResponse

The macroresponse command takes the following form:

```
<macroResp> ::= <OK> | <props> | <errormessage>
```

7.4 Transport

Note: This section needs some work and will be updated soon.

For each control function sent within a message an *Acknowledgement* control function is returned as a confirmation of correct receipt. (The use of a CRC field in each control function and a *NegativeAcknowledgement* control function to indicate control functions received in error is for further study.)

A *Send Sequence Number* field is included in each control function header. The value of this number allows the sender to determine that each control function sent has been correctly received. The *SendSequenceNumber* is incremented by one (modulo 128) for each control function sent of the same type within each session. The *Acknowledgement* control function is an exception. In this case this field shall contain the same *SendSequenceNumber* as the received control function to which the acknowledgement applies.

8. Profiles

The model provides a fairly low level command set and therefore a high level of control. For specific application special profiles can be created. The profiles result in a number of standardized *macro* sets. Implementation of just these macros will allow a simple interface.

The actual profiles are for further study below we have given some examples. The mapping to the model is for further study..

8.1 Trunk Gateway

Below are some examples given for macros that would fit in the Trunk profile.

8.1.1 SetupIPSession

A session that has to be set up like the example in Section 7.3.2.1 is the common mode of operation for these gateways.
One could define a SetupIPSession macro that would always include:

- trunk edge point
- RTP edge point
- echo canceller
- comfort noise
- transcoder
- encryptor

Note: operator specific additions like message playing upon connection setup or even advertisements in the session would be a modification on the *implementation* of the macro and would not change external behavior.

This can be signaled by sending a Macro command with the following in and out parameters:
The in parameters of relevance are then:

- SessionID
- Trunk circuit number
- RTP send address and port
- RTP side encoding
- RTP framing etc.
- encryption algorithm and key

Return values:

- RTP receive address and port.

If the parameter names in the resources are chosen well the same names can be used here.

8.1.2 TwoStage

A gateway that supports two stage dialing would first establish the identity of the user and query the number to be called and subsequently setup the connection.

A TwoStage macro would have the following parameters:

- SessionID
- Trunk circuit number

And would later send up an indication giving:

- userID
- calledNumber

A variant on the SetupIPSession could subsequently be used, the trunk circuit does not need to be signaled but some indication of the time left in case of pre-paid calling might be needed.

8.1.3 TrunkLoopBack

A trunk loopback in this model is equivalent to connecting two trunk edge points. A macro for this would only include:

- input trunk circuit ID
- output trunk circuit ID

8.2 Access Gateway

The access gateway has to perform in-band signaling before the call is actually proceeding.

8.2.1 Access

The Access macro would wait for a user to go off-hook, and dial some numbers.
The complexity of dialtones and digit maps would be hidden in the MG.
The dial plan could be a separate script in the processor which is downloaded in to the MG at startup.

The exact approach is for further study but one might setup a script that catches off-hook of all access ports and dynamically instantiates a new Access Macro on that line.

The Access macro would have the following parameters:

- SessionID
- line ID
- dial plan

The macro would send up an indication when a complete number is (believed to be) collected. This indication would include

- dialed number
- SessionID

The controller would subsequently issue a SetupIPSession like Section 8.1.1.

8.3 Residential gateway

A residential gateway is architecturally similar to the access gateway but may have a smaller size and not the processing power of a full scripting engine. It is for further study how this will differ.

8.4 NAS server

A NAS server that only has a modem pool will have one simple Macro

8.4.1 ConnectModem

ConnectModem logically connects a modem to a trunk line (for a pure NAS server the modems would already be connected. It would have the following parameters:

- Session ID
- trunk circuit
- IP address

Access control (passwords, CLI???) is for further discussion and will have significant impact on the shape of this macro.

8.5 Multipoint processor

8.6 IVR system

9. Encoding

The section describes the binary encoding scheme that is proposed. It is still work in progress!

9.1 Message Encoding

The general format for the encoding of messages is shown below.
Note that a protocol ID is not necessary because that has been negotiated during the registration,

<table>
<tr><td>Include Crypto</td><td>Message Length</td></tr>
<tr><td colspan="2">Crypto Token (optional)</td></tr>
<tr><td colspan="2">Message Contents</td></tr>
</table>

Include Crypto indicates if the *Crypto Token* is included in the message. It is single bit and is set to 1 to indicate the *Crypto Token* is included; otherwise, this bit is set to 0.

Message Length specifies the length of the *Message Contents* in octets; message length is a 15 bit unsigned integer quantity and is transmitted most significant bit first.

The *Crypto Token* the encoding of the crypto token is for further study; it will follow H.235 but may not use ASN.1 since we want to be able to handle real simple gateways.

Message Contents is a series of one or more Control Functions The message must contain an integral number of Control Functions and the maximum number included in a message varies as the length of the messages may vary. However, the total length of this field must be less than or equal to the value negotiated during the registration procedure.

9.2 Control Function Encoding

The encoding of Control Functions consist of two parts, a fixed header whose format is the same for all Control Functions and a variable portion consisting of elements that are Control Function specific as shown in the figure below.

ControlFunctionID
Session ID
SendSequenceNumber
Length
Elements

ControlFunctionID specifies the type of command. It is a single octet in length and takes a value from the table below:

Control FunctionID	Control Function
00	Ack
01	Connect
02	Modify
03	Delete
04	Query
05	ConnectConfirm
06	ModifyConfirm
07	DeleteConfirm
08	QueryResponse
09	Indication
10	Registration
11	RegistrationResp
12	Termination
13	Macro
14	MacroResponse

SessionID denotes the session to which the control function applies.

SendSequenceNumber –The length of the *Send Sequence Number* is one octet.

Length gives the total number of octets contained in the *elements* field. It is two octets in length and is transmitted most significant bit first

Elements give the parameters to the specified command. These elements consist of *ResourceIDs, Inter-ConnectionIDs, ConnectionIDs* and parameter lists.

9.3 Element Encodings

All elements share a common format as shown in figure below.

<table>
<tr><td>Type</td><td>Identifier Length</td></tr>
<tr><td colspan="2">Identifier</td></tr>
<tr><td colspan="2">Content Length</td></tr>
<tr><td colspan="2">Contents</td></tr>
</table>

Type specifies the category of the element. It is 3 bits in length and takes one of the values from the table below:

Element Category	Type
Standard Resource	0
Vendor Specific Resoruce	1
Interconnect	2
Connection	3
Parameter	4

All values not specified in the above table are reserved for future use.

The *Identifier Length* field gives the length of the element identifier in octets; this field is 5 bits in length.

The encoding of the *Identifier* field are dependent on the value of the *Type* field and are specified below.

The *Content* field is specific to the specified element and may not be present at all; this field gives the parameters associated with the element.

9.3.1 Standard Resources

Standard Resources have an *Identifier* format as shown below:

ModuleID
ResourceID
ResourceInstance

ModuleID identifies the module that contains the resource. This field is a single octet in length.

00	Reserved
01	EdgePoint
02	Control connection
03	Scripting
04	Signaling Backhaul
05	Trunk module
06	Access Module
07	DTMF module
08	IVR module
09	Audio processing
0a	Encryption
0b	Bridge Module
0c	
0d	
rest	

ResourceID gives the specific type of resource within the module and is a single octet in length.

The *ResourceInstance* gives the number of the resource. This is unique within the module/resource.

Note: Vendor specific resources may not be added to a standard module; rather, vendors may define a new vendor specific module that incorporates the standard features of an existing module.

ResourceTypeIDs follow the same form as ResourceIDs but with the ResourceInstance bits set to 0.

9.3.2 Vendor Specific Resources

The encoding of Vendor Specific Resources must allow for the identification of the specific vendor. To achieve this goal, the vendor specific resources have an *Identifier* format as shown below:

VendorID
ModuleID
ResourceID
ResourceInstance

The module numbering is vendor specific.

The *VendorID* determines which vendor specified the resource. . (The ID follows H.245) It is 4 octets in length.. The Vendor ID consists of three sub-fields as defined by H221NonStandard of ITU-T Recommendation H.225. It is encoded as follows:

The first octet of the Vendor ID specifies the t35Country code as defined in ITU-T Recommendation T.35.

The second octet specifies the t35Extension and is assigned nationally.

The third and forth octets are treated as an unsigned 16 bit integer quantity that specifies the manufacturer code. This field is assigned nationally.

9.3.3 InterConnect

The InterConnect element *Identifier* is a 32 bit unsigned integer that is transmitted most significant bit first.

9.3.4 Connection

The Connection element *Identifier* is a 32 bit unsigned integer that is transmitted most significant bit first.

10. Transactions

A master/slave relationship exists between the MGC and the MG. Control of the MG by the MGC is performed via the exchange of messages. Each sequence of message exchanges forms a transaction. Each transaction is independent of all others and can be initiated at any time.

10.1 Create a connection

Creation of the Connection from right to left in the example given in Figure 4 will be signalled as follows:

```
CONNECT Edgepoint (Type=Trunk, ID=????),
        EchoCanceler(Type=???,Delay=???),
        TransCoder(Codec1=MuLaw, Codec2=Q.723.1)
        StartBranch
           Encrypter(Type=Caesar, Key=123)
           Edgepoint(Type=RTP/IP, To=123.123.123.001:4012,
              MyAddress =123.123.123.100:5013)
        EndBranch
        StartBranch
        Encrypter(Type=DES, Key=123344)
        Edgepoint(Type=RTP/IP, To=100.100.100.001:4012,
              MyAddress =123.123.123.100:5016)
        EndBranch
```

In this example the *Create* command creates an Edge Point instance with the specified properties that is connected to a certain circuit on a trunk. An Inter-Connect will be created from this Edge Point to an echo canceller Pipe Processor Resource of designated type and delay. The output of that will be Inter-Connected to a transcoder converting G.711 to G.723.1. This will be Inter-Connected to two branches each with an encrypter Pipe Processor Resource Inter-Connected to an Edge Point.

When a *Connect* command is successful the *ConnectConfirm* control function is sent by the MG to the MGC.

10.2 Modify a resource

Using the Modify control function one could move the media stream to another address (e.g. to establish the media side of a call transfer)

The update of an encryption key is signalled as follows:

```
Modify (EncryptorID, Type=Caesar, Key=10)
```

10.3 Change or modification of a connection

A connection is modified by adding new branches to an interconnect and removal of branches.

10.4 Delete a connection or resource

See section 7.3.6

10.5 Request Information from MG

See Sections 7.2.7 and 7.2.8

10.6 Indication of events by MG

Indication events are sent asynchronously from the MG to the MGC. The MGC only needs to acknowledge these to comply with the protocol.

Some examples need to be added here.

10.7 MG Registration

A media gateway will be controlled by one controller at any given time. A registration will take place before the controller can control the resources in the gateway. At any given time one controller will have control the gateway, this is called the primary controller. Secondary controllers may be registered as well to allow fall back should the primary controller fail.

Note: Peer registrations may happen among MGC to build hierarchies of MGC to create complex services from MG pools. This is for further study.

Registration can be initiated by the controller as well as by the media gateway. In the first case, the controller has learned the address of the media gateway and issues a registration command. The registration command may also be sent to a multicast group by a media gateway in search of a controller. (After the registration phase the relationship between MG and controller is strictly client server.)

Depending on the issuer of the *Registration* command, several fields may be wildcarded.

By sending the parameter list back and forth using subsequent ResigstrationResponse commands, the MGC and MG may use the *Registration* command to negotiate the parameters of the control connection. In the registration process, the controller or media gateway involved in the negotiation does not repeat the entries which it agrees upon. This negotiation ends when all the parameters are agreed upon or when a *Terminate* command is received.

Note 1: The addresses depends on the transport network used, it may be possible that a controller can use two network technologies to control the media gateway (for instance ATM and IP).

Note 2: The buffers indicate the input buffer size on the control channel. This limits the size of the control packets and their rate.

Note 3: The protocol version that is used must be the same for all control sessions the media gateway has with its primary and secondary controllers.

10.8 Control Handover

There are three ways by which control can be handed over:

- **Timeout on heartbeat** : Should the media gateway discover that its controller does not respond within <ackTimer> on messages or heartbeat indications, the media gateway may conclude that its controller has died and may send a register command to any of its secondary controllers with the option Primary set.
- **Secondary discovers**: The alternatively, a secondary may discover the death of the primary controller and may issue a register command with the Primary flag set.
- **Voluntary handover**: The third option is that the controller hands over control itself. It names its successor by sending a Register message with the address of its successor filled in as controller. This successor must be registered as secondary.

Following a handover, the new primary may be unaware of the active calls on the media gateway. The new primary can use a series of Query commands to learn about the status of the calls. A similar relationship with the Gatekeeper may give the new controller additional information to complete the picture. Note: the best situation would be when the old primary would keep the secondary as hot standby so that all information is readily available.

11. Annex 1 Requirements for the Control to Media Interface

11.1 Source of Requirements

A number of media gateway requirements are known from the Tiphon Architecture Document - DTS/02002 v.0.3.1. We have selected the essential ones and have abstracted them in the following sections in addition to ones identified as part of the protocol discovery.

11.2 General Requirements

- Designed to minimize encoding and decoding complexity and therefore it will protocol message will be defined using a Binary Encoding Scheme.
- Communication across the interface be initiated by either the Media Gateway Controller or the Media Gateway.
- The protocol shall be extensible.
- Use appropriate methods to minimize traffic and delay (e.g. no DNS lookup but instead use IP address).
- Support Media Gateway and Media Gateway Controller registration and "Keep Alive" processing.
- Support optional vendor specific extensions
- Mechanism to support communication confirmation (e.g. reply options of acknowledge, acknowledge on error) and no confirmation (e.g. use timers and detect expiration).
- Initiated messages shall have available associated Success and Failure response messages.
- Provide a mechanism for reliable (redundant) operation.
- Support a mechanism to negotiate versions and extensions.
- Status requests and reports

11.3 Scope of Media Control

Media Control consists of four functions:

- <u>Connection Control</u> - Creation, modification, and deletion of media stream connections within the Media Gateway.
- <u>Media Stream Transformations</u> - 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.
- <u>Content Insertion</u> - Requests to the Media Gateway to insert content (tones and announcements) into the media streams, either on direct request from the Media Gateway Controller or beginning and ending with the detection of specified events within entity itself.
- <u>Event Driven</u> - Requests to the Media Gateway to report and possibly take actions upon detection of well-defined events within the media streams.

11.3.1 Connection Control

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

- circuit to packet (and vice versa) for both IP and ATM (e.g. H.323 Annex C operation)
- circuit to circuit (circuit side fallback)
- packet to packet (IP or ATM)

The Media Gateway will be notified through Connection Control of the capabilities of the connection based upon call setup and media negotiation (like H.245 negotiation):

- Ability to identify the IP, ATM, and/or circuit connection edge point involved in a connection. On the packet side, edge point descriptions shall reflect the content provided by *networkAddress* portion of the H.245 *NetworkAccessParameters* type. On the circuit side, it shall be possible to designate circuits using a hierarchical naming construct.
- It must be possible to wildcard the low-order portion of the edge point identifier (e.g. the port number for an IP transport address or the low-order term of a circuit identifier), with the intent that Media Gateway shall select an idle edgepoint instance and return its full identity to the controller. Wildcarding shall also be provided to allow a simultaneous reference to multiple endpoints.
- Ability to request the selection of ports for both RTP and RTCP reception on the packet side.
- Ability to specify unicast or multicast propagation of the media stream on the network side.
- Ability to specify the QOS parameters applicable on the packet side of a media stream connection.
- Ability to monitor QOS statistics for an established connection, or at least the accumulated statistics for each connection as it is taken down.
- Ability for the Media Gateway to autonomously report if the QOS on a given connection has fallen below a specified value.
- Ability to support circuit-to-circuit connections (the case of circuit-side fallback when the packet side is congested).
- Ability to support at least the circuit-side loopbacks needed for SS7 continuity testing.

11.3.2 Media stream transformations

This section has to do with the "steady-state" characteristics of the media streams.

- The Media Control protocol shall permit the payload type to be transmitted onward to the Media Gateway.
- The Media control protocol shall also allow the controller to pass explicit designation of the codec, packetization interval, and jitter buffer size for each media stream.
- In some cases, it will be necessary to specify the coding information on both sides of the connection.
- The protocol shall be able to specify whether silence suppression is to be used.
- The Media Gateway may be the point at which encryption is applied because the subscriber has requested confidentiality service across the packet network.
- The protocol shall be able to signal whether comfort noise is to be generated during silent periods.
- On the packet side, echo cancellation may be applied on a per-call basis.
- Typically much of this information must be specified at the same time that the connection is created. The Media control protocol shall allow for this possibility.
- However, it shall also be possible to change media handling instructions for an already-existing connection, in response, for instance, to an H.245 FlowControlCommand.
- A final requirement relating to media processing is that the Media Gateway control protocol shall support requests to initiate or terminate lawful interception of the content of a specified media stream.

11.3.3 Content Insertion

- The Media control protocol shall support the ability to request the playing of a specific tone or announcement at any point during the life of a connection.
- To reduce the messaging and other processing burden on the controller and improve response times, it is highly desirable that the Media Gateway control protocol go beyond requests to start and stop the playing of specified announcements or tones, to support at a minimum the specification of the conditions under which playout should stop. (This capability is a subset of the general requirement for the controller to be able to program the Media Gateway to detect and react to specific events associated with specific connections.)
- The Media control protocol shall also support the ability to request muting of a media stream in a specified direction.
- Finally, the Media control protocol shall support the ability for the controller to request the Media to insert and detect tones as required for SS7 continuity testing and other forms of testing.

11.3.4 Event Driven

- Even the narrowest definition of the scope of Media control requires the ability to instruct the Media Gateway to detect and act upon specified events. As a specific example, DTMF. The complete repertoire of events to be detected and actions to be taken as a result is for further study.

11.4 Other Requirements

11.4.1 Modularity and Extensibility

- Not all implementations may wish to support all of the possible extensions. Thus the protocol shall be modular, permitting light-weight implementations for specialized tasks where processing resources are constrained.
- The protocol shall provide the means whereby a controller can determine the capabilities supported by a particular Media Gateway.

11.4.2 Resource Management

- The Media control protocol shall provide the means for the controller to determine resource availability within the Media Gateway upon startup.
- Optionally, this capability may allow for queries during regular operation. The means by which remaining capacity is quantified is for further study.
- It shall be possible for the Media Gateway to indicate to the controller that it lacks sufficient resources to carry out a given command.
- It shall be possible for the controller to audit the commitment of resources to connections, to ensure that all commitments are valid.
- It shall be possible for the backup media gateway to rebuilt the connections for call set up by the primary from the media gateway information.
- It shall further be possible for the controller to order that specific resource assignments be cleared if it finds that they are invalid.
- It shall be possible for the Media Gateway to report changes in operational status of resources from in-service to out-of-service and vice versa.

11.4.3 Control Session Management

- The Media control protocol shall provide the means to start up and take down a control session between a specific controller and a specific Media Gateway. The ability for two controllers to make requests to the same Media Gateway at the same time is NOT a requirement.

- However, it shall be possible for a Media Gateway to establish a control session with an alternate controller if its primary controller becomes unavailable. It shall also be possible for a Media Gateway to establish an inactive control session with a standby controller. Control switchovers should occur without loss of connections already made within the Media Gateway. Means should be provided for the new controller to request a reset of all connections if it is unable to recover the existing connection states.
- It shall be possible for either the Media Gateway or the controller to detect loss of contact with the other party to the control session within a configurable time of the order of ten seconds or less. The appropriate actions to take upon detection of loss of contact are for further study.

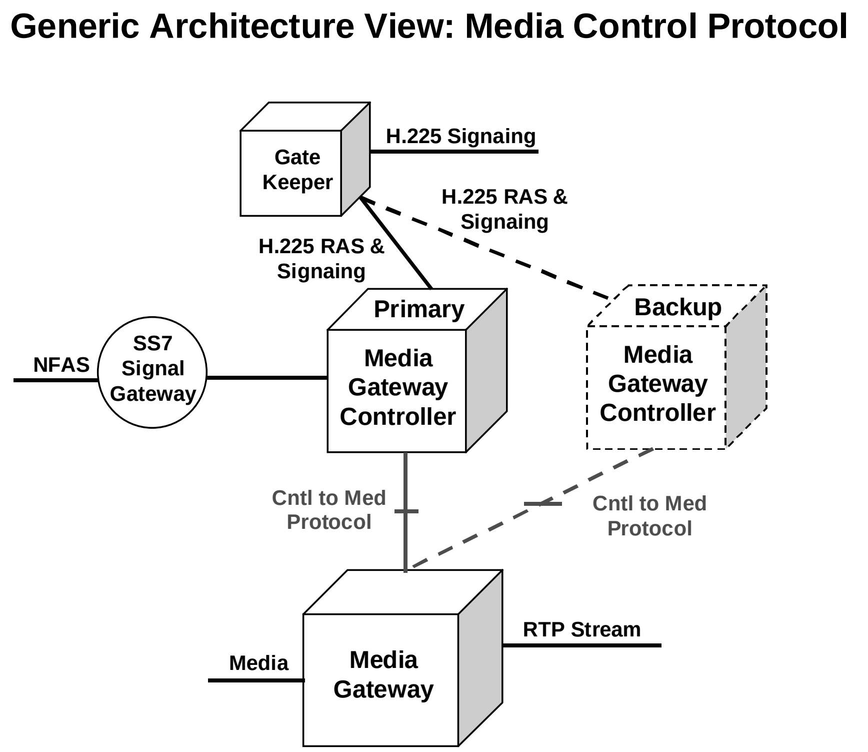
11.4.4 Control Session Security

The Media Gateway control protocol shall provide the means for mutual authentication at the start of a control session, and for preservation of the integrity and confidentiality of control messages once the session has started.

12. Annex 2 Additional Media Gateway Architecture Examples:

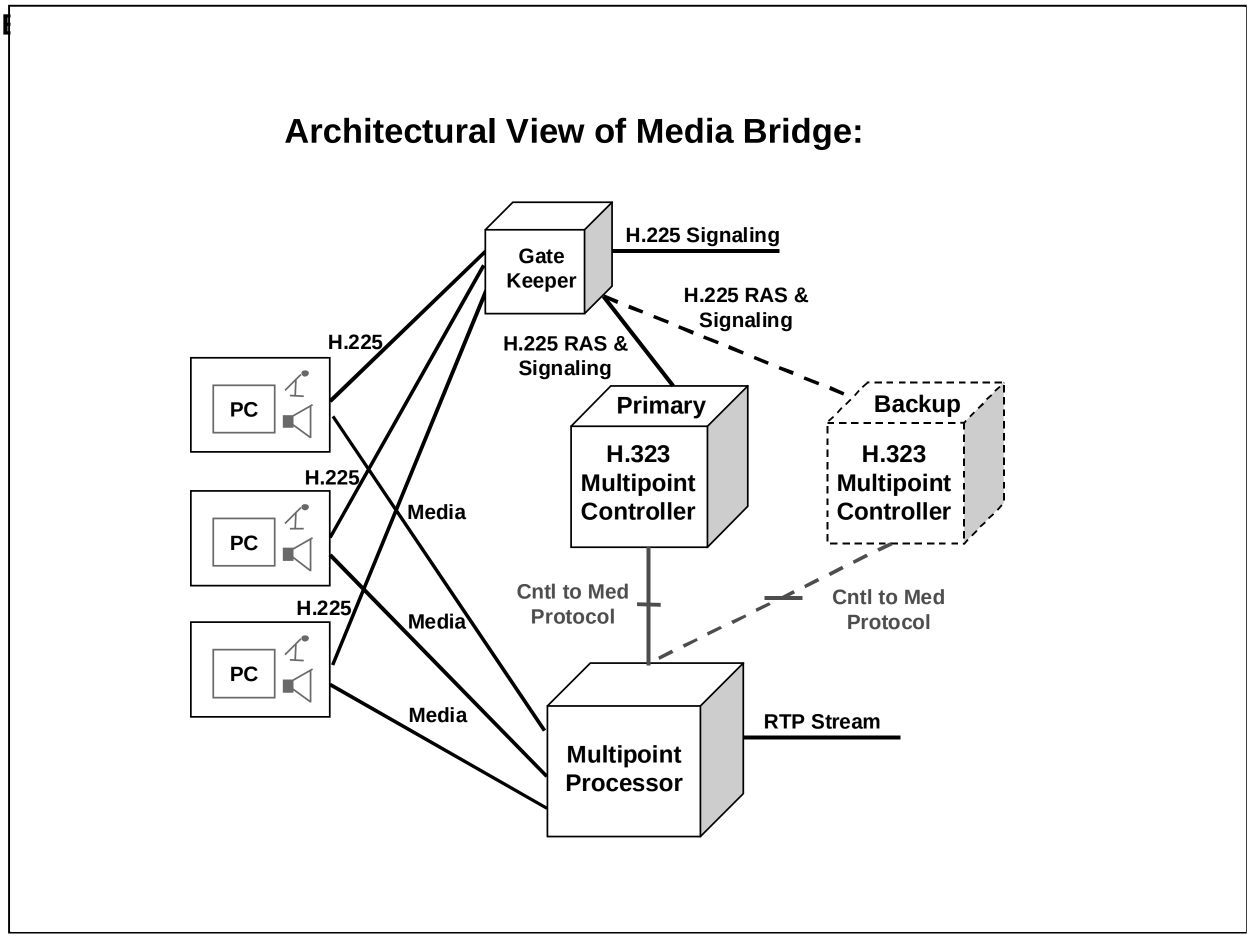
In this annex we describe some of the foreseen uses of the protocol. We address the following applications; trunk gateway, access gateway and conference bridge. Another not shown here would be a IVR System that would look a Trunk Gateway with only one terminal connected to it.

12.1 Trunk Gateway Example:



This example is a reprint of Figure 1 showing the application as trunk gateway. The media gateway is connected to a PCM trunk on one side and a packet (ATM or IP) connection on the other. The controller is connected with a gatekeeper on the H.323 side and a signaling gateway to the switched domain. The media gateway in use should have the following modules: transcoder, Echo canceller. Optional modules could be DTMF, IVR and Encryption.

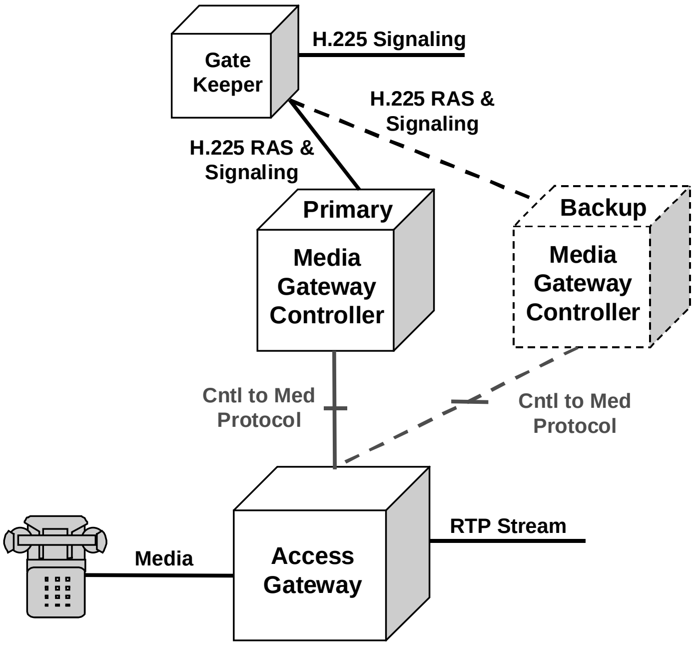
12.2 Media Bridge Example:



This exampe shows a conferencing application. The media gateway takes in this example the (H.323 defined name) Multipoint Processor,. the controller takes the name Multipoint Controller. Several terminals connect to the MP. The supported modules should be Bridge and optionally transcoder, DTMF and IVR..

12.3 Access Gateway Example:

Access Gateway Architecture View:



This example shows the access gateway. Depending the application it connects black telephones, BRI or a PRI interface to a packet line. The media gateway will have to pass in-band signaling to the controller, while out of band signaling may be sent over a different protocol or may be tunneled in the media control protocol. The following modules should be supported: transcoder, IVR, DTMF, Access or Channel Associated Signaling. Optional encryption.

13. Informative Annex 3 call flow example FastStart

The H.323 FastStart option allows an H.323 call to be set up in one round trip. The FastStart option circumvents the media negotiation by allowing the initiator to specify the RTP ports to use when the other side wants to send certain kinds of CODECs (e.g. G.711 on port 10000, G723.1 on port 10002, G.729 in port 10004).

The following example shows how the media control protocol is used to instruct the media gateway .

13.1 SCN originated call setup

The scenario assumes setup initiated from the SCN. (See Figure 8) This example is copied from the Tiphon draft document DTS 02002 v0.3.1. Below the picture the messages are named. In this example we focus in on some of the messages.

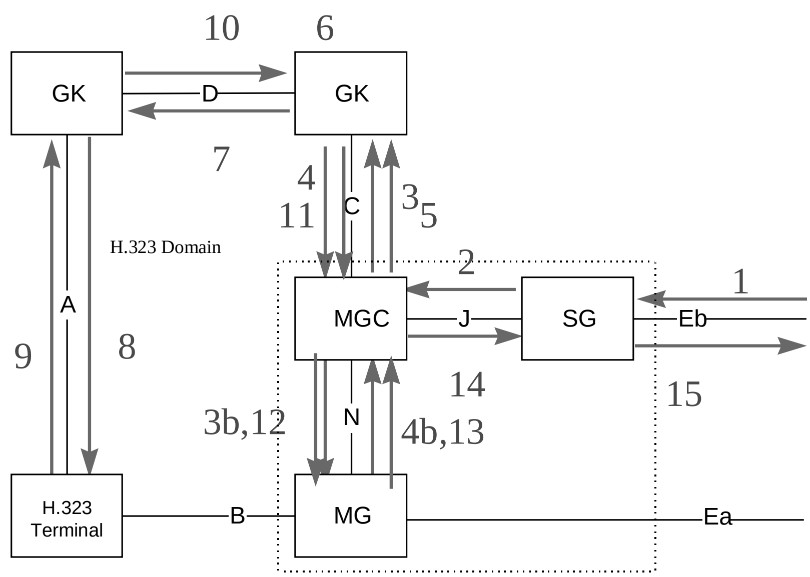


Figure 8. SCN originated call setup

1. ISUP Initial Address message
2. setup forwarded
3. admission request (RAS)
4. admission confirm
5. H.225 setup
6. map E.164 number to H.323 alias
7. H.225 (inter zone) setup to other GK
8. H.225 setup to Terminal (assume v2)
9. Terminal CONNECT

10,11. CONNECT forwarded

12 media gateway is told to forward the media stream (tells IP address, encoding) and trunk ID on PSTN side.

13 MG acks and gives its own RTP ports.

14 CONNECT

15 ISUP CON

at this point the call is connected end-to-end and media is streamed

Message 2 gives to the MGC the trunk circuit on which the connection will be sent to the switched circuit.
Upon receiving 2 the MGC will signal to the MG that it should prepare itself for the media. (Message 3b and 4b, not in the Tiphon example.)
Message 3b will instruct the MG to:

- accept data from the right trunk circuit, connect the appropriate transformations and fork the stream to the appropriate (in this case three) transcoders. Each transcoder sends its data to a nullResource. (we don't have a remote address to connect to.)
- Open up an RTP port for each of the supported resources (assuming availability of the appropriate resources) connect each port to an appropriate transcoder and other resources join these streams to eventually end to the appropriate trunk circuit.

Message 4b will acknowledge these creations and can give the numbers of the RTP ports used and can inform the MGC of the currently available codecs. The MG is now ready to stream media.
The connections now look like Figure 9 below.

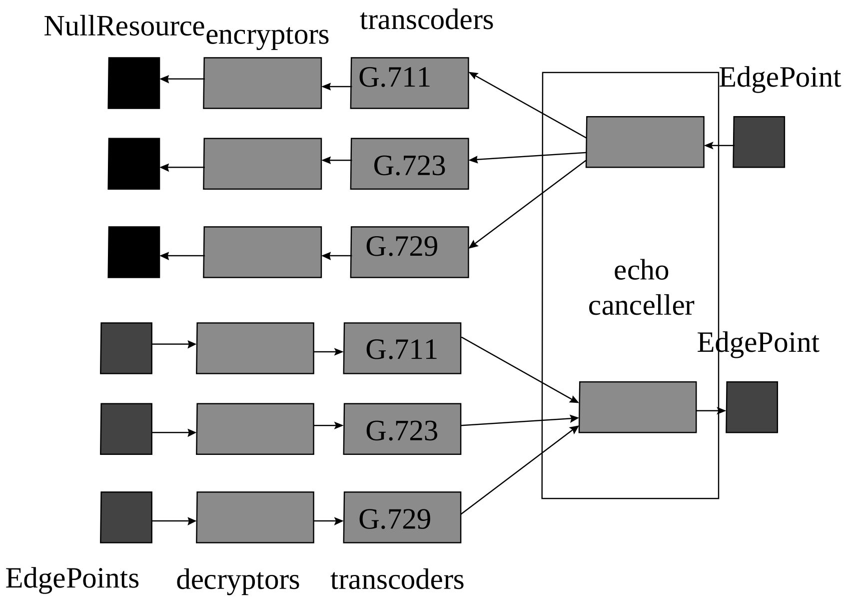


Figure 9. Media Gateway prepared

In Message 5 the MGC will signal the codecs it can send and the codecs it can receive along with their address and port number.
In Message 11 the MGC will receive the address it can send on, so it can instruct the MG to:

- connect the appropriate open output connected to nullResource to a real EdgePoint by adding a branch on the last interconnect to a newly created edgepoint.
- and subsequently removing the nullresource instances

The connection now looks like Figure 10.

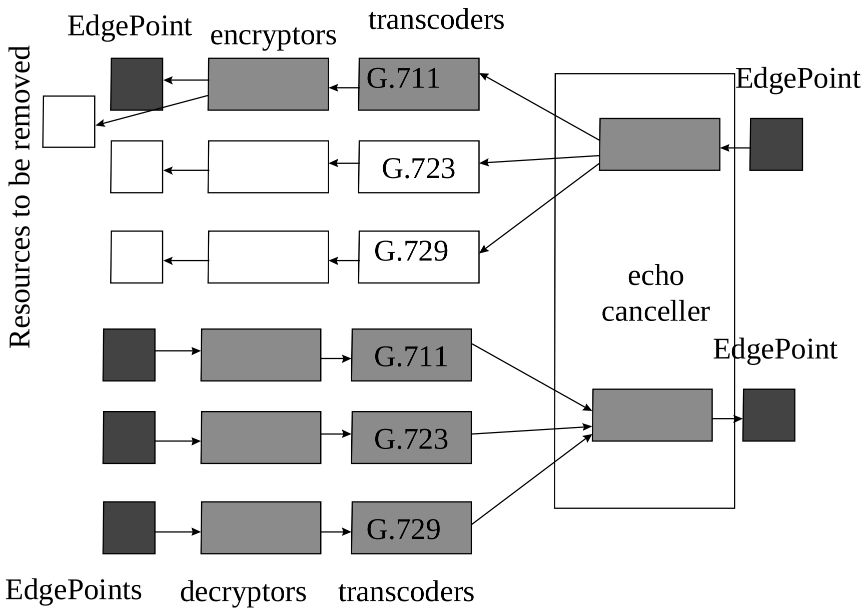


Figure 10. Reverse channel OK.

Of the RTP connections that are coming into the MG only one will be used. The MG will know which when packets start coming in on one of the RTP ports. At that moment the others can be removed. The MG could autonomously do this by using the Scripting Resource. A script could be downloaded to the MG that will catch an indication sent out by the EdgePoint that receives the first RTP packet Upon catching this indication, the script will remove the other edge points and thereby the branches until the join.

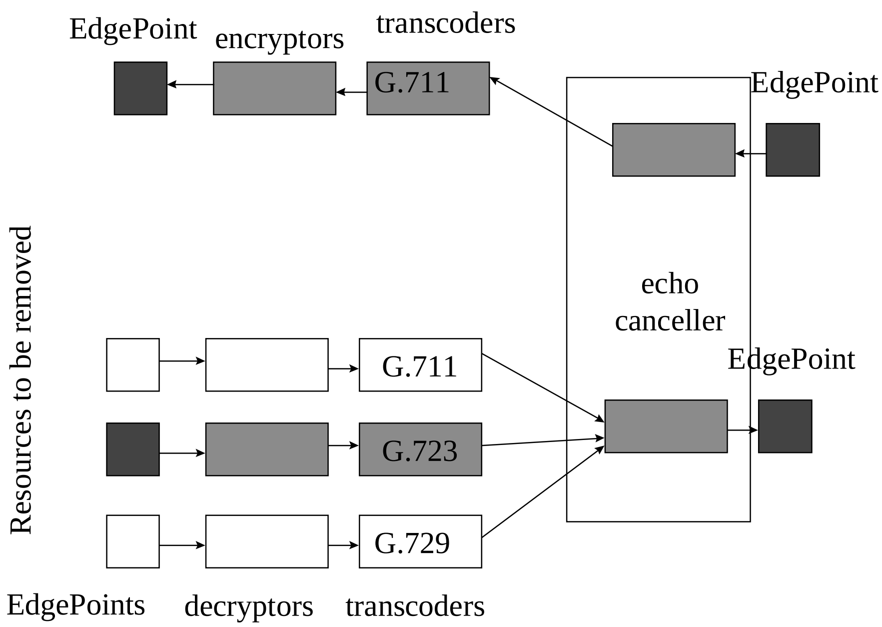


Figure 11. Script removes unused incoming streams.

14. Author's Address

Paul Sijben
Lucent Technologies
PO Box 18
1270 AA Huizen
the Netherlands
tel: +31 35 687 4774
mail: sijben@lucent.com

John Wachter
Mail: jwachter@lucent.com

Chia Li
Mail: chiali@lucent.com

Mike Buckley
Mail: mikebuckley@attmail.com

John Segers
Mail: jsegers@lucent.com

Tom Fritz
Mail: tefritz@lucent.com