Network Working Group   F.
Cuervo
Request for Comments: 3015   N.
Greene
Obsoletes: 2885, 2886   A.
Rayhan
Category: Standards Track   Nortel
Networks
                                                               C.
Huitema
                                                         Microsoft
Corporation
                                                               B.
Rosen

Marconi
                                                               J.
Segers
                                                         Lucent
Technologies
                                                         November
2000

                     Megaco Protocol Version 1.0

Status of this Memo

   This document specifies an Internet standards track protocol for
the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the
"Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is
unlimited.

Abstract

   This document defines the protocol used between elements of a
   physically decomposed multimedia gateway, i.e. a Media Gateway and
a
   Media Gateway Controller. The document is common text with ITU-T
   Recommendation H.248 and is a result of applying the changes in
RFC
   2886 to the text of RFC 2885.

   The protocol presented in this document meets the requirements for
a
   media gateway control protocol as presented in RFC 2805.

TABLE OF CONTENTS

1. SCOPE

   This document defines the protocol used between elements of a
   physically decomposed multimedia gateway. There are no functional
   differences from a system view between a decomposed gateway, with
   distributed sub-components potentially on more than one physical
   device, and a monolithic gateway such as described in H.246. This
   document does not define how gateways, multipoint control units or

interactive voice response units (IVRs) work. Instead it creates a

general framework that is suitable for these applications.

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

The protocol definition in this document is common text with ITU-T
Recommendation H.248. It meets the requirements documented in RFC
2805.

2. REFERENCES

2.1 Normative references

ATM Forum (1994): "User-Network Interface, Version 4.0".

ITU-T Recommendation H.225.0: "Call Signalling Protocols and Media
Stream Packetization for Packet Based Multimedia Communications
Systems".

ITU-T Recommendation H.235: "Security and encryption for H-Series
(H.323 and other H.245-based) multimedia terminals".

ITU-T Recommendation H.245: "Control Protocol for Multimedia
Communication".

ITU-T Recommendation H.323: "Packet Based Multimedia Communication
Systems".

ITU-T Recommendation I.363.1, "B-ISDN ATM Adaptation Layer
specification: Type 1 AAL".

ITU-T Recommendation I.363.2, "B-ISDN ATM Adaptation Layer
specification: Type 2 AAL".

ITU-T Recommendation I.363.5, "B-ISDN ATM Adaptation Layer
specification: Type 5 AAL".

ITU-T Recommendation I.363.5, "B-ISDN ATM Adaptation Layer
specification: Type 5 AAL".

ITU-T Recommendation I.366.1, "Segmentation and Reassembly Service
Specific Convergence Sublayer for the AAL type 2".

ITU-T Recommendation I.366.2, "AAL type 2 service specific
convergence sublayer for trunking".

ITU-T Recommendation I.371, "Traffic control and congestion
control
   in B-ISDN".

   ITU-T Recommendation Q.763, "Signalling System No. 7 - ISDN user
part
   formats and codes".

   ITU-T Recommendation Q.765, "Signalling System No. 7 - Application
   transport mechanism".

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

   ITU-T Recommendation Q.2630.1, "AAL Type 2 Signalling Protocol
   (Capability Set 1)".

   ITU-T Recommendation Q.2931, "Broadband Integrated Services
Digital
   Network (B-ISDN) - Digital Subscriber Signalling System No. 2 (DSS
2)
   - User-Network Interface (UNI) - Layer 3 specification for basic
   call/connection control".

   ITU-T Recommendation Q.2941.1, "Digital Subscriber Signalling
System
   No. 2 - Generic Identifier Transport".

   ITU-T Recommendation Q.2961, "Broadband integrated services
digital
   network (B-ISDN) - Digital subscriber signalling system no.2 (DSS
2)
   - additional traffic parameters".

   ITU-T Recommendation Q.2965.1, "Digital subscriber signalling
system
   No. 2 _ Support of Quality of Service classes."

   ITU-T Recommendation Q.2965.2, "Digital subscriber signalling
system
   No. 2 _ Signalling of individual Quality of Service parameters."

   ITU-T Recommendation Q.2961.2, "Digital subscriber signalling
system
   No. 2 - Additional traffic parameters: Support of ATM transfer
   capability in the broadband bearer capability information
element."

ITU-T Recommendation X.213, "Information technology - Open System
Interconnection - Network service definition plus Amendment 1
(08/1997), Addition of the Internet protocol address format
identifier".

ITU-T Recommendation V.76 (08/96), "Generic multiplexer using V.42
LAPM-based procedures".

ITU-T Recommendation X.680 (1997): "Information technology-
Abstract
Syntax Notation One (ASN.1): Specification of basic notation".

ITU-T Recommendation H.246 (1998), "Interworking of H-series
multimedia terminals with H-series multimedia terminals and
voice/voiceband terminals on GSTN and ISDN".

Rose, M. and D. Cass, "ISO Transport Service on top of the TCP,
Version 3", RFC 1006, May 1987.

Crocker, D. and P. Overell, "Augmented BNF for Syntax
Specifications:
ABNF", RFC 2234, November 1997.

Handley, M. and V. Jacobson, "SDP: Session Description Protocol",
RFC
2327, April 1998.

Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402,
November 1998.

Kent, S. and R. Atkinson, "IP Encapsulating Security Payload
(ESP)",
RFC 2406, November 1998.

2.2 Informative references

ITU-T Recommendation E.180/Q.35 (1998): "Technical characteristics
of
tones for the telephone service".

CCITT Recommendation G.711 (1988), "Pulse Code Modulation (PCM) of
voice frequencies".

ITU-T Recommendation H.221 (05/99),"Frame structure for a 64 to
1920
kbit/s channel in audiovisual teleservices".

ITU-T Recommendation H.223 (1996), "Multiplexing protocol for low
bit

rate multimedia communication".

Postel, J., "User Datagram Protocol", RFC 768, August 1980.

Postel, J., "Internet protocol", RFC 791, September 1981.

Postel, J., "TRANSMISSION CONTROL PROTOCOL", RFC 793, September
1981.

Simpson, W., "The Point-to-Point Protocol", RFC 1661, July 1994.

Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP:
A

Transport Protocol for Real-Time Applications", RFC 1889, January
1996.

Schulzrinne, H., "RTP Profile for Audio and Video Audio and Video
Conferences with Minimal Control", RFC 1890, January 1996.

Kent, S. and R. Atkinson, "Security Architecture for the Internet
Protocol", RFC 2401, November 1998.

Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6)
Specification", RFC 2460, December 1998.

Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP:
Session Initiation Protocol", RFC 2543, March 1999.

Greene, N., Ramalho, M. and B. Rosen, "Media Gateway control
protocol
architecture and requirements", RFC 2805, April 1999.

3. DEFINITIONS

Access Gateway: A type of gateway that provides a User to Network
Interface (UNI) such as ISDN.

Descriptor: A syntactic element of the protocol that groups
related
properties. For instance, the properties of a media flow on the
MG
can be set by the MGC by including the appropriate descriptor in a
command.

Media Gateway (MG): The media gateway converts media provided in
one
type of network to the format required in another type of network.
For example, a MG could terminate bearer channels from a switched
circuit network (e.g., DS0s) and media streams from a packet
network

(e.g., RTP streams in an IP network). This gateway may be capable
of
   processing audio, video and T.120 alone or in any combination, and
   will be capable of full duplex media translations. The MG may
also
   play audio/video messages and perform other IVR functions, or may
   perform media conferencing.

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

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

   Residential Gateway: A gateway that interworks an analogue line to
a
   packet network. A residential gateway typically contains one or
two
   analogue lines and is located at the customer premises.

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

   SCN NFAS Signalling Gateway: This function contains the SCN
   Signalling Interface that terminates SS7 or other signalling links
   where the call control channels are separated from bearer
channels.

   Stream: Bidirectional media or control flow received/sent by a
media
   gateway as part of a call or conference.

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

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

4. ABBREVIATIONS

   This recommendation defines the following terms.

   ATM   Asynchronous Transfer Mode
   CAS   Channel Associated Signalling

```
      DTMF        Dual Tone Multi-Frequency
      FAS         Facility Associated Signalling
      GSM         Global System for Mobile communications
      GW          GateWay
      IANA        Internet Assigned Numbers Authority
      IP          Internet Protocol
      ISUP        ISDN User Part
      IVR         Interactive Voice Response
      MG          Media Gateway
      MGC         Media Gateway Controller
      NFAS        Non-Facility Associated Signalling
      PRI         Primary Rate Interface
      PSTN        Public Switched Telephone Network
      QoS         Quality of Service
      RTP         Real-time Transport Protocol
      SCN         Switched Circuit Network
      SG          Signalling Gateway
      SS7         Signalling System No. 7
```

5. CONVENTIONS

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

6. CONNECTION MODEL

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

   A Termination sources and/or sinks one or more streams. In a
   multimedia conference, a Termination can be multimedia and sources or
   sinks multiple media streams. The media stream parameters, as well
   as modem, and bearer parameters are encapsulated within the
   Termination.

```
        +--------------------------------------------------------+
        |Media Gateway                                           |
        | +----------------------------------------------------+ |
        | |Context                              +------------+ | |
```

```
    | |                                    | Termination | |  |
    | |                                    |-------------| |  |
    | |    +-------------+       +->| SCN Bearer  |<---+->
    | |    | Termination |   +-----+  |  | Channel     | |  |
    | |    |-------------|   |     |---+  +-------------+ |  |
<-+--->| RTP Stream  |---|  *  |           | |
    | |    |             |   |     |---+  +-------------+ |  |
    | |    +-------------+   +-----+  |  | Termination | |  |
    | |                                    |  |-------------| |  |
    | |                              +->| SCN Bearer  |<---+->
    | |                                    |  Channel     | |  |
    | |                                    +-------------+ |  |
    |  +----------------------------------------------+  |
    |                                                       |
    |                                                       |
    |             +------------------------------+  |
    |             |Context                        |  |
    |  +-------------+  |                          +-------------+ | |
    |  | Termination |  |  +-----+    | Termination | | |
    |  |-------------|  |  |     |    |-------------| | |
<-+->| SCN Bearer  |  |  |  *  |------| SCN Bearer  |<---+->
    |  | Channel     |  |  |     |    |  Channel     | | |
    |  +-------------+  |  +-----+    +-------------+ | |
    |             +------------------------------+  |
    |                                                       |
    |                                                       |
    |  +------------------------------------------+  |
    |  |Context                                    |  |
    |  |  +-------------+  +-----+    +-------------+ | |
    |  |  | Termination |  |     |    | Termination | | |
    |  |  |-------------|  |     |    |-------------| | |
<-+--->| SCN Bearer  |---|  *  |------| SCN Bearer  |<---+->
    |  |  | Channel     |  |     |    |  Channel     | | |
    |  |  +-------------+  +-----+    +-------------+ | |
    |  +------------------------------------------+  |
    | _____ |
    +----------------------------------------------------+
```
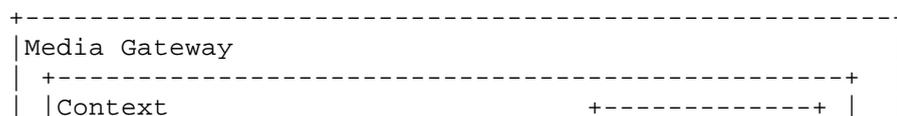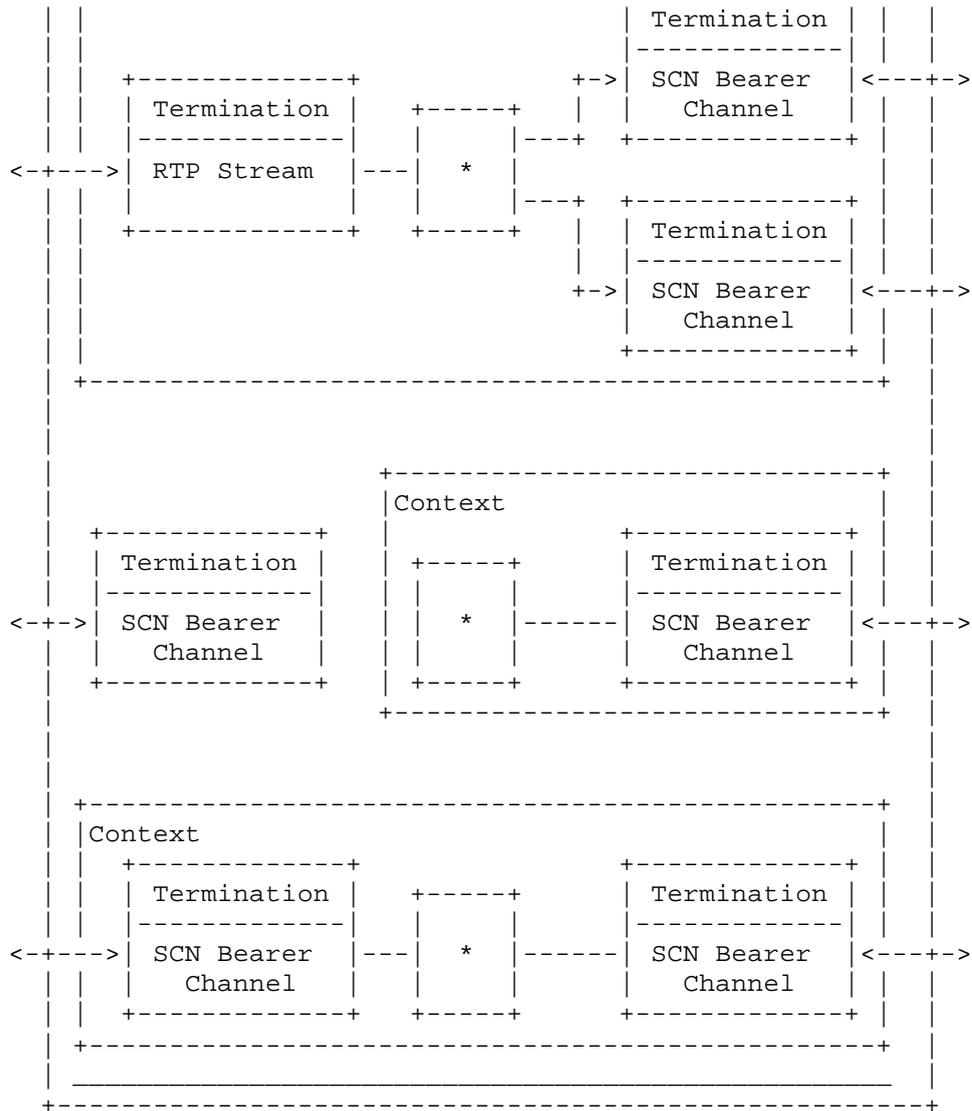
Figure 1: Example of H.248 Connection Model

   A Context is an association between a collection of Terminations.
   There is a special type of Context, the null Context, which
contains
   all Terminations that are not associated to any other Termination.
   For instance, in a decomposed access gateway, all idle lines are
   represented by Terminations in the null Context.

Figure 1 above is a graphical depiction of these concepts. The
diagram of Figure 1 gives several examples and is not meant to be
an
all-inclusive illustration. The asterisk box in each of the
Contexts
represents the logical association of Terminations implied by the
Context.

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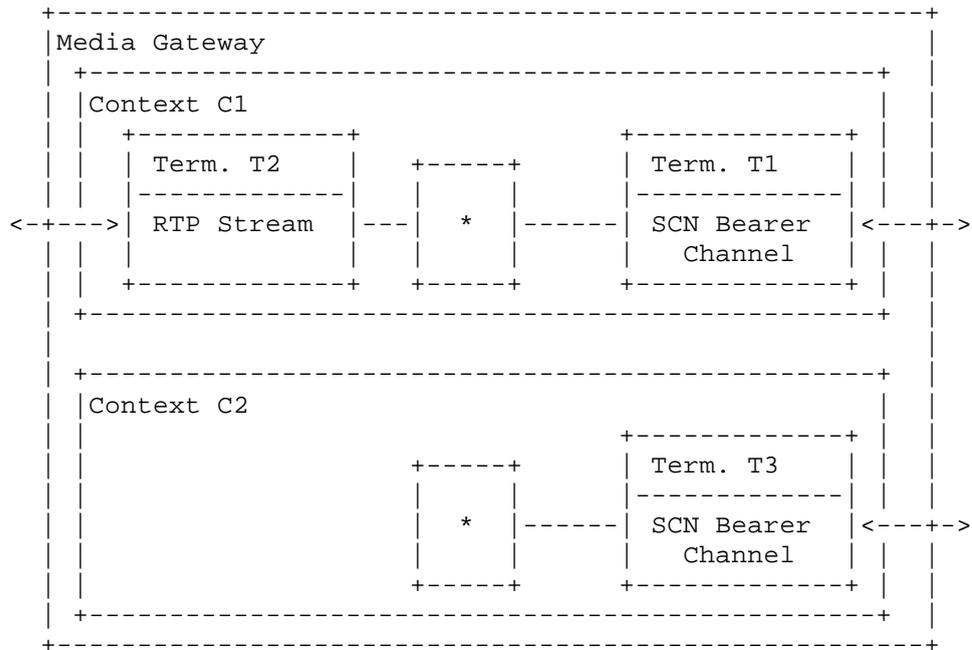in
T1 moving into Context C2, as shown below.

```
        +--------------------------------------------------+
        |Media Gateway                                     |
        | +----------------------------------------------+ |
        | |Context C1                                    | |
        | |  +------------+        +------------+ |  |
        | |  | Term. T2   |  +-----+        | Term. T1   | |  |
        | |  |------------|  |     |        |------------| |  |
    <-+--->| RTP Stream  |---|  *  |------| SCN Bearer  |<---+->
        | |  |            |  |     |        |   Channel   | |  |
        | |  +------------+  +-----+        +------------+ |  |
        | +----------------------------------------------+ |
        |                                                  |
        |                                                  |
        | +----------------------------------------------+ |
        | |Context C2                                    | |
        | |                                +------------+ |  |
        | |                  +-----+        | Term. T3   | |  |
        | |                  |     |        |------------| |  |
        | |                  |  *  |------| SCN Bearer  |<---+->
        | |                  |     |        |   Channel   | |  |
        | |                  +-----+        +------------+ |  |
        | +----------------------------------------------+ |
        +--------------------------------------------------+
```

   Figure 2: Example Call Waiting Scenario / Alerting Applied to T1

```
        +--------------------------------------------------+
        |Media Gateway                                     |
```

```
     |  +--------------------------------------------------+  |
     |  |Context C1                                        |  |
     |  |   +-------------+                                |  |
     |  |   | Term. T2    |      +-----+                   |  |
     |  |   |-------------|      |     |                   |  |
   <-+--->| RTP Stream  |---|  *  |                   |  |
     |  |   |             |      |     |                   |  |
     |  |   +-------------+      +-----+                   |  |
     |  +--------------------------------------------------+  |
     |                                                        |
     |  +--------------------------------------------------+  |
     |  |Context C2                                        |  |
     |  |   +-------------+                  +-------------+ |  |
     |  |   | Term. T1    |      +-----+      | Term. T3    | |  |
     |  |   |-------------|      |     |      |-------------| |  |
   <-+--->| SCN Bearer  |---|  *  |------| SCN Bearer  |<---+->
     |  |   |   Channel   |      |     |      |   Channel   | |  |
     |  |   +-------------+      +-----+      +-------------+ |  |
     |  +--------------------------------------------------+  |
     +--------------------------------------------------------+
```
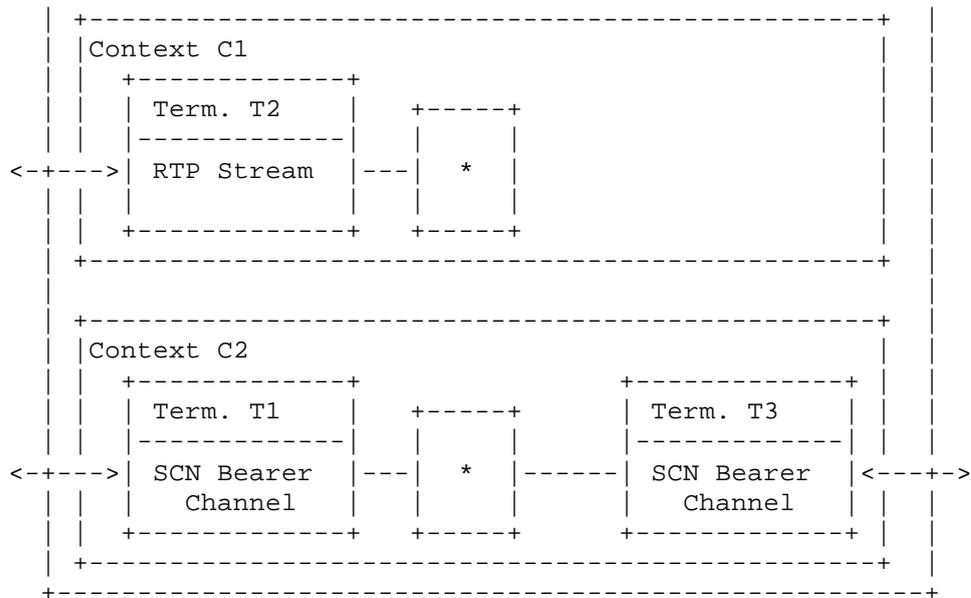
        Figure 3. Example Call Waiting Scenario / Answer by T1

6.1 Contexts

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

   There is a special Context called the null Context. It contains
   Terminations that are not associated to any other Termination.
   Terminations in the null Context can have their parameters
examined
   or modified, and may have events detected on them.

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

   The maximum number of Terminations in a Context is a MG property.
   Media gateways that offer only point-to-point connectivity might

allow at most two Terminations per Context. Media gateways that
support multipoint conferences might allow three or more
terminations
per Context.

6.1.1 Context Attributes and Descriptors

The attributes of Contexts are:

* ContextID.

* The topology (who hears/sees whom).
   The topology of a Context describes the flow of media between
the
   Terminations within a Context. In contrast, the mode of a
   Termination (send/receive/_) describes the flow of the media at
   the ingress/egress of the media gateway.

* The priority is used for a context in order to provide the MG
with
   information about a certain precedence handling for a context.
The
   MGC can also use the priority to control autonomously the
traffic
   precedence in the MG in a smooth way in certain situations
(e.g.
   restart), when a lot of contexts must be handled
simultaneously.

* An indicator for an emergency call is also provided to allow a
   preference handling in the MG.

6.1.2 Creating, Deleting and Modifying Contexts

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

6.2 Terminations

A Termination is a logical entity on a MG that sources and/or
sinks
media and/or control streams. A Termination is described by a
number
of characterizing Properties, which are grouped in a set of
Descriptors that are included in commands. Terminations have
unique
identities (TerminationIDs), assigned by the MG at the time of
their
creation.

   Terminations representing physical entities 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, such as RTP
   flows, would usually exist only for the duration of their use.

   Ephemeral Terminations are created by means of an Add command.
They
   are destroyed by means of a Subtract command. In contrast, when a
   physical Termination is Added to or Subtracted from a Context, it
is
   taken from or to the null Context, respectively.

   Terminations may have signals applied to them. Signals are MG
   generated media streams such as tones and announcements as well as
   line signals such as hookswitch. Terminations may be programmed
to
   detect Events, the occurrence of which can trigger notification
   messages to the MGC, or action by the MG. Statistics may be
   accumulated on a Termination. Statistics are reported to the MGC
   upon request (by means of the AuditValue command, see section
7.2.5)
   and when the Termination is taken out of the call it is in.

   Multimedia gateways may process multiplexed media streams. For
   example, Recommendation H.221 describes a frame structure for
   multiple media streams multiplexed on a number of digital 64
kbit/s
   channels. Such a case is handled in the connection model in the
   following way. For every bearer channel that carries part of the
   multiplexed streams, there is a Termination. The Terminations
that
   source/sink the digital channels are connected to a separate
   Termination called the multiplexing Termination. This Termination
   describes the multiplex used (e.g. how the H.221 frames are
carried
   over the digital channels used). The MuxDescriptor is used to
this
   end. If multiple media are carried, this Termination contains
   multiple StreamDescriptors. The media streams can be associated
with
   streams sourced/sunk by other Terminations in the Context.

   Terminations may be created which represent multiplexed bearers,
such
   as an ATM AAL Type 2 bearer. When a new multiplexed bearer is to
be
   created, an ephemeral termination is created in a context
established
   for this purpose. When the termination is subtracted, the
   multiplexed bearer is destroyed.

6.2.1 Termination Dynamics

   The protocol can be used to create new Terminations and to modify
   property values of existing Terminations. These modifications
   include the possibility of adding or removing events and/or
signals.
   The Termination properties, and events and signals are described
in

   the ensuing sections. An MGC can only release/modify terminations
and
   the resources that the termination represents which it has
previously
   seized via, e.g., the Add command.

6.2.2 TerminationIDs

   Terminations are referenced by a TerminationID, which is an
arbitrary
   schema chosen by the MG.

   TerminationIDs of physical Terminations are provisioned in the
Media
   Gateway. The TerminationIDs may be chosen to have structure. For
   instance, a TerminationID may consist of trunk group and a trunk
   within the group.

   A wildcarding mechanism using two types of wildcards can be used
with
   TerminationIDs. The two wildcards are ALL and CHOOSE. The former
is
   used to address multiple Terminations at once, while the latter is
   used to indicate to a media gateway that it must select a
Termination
   satisfying the partially specified TerminationID. This allows,
for
   instance, that a MGC instructs a MG to choose a circuit within a
   trunk group.

   When ALL is used in the TerminationID of a command, the effect is
   identical to repeating the command with each of the matching
   TerminationIDs. Since each of these commands may generate a
   response, the size of the entire response may be large. If
   individual responses are not required, a wildcard response may be
   requested. In such a case, a single response is generated, which
   contains the UNION of all of the individual responses which
otherwise
   would have been generated, with duplicate values suppressed. For
   instance, given a Termination Ta with properties p1=a, p2=b and

Termination Tb with properties p2=c, p3=d, a UNION response would
consist of a wildcarded TerminationId and the sequence of
properties
p1=a, p2=b,c and p3=d. Wildcard response may be particularly
useful
in the Audit commands.

The encoding of the wildcarding mechanism is detailed in Annexes A
and B.

6.2.3 Packages

Different types of gateways may implement Terminations that have
widely differing characteristics. Variations in Terminations are
accommodated in the protocol by allowing Terminations to have
optional Properties, Events, Signals and Statistics implemented by
MGs.

In order to achieve MG/MGC interoperability, such options are
grouped
into Packages, and a Termination realizes a set of such Packages.
More information on definition of packages can be found in section
12. An MGC can audit a Termination to determine which Packages it
realizes.

Properties, Events, Signals and Statistics defined in Packages, as
well as parameters to them, are referenced by identifiers (Ids).
Identifiers are scoped. For each package, PropertyIds, EventIds,
SignalIds, StatisticsIds and ParameterIds have unique name spaces
and
the same identifier may be used in each of them. Two PropertyIds
in
different packages may also have the same identifier, etc.

6.2.4 Termination Properties and Descriptors

Terminations have properties. The properties have unique
PropertyIDs. Most properties have default values, which are
explicitly defined in this standard or in a package (see Section
12)
or set by provisioning. If not provisioned otherwise, all
descriptors except TerminationState and LocalControl default to
empty/"no value" when a Termination is first created or returned
to
the null Context. The default contents of the two exceptions are
described in sections 7.1.5 and 7.1.7.

There are a number of common properties for Terminations and
properties specific to media streams. The common properties are
also

called the termination state properties. For each media stream,
there are local properties and properties of the received and
transmitted flows.

Properties not included in the base protocol are defined in
Packages.
These properties are referred to by a name consisting of the
PackageName and a PropertyId. Most properties have default values
described in the Package description. Properties may be read- only
or
read/write. The possible values of a property may be audited, as
can
their current values. For properties that are read/write, the MGC
can set their values. A property may be declared as "Global"
which
has a single value shared by all terminations realizing the
package.
Related properties are grouped into descriptors for convenience.

When a Termination is Added to a Context, the value of its
read/write
properties can be set by including the appropriate descriptors as
parameters to the Add command. Properties not mentioned in the
command retain their prior values. Similarly, a property of a
Termination in a Context may have its value changed by the Modify
command. Properties not mentioned in the Modify command retain
their
prior values. Properties may also have their values changed when a

Termination is moved from one Context to another as a result of a
Move command. In some cases, descriptors are returned as output
from
a command.

The following table lists all of the possible Descriptors and
their
use. Not all descriptors are legal as input or output parameters
to
every command.

```
   +-----------------+--------------------------------------------------+
   | Descriptor Name | Description                                      |
   |-----------------|--------------------------------------------------|
   | Modem           | Identifies modem type and properties             |
   |                 | when applicable.                                 |
   | Mux             | Describes multiplex type for multimedia          |
   |                 | terminations (e.g. H.221, H.223, H.225.0)        |
   |                 | and Terminations forming the input mux.          |
   | Media           | A list of media stream specifications            |
   |                 | (see 7.1.4).                                     |
   | TerminationState | Properties of a Termination (which can be       |
```

		defined in Packages) that are not stream
		specific.
Stream	A list of remote/local/localControl
		descriptors for a single stream.
Local	Contains properties that specify the media
		flows that the MG receives from the remote
		entity.
Remote	Contains properties that specify the media
		flows that the MG sends to the remote entity.
LocalControl	Contains properties (which can be defined in
		packages) that are of interest between the MG
		and the MGC.
Events	Describes events to be detected by the MG and
		what to do when an event is detected.
EventBuffer	Describes events to be detected by the MG
		when Event Buffering is active.
Signals	Describes signals and/or actions to be
		applied (e.g. Busy Tone) to the Terminations.
Audit	In Audit commands, identifies which
		information is desired.
Packages	In AuditValue, returns a list of Packages
		realized by Termination.
DigitMap	Defines patterns against which sequences of a
		specified set of events are to be matched so
		they can be reported as a group rather than
		singly.
ServiceChange	In ServiceChange, what, why service change
		occurred, etc.
ObservedEvents	In Notify or AuditValue, report of events
		observed.

```
   | Statistics          | In Subtract and Audit, Report of Statistics
|
   |                     | kept on a Termination.
|
   +---------------------------------------------------------------
-+
```

6.2.5 Root Termination

   Occasionally, a command must refer to the entire gateway, rather
than
   a termination within it. A special TerminationID, "Root" is
reserved
   for this purpose. Packages may be defined on Root. Root thus may
   have properties, events and statistics (signals are not
appropriate
   for root). Accordingly, the root TerminationID may appear in:

   * a Modify command - to change a property or set an event
   * a Notify command - to report an event
   * an AuditValue return - to examine the values of properties and
      statistics implemented on root
   * an AuditCapability - to determine what properties of root are
      implemented
   * a ServiceChange - to declare the gateway in or out of service.

   Any other use of the root TerminationID is an error.

7. COMMANDS

   The protocol provides commands for manipulating the logical
entities
   of the protocol connection model, Contexts and Terminations.

   Commands provide control at the finest level of granularity
supported
   by the protocol. For example, Commands exist to add Terminations
to
   a Context, modify Terminations, subtract Terminations from a
Context,
   and audit properties of Contexts or Terminations. Commands provide
   for complete control of the properties of Contexts and
Terminations.
   This includes specifying which events a Termination is to report,
   which signals/actions are to be applied to a Termination and
   specifying the topology of a Context (who hears/sees whom).

   Most commands are for the specific use of the Media Gateway

Controller as command initiator in controlling Media Gateways as
command responders. The exceptions are the Notify and
ServiceChange
commands: Notify is sent from Media Gateway to Media Gateway
Controller, and ServiceChange may be sent by either entity. Below
is
an overview of the commands; they are explained in more detail in
section 7.2.

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

2. Modify. The Modify command modifies the properties, events and
   signals of a termination.

3. Subtract. The Subtract command disconnects a Termination from
its
   Context and returns statistics on the Termination's
participation
   in the Context. The Subtract command on the last Termination
in a
   Context deletes the Context.

4. Move. The Move command atomically moves a Termination to
another
   context.

5. AuditValue. The AuditValue command returns the current state of
   properties, events, signals and statistics of Terminations.

6. AuditCapabilities. The AuditCapabilities command returns all
the
   possible values for Termination properties, events and signals
   allowed by the Media Gateway.

7. Notify. The Notify command allows the Media Gateway to inform
the
   Media Gateway Controller of the occurrence of events in the
Media
   Gateway.

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

to announce its availability to an MGC (registration), and to
notify the MGC of impending or completed restart of the MG. The
MGC may announce a handover to the MG by sending it a
ServiceChange command. The MGC may also use ServiceChange to
instruct the MG to take a Termination or group of Terminations
in
or out of service.

   These commands are detailed in sections 7.2.1 through 7.2.8

7.1 Descriptors

   The parameters to a command are termed Descriptors. A Descriptor
   consists of a name and a list of items. Some items may have
values.
   Many Commands share common Descriptors. This subsection
enumerates
   these Descriptors. Descriptors may be returned as output from a
   command. In any such return of descriptor contents, an empty
   descriptor is represented by its name unaccompanied by any list.
   Parameters and parameter usage specific to a given Command type
are
   described in the subsection that describes the Command.

7.1.1 Specifying Parameters

   Command parameters are structured into a number of descriptors. In
   general, the text format of descriptors is
   DescriptorName={parm=value, parm=value_.}.

   Parameters may be fully specified, over-specified or under-
specified:

   1. Fully specified parameters have a single, unambiguous value
that
      the command initiator is instructing the command responder to
use
      for the specified parameter.

   2. Under-specified parameters, using the CHOOSE value, allow the
      command responder to choose any value it can support.

   3. Over-specified parameters have a list of potential values. The
      list order specifies the command initiator's order of
preference
      of selection. The command responder chooses one value from the
      offered list and returns that value to the command initiator.

   If a required descriptor other than the Audit descriptor is
   unspecified (i.e., entirely absent) from a command, the previous
   values set in that descriptor for that termination, if any, are

retained. A missing Audit descriptor is equivalent to an empty
Audit
   Descriptor. The behavior of the MG with respect to unspecified
   parameters within a descriptor varies with the descriptor
concerned,
   as indicated in succeeding sections. Whenever a parameter is
   underspecified or overspecified, the descriptor containing the
value
   chosen by the responder is included as output from the command.

   Each command specifies the TerminationId the command operates on.
   This TerminationId may be "wildcarded". When the TerminationId of
a
   command is wildcarded, the effect shall be as if the command was
   repeated with each of the TerminationIds matched.

7.1.2 Modem Descriptor

   The Modem descriptor specifies the modem type and parameters, if
any,
   required for use in e.g. H.324 and text conversation. The
descriptor
   includes the following modem types: V.18, V.22, V.22bis, V.32,
   V.32bis, V.34, V.90, V.91, Synchronous ISDN, and allows for
   extensions. By default, no modem descriptor is present in a
   Termination.

7.1.3 Multiplex Descriptor

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

   * H.221
   * H.223,
   * H.226,
   * V.76,
   * Possible Extensions

   and a set of TerminationIDs representing the multiplexed inputs,
in
   order. For example:

        Mux = H.221{ MyT3/1/2, MyT3/2/13, MyT3/3/6, MyT3/21/22}

7.1.4 Media Descriptor

The Media Descriptor specifies the parameters for all the media
streams. These parameters are structured into two descriptors, a
Termination State Descriptor, which specifies the properties of a
termination that are not stream dependent, and one or more Stream
Descriptors each of which describes a single media stream.

A stream is identified by a StreamID. The StreamID is used to
link
the streams in a Context that belong together. Multiple streams
exiting a termination shall be synchronized with each other.
Within
the Stream Descriptor, there are up to three subsidiary
descriptors,
LocalControl, Local, and Remote. The relationship between these
descriptors is thus:

```
Media Descriptor
      TerminationStateDescriptor
      Stream Descriptor
               LocalControl Descriptor
               Local Descriptor
               Remote Descriptor
```

As a convenience a LocalControl, Local, or Remote descriptor may
be
included in the Media Descriptor without an enclosing Stream
descriptor. In this case, the StreamID is assumed to be 1.

7.1.5 Termination State Descriptor

The Termination State Descriptor contains the ServiceStates
property,
the EventBufferControl property and properties of a termination
(defined in Packages) that are not stream specific.

The ServiceStates property describes the overall state of the
termination (not stream-specific). A Termination can be in one of
the following states: "test", "out of service", or "in service".
The
"test" state indicates that the termination is being tested. The
state "out of service" indicates that the termination cannot be
used

for traffic. The state "in service" indicates that a termination
can
be used or is being used for normal traffic. "in service" is the
default state.

Values assigned to Properties may be simple values
(integer/string/enumeration) or may be underspecified, where more
than one value is supplied and the MG may make a choice:

*   Alternative Values: multiple values in a list, one of which
must
        be selected
    *   Ranges: minimum and maximum values, any value between min and
max
        must be selected, boundary values included
    *   Greater Than/Less Than: value must be greater/less than
specified
        value
    *   CHOOSE Wildcard: the MG chooses from the allowed values for the
        property

    The EventBufferControl property specifies whether events are
    buffered following detection of an event in the Events Descriptor,
or
    processed immediately. See section 7.1.9 for details.

7.1.6 Stream Descriptor

    A Stream descriptor specifies the parameters of a single bi-
    directional stream. These parameters are structured into three
    descriptors: one that contains termination properties specific to
a
    stream and one each for local and remote flows. The Stream
Descriptor
    includes a StreamID which identifies the stream. Streams are
created
    by specifying a new StreamID on one of the terminations in a
Context.
    A stream is deleted by setting empty Local and Remote descriptors
for
    the stream with ReserveGroup and ReserveValue in LocalControl set
to
    "false" on all terminations in the context that previously
supported
    that stream.

    StreamIDs are of local significance between MGC and MG and they
are
    assigned by the MGC. Within a context, StreamID is a means by
which
    to indicate which media flows are interconnected: streams with
the
    same StreamID are connected.

    If a termination is moved from one context to another, the effect
on
    the context to which the termination is moved is the same as in
the
    case that a new termination were added with the same StreamIDs as
the
    moved termination.

7.1.7 LocalControl Descriptor

   The LocalControl Descriptor contains the Mode property, the
   ReserveGroup and ReserveValue properties and properties of a
   termination (defined in Packages) that are stream specific, and are
   of interest between the MG and the MGC. Values of properties may be
   underspecified as in section 7.1.1.

   The allowed values for the mode property are send-only, receive-only,
   send/receive, inactive and loop-back. "Send" and "receive" are with
   respect to the exterior of the context, so that, for example, a
   stream set to mode=sendonly does not pass received media into the
   context. Signals and Events are not affected by mode.

   The boolean-valued Reserve properties, ReserveValue and ReserveGroup,
   of a Termination indicate what the MG is expected to do when it
   receives a local and/or remote descriptor.

   If the value of a Reserve property is True, the MG SHALL reserve
   resources for all alternatives specified in the local and/or remote
   descriptors for which it currently has resources available. It SHALL
   respond with the alternatives for which it reserves resources. If it
   cannot not support any of the alternatives, it SHALL respond with a
   reply to the MGC that contains empty local and/or remote
   descriptors.

   If the value of a Reserve property is False, the MG SHALL choose one
   of the alternatives specified in the local descriptor (if present)
   and one of the alternatives specified in the remote descriptor (if
   present). If the MG has not yet reserved resources to support the
   selected alternative, it SHALL reserve the resources. If, on the
   other hand, it already reserved resources for the Termination
   addressed (because of a prior exchange with ReserveValue and/or
   ReserveGroup equal to True), it SHALL release any excess resources it
   reserved previously. Finally, the MG shall send a reply to the MGC
   containing the alternatives for the local and/or remote descriptor
   that it selected. If the MG does not have sufficient resources to
   support any of the alternatives specified, is SHALL respond with
   error 510 (insufficient resources).

   The default value of ReserveValue and ReserveGroup is False. More

information on the use of the two Reserve properties is provided in
section 7.1.8.

A new setting of the LocalControl Descriptor completely replaces the
previous setting of that descriptor in the MG. Thus to retain
information from the previous setting the MGC must include that
information in the new setting. If the MGC wishes to delete some

information from the existing descriptor, it merely resends the
descriptor (in a Modify command) with the unwanted information
stripped out.

7.1.8 Local and Remote Descriptors

The MGC uses Local and Remote descriptors to reserve and commit MG
resources for media decoding and encoding for the given Stream(s) and
Termination to which they apply. The MG includes these descriptors
in its response to indicate what it is actually prepared to support.
The MG SHALL include additional properties and their values in its
response if these properties are mandatory yet not present in the
requests made by the MGC (e.g., by specifying detailed video encoding
parameters where the MGC only specified the payload type).

Local refers to the media received by the MG and Remote refers to the
media sent by the MG.

When text encoding the protocol, the descriptors consist of session
descriptions as defined in SDP (RFC2327). In session descriptions
sent from the MGC to the MG, the following exceptions to the syntax
of RFC 2327 are allowed:

* the "s=", "t=" and "o=" lines are optional,
* the use of CHOOSE is allowed in place of a single parameter value,
   and
* the use of alternatives is allowed in place of a single parameter
   value.

When multiple session descriptions are provided in one descriptor,
the "v=" lines are required as delimiters; otherwise they are
optional in session descriptions sent to the MG. Implementations

shall accept session descriptions that are fully conformant to
RFC2327. When binary encoding the protocol the descriptor consists
of
groups of properties (tag-value pairs) as specified in Annex C.
Each
such group may contain the parameters of a session description.

Below, the semantics of the local and remote descriptors are
specified in detail. The specification consists of two parts.
The
first part specifies the interpretation of the contents of the
descriptor. The second part specifies the actions the MG must
take
upon receiving the local and remote descriptors. The actions to
be
taken by the MG depend on the values of the ReserveValue and
ReserveGroup properties of the LocalControl descriptor.

Either the local or the remote descriptor or both may be

* unspecified (i.e., absent),
* empty,
* underspecified through use of CHOOSE in a property value,
* fully specified, or
* overspecified through presentation of multiple groups of
  properties and possibly multiple property values in one or more
of
  these groups.

Where the descriptors have been passed from the MGC to the MG,
they
are interpreted according to the rules given in section 7.1.1,
with
the following additional comments for clarification:

(a) An unspecified Local or Remote descriptor is considered to be
a
missing mandatory parameter. It requires the MG to use whatever
was
last specified for that descriptor. It is possible that there was
no
previously-specified value, in which case the descriptor concerned
is
ignored in further processing of the command.

(b) An empty Local (Remote) descriptor in a message from the MGC
signifies a request to release any resources reserved for the
media

flow received (sent).

   (c) If multiple groups of properties are present in a Local or
Remote
   descriptor or multiple values within a group, the order of
preference
   is descending.

   (d) Underspecified or overspecified properties within a group of
   properties sent by the MGC are requests for the MG to choose one
or
   more values which it can support for each of those properties. In
   case of an overspecified property, the list of values is in
   descending order of preference.

   Subject to the above rules, subsequent action depends on the
values
   of the ReserveValue and ReserveGroup properties in LocalControl.

   If ReserveGroup is true, the MG reserves the resources required to
   support any of the requested property group alternatives that it
can
   currently support. If ReserveValue is true, the MG reserves the
   resources required to support any of the requested property value
   alternatives that it can currently support.

   NOTE - If a Local or Remote descriptor contains multiple groups
of
   properties, and ReserveGroup is true, then the MG is requested to
   reserve resources so that it can decode or encode the media stream
   according to any of the alternatives. For instance, if the Local
   descriptor contains two groups of properties, one specifying

   packetized G.711 A-law audio and the other G.723.1 audio, the MG
   reserves resources so that it can decode one audio stream encoded
in
   either G.711 A-law format or G.723.1 format. The MG does not have
to
   reserve resources to decode two audio streams simultaneously, one
   encoded in G.711 A-law and one in G.723.1. The intention for the
use
   of ReserveValue is analogous.

   If ReserveGroup is true or ReserveValue is true, then the
following
   rules apply.

   * If the MG has insufficient resources to support all
alternatives
      requested by the MGC and the MGC requested resources in both
Local
      and Remote, the MG should reserve resources to support at least

one alternative each within Local and Remote.

   * If the MG has insufficient resources to support at least one
      alternative within a Local (Remote) descriptor received from
the
      MGC, it shall return an empty Local (Remote) in response.

   * In its response to the MGC, when the MGC included Local and
Remote
      descriptors, the MG SHALL include Local and Remote descriptors
for
      all groups of properties and property values it reserved
resources
      for. If the MG is incapable of supporting at least one of the
      alternatives within the Local (Remote) descriptor received from
      the MGC, it SHALL return an empty Local (Remote) descriptor.

   * If the Mode property of the LocalControl descriptor is
RecvOnly,
      SendRecv, or Loopback, the MG must be prepared to receive media
      encoded according to any of the alternatives included in its
      response to the MGC.

   * If ReserveGroup is False and ReserveValue is false, then the MG
      SHOULD apply the following rules to resolve Local and Remote to
a
      single alternative each:

   * The MG chooses the first alternative in Local for which it is
able
      to support at least one alternative in Remote.

   * If the MG is unable to support at least one Local and one
Remote
      alternative, it returns Error 510 (Insufficient Resources).

   * The MG returns its selected alternative in each of Local and
      Remote.

   A new setting of a Local or Remote Descriptor completely replaces
the
   previous setting of that descriptor in the MG. Thus to retain
   information from the previous setting the MGC must include that
   information in the new setting. If the MGC wishes to delete some

   information from the existing descriptor, it merely resends the
   descriptor (in a Modify command) with the unwanted information
   stripped out.

7.1.9 Events Descriptor

   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 test results, and on-hook and off-
hook
transitions.

Each event in the descriptor contains the Event name, an optional
streamID, an optional KeepActive flag, and optional parameters.
The
Event name consists of a Package Name (where the event is defined)
and an EventID. The ALL wildcard may be used for the EventID,
indicating that all events from the specified package have to be
detected. The default streamID is 0, indicating that the event to
be
detected is not related to a particular media stream. Events can
have parameters. This allows a single event description to have
some
variation in meaning without creating large numbers of individual
events. Further event parameters are defined in the package.

If a digit map completion event is present or implied in the
EventsDescriptor, the EventDM parameter is used to carry either
the
name or the value of the associated digit map. See section 7.1.14
for further details.

When an event is processed against the contents of an active
Events
descriptor and found to be present in that descriptor
("recognized"),
the default action of the MG is to send a Notify command to the
MG.
Notification may be deferred if the event is absorbed into the
current dial string of an active digit map (see section 7.1.14).
Any
other action is for further study. Moreover, event recognition
may
cause currently active signals to stop, or may cause the current
Events and/or Signals descriptor to be replaced, as described at
the
end of this section.

If the value of the EventBufferControl property equals LockStep,
following detection of such an event, normal handling of events is
suspended. Any event which is subsequently detected and occurs in
the
EventBuffer Descriptor is added to the end of the EventBuffer (a
FIFO
queue), along with the time that it was detected.   The MG SHALL
wait
for a new EventsDescriptor to be loaded. A new EventsDescriptor
can
be loaded either as the result of receiving a command with a new
EventsDescriptor, or by activating an embedded EventsDescriptor.

   If EventBufferControl equals Off, the MG continues processing based
   on the active EventsDescriptor.

   In the case that an embedded EventsDescriptor being activated, the MG
   continues event processing based on the newly activated
   EventsDescriptor (Note - for purposes of EventBuffer handling,
   activation of an embedded EventsDescriptor is equivalent to receipt
   of a new EventsDescriptor).

   When the MG receives a command with a new EventsDescriptor, one or
   more events may have been buffered in the EventBuffer in the MG. The
   value of EventBufferControl then determines how the MG treats such
   buffered events.

   Case 1

   If EventBufferControl equals LockStep and the MG receives a new
   EventsDescriptor it will check the FIFO EventBuffer and take the
   following actions:

   1. If the EventBuffer is empty, the MG waits for detection of events
      based on the new EventsDescriptor.

   2. If the EventBuffer is non-empty, the MG processes the FIFO queue
      starting with the first event:

      a) If the event in the queue is in the events listed in the new
         EventsDescriptor, the MG acts on the event and removes the
         event from the EventBuffer. The time stamp of the Notify shall
         be the time the event was actually detected. The MG then waits
         for a new EventsDescriptor. While waiting for a new
         EventsDescriptor, any events detected that appear in the
         EventsBufferDescriptor will be placed in the EventBuffer. When
         a new EventsDescriptor is received, the event processing will
         repeat from step 1.

      b) If the event is not in the new EventsDescriptor, the MG SHALL
         discard the event and repeat from step 1.

   Case 2

   If EventBufferControl equals Off and the MG receives a new
   EventsDescriptor, it processes new events with the new

EventsDescriptor.

If the MG receives a command instructing it to set the value of
EventBufferControl to Off, all events in the EventBuffer SHALL be
discarded.

The MG may report several events in a single Transaction as long as
this does not unnecessarily delay the reporting of individual events.

For procedures regarding transmitting the Notify command, refer to
the appropriate annex for specific transport considerations.

The default value of EventBufferControl is Off.

Note - Since the EventBufferControl property is in the
TerminationStateDescriptor, the MG might receive a command that
changes the EventBufferControl property and does not include an
EventsDescriptor.

Normally, recognition of an event shall cause any active signals to
stop. When KeepActive is specified in the event, the MG shall not
interrupt any signals active on the Termination on which the event is
detected.

An event can include an Embedded Signals descriptor and/or an
Embedded Events Descriptor which, if present, replaces the current
Signals/Events descriptor when the event is recognized. It is
possible, for example, to specify that the dial-tone Signal be
generated when an off-hook Event is recognized, or that the dial-tone
Signal be stopped when a digit is recognized. A media gateway
controller shall not send EventsDescriptors with an event both marked
KeepActive and containing an embedded SignalsDescriptor.

Only one level of embedding is permitted. An embedded
EventsDescriptor SHALL NOT contain another embedded EventsDescriptor;
an embedded EventsDescriptor may contain an embedded
SignalsDescriptor.

An EventsDescriptor received by a media gateway replaces any previous
Events Descriptor. Event notification in process shall complete, and
events detected after the command containing the new EventsDescriptor

executes, shall be processed according to the new
EventsDescriptor.

7.1.10 EventBuffer Descriptor

   The EventBuffer Descriptor contains a list of events, with their
   parameters if any, that the MG is requested to detect and buffer
when
   EventBufferControl equals LockStep (see 7.1.9).

7.1.11 Signals Descriptor

   A SignalsDescriptor is a parameter that contains the set of
signals
   that the Media Gateway is asked to apply to a Termination. A
   SignalsDescriptor contains a number of signals and/or sequential

   signal lists. A SignalsDescriptor may contain zero signals and
   sequential signal lists. Support of sequential signal lists is
   optional.

   Signals are defined in packages. Signals shall be named with a
   Package name (in which the signal is defined) and a SignalID. No
   wildcard shall be used in the SignalID. Signals that occur in a
   SignalsDescriptor have an optional StreamID parameter (default is
0,
   to indicate that the signal is not related to a particular media
   stream), an optional signal type (see below), an optional duration
   and possibly parameters defined in the package that defines the
   signal. This allows a single signal to have some variation in
   meaning, obviating the need to create large numbers of individual
   signals.

   Finally, the optional parameter "notifyCompletion" allows a MGC to
   indicate that it wishes to be notified when the signal finishes
   playout. The possible cases are that the signal timed out, that
it
   was interrupted by an event, that it was halted when a Signals
   Descriptor was replaced, or that it stopped or never started for
   other reasons. If the notifyCompletion parameter is not included
in
   a Signals Descriptor, notification is generated only if the signal
   stopped or was never started for other reasons. For reporting to
   occur, the signal completion event (see section E.1.2) must be
   enabled in the currently active Events Descriptor.

   The duration is an integer value that is expressed in hundredths
of a
   second.

   There are three types of signals:

* on/off - the signal lasts until it is turned off,
       * timeout - the signal lasts until it is turned off or a specific
          period of time elapses,
       * brief - the signal duration is so short that it will stop on
its
          own unless a new signal is applied that causes it to stop; no
          timeout value is needed.

       If the signal type is specified in a SignalsDescriptor, it
overrides
       the default signal type (see Section 12.1.4). If duration is
       specified for an on/off signal, it SHALL be ignored.

       A sequential signal list consists of a signal list identifier, a
       sequence of signals to be played sequentially, and a signal type.
       Only the trailing element of the sequence of signals in a
sequential
       signal list may be an on/off signal. If the trailing element of
the
       sequence is an on/off signal, the signal type of the sequential
       signal list shall be on/off as well. If the sequence of signals
in a

       sequential signal list contains signals of type timeout and the
       trailing element is not of type on/off, the type of the sequential
       signal list SHALL be set to timeout. The duration of a sequential
       signal list with type timeout is the sum of the durations of the
       signals it contains. If the sequence of signals in a sequential
       signal list contains only signals of type brief, the type of the
       sequential signal list SHALL be set to brief. A signal list is
       treated as a single signal of the specified type when played out.

       Multiple signals and sequential signal lists in the same
       SignalsDescriptor shall be played simultaneously.

       Signals are defined as proceeding from the termination towards the
       exterior of the Context unless otherwise specified in a package.
When
       the same Signal is applied to multiple Terminations within one
       Transaction, the MG should consider using the same resource to
       generate these Signals.

       Production of a Signal on a Termination is stopped by application
of
       a new SignalsDescriptor, or detection of an Event on the
Termination
       (see section 7.1.9).

       A new SignalsDescriptor replaces any existing SignalsDescriptor.
Any
       signals applied to the Termination not in the replacement
descriptor

shall be stopped, and new signals are applied, except as follows.
Signals present in the replacement descriptor and containing the
KeepActive flagshall be continued if they are currently playing and
have not already completed. If a replacement signal descriptor
contains a signal that is not currently playing and contains the
KeepActive flag, that signal SHALL be ignored. If the replacement
descriptor contains a sequential signal list with the same identifier
as the existing descriptor, then

   * the signal type and sequence of signals in the sequential signal
      list in the replacement descriptor shall be ignored, and

   * the playing of the signals in the sequential signal list in the
      existing descriptor shall not be interrupted.

7.1.12 Audit Descriptor

   The Audit Descriptor specifies what information is to be audited. The
   Audit Descriptor specifies the list of descriptors to be returned.
   Audit may be used in any command to force the return of a descriptor
   even if the descriptor in the command was not present, or had no
   underspecified parameters. Possible items in the Audit Descriptor
   are:

```
          +----------------+
          | Modem          |
          |----------------|
          | Mux            |
          |----------------|
          | Events         |
          |----------------|
          | Media          |
          |----------------|
          | Signals        |
          |----------------|
          | ObservedEvents |
          |----------------|
          | DigitMap       |
          |----------------|
          | Statistics     |
          |----------------|
          | Packages       |
          |----------------|
          | EventBuffer    |
          +----------------+
```

Audit may be empty, in which case, no descriptors are returned. This
   is useful in Subtract, to inhibit return of statistics, especially
   when using wildcard.

7.1.13 ServiceChange Descriptor

   The ServiceChangeDescriptor contains the following parameters:

   *   ServiceChangeMethod
   *   ServiceChangeReason
   *   ServiceChangeAddress
   *   ServiceChangeDelay
   *   ServiceChangeProfile
   *   ServiceChangeVersion
   *   ServiceChangeMGCId
   *   TimeStamp
   *   Extension.

   See section 7.2.8.

7.1.14 DigitMap Descriptor

7.1.14.1 DigitMap Definition, Creation, Modification and Deletion

   A DigitMap is a dialing plan resident in the Media Gateway used
for
   detecting and reporting digit events received on a Termination.
The
   DigitMap Descriptor contains a DigitMap name and the DigitMap to
be
   assigned. A digit map may be preloaded into the MG by management
   action and referenced by name in an EventsDescriptor, may be
defined
   dynamically and subsequently referenced by name, or the actual
   digitmap itself may be specified in the EventsDescriptor. It is
   permissible for a digit map completion event within an Events
   Descriptor to refer by name to a DigitMap which is defined by a
   DigitMap Descriptor within the same command, regardless of the
   transmitted order of the respective descriptors.

   DigitMaps defined in a DigitMapDescriptor can occur in any of the
   standard Termination manipulation Commands of the protocol. A
   DigitMap, once defined, can be used on all Terminations specified
by

the (possibly wildcarded) TerminationID in such a command. DigitMaps
   defined on the root Termination are global and can be used on every
   Termination in the MG, provided that a DigitMap with the same name
   has not been defined on the given Termination. When a DigitMap is
   defined dynamically in a DigitMap Descriptor:

   * A new DigitMap is created by specifying a name that is not yet
      defined. The value shall be present.

   * A DigitMap value is updated by supplying a new value for a name
      that is already defined. Terminations presently using the
      digitmap shall continue to use the old definition; subsequent
      EventsDescriptors specifying the name, including any
      EventsDescriptor in the command containing the DigitMap
      descriptor, shall use the new one.

   * A DigitMap is deleted by supplying an empty value for a name that
      is already defined. Terminations presently using the digitmap
      shall continue to use the old definition.

7.1.14.2 DigitMap Timers

   The collection of digits according to a DigitMap may be protected by
   three timers, viz. a start timer (T), short timer (S), and long timer
   (L).

   1. The start timer (T) is used prior to any digits having been
      dialed.

   2. If the Media Gateway can determine that at least one more digit is
      needed for a digit string to match any of the allowed patterns in
      the digit map, then the interdigit timer value should be set to a
      long (L) duration (e.g. 16 seconds).

   3. If the digit string has matched one of the patterns in a digit
      map, but it is possible that more digits could be received which
      would cause a match with a different pattern, then instead of
      reporting the match immediately, the MG must apply the short timer
      (S) and wait for more digits.

   The timers are configurable parameters to a DigitMap. The Start

timer is started at the beginning of every digit map use, but can
be
   overridden.

7.1.14.3 DigitMap Syntax

   The formal syntax of the digit map is described by the DigitMap
rule
   in the formal syntax description of the protocol (see Annex A and
   Annex B). A DigitMap, according to this syntax, is defined either
by
   a string or by a list of strings. Each string in the list is an
   alternative event sequence, specified either as a sequence of
digit
   map symbols or as a regular expression of digit map symbols.
These
   digit map symbols, the digits "0" through "9" and letters "A"
through
   a maximum value depending on the signalling system concerned, but
   never exceeding "K", correspond to specified events within a
package
   which has been designated in the Events Descriptor on the
termination
   to which the digit map is being applied. (The mapping between
events
   and digit map symbols is defined in the documentation for packages
   associated with channel-associated signalling systems such as
DTMF,
   MF, or R2. Digits "0" through "9" MUST be mapped to the
   corresponding digit events within the signalling system concerned.
   Letters should be allocated in logical fashion, facilitating the
use
   of range notation for alternative events.)

   The letter "x" is used as a wildcard, designating any event
   corresponding to symbols in the range "0"-"9". The string may
also
   contain explicit ranges and, more generally, explicit sets of
   symbols, designating alternative events any one of which satisfies
   that position of the digit map. Finally, the dot symbol "."
stands
   for zero or more repetitions of the event selector (event, range
of
   events, set of alternative events, or wildcard) that precedes it.
As
   a consequence of the third timing rule above, inter-event timing
   while matching a terminal dot symbol uses the short timer by
default.

   In addition to these event symbols, the string may contain "S" and
   "L" inter-event timing specifiers and the "Z" duration modifier.
"S"
   and "L" respectively indicate that the MG should use the short (S)

timer or the long (L) timer for subsequent events, over-riding the
timing rules described above. If an explicit timing specifier is
in
effect in one alternative event sequence, but none is given in any
other candidate alternative, the timer value set by the explicit
timing specifier must be used. If all sequences with explicit
timing
controls are dropped from the candidate set, timing reverts to the
default rules given above. Finally, if conflicting timing
specifiers
are in effect in different alternative sequences, the results are
undefined.

A "Z" designates a long duration event: placed in front of the
symbol(s) designating the event(s) which satisfy a given digit
position, it indicates that that position is satisfied only if the
duration of the event exceeds the long-duration threshold. The
value
of this threshold is assumed to be provisioned in the MG.

7.1.14.4 DigitMap Completion Event

A digit map is active while the events descriptor which invoked it
is
active and it has not completed. A digit map completes when:

* a timer has expired, or

* an alternative event sequence has been matched and no other
  alternative event sequence in the digit map could be matched
  through detection of an additional event (unambiguous match),
or

* an event has been detected such that a match to a complete
  alternative event sequence of the digit map will be impossible
no
  matter what additional events are received.

Upon completion, a digit map completion event as defined in the
package providing the events being mapped into the digit map shall
be
generated. At that point the digit map is deactivated. Subsequent
events in the package are processed as per the currently active
event
processing mechanisms.

7.1.14.5 DigitMap Procedures

Pending completion, successive events shall be processed according
to
the following rules:

1. The "current dial string", an internal variable, is initially
   empty. The set of candidate alternative event sequences
includes
   all of the alternatives specified in the digit map.

   2. At each step, a timer is set to wait for the next event, based
      either on the default timing rules given above or on explicit
      timing specified in one or more alternative event sequences. If
      the timer expires and a member of the candidate set of
      alternatives is fully satisfied, a timeout completion with full
      match is reported. If the timer expires and part or none of
any
      candidate alternative is satisfied, a timeout completion with
      partial match is reported.

   3. If an event is detected before the timer expires, it is mapped
to
      a digit string symbol and provisionally added to the end of the
      current dial string. The duration of the event (long or not
long)
      is noted if and only if this is relevant in the current symbol
      position (because at least one of the candidate alternative
event
      sequences includes the "Z" modifier at this position in the
      sequence).

   4. The current dial string is compared to the candidate
alternative
      event sequences. If and only if a sequence expecting a long-
      duration event at this position is matched (i.e. the event had
      long duration and met the specification for this position),
then
      any alternative event sequences not specifying a long duration
      event at this position are discarded, and the current dial
string
      is modified by inserting a "Z" in front of the symbol
representing
      the latest event. Any sequence expecting a long-duration event
at
      this position but not matching the observed event is discarded
      from the candidate set.   If alternative event sequences not
      specifying a long duration event in the given position remain
in
      the candidate set after application of the above rules, the
      observed event duration is treated as irrelevant in assessing
      matches to them.

   5. If exactly one candidate remains and it has been fully matched,
a
      completion event is generated indicating an unambiguous match.
If
      no candidates remain, the latest event is removed from the
current
      dial string and a completion event is generated indicating full
      match if one of the candidates from the previous step was fully

satisfied before the latest event was detected, or partial
match
   otherwise. The event removed from the current dial string will
   then be reported as per the currently active event processing
   mechanisms.

   6. If no completion event is reported out of step 5, processing
      returns to step 2.

7.1.14.6 DigitMap Activation

   A digit map is activated whenever a new event descriptor is
applied
   to the termination or embedded event descriptor is activated, and
   that event descriptor contains a digit map completion event which
   itself contains a digit map parameter. Each new activation of a
   digit map begins at step 1 of the above procedure, with a clear
   current dial string. Any previous contents of the current dial
   string from an earlier activation are lost.

7.1.14.7 Interaction of DigitMap and Event Processing

   While the digit map is activated, detection is enabled for all
events
   defined in the package containing the specified digit map
completion
   event. Normal event behaviour (e.g. stopping of signals unless
the
   digit completion event has the KeepActive flag enabled) continues
to
   apply for each such event detected, except that:

   * the events in the package containing the specified digit map
      completion event other than the completion event itself are not
      individually notified, and

   * an event that triggers a partial match completion event is not
      recognized and therefore has no side effects until reprocessed
      following the recognition of the digit map completion event.

7.1.14.8 Wildcards

   Note that if a package contains a digit map completion event, then
an
   event specification consisting of the package name with a
wildcarded
   ItemID (Property Name) will activate a digit map if the event

includes a digit map parameter. Regardless of whether a digit map is
    activated, if the package also contains the digit events themselves,
    this form of event specification will cause the individual events to
    be reported to the MGC as they are detected.

7.1.14.9 Example

    As an example, consider the following dial plan:

    +-----------------------+---+
    | 0 | Local operator |
    |-----------------------+---|
    | 00 | Long distance operator |
    |-----------------------+---|
    | xxxx | Local extension number (starts with 1-7) |
    |-----------------------+---|
    | 8xxxxxxx | Local number |
    |-----------------------+---|
    | #xxxxxxx | Off-site extension |
    |-----------------------+---|
    | *xx | Star services |
    |-----------------------+---|

```
   | 91xxxxxxxxxx             | Long distance number
|
   |----------------------+--------------------------------------
--|
   | 9011 + up to 15 digits | International number
|
   +----------------------+--------------------------------------
--+
```

   If the DTMF detection package described in Annex E (section E.6)
is
   used to collect the dialled digits, then the dialling plan shown
   above results in the following digit map:

      (0| 00|[1-7]xxx|8xxxxxxx|Fxxxxxxx|Exx|91xxxxxxxxxx|9011x.)

7.1.15 Statistics Descriptor

   The Statistics parameter provides information describing the
status
   and usage of a Termination during its existence within a specific
   Context. There is a set of standard statistics kept for each
   termination where appropriate (number of octets sent and received
for
   example). The particular statistical properties that are reported
   for a given Termination are determined by the Packages realized by
   the Termination. By default, statistics are reported when the
   Termination is Subtracted from the Context. This behavior can be
   overridden by including an empty AuditDescriptor in the Subtract
   command. Statistics may also be returned from the AuditValue
   command, or any Add/Move/Modify command using the Audit
descriptor.
   Statistics are cumulative; reporting Statistics does not reset
them.
   Statistics are reset when a Termination is Subtracted from a
Context.

7.1.16 Packages Descriptor

   Used only with the AuditValue command, the PackageDescriptor
returns
   a list of Packages realized by the Termination.

7.1.17 ObservedEvents Descriptor

   ObservedEvents is supplied with the Notify command to inform the
MGC
   of which event(s) were detected. Used with the AuditValue
command,
   the ObservedEventsDescriptor returns events in the event buffer
which
```

have not been Notified. ObservedEvents contains the RequestIdentifier
   of the EventsDescriptor that triggered the notification, the event(s)
   detected and the detection time(s). Detection times are reported with
   a precision of hundredths of a second.  Time is expressed in UTC.

7.1.18  Topology Descriptor

   A topology descriptor is used to specify flow directions between
   terminations in a Context.  Contrary to the descriptors in previous
   sections, the topology descriptor applies to a Context instead of a
   Termination.  The default topology of a Context is that each
   termination's transmission is received by all other terminations. The
   Topology Descriptor is optional to implement.

   The Topology Descriptor occurs before the commands in an action. It
   is possible to have an action containing only a Topology Descriptor,
   provided that the context to which the action applies already exists.

   A topology descriptor consists of a sequence of triples of the form
   (T1, T2, association). T1 and T2 specify Terminations within the
   Context, possibly using the ALL or CHOOSE wildcard.  The association
   specifies how media flows between these two Terminations as follows.

   *  (T1, T2, isolate) means that the Terminations matching T2 do not
      receive media from the Terminations matching T1, nor vice versa.

   *  (T1, T2, oneway) means that the Terminations that match T2 receive
      media from the Terminations matching T1, but not vice versa. In
      this case use of the ALL wildcard such that there are Terminations
      that match both T1 and T2 is not allowed.

   *  (T1, T2, bothway) means that the Terminations matching T2 receive
      media from the Terminations matching T1, and vice versa.  In this
      case it is allowed to use wildcards such that there are
      Terminations that match both T1 and T2.  However, if there is a
      Termination that matches both, no loopback is introduced.

   CHOOSE wildcards may be used in T1 and T2 as well, under the
   following restrictions:

   *  the action (see section 8) of which the topology descriptor is

part contains an Add command in which a CHOOSE wildcard is
used;

   *  if a CHOOSE wildcard occurs in T1 or T2, then a partial name
SHALL
      NOT be specified.

   The CHOOSE wildcard in a topology descriptor matches the
   TerminationID that the MG assigns in the first Add command that
uses
   a CHOOSE wildcard in the same action.  An existing Termination
that
   matches T1 or T2 in the Context to which a Termination is added,
is
   connected to the newly added Termination as specified by the
topology
   descriptor. The default association when a termination is not
   mentioned in the Topology descriptor is bothway (if T3 is added to
a
   context with T1 and T2 with topology (T3,T1,oneway) it will be
   connected bothway to T2).

   The figure below and the table following it show some examples of
the
   effect of including topology descriptors in actions.  In these
   examples it is assumed that the topology descriptors are applied
in
   sequence.

```
     +----------------+  +----------------+  +----------------+
     |      +----+    |  |      +----+    |  |      +----+    |
     |      | T2 |    |  |      | T2 |    |  |      | T2 |    |
     |      +----+    |  |      +----+    |  |      +----+    |
     |      ^   ^     |  |        ^       |  |        ^       |
     |      |   |     |  |        |       |  |        |       |
     |    +--+ +--+   |  |       +---+    |  |       +--+     |
     |    |      |    |  |        |       |  |        |       |
     |    v      v    |  |        v       |  |        |       |
     | +----+  +----+ |  | +----+  +----+ |  | +----+  +----+ |
     | | T1 |<-->| T3 | |  | | T1 |<-->| T3 | |  | | T1 |<-->| T3 | |
     | +----+  +----+ |  | +----+  +----+ |  | +----+  +----+ |
     +----------------+  +----------------+  +----------------+
      1. No Topology Desc.  2. T1, T2 Isolate   3. T3, T2 oneway
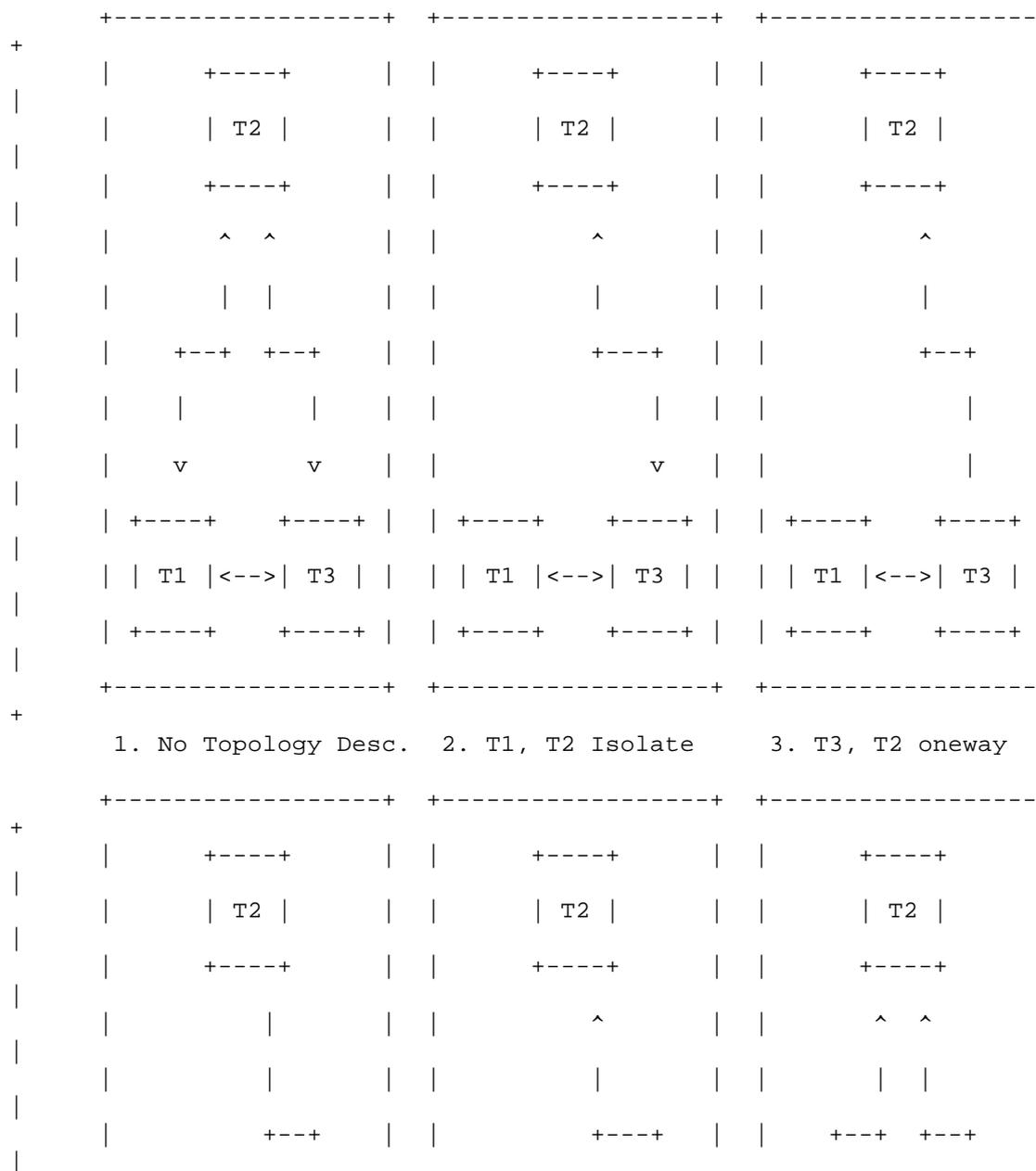
     +----------------+  +----------------+  +----------------+
     |      +----+    |  |      +----+    |  |      +----+    |
     |      | T2 |    |  |      | T2 |    |  |      | T2 |    |
     |      +----+    |  |      +----+    |  |      +----+    |
     |        |       |  |        ^       |  |        ^  ^    |
     |        |       |  |        |       |  |        |  |    |
     |       +--+     |  |       +---+    |  |       +--+ +--+ |
     |        |       |  |        |       |  |        |      |
```

```
        |              |    |  |                |   |  |     |          |
   |
        |              |    v  |  |                v   |  |  v          v
   |
   |    | +----+    +----+ |  | +----+    +----+ |  | +----+    +----+
   |    | | T1 |<-->| T3 | |  | | T1 |<-->| T3 | |  | | T1 |<-->| T3 |
   |    | +----+    +----+ |  | +----+    +----+ |  | +----+    +----+
   |    +-----------------+  +-----------------+  +------------------
   +
        4. T2, T3 oneway      5. T2, T3 bothway     6. T1, T2 bothway
```

                  Figure 4: A Sequence Of Example Topologies

```
      +---------+-------------------------------------------------
   +
      | Topology |   Description
   |
      |---------+-------------------------------------------------
   |
      |    1     | No topology descriptors
   |
      |---------+-------------------------------------------------
   |
      | When no topology descriptors are included, all terminations
   |
      | have a both way connection to all other terminations.
   |
      |---------+-------------------------------------------------
   |
      |    2     | T1, T2, Isolate
   |
```

```
          |---------+-------------------------------------------------
|         | Removes the connection between T1 and T2.
|         | T3 has a both way connection with both T1 and T2.  T1 and
|         | T2 have bothway connection to T3.
|         |---------+-------------------------------------------------
|         |    3    | T3, T2, oneway
|         |---------+-------------------------------------------------
|         | A oneway connection from T3 to T2 (i.e. T2 receives media
|         | flow from T3).  A bothway connection between T1 and T3.
|         |---------+-------------------------------------------------
|         |    4    | T2, T3, oneway
|         |---------+-------------------------------------------------
|         | A oneway connection between T2 to T3.
|         | T1 and T3 remain bothway connected
|         |---------+-------------------------------------------------
|         |    5    | T2, T3 bothway
|         |---------+-------------------------------------------------
|         | T2 is bothway connected to T3.  This results in the same
|         | as 2.
|         |---------+-------------------------------------------------
|         |    6    | T1, T2 bothway (T2, T3 bothway and T1,T3 bothway
|         |         | may be implied or explicit).
|         |---------+-------------------------------------------------
|         | All terminations have a bothway connection to all other
|         | terminations.
|         |---------+-------------------------------------------------
|         | A oneway connection must implemented in such a way that the
|         | other Terminations in the Context are not aware of the
|         | change in topology.
|         +-------------------------------------------------
|
7.2 Command Application Programming Interface
```

Following is an Application Programming Interface (API) describing
the Commands of the protocol. This API is shown to illustrate the
Commands and their parameters and is not intended to specify
implementation (e.g. via use of blocking function calls). It
describes the input parameters in parentheses after the command
name
and the return values in front of the Command. This is only for

descriptive purposes; the actual Command syntax and encoding are
specified in later subsections. The order of parameters to
commands
is not fixed. Descriptors may appear as parameters to commands in
any order. The descriptors SHALL be processed in the order in
which
they appear.

All parameters enclosed by square brackets ([. . .]) are
considered
optional.

7.2.1 Add

The Add Command adds a Termination to a Context.

TerminationID
[,MediaDescriptor]
[,ModemDescriptor]
[,MuxDescriptor]
[,EventsDescriptor]
[,SignalsDescriptor]
[,DigitMapDescriptor]
[,ObservedEventsDescriptor]
[,EventBufferDescriptor]
[,StatisticsDescriptor]
[,PackagesDescriptor]
        Add(TerminationID
      [, MediaDescriptor]
      [, ModemDescriptor]
      [, MuxDescriptor]
      [, EventsDescriptor]
      [, SignalsDescriptor]
      [, DigitMapDescriptor]
      [, AuditDescriptor]
)

The TerminationID specifies the termination to be added to the
Context. The Termination is either created, or taken from the
null
Context. For an existing Termination, the TerminationID would be
specific. For a Termination that does not yet exist, the
TerminationID is specified as CHOOSE in the command. The new

TerminationID will be returned. Wildcards may be used in an Add, but
   such usage would be unusual. If the wildcard matches more than one
   TerminationID, all possible matches are attempted, with results
   reported for each one. The order of attempts when multiple
   TerminationIDs match is not specified.

   The optional MediaDescriptor describes all media streams.

   The optional ModemDescriptor and MuxDescriptor specify a modem and
   multiplexer if applicable. For convenience, if a Multiplex Descriptor
   is present in an Add command and lists any Terminations that are not
   currently in the Context, such Terminations are added to the context
   as if individual Add commands listing the Terminations were invoked.
   If an error occurs on such an implied Add, error 471 - Implied Add
   for Multiplex failure shall be returned and further processing of the
   command shall cease.

   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, defines a
   DigitMap definition that may be used in an EventsDescriptor.

   The AuditDescriptor is optional. If present, the command will return
   descriptors as specified in the AuditDescriptor.

   All descriptors that can be modified could be returned by MG if a
   parameter was underspecified or overspecified. ObservedEvents,
   Statistics, and Packages, and the EventBuffer Descriptors are
   returned only if requested in the AuditDescriptor.

   Add SHALL NOT be used on a Termination with a serviceState of
   "OutofService".

7.2.2 Modify

The Modify Command modifies the properties of a Termination.

```
TerminationID
[,MediaDescriptor]
[,ModemDescriptor]
[,MuxDescriptor]
[,EventsDescriptor]
[,SignalsDescriptor]
[,DigitMapDescriptor]
[,ObservedEventsDescriptor]
[,EventBufferDescriptor]
[,StatisticsDescriptor]
[,PackagesDescriptor]
      Modify( TerminationID
      [, MediaDescriptor]
      [, ModemDescriptor]
      [, MuxDescriptor]
```

```
      [, EventsDescriptor]
      [, SignalsDescriptor]
      [, DigitMapDescriptor]
      [, AuditDescriptor]
      )
```

The TerminationID may be specific if a single Termination in the
Context is to be modified. Use of wildcards in the TerminationID may
be appropriate for some operations. If the wildcard matches more than
one TerminationID, all possible matches are attempted, with results
reported for each one. The order of attempts when multiple
TerminationIDs match is not specified. The CHOOSE option is an error,
as the Modify command may only be used on existing Terminations.

The remaining parameters to Modify are the same as those to Add.
Possible return values are the same as those to Add.

7.2.3 Subtract

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

```
TerminationID
[,MediaDescriptor]
[,ModemDescriptor]
[,MuxDescriptor]
[,EventsDescriptor]
[,SignalsDescriptor]
```

```
      [,DigitMapDescriptor]
      [,ObservedEventsDescriptor]
      [,EventBufferDescriptor]
      [,StatisticsDescriptor]
      [,PackagesDescriptor]
           Subtract(TerminationID
           [, AuditDescriptor]
           )
```

   TerminationID in the input parameters represents the Termination
that
   is being subtracted. The TerminationID may be specific or may be
a
   wildcard value indicating that all (or a set of related)
Terminations
   in the Context of the Subtract Command are to be subtracted. If
the
   wildcard matches more than one TerminationID, all possible matches
   are attempted, with results reported for each one. The order of
   attempts when multiple TerminationIDs match is not specified.

   The use of CHOOSE in the TerminationID is an error, as the
Subtract
   command may only be used on existing Terminations.

   ALL may be used as the ContextID as well as the TerminationId in a
   Subtract, which would have the effect of deleting all contexts,
   deleting all ephemeral terminations, and returning all physical
   terminations to Null context.

   By default, 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.

   The AuditDescriptor is optional. If present, the command will
return
   descriptors as specified in the AuditDescriptor.   Possible return
   values are the same as those to Add.

   When a provisioned Termination is Subtracted from a context, its
   property values shall revert to:

   * the default value, if specified for the property and not
      overridden by provisioning,
   * otherwise, the provisioned value.

7.2.4 Move

   The Move Command moves a Termination to another Context from its
   current Context in one atomic operation. The Move command is the
   only command that refers to a Termination in a Context different
from
   that to which the command is applied. The Move command shall not
be
   used to move Terminations to or from the null Context.

   TerminationID
   [,MediaDescriptor]
   [,ModemDescriptor]
   [,MuxDescriptor]
   [,EventsDescriptor]
   [,SignalsDescriptor]
   [,DigitMapDescriptor]
   [,ObservedEventsDescriptor]
   [,EventBufferDescriptor]
   [,StatisticsDescriptor]
   [,PackagesDescriptor]
         Move(TerminationID
         [, MediaDescriptor]
         [, ModemDescriptor]
         [, MuxDescriptor]
         [, EventsDescriptor]
         [, SignalsDescriptor]
         [, DigitMapDescriptor]

         [, AuditDescriptor]
)

   The TerminationID specifies the Termination to be moved. It may
be
   wildcarded, but CHOOSE shall not be used in the TerminationID. If
   the wildcard matches more than one TerminationID, all possible
   matches are attempted, with results reported for each one. The
order
   of attempts when multiple TerminationIDs match is not specified.
By
   convention, the Termination is subtracted from its previous
Context.
   The Context to which the Termination is moved is indicated by the
   target ContextId in the Action. If the last remaining Termination
is
   moved out of a Context, the Context is deleted.

   The remaining descriptors are processed as in the Modify Command.
The
   AuditDescriptor with the Statistics option, for example, would
return
   statistics on the Termination just prior to the Move. Possible

descriptors returned from Move are the same as for Add.

Move SHALL NOT be used on a Termination with a serviceState of
"OutofService".

7.2.5 AuditValue

The AuditValue Command returns the current values of properties,
events, signals and statistics associated with Terminations.

```
TerminationID
[,MediaDescriptor]
[,ModemDescriptor]
[,MuxDescriptor]
[,EventsDescriptor]
[,SignalsDescriptor]
[,DigitMapDescriptor]
[,ObservedEventsDescriptor]
[,EventBufferDescriptor]
[,StatisticsDescriptor]
[,PackagesDescriptor]
      AuditValue(TerminationID,
      AuditDescriptor
      )
```

TerminationID may be specific or wildcarded. If the wildcard
matches
more than one TerminationID, all possible matches are attempted,
with
results reported for each one. The order of attempts when
multiple
TerminationIDs match is not specified. If a wildcarded response is
requested, only one command return is generated, with the contents

containing the union of the values of all Terminations matching
the
wildcard. This convention may reduce the volume of data required
to
audit a group of Terminations. Use of CHOOSE is an error.

The appropriate descriptors, with the current values for the
Termination, are returned from AuditValue. Values appearing in
multiple instances of a descriptor are defined to be alternate
values
supported, with each parameter in a descriptor considered
independent.

ObservedEvents returns a list of events in the EventBuffer. If
the

ObservedEvents descriptor is audited while a DigitMap is active, the
returned ObservedEvents descriptor also includes a digit map
completion event that shows the current dial string but does not show
a termination method.

EventBuffer returns the set of events and associated parameter values
currently enabled in the EventBufferDescriptor. PackagesDescriptor
returns a list of packages realized by the Termination.
DigitMapDescriptor returns the name or value of the current DigitMap
for the Termination. DigitMap requested in an AuditValue command
with TerminationID ALL returns all DigitMaps in the gateway.
Statistics returns the current values of all statistics being kept on
the Termination. Specifying an empty Audit Descriptor results in
only the TerminationID being returned. This may be useful to get a
list of TerminationIDs when used with wildcard. Annexes A and B
provide a special syntax for presenting such a list in condensed
form, such that the AuditValue command tag does not have to be
repeated for each TerminationID.

AuditValue results depend on the Context, viz. specific, null, or
wildcarded. The TerminationID may be specific, or wildcarded. (Note
that ContextID All does not include the null Context.) The following
illustrates other information that can be obtained with the Audit
Command:

```
+-----------+---------------+-----------------------------------+
| ContextID | TerminationID | Information Obtained              |
+-----------+---------------+-----------------------------------+
```

```
   | Specific  | wildcard      | Audit of matching Terminations in a |
   |           |               | Context                             |
   +-----------+--------------+-------------------------------------+
   | Specific  | specific      | Audit of a single Termination in a  |
   |           |               | Context                             |
   +-----------+--------------+-------------------------------------+
   | Null      | Root          | Audit of Media Gateway state and    |
   |           |               | events                              |
   +-----------+--------------+-------------------------------------+
   | Null      | wildcard      | Audit of all matching Terminations  |
   |           |               | in the Null Context                 |
   +-----------+--------------+-------------------------------------+
   | Null      | specific      | Audit of a single Termination       |
   |           |               | outside of any Context              |
   +-----------+--------------+-------------------------------------+
   | All       | wildcard      | Audit of all matching Terminations  |
   |           |               | and the Context to which they are   |
   |           |               | associated                          |
   +-----------+--------------+-------------------------------------+
   | All       | Root          | List of all ContextIds              |
   +-----------+--------------+-------------------------------------+
```

7.2.6 AuditCapabilities

   The AuditCapabilities Command returns the possible values of
   properties, events, signals and statistics associated with
   Terminations.

   TerminationID
   [,MediaDescriptor]
   [,ModemDescriptor]
   [,MuxDescriptor]
   [,EventsDescriptor]
   [,SignalsDescriptor]
   [,ObservedEventsDescriptor]
   [,EventBufferDescriptor]
   [,StatisticsDescriptor]
        AuditCapabilities(TerminationID,
        AuditDescriptor
)

The appropriate descriptors, with the possible values for the
Termination are returned from AuditCapabilities. Descriptors may
be
repeated where there are multiple possible values. If a
wildcarded
response is requested, only one command return is generated, with
the

contents containing the union of the values of all Terminations
matching the wildcard. This convention may reduce the volume of
data
required to audit a group of Terminations.

Interpretation of what capabilities are requested for various
values
of ContextID and TerminationID is the same as in AuditValue.

The EventsDescriptor returns the list of possible events on the
Termination together with the list of all possible values for the
EventsDescriptor Parameters. EventBufferDescriptor returns the
same
information as EventsDescriptor. The SignalsDescriptor returns
the
list of possible signals that could be applied to the Termination
together with the list of all possible values for the Signals
Parameters. StatisticsDescriptor returns the names of the
statistics
being kept on the termination. ObservedEventsDescriptor returns
the
names of active events on the termination. DigitMap and Packages
are
not legal in AuditCapability.

7.2.7 Notify

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

          Notify(TerminationID,
          ObservedEventsDescriptor,
          [ErrorDescriptor]
)

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

The ObservedEventsDescriptor contains the RequestID and a list of
events that the Media Gateway detected in the order that they were
detected. Each event in the list is accompanied by parameters
associated with the event and an indication of the time that the
event was detected. Procedures for sending Notify commands with

RequestID equal to 0 are for further study.

Notify Commands with RequestID not equal to 0 shall occur only as
the
result of detection of an event specified by an Events Descriptor
which is active on the termination concerned.

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 triggering EventsDescriptor
or
embedded events descriptor unless the RequestID is 0 (which is for
further study).

7.2.8 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.   The Media Gateway Controller may indicate that
Termination(s) shall be taken out of or returned to service. The
Media Gateway may notify the MGC that the capability of a
Termination
has changed. It also allows a MGC to hand over control of a MG to
another MGC.

TerminationID,
[ServiceChangeDescriptor]
     ServiceChange(TerminationID,
     ServiceChangeDescriptor
)

The TerminationID parameter specifies the Termination(s) that are
taken out of or returned to service. Wildcarding of Termination
names is permitted, with the exception that the CHOOSE mechanism
shall not be used. Use of the "Root" TerminationID indicates a
ServiceChange affecting the entire Media Gateway.

The ServiceChangeDescriptor contains the following parameters as
required:

*   ServiceChangeMethod
*   ServiceChangeReason
*   ServiceChangeDelay
*   ServiceChangeAddress
*   ServiceChangeProfile
*   ServiceChangeVersion
*   ServiceChangeMgcId

*   TimeStamp

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 on the
Termination(s) affected by the serviceChange command. The MG
should set termination serviceState at the expiry of
ServiceChangeDelay or the removal of the termination from an
active context (whichever is first), to "out of service".

2) Forced - indicates that the specified Terminations were taken
abruptly out of service and any established connections associated
with them were lost. The MGC is responsible for cleaning up the
context (if any) with which the failed termination is associated.
At a minimum the termination shall be subtracted from the context.
The termination serviceState should be "out of service".

3) Restart - indicates that service will be restored on the specified
Terminations after expiration of the ServiceChangeDelay. The
serviceState should be set to "inService" upon expiry of
ServiceChangeDelay.

4) Disconnected - always applied with the Root TerminationID,
indicates that the MG lost communication with the MGC, but it was
subsequently restored. Since MG state may have changed, the MGC
may wish to use the Audit command to resynchronize its state with
the MG's.

5) Handoff - sent from the MGC to the MG, this reason indicates that
the MGC is going out of service and a new MGC association must be
established. Sent from the MG to the MGC, this indicates that the

MG is attempting to establish a new association in accordance
with
        a Handoff received from the MGC with which it was previously
        associated.

    6) Failover - sent from MG to MGC to indicate the primary MG is
out
        of service and a secondary MG is taking over.

    7) Another value whose meaning is mutually understood between the
MG
        and the MGC.

    The ServiceChangeReason parameter specifies the reason why the
    ServiceChange has or will occur. It consists of an alphanumeric
    token (IANA registered) and, optionally, an explanatory string.

    The optional ServiceChangeAddress parameter specifies the address
    (e.g., IP port number for IP networks) to be used for subsequent
    communications. It can be specified in the input parameter
    descriptor or the returned result descriptor.
ServiceChangeAddress
    and ServiceChangeMgcId parameters must not both be present in the
    ServiceChangeDescriptor or the ServiceChangeResultDescriptor. The
    ServiceChangeAddress provides an address to be used within the
    context of the association currently being negotiated, while the
    ServiceChangeMgcId provides an alternate address where the MG
should
    seek to establish another association.

    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. For
"graceful"
    only, a null delay means the MG must not set serviceState "out of
    service" until the termination is in the null context.

    The optional ServiceChangeProfile parameter specifies the Profile
(if
    any) of the protocol supported. The ServiceChangeProfile includes
    the version of the profile supported.

    The optional ServiceChangeVersion parameter contains the protocol
    version and is used if protocol version negotiation occurs (see
    section 11.3).

The optional TimeStamp parameter specifies the actual time as kept by
   the sender. It can be used by the responder to determine how its
   notion of time differs from that of its correspondent. TimeStamp is
   sent with a precision of hundredths of a second, and is expressed in
   UTC.

   The optional Extension parameter may contain any value whose meaning
   is mutually understood by the MG and MGC.

   A ServiceChange Command specifying the "Root" for the TerminationID
   and ServiceChangeMethod equal to Restart 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 optionally some number of alternate Media
   Gateway Controllers. Acknowledgement of the ServiceChange Command
   completes the registration process, except when the MGC has returned
   an alternative ServiceChangeMgcId as described in the following
   paragraph. The MG may specify the transport ServiceChangeAddress to
   be used by the MGC for sending messages in the ServiceChangeAddress
   parameter in the input ServiceChangeDescriptor. The MG may specify an
   address in the ServiceChangeAddress parameter of the ServiceChange
   request, and the MGC may also do so in the ServiceChange reply. In
   either case, the recipient must use the supplied address as the
   destination for all subsequent transaction requests within the
   association. At the same time, as indicated in section 9,
   transaction replies and pending indications must be sent to the
   address from which the corresponding requests originated. This must
   be done even if it implies extra messaging because commands and
   responses cannot be packed together. The TimeStamp parameter shall be
   sent with a registration command and its response.

   The Media Gateway Controller may return an ServiceChangeMgcId
   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 shall reissue the ServiceChange
command
   to the new Media Gateway Controller.   The Gateway specified in an
   ServiceChangeMgcId, if provided, shall be contacted before any
   further alternate MGCs. On a HandOff message from MGC to MG, the
   ServiceChangeMgcId is the new MGC that will take over from the
   current MGC.

   The return from ServiceChange is empty except when the Root
   terminationID is used. In that case it includes the following
   parameters as required:

   * ServiceChangeAddress, if the responding MGC wishes to specify a
      new destination for messages from the MG for the remainder of
the
      association;

   * ServiceChangeMgcId, if the responding MGC does not wish to
sustain
      an association with the MG;

   * ServiceChangeProfile, if the responder wishes to negotiate the
      profile to be used for the association;

   * ServiceChangeVersion, if the responder wishes to negotiate the
      version of the protocol to be used for the association.

   The following ServiceChangeReasons are defined. This list may be
   extended by an IANA registration as outlined in section 13.3.

        900 Service Restored
        901 Cold Boot
        902 Warm Boot
        903 MGC Directed Change
        904 Termination malfunctioning
        905 Termination taken out of service
        906 Loss of lower layer connectivity (e.g. downstream sync)
        907 Transmission Failure
        908 MG Impending Failure
        909 MGC Impending Failure
        910 Media Capability Failure
        911 Modem Capability Failure
        912 Mux Capability Failure
        913 Signal Capability Failure
        914 Event Capability Failure
        915 State Loss

7.2.9 Manipulating and Auditing Context Attributes

The commands of the protocol as discussed in the preceding
sections
apply to terminations. This section specifies how contexts are
manipulated and audited.

Commands are grouped into actions (see section 8). An action
applies
to one context. In addition to commands, an action may contain
context manipulation and auditing instructions.

An action request sent to a MG may include a request to audit
attributes of a context. An action may also include a request to
change the attributes of a context.

The context properties that may be included in an action reply are
used to return information to a MGC. This can be information
requested by an audit of context attributes or details of the
effect
of manipulation of a context.

If a MG receives an action which contains both a request to audit
context attributes and a request to manipulate those attributes,
the
response SHALL include the values of the attributes after
processing
the manipulation request.

7.2.10 Generic Command Syntax

The protocol can be encoded in a binary format or in a text
format.
MGCs should support both encoding formats. MGs may support both
formats.

The protocol syntax for the binary format of the protocol is
defined
in Annex A. Annex C specifies the encoding of the Local and
Remote
descriptors for use with the binary format.

A complete ABNF of the text encoding of the protocol per RFC2234
is
given in Annex B. SDP is used as the encoding of the Local and
Remote Descriptors for use with the text encoding as modified in
section 7.1.8.

7.3 Command Error Codes

Errors consist of an IANA registered error code and an explanatory
string. Sending the explanatory string is optional.
Implementations
are encouraged to append diagnostic information to the end of the
string.

   When a MG reports an error to a MGC, it does so in an error
   descriptor. An error descriptor consists of an error code and
   optionally the associated explanatory string.

   The identified error codes are:

        400 - Bad Request
        401 - Protocol Error
        402 - Unauthorized
        403 - Syntax Error in Transaction
        406 - Version Not Supported
        410 - Incorrect identifier
        411 - The transaction refers to an unknown ContextId
        412 - No ContextIDs available

        421 - Unknown action or illegal combination of actions
        422 - Syntax Error in Action
        430 - Unknown TerminationID
        431 - No TerminationID matched a wildcard
        432 - Out of TerminationIDs or No TerminationID available
        433 - TerminationID is already in a Context
        440 - Unsupported or unknown Package
        441 - Missing RemoteDescriptor
        442 - Syntax Error in Command
        443 - Unsupported or Unknown Command
        444 - Unsupported or Unknown Descriptor
        445 - Unsupported or Unknown Property
        446 - Unsupported or Unknown Parameter
        447 - Descriptor not legal in this command
        448 - Descriptor appears twice in a command
        450 - No such property in this package
        451 - No such event in this package
        452 - No such signal in this package
        453 - No such statistic in this package
        454 - No such parameter value in this package
        455 - Parameter illegal in this Descriptor
        456 - Parameter or Property appears twice in this Descriptor
        471 - Implied Add for Multiplex failure

        500 - Internal Gateway Error
        501 - Not Implemented
        502 - Not ready.
        503 - Service Unavailable
        504 - Command Received from unauthorized entity
        505 - Command Received before Restart Response
        510 - Insufficient resources
        512 - Media Gateway unequipped to detect requested Event
        513 - Media Gateway unequipped to generate requested Signals

514 - Media Gateway cannot send the specified announcement
                  515 - Unsupported Media Type
                  517 - Unsupported or invalid mode
                  518 - Event buffer full
                  519 - Out of space to store digit map
                  520 - Media Gateway does not have a digit map
                  521 - Termination is "ServiceChangeing"
                  526 - Insufficient bandwidth
                  529 - Internal hardware failure
                  530 - Temporary Network failure
                  531 - Permanent Network failure
                  581 - Does Not Exist

8. TRANSACTIONS

   Commands between the Media Gateway Controller and 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
   specifies 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.

```
    +-----------------------------------------------------------+
    | Transaction x                                             |
    |   +----------------------------------------------------+  |
    |   | Action 1                                           |  |
    |   | +---------+  +---------+  +---------+  +---------+  |  |
    |   | | Command |  | Command |  | Command |  | Command | |  |
    |   | |    1    |  |    2    |  |    3    |  |    4    | |  |
    |   | +---------+  +---------+  +---------+  +---------+  |  |
    |   +----------------------------------------------------+  |
    |                                                           |
    |   +----------------------------------------------------+  |
    |   | Action 2                                           |  |
    |   | +---------+                                        |  |
    |   | | Command |                                        |  |
    |   | |    1    |                                        |  |
    |   | +---------+                                        |  |
    |   +----------------------------------------------------+  |
    |                                                           |
    |   +----------------------------------------------------+  |
    |   | Action 3                                           |  |
    |   | +---------+  +---------+  +---------+              |  |
    |   | | Command |  | Command |  | Command |              |  |
    |   | |    1    |  |    2    |  |    3    |              |  |
    |   | +---------+  +---------+  +---------+              |  |
    |   +----------------------------------------------------+  |
    +-----------------------------------------------------------+
```

           Figure 5 Transactions, Actions and Commands

    Transactions are presented as TransactionRequests. Corresponding
    responses to a TransactionRequest are received in a single reply,
    possibly preceded by a number of TransactionPending messages (see
    section 8.2.3).

    Transactions guarantee ordered Command processing. That is,
Commands
    within a Transaction are executed sequentially. Ordering of
    Transactions is NOT guaranteed - transactions may be executed in
any
    order, or simultaneously.

    At the first failing Command in a Transaction, processing of the
    remaining Commands in that Transaction stops. If a command
contains
    a wildcarded TerminationID, the command is attempted with each of
the
    actual TerminationIDs matching the wildcard. A response within
the
    TransactionReply is included for each matching TerminationID, even
if
    one or more instances generated an error. If any TerminationID
    matching a wildcard results in an error when executed, any
commands
    following the wildcarded command are not attempted.

Commands may be marked as "Optional" which can override this
behaviour - if a command marked as Optional results in an error,
subsequent commands in the Transaction will be executed. If a
command fails, the MG shall as far as possible restore the state that
existed prior to the attempted execution of the command before
continuing with command processing.

A TransactionReply includes the results for all of the Commands in
the corresponding TransactionRequest. The TransactionReply includes
the return values for the Commands that were executed successfully,
and the Command and error descriptor for any Command that failed.
TransactionPending is used to periodically notify the receiver that a
Transaction has not completed yet, but is actively being processed.

Applications SHOULD implement an application level timer per
transaction. Expiration of the timer should cause a retransmission
of the request. Receipt of a Reply should cancel the timer. Receipt
of Pending should restart the timer.

8.1 Common Parameters

8.1.1 Transaction Identifiers

Transactions are identified by a TransactionID, which is assigned by
sender and is unique within the scope of the sender. A response
containing an error descriptor to indicate that the TransactionID is
missing in a request shall use TransactionID 0 in the corresponding
TransactionReply.

8.1.2 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 shall use the ContextID supplied by the
Media Gateway in all subsequent Transactions relating to that
Context. The protocol makes reference to a distinguished value that
may be used by the Media Gateway Controller when referring to a
Termination that is currently not associated with a Context, namely
the null ContextID.

The CHOOSE wildcard is used to request that the Media Gateway create
a new Context. The MGC shall not use partially specified ContextIDs
containing the CHOOSE wildcard.

The MGC may use the ALL wildcard to address all Contexts on the MG.
The null Context is not included when the ALL wildcard is used.

8.2 Transaction Application Programming Interface

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

8.2.1 TransactionRequest

   The TransactionRequest is invoked by the sender. There is one
   Transaction per request invocation. A request contains one or more
   Actions, each of which specifies its target Context and one or more
   Commands per Context.

         TransactionRequest(TransactionId {
                ContextID {Command _ Command},
                              . . .
                ContextID {Command _ Command } })

   The TransactionID parameter must specify a value for later
   correlation with the TransactionReply or TransactionPending response
   from the receiver.

   The ContextID parameter must specify a value 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 Command parameter represents one of the Commands mentioned in
the
"Command Details" subsection titled "Application Programming
Interface".

8.2.2 TransactionReply

The TransactionReply is invoked by the receiver. There is one
reply
invocation per transaction. A reply contains one or more Actions,
each of which must specify its target Context and one or more
Responses per Context.

```
        TransactionReply(TransactionID {
               ContextID { Response _ Response },
                             . . .
               ContextID { Response _ Response } })
```

The TransactionID parameter must be the same as that of the
corresponding TransactionRequest.

The ContextID parameter must specify a value to pertain to all
Responses for the action. The ContextID may be specific or null.

Each of the Response parameters represents a return value as
mentioned in section 7.2, or an error descriptor if the command
execution encountered an error. Commands after the point of
failure
are not processed and, therefore, Responses are not issued for
them.

An exception to this occurs if a command has been marked as
optional
in the Transaction request. If the optional command generates an
error, the transaction still continues to execute, so the Reply
would, in this case, have Responses after an Error.

If the receiver encounters an error in processing a ContextID, the
requested Action response will consist of the context ID and a
single
error descriptor, 422 Syntax Error in Action.

If the receiver encounters an error such that it cannot determine
a
legal Action, it will return a TransactionReply consisting of the
TransactionID and a single error descriptor, 422 Syntax Error in
Action. If the end of an action cannot be reliably determined but
one
or more Actions can be parsed, it will process them and then send
422

Syntax Error in Action as the last action for the transaction. If
the receiver encounters an error such that is cannot determine a
legal Transaction, it will return a TransactionReply with a null
TransactionID and a single error descriptor (403 Syntax Error in
Transaction).

If the end of a transaction can not be reliably determined and one
or
more Actions can be parsed, it will process them and then return
403
Syntax Error in Transaction as the last action reply for the
transaction. If no Actions can be parsed, it will return 403
Syntax
Error in Transaction as the only reply

If the terminationID cannot be reliably determined it will send
442
Syntax Error in Command as the action reply.

If the end of a command cannot be reliably determined it will
return
442 Syntax Error in Command as the reply to the last action it can
parse.

8.2.3 TransactionPending

The receiver invokes the TransactionPending. A TransactionPending
indicates that the Transaction is actively being processed, but
has
not been completed. It is used to prevent the sender from
assuming
the TransactionRequest was lost where the Transaction will take
some
time to complete.

        TransactionPending(TransactionID { })

The TransactionID parameter must be the same as that of the
corresponding TransactionRequest. A property of root
(normalMGExecutionTime) is settable by the MGC to indicate the
interval within which the MGC expects a response to any
transaction
from the MG. Another property (normalMGCExecutionTime) is
settable
by the MGC to indicate the interval within which the MG should
expects a response to any transaction from the MGC. Senders may
receive more than one TransactionPending for a command. If a

duplicate request is received when pending, the responder may send a

duplicate pending immediately, or continue waiting for its timer to

trigger another Transaction Pending.

8.3 Messages

Multiple Transactions can be concatenated into a Message. Messages
have a header, which includes the identity of the sender. The Message
Identifier (MID) of a message is set to a provisioned name (e.g.
domain address/domain name/device name) of the entity transmitting
the message. Domain name is a suggested default.

Every Message contains a Version Number identifying the version of
the protocol the message conforms to. Versions consist of one or two
digits, beginning with version 1 for the present version of the
protocol.

The transactions in a message are treated independently. There is no
order implied, there is no application or protocol acknowledgement of
a message.

9. TRANSPORT

The transport mechanism for the protocol should allow the reliable
transport of transactions between an MGC and MG. The transport shall
remain independent of what particular commands are being sent and
shall be applicable to all application states. There are several
transports defined for the protocol, which are defined in normative
Annexes to this document. Additional Transports may be defined as
additional annexes in subsequent editions of this document, or in

separate documents. For transport of the protocol over IP, MGCs
shall implement both TCP and UDP/ALF, an MG shall implement TCP or
UDP/ALF or both.

The MG is provisioned with a name or address (such as DNS name or IP
address) of a primary and zero or more secondary MGCs (see section
7.2.8) that is the address the MG uses to send messages to the MGC.
If TCP or UDP is used as the protocol transport and the port to which
the initial ServiceChange request is to be sent is not otherwise

known, that request should be sent to the default port number for the
   protocol. This port number is 2944 for text-encoded operation or
   2945 for binary-encoded operation, for either UDP or TCP. The MGC
   receives the message containing the ServiceChange request from the MG
   and can determine the MG's address from it. As described in section
   7.2.8, either the MG or the MGC may supply an address in the
   ServiceChangeAddress parameter to which subsequent transaction
   requests must be addressed, but responses (including the response to
   the initial ServiceChange request) must always be sent back to the
   address which was the source of the corresponding request.

9.1 Ordering of Commands

   This document does not mandate that the underlying transport protocol
   guarantees the sequencing of transactions sent to an entity. This
   property tends to maximize the timeliness of actions, but it has a
   few drawbacks. For example:

   * Notify commands may be delayed and arrive at the MGC after the
      transmission of a new command changing the EventsDescriptor

   * If a new command is transmitted before a previous one is
      acknowledged, there is no guarantee that prior command will be
      executed before the new one.

   Media Gateway Controllers that want to guarantee consistent operation
   of the Media Gateway may use the following rules. These rules are
   with respect to commands that are in different transactions.
   Commands that are in the same transaction are executed in order (see
   section 8).

   1. When a Media Gateway handles several Terminations, commands
      pertaining to the different Terminations may be sent in parallel,
      for example following a model where each Termination (or group of
      Terminations) is controlled by its own process or its own thread.

   2. On a Termination, there should normally be at most one outstanding
      command (Add or Modify or Move), unless the outstanding commands
      are in the same transaction. However, a Subtract command may be

issued at any time. In consequence, a Media Gateway may sometimes
receive a Modify command that applies to a previously subtracted
Termination. Such commands should be ignored, and an error code
should be returned.

   3. On a given Termination, there should normally be at most one
      outstanding Notify command at any time.

   4. In some cases, an implicitly or explicitly wildcarded Subtract
      command that applies to a group of Terminations may step in front
      of a pending Add command. The Media Gateway Controller should
      individually delete all Terminations for which an Add command was
      pending at the time of the global Subtract command. Also, new Add
      commands for Terminations named by the wild-carding (or implied in
      a Multiplex descriptor) should not be sent until the wild-carded
      Subtract command is acknowledged.

   5. AuditValue and AuditCapability are not subject to any sequencing.

   6. ServiceChange shall always be the first command sent by a MG as
      defined by the restart procedure. Any other command or response
      must be delivered after this ServiceChange command.

   These rules do not affect the command responder, which should always
   respond to commands.

9.2 Protection against Restart Avalanche

   In the event that a large number of Media Gateways are powered on
   simultaneously and they were to all initiate a ServiceChange
   transaction, the Media Gateway Controller would very likely be
   swamped, leading to message losses and network congestion during the
   critical period of service restoration. In order to prevent such
   avalanches, the following behavior is suggested:

   1. When a Media Gateway is powered on, it should initiate a restart
      timer to a random value, uniformly distributed between 0 and a
      maximum waiting delay (MWD). Care should be taken to avoid
      synchronicity of the random number generation between multiple
      Media Gateways that would use the same algorithm.

   2. The Media Gateway should then wait for either the end of this
      timer or the detection of a local user activity, such as for
      example an off-hook transition on a residential Media Gateway.

   3. When the timer elapses, or when an activity is detected, the Media
      Gateway should initiate the restart procedure.

   The restart procedure simply requires the MG to guarantee that the
   first message that the Media Gateway Controller sees from this MG is
   a ServiceChange message informing the Media Gateway Controller about
   the restart.

   Note - The value of MWD is a configuration parameter that depends on
   the type of the Media Gateway. The following reasoning may be used to
   determine the value of this delay on residential gateways.

   Media Gateway Controllers are typically dimensioned to handle the
   peak hour traffic load, during which, in average, 10% of the lines
   will be busy, placing calls whose average duration is typically 3
   minutes. The processing of a call typically involves 5 to 6 Media
   Gateway Controller transactions between each Media Gateway and the
   Media Gateway Controller. This simple calculation shows that the
   Media Gateway Controller is expected to handle 5 to 6 transactions
   for each Termination, every 30 minutes on average, or, to put it
   otherwise, about one transaction per Termination every 5 to 6 minutes
   on average. This suggests that a reasonable value of MWD for a
   residential gateway would be 10 to 12 minutes. In the absence of
   explicit configuration, residential gateways should adopt a value of
   600 seconds for MWD.

   The same reasoning suggests that the value of MWD should be much
   shorter for trunking gateways or for business gateways, because they
   handle a large number of Terminations, and also because the usage
   rate of these Terminations is much higher than 10% during the peak
   busy hour, a typical value being 60%. These Terminations, during the
   peak hour, are this expected to contribute about one transaction per
   minute to the Media Gateway Controller load. A reasonable algorithm
   is to make the value of MWD per "trunk" Termination six times shorter
   than the MWD per residential gateway, and also inversely proportional
   to the number of Terminations that are being restarted. For example
   MWD should be set to 2.5 seconds for a gateway that handles a T1
   line, or to 60 milliseconds for a gateway that handles a T3 line.

10. SECURITY CONSIDERATIONS

This section covers security when using the protocol in an IP
environment.

10.1 Protection of Protocol Connections

A security mechanism is clearly needed to prevent unauthorized
entities from using the protocol defined in this document for
setting
up unauthorized calls or interfering with authorized calls. The
security mechanism for the protocol when transported over IP
networks
is IPsec [RFC2401 to RFC2411].

The AH header [RFC2402] affords data origin authentication,
connectionless integrity and optional anti-replay protection of
messages passed between the MG and the MGC. The ESP header
[RFC2406]
provides confidentiality of messages, if desired. For instance,
the
ESP encryption service should be requested if the session
descriptions are used to carry session keys, as defined in SDP.

Implementations of the protocol defined in this document employing
the ESP header SHALL comply with section 5 of [RFC2406], which
defines a minimum set of algorithms for integrity checking and
encryption. Similarly, implementations employing the AH header
SHALL
comply with section 5 of [RFC2402], which defines a minimum set of
algorithms for integrity checking using manual keys.

Implementations SHOULD use IKE [RFC2409] to permit more robust
keying
options. Implementations employing IKE SHOULD support
authentication
with RSA signatures and RSA public key encryption.

10.2 Interim AH scheme

Implementation of IPsec requires that the AH or ESP header be
inserted immediately after the IP header. This cannot be easily
done
at the application level. Therefore, this presents a deployment
problem for the protocol defined in this document where the
underlying network implementation does not support IPsec.

As an interim solution, an optional AH header is defined within
the
H.248 protocol header. The header fields are exactly those of the
SPI, SEQUENCE NUMBER and DATA fields 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 a 16 bit UDP destination port encoded as 10 hex
digits.
   When the interim AH 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 the H.248 protocol SHALL implement IPsec where
the
   underlying operating system and the transport network supports
IPsec.
   Implementations of the protocol using IPv4 SHALL implement the
   interim AH scheme. However, this interim scheme SHALL NOT be used
   when the underlying network layer supports IPsec. IPv6
   implementations are assumed to support IPsec and SHALL NOT use the
   interim AH scheme.

   All implementations of the interim AH mechanism SHALL comply with
   section 5 of [RFC2402] which defines a minimum set of algorithms
for
   integrity checking using manual keys.

   The interim AH interim scheme does not provide protection against
   eavesdropping; thus forbidding third parties from monitoring the
   connections set up by a given termination. Also, it does not
provide
   protection against replay attacks. 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.

10.3 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 Remote
Descriptor.
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.

11. MG-MGC CONTROL INTERFACE

The control association between MG and MGC is initiated at MG cold
start, and announced by a ServiceChange message, but can be
changed
by subsequent events, such as failures or manual service events.
While the protocol does not have an explicit mechanism to support

multiple MGCs controlling a physical MG, it has been designed to
support the multiple logical MG (within a single physical MG) that
can be associated with different MGCs.

11.1 Multiple Virtual MGs

A physical Media Gateway may be partitioned into one or more
Virtual
MGs. A virtual MG consists of a set of statically partitioned
physical Terminations and/or sets of ephemeral Terminations. A
physical Termination is controlled by one MGC. The model does not
require that other resources be statically allocated, just
Terminations. The mechanism for allocating Terminations to
virtual
MGs is a management method outside the scope of the protocol.
Each
of the virtual MGs appears to the MGC as a complete MG client.

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

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

If recovering from the event requires manipulation of a physical
interface's state, only one MGC should do so. These issues are
resolved by allowing any of the MGCs to create EventsDescriptors
to
be notified of such events, but only one MGC can have read/write
access to the physical interface properties; all other MGCs have

read-only access. The management mechanism is used to designate
which MGC has read/write capability, and is designated the Master
MGC.

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

ServiceChange may only be applied to a Termination or set of
Terminations partitioned to the Virtual MG or created (in the case
of
ephemeral Terminations) by that Virtual MG.

11.2 Cold Start

A MG is pre-provisioned by a management mechanism outside the
scope
of this protocol with a Primary and (optionally) an ordered list
of
Secondary MGCs. Upon a cold start of the MG, it will issue a
ServiceChange command with a "Restart" method, on the Root
Termination to its primary MGC. If the MGC accepts the MG, it
will
send a Transaction Accept, with the ServiceChangeMgcId set to
itself.
If the MG receives an ServiceChangeMgcId not equal to the MGC it
contacted, it sends a ServiceChange to the MGC specified in the
ServiceChangeMgcId. It continues this process until it gets a
controlling MGC to accept its registration, or it fails to get a
reply. Upon failure to obtain a reply, either from the Primary
MGC,
or a designated successor, the MG tries its pre-provisioned
Secondary
MGCs, in order. If the MG is unable to establish a control
relationship with any MGC, it shall wait a random amount of time
as
described in section 9.2 and then start contacting its primary,
and
if necessary, its secondary MGCs again.

It is possible that the reply to a ServiceChange with Restart will
be
lost, and a command will be received by the MG prior to the
receipt
of the ServiceChange response. The MG shall issue error 505 –
Command Received before Restart Response.

11.3 Negotiation of Protocol Version

The first ServiceChange command from an MG shall contain the
version
number of the protocol supported by the MG in the
ServiceChangeVersion parameter. Upon receiving such a message, if
the
MGC supports only a lower version, then the MGC shall send a
ServiceChangeReply with the lower version and thereafter all the
messages between MG and MGC shall conform to the lower version of
the
protocol. If the MG is unable to comply and it has established

   a transport connection to the MGC, it should close that connection.
   In any event, it should reject all subsequent requests from the MGC
   with Error 406 Version Not supported.

   If the MGC supports a higher version than the MG but is able to
   support the lower version proposed by the MG, it shall send a
   ServiceChangeReply with the lower version and thereafter all the
   messages between MG and MGC shall conform to the lower version of the
   protocol. If the MGC is unable to comply, it shall reject the
   association, with Error 406 Version Not Supported.

   Protocol version negotiation may also occur at "handoff" and
   "failover" ServiceChanges.

   When extending the protocol with new versions, the following rules
   should be followed.

   1. Existing protocol elements, i.e., procedures, parameters,
      descriptor, property, values, should not be changed unless a
      protocol error needs to be corrected or it becomes necessary to
      change the operation of the service that is being supported by the
      protocol.

   2. The semantics of a command, a parameter, descriptor, property,
      value should not be changed.

   3. Established rules for formatting and encoding messages and
      parameters should not be modified.

   4. When information elements are found to be obsolete they can be
      marked as not used. However, the identifier for that information
      element will be marked as reserved. In that way it can not be used
      in future versions.

11.4 Failure of an MG

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

   Allowing the MGC to send duplicate messages to both MGs accommodates
   pairs of MGs that are capable of redundant failover of one of the

MGs. Only the Working MG shall accept or reject transactions.
Upon
   failover, the Primary MG sends a ServiceChange command with a
   "Failover" method and a "MG Impending Failure" reason. The MGC
then

   uses the secondary MG as the active MG. When the error condition
is
   repaired, the Working MG can send a "ServiceChange" with a
"Restart"
   method.

11.5 Failure of an MGC

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

   In partial failure, or manual maintenance reasons, an MGC may wish
to
   direct its controlled MGs to use a different MGC. To do so, it
sends
   a ServiceChange method to the MG with a "HandOff" method, and its
   designated replacement in ServiceChangeMgcId. The MG should send a
   ServiceChange message with a "Handoff" method and a "MGC directed
   change" reason to the designated MGC. If it fails to get a reply,
or
   fails to see an Audit command subsequently, it should behave as if
   its MGC failed, and start contacting secondary MGCs. If the MG is
   unable to establish a control relationship with any MGC, it shall
   wait a random amount of time as described in section 9.2 and then
   start contacting its primary, and if necessary, its secondary MGCs
   again.

   No recommendation is made on how the MGCs involved in the Handoff
   maintain state information; this is considered to be out of scope
of
   this recommendation. The MGC and MG may take the following steps
when
   Handoff occurs. When the MGC initiates a HandOff, the handover
   should be transparent to Operations on the Media Gateway.
   Transactions can be executed in any order, and could be in
progress
   when the ServiceChange is executed. Accordingly, commands in

progress continue, transaction replies are sent to the new MGC
(after
   a new control association is established), and the MG should
expect
   outstanding transaction replies from the new MGC. No new messages
   shall be sent to the new MGC until the control association is
   established. Repeated transaction requests shall be directed to
the
   new MGC. The MG shall maintain state on all terminations and
   contexts.

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

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

12. PACKAGE DEFINITION

   The primary mechanism for extension is by means of Packages.
Packages
   define additional Properties, Events, Signals and Statistics that
may
   occur on Terminations.

   Packages defined by IETF will appear in separate RFCs.

   Packages defined by ITU-T may appear in the relevant
recommendations
   (e.g. as annexes).

   1. A public document or a standard forum document, which can be
      referenced as the document that describes the package following
      the guideline above, should be specified.

   2. The document shall specify the version of the Package that it
      describes.

   3. The document should be available on a public web server and
should
      have a stable URL. The site should provide a mechanism to
provide

comments and appropriate responses should be returned.

12.1 Guidelines for defining packages

   Packages define Properties, Events, Signals, and Statistics.

   Packages may also define new error codes according to the
guidelines
   given in section 13.2. This is a matter of documentary
convenience:
   the package documentation is submitted to IANA in support of the
   error code registration. If a package is modified, it is
unnecessary
   to provide IANA with a new document reference in support of the
error
   code unless the description of the error code itself is modified.

   Names of all such defined constructs shall consist of the
PackageID
   (which uniquely identifies the package) and the ID of the item
(which
   uniquely identifies the item in that package). In the text
encoding
   the two shall be separated by a forward slash ("/") character.
   Example: togen/playtone is the text encoding to refer to the play
   tone signal in the tone generation package.

   A Package will contain the following sections:

12.1.1 Package

   Overall description of the package, specifying:

   *   Package Name: only descriptive,
   *   PackageID: Is an identifier
   *   Description:
   *   Version: A new version of a package can only add additional
       Properties, Events, Signals, Statistics and new possible values
       for an existing parameter described in the original package. No
       deletions or modifications shall be allowed. A version is an
       integer in the range from 1 to 99.

   *   Extends (Optional): A package may extend an existing package.
The
       version of the original package must be specified. When a
package
       extends another package it shall only add additional
Properties,

Events, Signals, Statistics and new possible values for an
existing parameter described in the original package. An
extended
package shall not redefine or overload a name defined in the
original package. Hence, if package B version 1 extends
package A
version 1, version 2 of B will not be able to extend the A
version
2 if A version 2 defines a name already in B version 1.

12.1.2 Properties

   Properties defined by the package, specifying:

   *   Property Name: only descriptive.
   *   PropertyID: Is an identifier
   *   Description:
   *   Type: One of:
       String: UTF-8 string
       Integer: 4 byte signed integer
       Double: 8 byte signed integer
       Character: Unicode UTF-8 encoding of a single letter.
                  Could be more than one octet.
          Enumeration: One of a list of possible unique values (See
12.3)
          Sub-list: A list of several values from a list
          Boolean

   *   Possible Values:
   *   Defined in: Which Megaco descriptor the property is defined in.
       LocalControl is for stream dependent properties.
TerminationState
       is for stream independent properties. These are expected to be
       the most common cases, but it is possible for properties to be
       defined in other descriptors.

   *   Characteristics: Read / Write or both, and (optionally),
global:
       Indicates whether a property is read-only, or read-write, and
if
       it is global. If Global is omitted, the property is not
global.
       If a property is declared as global, the value of the property
is
       shared by all terminations realizing the package.

12.1.3 Events

   Events defined by the package, specifying:

* Event name: only descriptive.
    * EventID: Is an identifier
    * Description:
    * EventsDescriptor Parameters: Parameters used by the MGC to
       configure the event, and found in the EventsDescriptor. See
       section 12.2.

    * ObservedEventsDescriptor Parameters: Parameters returned to the
       MGC in Notify requests and in replies to command requests from
       the MGC that audit ObservedEventsDescriptor, and found in the
       ObservedEventsDescriptor. See section 12.2.

12.1.4 Signals

    * Signals defined by the package, specifying:
    * Signal Name: only descriptive.
    * SignalID: Is an identifier. SignalID is used in a
       SignalsDescriptor
    * Description
    * SignalType: One of:
       - OO (On/Off)
       - TO (TimeOut)
       - BR (Brief)

       Note - SignalType may be defined such that it is dependent
on
       the value of one or more parameters. Signals that would be
       played with SignalType BR or TO should have a default
duration.
       The package has to define the default duration and
signalType.

    * Duration: in hundredths of seconds
    * Additional Parameters: See section 12.2

12.1.5 Statistics

    Statistics defined by the package, specifying:

    * Statistic name: only descriptive.

    * StatisticID: Is an identifier. StatisticID is used in a
       StatisticsDescriptor.
    * Description
    * Units: unit of measure, e.g. milliseconds, packets.

12.1.6 Procedures

    Additional guidance on the use of the package.

12.2 Guidelines to defining Properties, Statistics and Parameters to

Events and Signals.

   * Parameter Name: only descriptive
   * ParameterID: Is an identifier
   * Type: One of:
        String: UTF-8 octet string
        Integer: 4 octet signed integer
        Double: 8 octet signed integer
        Character: Unicode UTF-8 encoding of a single letter. Could
be
        more than one octet.
        Enumeration: One of a list of possible unique values (See
12.3)
        Sub-list: A list of several values from a list (not
supported
        for statistics)
        Boolean

   * Possible values:
   * Description:

12.3 Lists

   Possible values for parameters include enumerations. Enumerations
   may be defined in a list. It is recommended that the list be IANA
   registered so that packages that extend the list can be defined
   without concern for conflicting names.

12.4 Identifiers

   Identifiers in text encoding shall be strings of up to 64
characters,
   containing no spaces, starting with an alphanumeric character and
   consisting of alphanumeric characters and / or digits, and
possibly
   including the special character underscore ("_").

   Identifiers in binary encoding are 2 octets long.

   Both text and binary values shall be specified for each
identifier,
   including identifiers used as values in enumerated types.

12.5 Package Registration

   A package can be registered with IANA for interoperability
reasons.
   See section 13 for IANA considerations.

13. IANA CONSIDERATIONS

13.1 Packages

   The following considerations SHALL be met to register a package with
   IANA:

   1. A unique string name, unique serial number and version number is
      registered for each package. The string name is used with text
      encoding. The serial number shall be used with binary encoding.
      Serial Numbers 60000-64565 are reserved for private use. Serial
      number 0 is reserved.

   2. A contact name, email and postal addresses for that contact shall
      be specified. The contact information shall be updated by the
      defining organization as necessary.

   3. A reference to a document that describes the package, which should
      be public:

      The document shall specify the version of the Package that it
      describes.

      If the document is public, it should be located on a public web
      server and should have a stable URL. The site should provide a
      mechanism to provide comments and appropriate responses should be
      returned.

   4. Packages registered by other than recognized standards bodies
      shall have a minimum package name length of 8 characters.

   5. All other package names are first come-first served if all other
      conditions are met

13.2 Error Codes

   The following considerations SHALL be met to register an error code
   with IANA:

   1. An error number and a one line (80 character maximum) string is
      registered for each error.

2. A complete description of the conditions under which the error
is
       detected shall be included in a publicly available document.
The
       description shall be sufficiently clear to differentiate the
error
       from all other existing error codes.

    3. The document should be available on a public web server and
should
       have a stable URL.

    4. Error numbers registered by recognized standards bodies shall
have
       3 or 4 character error numbers.

    5. Error numbers registered by all other organizations or
individuals
       shall have 4 character error numbers.

    6. An error number shall not be redefined, nor modified except by
the
       organization or individual that originally defined it, or their
       successors or assigns.

13.3 ServiceChange Reasons

    The following considerations SHALL be met to register service
change
    reason with IANA:

    1. A one phrase, 80-character maximum, unique reason code is
       registered for each reason.

    2. A complete description of the conditions under which the reason
is
       used is detected shall be included in a publicly available
       document. The description shall be sufficiently clear to
       differentiate the reason from all other existing reasons.

    3. The document should be available on a public web server and
should
       have a stable URL.

ANNEX A: BINARY ENCODING OF THE PROTOCOL (NORMATIVE)

    This Annex specifies the syntax of messages using the notation
    defined in ASN.1 [ITU-T Recommendation X.680 (1997): Information
    Technology - Abstract Syntax Notation One (ASN.1) - Specification
of
    basic notation.]. Messages shall be encoded for transmission by
    applying the basic encoding rules specified in [ITU-T
Recommendation
    X.690(1994) Information Technology - ASN.1 Encoding Rules:
    Specification of Basic Encoding Rules (BER)].

A.1 Coding of wildcars

   The use of wildcards ALL and CHOOSE is allowed in the protocol. This
   allows a MGC to partially specify Termination IDs and let the MG
   choose from the values that conform to the partial specification.
   Termination IDs may encode a hierarchy of names. This hierarchy is
   provisioned. For instance, a TerminationID may consist of a trunk
   group, a trunk within the group and a circuit. Wildcarding must be
   possible at all levels. The following paragraphs explain how this is
   achieved.

   The ASN.1 description uses octet strings of up to 8 octets in length
   for Termination IDs. This means that Termination IDs consist of at
   most 64 bits. A fully specified Termination ID may be preceded by a
   sequence of wildcarding fields. A wildcarding field is one octet in
   length. Bit 7 (the most significant bit) of this octet specifies
   what type of wildcarding is invoked: if the bit value equals 1, then
   the ALL wildcard is used; if the bit value if 0, then the CHOOSE
   wildcard is used. Bit 6 of the wildcarding field specifies whether
   the wildcarding pertains to one level in the hierarchical naming
   scheme (bit value 0) or to the level of the hierarchy specified in
   the wildcarding field plus all lower levels (bit value 1). Bits 0
   through 5 of the wildcarding field specify the bit position in the
   Termination ID at which the starts.

   We illustrate this scheme with some examples. In these examples, the
   most significant bit in a string of bits appears on the left hand
   side.

   Assume that Termination IDs are three octets long and that each octet
   represents a level in a hierarchical naming scheme. A valid
   Termination ID is

         00000001 00011110 01010101.

   Addressing ALL names with prefix 00000001 00011110 is done as
   follows:

         wildcarding field: 10000111
         Termination ID: 00000001 00011110 xxxxxxxx.

The values of the bits labeled "x" is irrelevant and shall be ignored
by the receiver.

Indicating to the receiver that is must choose a name with 00011110
as the second octet is done as follows:

        wildcarding fields: 00010111 followed by 00000111
        Termination ID: xxxxxxxx 00011110 xxxxxxxx.

The first wildcard field indicates a CHOOSE wildcard for the level in
the naming hierarchy starting at bit 23, the highest level in our
assumed naming scheme. The second wildcard field indicates a CHOOSE
wildcard for the level in the naming hierarchy starting at bit 7, the
lowest level in our assumed naming scheme.

Finally, a CHOOSE-wildcarded name with the highest level of the name
equal to 00000001 is specified as follows:

        wildcard field: 01001111
        Termination ID: 0000001 xxxxxxxx xxxxxxxx .

Bit value 1 at bit position 6 of the first octet of the wildcard
field indicates that the wildcarding pertains to the specified level
in the naming hierarchy and all lower levels.

Context IDs may also be wildcarded. In the case of Context IDs,
however, specifying partial names is not allowed. Context ID 0x0
SHALL be used to indicate the NULL Context, Context ID 0xFFFFFFFE
SHALL be used to indicate a CHOOSE wildcard, and Context ID
0xFFFFFFFF SHALL be used to indicate an ALL wildcard.

TerminationID 0xFFFFFFFFFFFFFFFF SHALL be used to indicate the ROOT
Termination.

A.2 ASN.1 syntax specification

This section contains the ASN.1 specification of the H.248 protocol
syntax.

NOTE - In case a transport mechanism is used that employs
application
    level framing, the definition of Transaction below changes. Refer
to
    the annex defining the transport mechanism for the definition that
    applies in that case.

    NOTE - The ASN.1 specification below contains a clause defining
    TerminationIDList as a sequence of TerminationIDs. The length of
    this sequence SHALL be one, except possibly when used in
    contextAuditResult.

    MEDIA-GATEWAY-CONTROL DEFINITIONS AUTOMATIC TAGS::= BEGIN

    MegacoMessage ::= SEQUENCE
    {
        authHeader   AuthenticationHeader OPTIONAL,
        mess   Message
    }
    AuthenticationHeader ::= SEQUENCE
    {
        secParmIndex   SecurityParmIndex,
        seqNum   SequenceNum,
        ad   AuthData
    }

    SecurityParmIndex ::= OCTET STRING(SIZE(4))

    SequenceNum   ::= OCTET STRING(SIZE(4))

    AuthData   ::= OCTET STRING (SIZE (12..32))

    Message ::= SEQUENCE
    {   version   INTEGER(0..99),
        -- The version of the protocol defined here is equal to 1.
        mId   MId,   -- Name/address of message originator
        messageBody   CHOICE
        {
                messageError   ErrorDescriptor,
                transactions   SEQUENCE OF Transaction
        },
        ...
    }

    MId ::= CHOICE
    {

```
        ip4Address                     IP4Address,
        ip6Address                     IP6Address,
        domainName                     DomainName,
        deviceName                     PathName,
        mtpAddress                     OCTET STRING(SIZE(2)),
     -- Addressing structure of mtpAddress:
     --        15                 0
     --        | PC         | NI |
     --        14 bits    2 bits
          ...
     }

     DomainName ::= SEQUENCE
     {
```

```
        name            IA5String,
        -- The name starts with an alphanumeric digit followed by a
        -- sequence of alphanumeric digits, hyphens and dots.  No two
        -- dots shall occur consecutively.
        portNumber      INTEGER(0..65535) OPTIONAL
     }

     {
        address         OCTET STRING (SIZE(4)),
        portNumber      INTEGER(0..65535) OPTIONAL
     }

     IP6Address ::= SEQUENCE
     {
        address         OCTET STRING (SIZE(16)),
        portNumber      INTEGER(0..65535) OPTIONAL
     }

     PathName ::= IA5String(SIZE (1..64))
     -- See section A.3

     Transaction ::= CHOICE
     {
        transactionRequest      TransactionRequest,
        transactionPending      TransactionPending,
        transactionReply        TransactionReply,
        transactionResponseAck  TransactionResponseAck,
            -- use of response acks is dependent on underlying
     transport
        ...
     }

     TransactionId ::= INTEGER(0..4294967295)  -- 32 bit unsigned
integer

     TransactionRequest ::= SEQUENCE
     {
```

```
      transactionId          TransactionId,
      actions                SEQUENCE OF ActionRequest,
      ...
   }

   TransactionPending ::= SEQUENCE
   {
      transactionId          TransactionId,
      ...
   }

   TransactionReply ::= SEQUENCE
```

```
   {
      transactionId          TransactionId,
      immAckRequired         NULL OPTIONAL,  transactionResult
      CHOICE
      {
          transactionError   ErrorDescriptor,
          actionReplies      SEQUENCE OF ActionReply
      },
      ...
   }

   TransactionResponseAck ::= SEQUENCE
   {
      firstAck       TransactionId,
      lastAck        TransactionId OPTIONAL
   }

   ErrorDescriptor ::= SEQUENCE
   {
      errorCode      ErrorCode,
      errorText      ErrorText OPTIONAL
   }

   ErrorCode ::= INTEGER(0..65535)
   -- See section 13 for IANA considerations w.r.t. error codes

   ErrorText ::= IA5String

   ContextID ::= INTEGER(0..4294967295)

   -- Context NULL Value: 0
   -- Context CHOOSE Value: 429467294 (0xFFFFFFFE)
   -- Context ALL Value: 4294967295 (0xFFFFFFFF)


   ActionRequest ::= SEQUENCE
   {
      contextId              ContextID,
      contextRequest         ContextRequest OPTIONAL,
```

```
          contextAttrAuditReq       ContextAttrAuditRequest OPTIONAL,
          commandRequests           SEQUENCE OF CommandRequest
     }

     ActionReply ::= SEQUENCE
     {
          contextId                 ContextID,
          errorDescriptor           ErrorDescriptor OPTIONAL,
          contextReply              ContextRequest OPTIONAL,
```

```
          commandReply              SEQUENCE OF CommandReply
     }

     ContextRequest ::= SEQUENCE
     {
          priority                  INTEGER(0..15) OPTIONAL,
          emergency                 BOOLEAN OPTIONAL,
          topologyReq               SEQUENCE OF TopologyRequest OPTIONAL,
          ...
     }

     ContextAttrAuditRequest ::= SEQUENCE
     {
     topology    NULL OPTIONAL,
          emergency       NULL OPTIONAL,
          priority        NULL OPTIONAL,
          ...
     }

     CommandRequest ::= SEQUENCE
     {
          command                   Command,
          optional                  NULL OPTIONAL,
          wildcardReturn            NULL OPTIONAL,
          ...
     }

     Command ::= CHOICE
     {
          addReq                    AmmRequest,
          moveReq                   AmmRequest,
          modReq                    AmmRequest,
          -- Add, Move, Modify requests have the same parameters
          subtractReq               SubtractRequest,
          auditCapRequest           AuditRequest,
          auditValueRequest         AuditRequest,
          notifyReq                 NotifyRequest,
          serviceChangeReq          ServiceChangeRequest,
          ...
     }

     CommandReply ::= CHOICE
```

```
     {
            addReply                 AmmsReply,
            moveReply                AmmsReply,
            modReply                 AmmsReply,
            subtractReply            AmmsReply,
            -- Add, Move, Modify, Subtract replies have the same
parameters
```

```
            auditCapReply            AuditReply,
            auditValueReply          AuditReply,
            notifyReply              NotifyReply,
            serviceChangeReply       ServiceChangeReply,
            ...
     }

     TopologyRequest ::= SEQUENCE
     {
            terminationFrom          TerminationID,
            terminationTo            TerminationID,
            topologyDirection        ENUMERATED
            {
                    bothway(0),
                    isolate(1),
                    oneway(2)
            }
     }

     AmmRequest ::= SEQUENCE
     {
            terminationID            TerminationIDList,
            descriptors              SEQUENCE OF    AmmDescriptor,
            -- At most one descriptor of each type (see AmmDescriptor)
            -- allowed in the sequence.
            ...
     }

     AmmDescriptor ::= CHOICE
     {
            mediaDescriptor               MediaDescriptor,
            modemDescriptor               ModemDescriptor,
            muxDescriptor                 MuxDescriptor,
            eventsDescriptor              EventsDescriptor,
            eventBufferDescriptor         EventBufferDescriptor,
            signalsDescriptor             SignalsDescriptor,
            digitMapDescriptor            DigitMapDescriptor,
            auditDescriptor               AuditDescriptor,
            ...
     }

     AmmsReply ::= SEQUENCE
     {
            terminationID            TerminationIDList,
```

```
        terminationAudit       TerminationAudit OPTIONAL,
        ...
    }
```

```
    SubtractRequest ::= SEQUENCE
    {
        terminationID          TerminationIDList,
        auditDescriptor        AuditDescriptor OPTIONAL,
        ...
    }

    AuditRequest ::= SEQUENCE
    {
        terminationID          TerminationID,
        auditDescriptor        AuditDescriptor,
        ...
    }

    AuditReply ::= SEQUENCE
    {
        terminationID          TerminationID,
        auditResult            AuditResult,
        ...
    }

    AuditResult ::= CHOICE
    {
        contextAuditResult     TerminationIDList,
        terminationAuditResult  TerminationAudit
    }


    TerminationAudit ::= SEQUENCE OF AuditReturnParameter

    AuditReturnParameter ::= CHOICE
    {
        errorDescriptor                ErrorDescriptor,
        mediaDescriptor                MediaDescriptor,
        modemDescriptor                ModemDescriptor,
        muxDescriptor                  MuxDescriptor,
        eventsDescriptor               EventsDescriptor,
        eventBufferDescriptor          EventBufferDescriptor,
        signalsDescriptor              SignalsDescriptor,
        digitMapDescriptor             DigitMapDescriptor,
        observedEventsDescriptor       ObservedEventsDescriptor,
        statisticsDescriptor           StatisticsDescriptor,
        packagesDescriptor             PackagesDescriptor,
        emptyDescriptors               AuditDescriptor,
        ...
    }
```

```
AuditDescriptor ::= SEQUENCE
{
     auditToken      BIT STRING
     {
             muxToken(0), modemToken(1), mediaToken(2),
             eventsToken(3), signalsToken(4),
             digitMapToken(5), statsToken(6),
             observedEventsToken(7),
             packagesToken(8), eventBufferToken(9)
     } OPTIONAL,
     ...
}

NotifyRequest ::= SEQUENCE
{
     terminationID                 TerminationIDList,
     observedEventsDescriptor      ObservedEventsDescriptor,
     errorDescriptor               ErrorDescriptor OPTIONAL,
     ...
}

NotifyReply ::= SEQUENCE
{
     terminationID                 TerminationIDList OPTIONAL,
     errorDescriptor               ErrorDescriptor OPTIONAL,
     ...
}

ObservedEventsDescriptor ::= SEQUENCE
{
     requestId                     RequestID,
     observedEventLst              SEQUENCE OF ObservedEvent
}

ObservedEvent ::= SEQUENCE
{
     eventName                     EventName,
     streamID                      StreamID OPTIONAL,
     eventParList                  SEQUENCE OF EventParameter,
     timeNotation                  TimeNotation OPTIONAL,
     ...
}

EventName ::= PkgdName

EventParameter ::= SEQUENCE
{
     eventParameterName            Name,
```

```
      value                           Value
   }

   ServiceChangeRequest ::= SEQUENCE
   {
       terminationID                   TerminationIDList,
       serviceChangeParms              ServiceChangeParm,
       ...
   }

   ServiceChangeReply ::= SEQUENCE
   {
       terminationID                   TerminationIDList,
       serviceChangeResult             ServiceChangeResult,
       ...
   }

   -- For ServiceChangeResult, no parameters are mandatory.  Hence
the
   -- distinction between ServiceChangeParm and ServiceChangeResParm.

   ServiceChangeResult ::= CHOICE
   {
       errorDescriptor                 ErrorDescriptor,
       serviceChangeResParms           ServiceChangeResParm
   }

   WildcardField ::= OCTET STRING(SIZE(1))

   TerminationID ::= SEQUENCE
   {
       wildcard        SEQUENCE OF WildcardField,
       id              OCTET STRING(SIZE(1..8)),
       ...
   }
   -- See Section A.1 for explanation of wildcarding mechanism.
   -- Termination ID 0xFFFFFFFFFFFFFFFF indicates the ROOT
Termination.

   TerminationIDList ::= SEQUENCE OF TerminationID

   MediaDescriptor ::= SEQUENCE
   {

       termStateDescr  TerminationStateDescriptor OPTIONAL,
       streams         CHOICE
               {
                       oneStream       StreamParms,
                       multiStream     SEQUENCE OF StreamDescriptor
               },
```

```
        ...
    }

    StreamDescriptor ::= SEQUENCE
    {
        streamID                        StreamID,
        streamParms                     StreamParms
    }

    StreamParms ::= SEQUENCE
    {
        localControlDescriptor    LocalControlDescriptor OPTIONAL,
        localDescriptor           LocalRemoteDescriptor OPTIONAL,
        remoteDescriptor          LocalRemoteDescriptor OPTIONAL,
        ...
    }

    LocalControlDescriptor ::= SEQUENCE
    {
        streamMode      StreamMode OPTIONAL,
        reserveValue    BOOLEAN,
        reserveGroup    BOOLEAN,
        propertyParms   SEQUENCE OF PropertyParm,
        ...
    }

    StreamMode ::= ENUMERATED
    {
        sendOnly(0),
        recvOnly(1),
        sendRecv(2),
        inactive(3),
        loopBack(4),
                ...
    }

    -- In PropertyParm, value is a SEQUENCE OF octet string.  When sent
    -- by an MGC the interpretation is as follows:
    -- empty sequence means CHOOSE
    -- one element sequence specifies value
    -- If the sublist field is not selected, a longer sequence means
    -- "choose one of the values" (i.e. value1 OR value2 OR ...)
    -- If the sublist field is selected,
    -- a sequence with more than one element encodes the value of a
    -- list-valued property (i.e. value1 AND value2 AND ...).
    -- The relation field may only be selected if the value sequence
    -- has length 1.  It indicates that the MG has to choose a value
    -- for the property. E.g., x > 3 (using the greaterThan
```

```
     -- value for relation) instructs the MG to choose any value larger
     -- than 3 for property x.
     -- The range field may only be selected if the value sequence
     -- has length 2.  It indicates that the MG has to choose a value
     -- in the range between the first octet in the value sequence and
     -- the trailing octet in the value sequence, including the
     -- boundary values.
     -- When sent by the MG, only responses to an AuditCapability
request
     -- may contain multiple values, a range, or a relation field.

     PropertyParm ::= SEQUENCE
     {
          name            PkgdName,
          value           SEQUENCE OF OCTET STRING,
          extraInfo       CHOICE
                   {
                           relation        Relation,
                           range           BOOLEAN,
                           sublist         BOOLEAN
                   } OPTIONAL,
          ...
     }


     Name ::= OCTET STRING(SIZE(2))


     PkgdName ::= OCTET STRING(SIZE(4))
     -- represents Package Name (2 octets) plus Property Name (2
octets)
     -- To wildcard a package use 0xFFFF for first two octets, choose
     -- is not allowed. To reference native property tag specified in
     -- Annex C, use 0x0000 as first two octets.
     -- Wildcarding of Package Name is permitted only if Property Name
is
     -- also wildcarded.


     Relation ::= ENUMERATED
     {
          greaterThan(0),
          smallerThan(1),
          unequalTo(2),
          ...
     }


     LocalRemoteDescriptor ::= SEQUENCE
     {
          propGrps        SEQUENCE OF PropertyGroup,
          ...
     }


     PropertyGroup ::= SEQUENCE OF PropertyParm
```

```
TerminationStateDescriptor ::= SEQUENCE
{
    propertyParms           SEQUENCE OF PropertyParm,
    eventBufferControl      EventBufferControl OPTIONAL,
    serviceState            ServiceState OPTIONAL,
    ...
}

EventBufferControl ::= ENUMERATED
{
    off(0),
    lockStep(1),
    ...
}

ServiceState ::= ENUMERATED
{
    test(0),
    outOfSvc(1),
    inSvc(2),
     ...
}

MuxDescriptor   ::= SEQUENCE
{
    muxType                 MuxType,
    termList                SEQUENCE OF TerminationID,
    nonStandardData         NonStandardData OPTIONAL,
    ...
}

MuxType ::= ENUMERATED
{
    h221(0),
    h223(1),
    h226(2),
    v76(3),
    ...
}

StreamID ::= INTEGER(0..65535)  -- 16 bit unsigned integer

EventsDescriptor ::= SEQUENCE
{
    requestID               RequestID,
    eventList               SEQUENCE OF RequestedEvent,
    ...
```

```
    }

    RequestedEvent ::= SEQUENCE
    {
        pkgdName                PkgdName,
        streamID                StreamID OPTIONAL,
        eventAction             RequestedActions OPTIONAL,
        evParList               SEQUENCE OF EventParameter,
        ...
    }

    RequestedActions ::= SEQUENCE
    {
        keepActive              BOOLEAN,
        eventDM                 EventDM OPTIONAL,
        secondEvent             SecondEventsDescriptor OPTIONAL,
        signalsDescriptor       SignalsDescriptor OPTIONAL,
        ...
    }


    EventDM ::= CHOICE
    {   digitMapName    DigitMapName,
        digitMapValue   DigitMapValue
    }

    SecondEventsDescriptor ::= SEQUENCE
    {
        requestID               RequestID,
        eventList               SEQUENCE OF SecondRequestedEvent,
        ...
    }

    SecondRequestedEvent ::= SEQUENCE
    {
        pkgdName                PkgdName,
        streamID                StreamID OPTIONAL,
        eventAction             SecondRequestedActions OPTIONAL,
        evParList               SEQUENCE OF EventParameter,
        ...
    }

    SecondRequestedActions ::= SEQUENCE
    {
        keepActive              BOOLEAN,
        eventDM                 EventDM OPTIONAL,
        signalsDescriptor       SignalsDescriptor OPTIONAL,
        ...
    }
```

```
    }
```

```
EventBufferDescriptor ::= SEQUENCE OF EventSpec

EventSpec ::= SEQUENCE
{
    eventName              EventName,
    streamID               StreamID OPTIONAL,
    eventParList           SEQUENCE OF EventParameter,
    ...
}

SignalsDescriptor ::= SEQUENCE OF SignalRequest

SignalRequest ::=CHOICE
{
    signal          Signal,
    seqSigList      SeqSigList,
    ...
}

SeqSigList ::= SEQUENCE
{
    id              INTEGER(0..65535),
    signalList      SEQUENCE OF Signal
}

Signal ::= SEQUENCE
{
    signalName             SignalName,
    streamID               StreamID OPTIONAL,
    sigType                SignalType OPTIONAL,
    duration               INTEGER (0..65535) OPTIONAL,
    notifyCompletion       NotifyCompletion OPTIONAL,
    keepActive             BOOLEAN OPTIONAL,
    sigParList             SEQUENCE OF SigParameter,
    ...
}

SignalType ::= ENUMERATED
{
    brief(0),
    onOff(1),
    timeOut(2),
    ...
}

SignalName ::= PkgdName
```

```
NotifyCompletion ::= BIT STRING
{
    onTimeOut(0), onInterruptByEvent(1),
```

```
         onInterruptByNewSignalDescr(2), otherReason(3)
    }

    SigParameter ::= SEQUENCE
    {
         sigParameterName                Name,
         value                           Value
    }

    RequestID ::= INTEGER(0..4294967295)   -- 32 bit unsigned integer

    ModemDescriptor ::= SEQUENCE
    {
         mtl                    SEQUENCE OF ModemType,
         mpl                    SEQUENCE OF PropertyParm,
         nonStandardData        NonStandardData OPTIONAL
    }

    ModemType ::= ENUMERATED
    {
         v18(0),
         v22(1),
         v22bis(2),
         v32(3),
         v32bis(4),
         v34(5),
         v90(6),
         v91(7),
         synchISDN(8),
         ...
    }

    DigitMapDescriptor ::= SEQUENCE
    {
         digitMapName           DigitMapName    OPTIONAL,
         digitMapValue          DigitMapValue   OPTIONAL
    }

    DigitMapName ::= Name

    DigitMapValue ::= SEQUENCE
    {
         startTimer             INTEGER(0..99) OPTIONAL,
         shortTimer             INTEGER(0..99) OPTIONAL,
```

```
         longTimer              INTEGER(0..99) OPTIONAL,
         digitMapBody           IA5String,
         -- See Section A.3 for explanation of digit map syntax
         ...
    }

    ServiceChangeParm ::= SEQUENCE
```

```
    {
          serviceChangeMethod      ServiceChangeMethod,
          serviceChangeAddress     ServiceChangeAddress OPTIONAL,
          serviceChangeVersion     INTEGER(0..99) OPTIONAL,
          serviceChangeProfile     ServiceChangeProfile OPTIONAL,
          serviceChangeReason      Value,
          serviceChangeDelay       INTEGER(0..4294967295) OPTIONAL,
                                       -- 32 bit unsigned integer
          serviceChangeMgcId       MId OPTIONAL,
          timeStamp                TimeNotation OPTIONAL,
          nonStandardData          NonStandardData OPTIONAL,
    }

    ServiceChangeAddress ::= CHOICE
    {
          portNumber      INTEGER(0..65535), -- TCP/UDP port number
          ip4Address      IP4Address,
          ip6Address      IP6Address,
          domainName      DomainName,
          deviceName      PathName,
          mtpAddress      OCTET STRING(SIZE(2)),
          ...
    }

    ServiceChangeResParm ::= SEQUENCE
    {
          serviceChangeMgcId       MId OPTIONAL,
          serviceChangeAddress     ServiceChangeAddress OPTIONAL,
          serviceChangeVersion     INTEGER(0..99) OPTIONAL,
          serviceChangeProfile     ServiceChangeProfile OPTIONAL,
          ...
    }

    ServiceChangeMethod ::= ENUMERATED
    {
          failover(0),
          forced(1),
          graceful(2),
          restart(3),
          disconnected(4),
          handOff(5),
```

```
          ...
    }

    ServiceChangeProfile ::= SEQUENCE
    {
          profileName     Name,
          version         INTEGER(0..99)
    }

    PackagesDescriptor ::= SEQUENCE OF PackagesItem
```

```
PackagesItem ::= SEQUENCE
{
     packageName            Name,
     packageVersion  INTEGER(0..99),
     ...
}

StatisticsDescriptor ::= SEQUENCE OF StatisticsParameter

StatisticsParameter ::= SEQUENCE
{
     statName               PkgdName,
     statValue              Value
}

NonStandardData ::= SEQUENCE
{
     nonStandardIdentifier   NonStandardIdentifier,
     data                    OCTET STRING
}

NonStandardIdentifier                   ::= CHOICE
{
     object                 OBJECT IDENTIFIER,
     h221NonStandard        H221NonStandard,
     experimental           IA5String(SIZE(8)),
 -- first two characters should be "X-" or "X+"
     ...
}

H221NonStandard ::= SEQUENCE
{    t35CountryCode1   INTEGER(0..255),
     t35CountryCode2   INTEGER(0..255), -- country, as per T.35
     t35Extension      INTEGER(0..255), -- assigned nationally
     manufacturerCode  INTEGER(0..65535), -- assigned nationally
     ...
}
```

```
TimeNotation ::= SEQUENCE
{
     date            IA5String(SIZE(8)),  -- yyyymmdd format
     time            IA5String(SIZE(8))  -- hhmmssss format
}

Value ::= OCTET STRING


END
```

A.3 Digit maps and path names

From a syntactic viewpoint, digit maps are strings with syntactic
restrictions imposed upon them. The syntax of valid digit maps is
specified in ABNF [RFC 2234]. The syntax for digit maps presented
in
this section is for illustrative purposes only. The definition of
digitMap in Annex B takes precedence in the case of differences
between the two.

```
digitMap = (digitString / LWSP "(" LWSP digitStringList LWSP ")"
LWSP)
digitStringList = digitString *( LWSP "/" LWSP digitString )
digitString = 1*(digitStringElement)
digitStringElement = digitPosition [DOT]
digitPosition = digitMapLetter / digitMapRange
digitMapRange = ("x" / LWSP "[" LWSP digitLetter LWSP "]" LWSP)
digitLetter = *((DIGIT "-" DIGIT) /digitMapLetter)
digitMapLetter = DIGIT                 ;digits 0-9
     / %x41-4B / %x61-6B               ;a-k and A-K
     / "L"    / "S"                    ;Inter-event timers
                                       ;(long, short)
     / "Z"                            ;Long duration event
DOT = %x2E ; "."
LWSP = *(WSP / COMMENT / EOL)
WSP = SP / HTAB
COMMENT = ";" *(SafeChar / RestChar / WSP) EOL
EOL = (CR [LF]) / LF
SP = %x20
HTAB = %x09
CR = %x0D
LF = %x0A
SafeChar = DIGIT / ALPHA / "+" / "-" / "&" / "!" / "_" / "/" /
 "'" / "?" / "@" / "^" / "`" / "~" / "*" / "$" / "\" /
"(" / ")" / "%" / "."
RestChar = ";" / "[" / "]" / "{" / "}" / ":" / "," / "#" /
            "<" / ">" / "=" / %x22
```

```
DIGIT = %x30-39                   ; digits 0 through 9
ALPHA = %x41-5A / %x61-7A    ; A-Z, a-z
```
A path name is also a string with syntactic restrictions imposed
upon it. The ABNF production defining it is copied from Annex B.

```
PathName = NAME *(["/"] ["*"] ["@"] (ALPHA / DIGIT)) ["*"]
NAME = ALPHA *63(ALPHA / DIGIT / "_" )
```

ANNEX B TEXT ENCODING OF THE PROTOCOL (NORMATIVE)

B.1 Coding of wildcards

   In a text encoding of the protocol, while TerminationIDs are
   arbitrary, by judicious choice of names, the wildcard character,
"*"

may be made more useful. When the wildcard character is encountered,
it will "match" all TerminationIDs having the same previous and
following characters (if appropriate). For example, if there were
TerminationIDs of R13/3/1, R13/3/2 and R13/3/3, the TerminationID
R13/3/* would match all of them. There are some circumstances where
ALL Terminations must be referred to. The TerminationID "*"
suffices, and is referred to as ALL. The CHOOSE TerminationID "$" may
be used to signal to the MG that it has to create an ephemeral
Termination or select an idle physical Termination.

B.2 ABNF specification

The protocol syntax is presented in ABNF according to RFC2234.

; Boolean values, indicated in the text as True and False, are
; encoded as "On" and "Off", respectively, in the ABNF.

```
megacoMessage        = LWSP [authenticationHeader SEP ] message

authenticationHeader = AuthToken EQUAL SecurityParmIndex COLON
                       SequenceNum COLON AuthData

SecurityParmIndex    = "0x" 8(HEXDIG)
SequenceNum          = "0x" 8(HEXDIG)
AuthData             = "0x" 24*64(HEXDIG)

message   = MegacopToken SLASH Version SEP mId SEP messageBody
; The version of the protocol defined here is equal to 1.

messageBody          = ( errorDescriptor / transactionList )

transactionList      = 1*( transactionRequest / transactionReply /
                       transactionPending / transactionResponseAck
)
;Use of response acks is dependent on underlying transport
```

```
transactionPending   = PendingToken EQUAL TransactionID LBRKT
RBRKT

transactionResponseAck = ResponseAckToken LBRKT transactionAck
                                   *(COMMA transactionAck) RBRKT
transactionAck = transactionID / (transactionID "-" transactionID)

transactionRequest   = TransToken EQUAL TransactionID LBRKT
                       actionRequest *(COMMA actionRequest) RBRKT

actionRequest        = CtxToken EQUAL ContextID LBRKT ((
                       contextRequest [COMMA  commandRequestList])
```

```
                             / commandRequestList) RBRKT

    contextRequest           = ((contextProperties [COMMA
contextAudit])
                             / contextAudit)

    contextProperties    = contextProperty *(COMMA contextProperty)

    ; at-most-once
    contextProperty  = (topologyDescriptor / priority /
EmergencyToken)

    contextAudit     = ContextAuditToken LBRKT
                         contextAuditProperties *(COMMA
                         contextAuditProperties) RBRKT

    ; at-most-once
    contextAuditProperties = ( TopologyToken / EmergencyToken /
                               PriorityToken )

    commandRequestList= ["O-"] commandRequest *(COMMA ["O-"]
    commandRequest)

    commandRequest       = ( ammRequest / subtractRequest / auditRequest
                           / notifyRequest / serviceChangeRequest)

    transactionReply      = ReplyToken EQUAL TransactionID LBRKT
                            [ ImmAckRequiredToken COMMA]
                            ( errorDescriptor / actionReplyList ) RBRKT

    actionReplyList       = actionReply *(COMMA actionReply )

    actionReply           = CtxToken EQUAL ContextID LBRKT
                            ( errorDescriptor / commandReply ) RBRKT

    commandReply          = (( contextProperties [COMMA commandReplyList]
)
                            / commandReplyList )
```

```
    commandReplyList      = commandReplys *(COMMA commandReplys )

    commandReplys         = (serviceChangeReply / auditReply /
ammsReply
                            / notifyReply )

    ;Add Move and Modify have the same request parameters
    ammRequest            = (AddToken / MoveToken / ModifyToken ) EQUAL
                            TerminationID [LBRKT ammParameter *(COMMA
                            ammParameter) RBRKT]
```

```
     ;at-most-once
     ammParameter            = (mediaDescriptor / modemDescriptor /
                                muxDescriptor / eventsDescriptor /
                                signalsDescriptor / digitMapDescriptor /
                                eventBufferDescriptor / auditDescriptor)

     ammsReply               = (AddToken / MoveToken / ModifyToken /
                                SubtractToken ) EQUAL TerminationID [
LBRKT
                                terminationAudit RBRKT ]

     subtractRequest         = ["W-"] SubtractToken EQUAL TerminationID
                                [ LBRKT auditDescriptor RBRKT]

     auditRequest            = ["W-"] (AuditValueToken / AuditCapToken )
                                EQUAL TerminationID LBRKT auditDescriptor
RBRKT

     auditReply              = (AuditValueToken / AuditCapToken )
                                ( contextTerminationAudit  / auditOther)

     auditOther              = EQUAL TerminationID LBRKT
                                terminationAudit RBRKT

     terminationAudit        = auditReturnParameter *(COMMA
                                auditReturnParameter)

     contextTerminationAudit = EQUAL CtxToken ( terminationIDList /
                                  LBRKT errorDescriptor RBRKT )

     auditReturnParameter = (mediaDescriptor / modemDescriptor /
                              muxDescriptor / eventsDescriptor /
                              signalsDescriptor / digitMapDescriptor /
                        observedEventsDescriptor / eventBufferDescriptor
/
                              statisticsDescriptor / packagesDescriptor
/
                               errorDescriptor / auditItem )

     auditDescriptor         = AuditToken LBRKT [ auditItem
                                *(COMMA auditItem) ] RBRKT
```

```
     notifyRequest           = NotifyToken EQUAL TerminationID
                                LBRKT ( observedEventsDescriptor
                                    [ COMMA errorDescriptor ] ) RBRKT

     notifyReply             = NotifyToken EQUAL TerminationID
                                [ LBRKT errorDescriptor RBRKT ]

     serviceChangeRequest = ServiceChangeToken EQUAL TerminationID
                                LBRKT serviceChangeDescriptor RBRKT
```

```
    serviceChangeReply   = ServiceChangeToken EQUAL TerminationID
                           [LBRKT (errorDescriptor /
                           serviceChangeReplyDescriptor) RBRKT]


    errorDescriptor    = ErrorToken EQUAL ErrorCode
                         LBRKT [quotedString] RBRKT


    ErrorCode            = 1*4(DIGIT) ; could be extended


    TransactionID        = UINT32


    mId                  = (( domainAddress / domainName )
                           [":" portNumber]) / mtpAddress / deviceName

    ; ABNF allows two or more consecutive "." although it is
meaningless
    ; in a domain name.
    domainName           = "<" (ALPHA / DIGIT) *63(ALPHA / DIGIT / "-"
/
                           ".") ">"
    deviceName           = pathNAME

    ;The values 0x0, 0xFFFFFFFE and 0xFFFFFFFF are reserved.
    ContextID            = (UINT32 / "*" / "-" / "$")

    domainAddress        = "[" (IPv4address / IPv6address) "]"
    ;RFC2373 contains the definition of IP6Addresses.
    IPv6address          = hexpart [ ":" IPv4address ]
    IPv4address          = V4hex DOT V4hex DOT V4hex DOT V4hex
    V4hex                = 1*3(DIGIT) ; "0".."225"
    ; this production, while occurring in RFC2373, is not referenced
    ; IPv6prefix          = hexpart SLASH 1*2DIGIT
    hexpart        = hexseq "::" [ hexseq ] / "::" [ hexseq ] /
hexseq
    hexseq               = hex4 *( ":" hex4)
    hex4                 = 1*4HEXDIG

    portNumber           = UINT16

    ; An mtp address is two octets long
```

```
    mtpAddress           = MTPToken LBRKT octetString RBRKT

    terminationIDList    = LBRKT TerminationID *(COMMA TerminationID)
    RBRKT

    ; Total length of pathNAME must not exceed 64 chars.
    pathNAME        = ["*"] NAME *("/" / "*"/ ALPHA / DIGIT /"_" / "$"
)
                           ["@" pathDomainName ]
```

```
      ; ABNF allows two or more consecutive "." although it is
meaningless
      ; in a path domain name.
      pathDomainName        = (ALPHA / DIGIT / "*" )
                              *63(ALPHA / DIGIT / "-" / "*" / ".")

      TerminationID         = "ROOT" / pathNAME / "$" / "*"

      mediaDescriptor = MediaToken LBRKT mediaParm *(COMMA mediaParm)
                         RBRKT

      ; at-most-once per item
      ; and either streamParm or streamDescriptor but not both
      mediaParm             = (streamParm / streamDescriptor /
                               terminationStateDescriptor)

      ; at-most-once
      streamParm            = ( localDescriptor / remoteDescriptor /
                               localControlDescriptor )

      streamDescriptor      = StreamToken EQUAL StreamID LBRKT streamParm
                               *(COMMA streamParm) RBRKT

      localControlDescriptor = LocalControlToken LBRKT localParm
                                *(COMMA localParm) RBRKT

      ; at-most-once per item
      localParm             = ( streamMode / propertyParm /
      reservedValueMode
          / reservedGroupMode )

      reservedValueMode      = ReservedValueToken EQUAL ( "ON" / "OFF"
)
      reservedGroupMode      = ReservedGroupToken EQUAL ( "ON" / "OFF"
)

      streamMode            = ModeToken EQUAL streamModes

      streamModes           = (SendonlyToken / RecvonlyToken /
      SendrecvToken /
                              InactiveToken / LoopbackToken )
```

```
      propertyParm          = pkgdName parmValue
      parmValue             = (EQUAL alternativeValue/ INEQUAL VALUE)
      alternativeValue      = ( VALUE
                               / LSBRKT VALUE *(COMMA VALUE) RSBRKT
                                 ; sublist (i.e. A AND B AND ...)
                               / LBRKT VALUE *(COMMA VALUE) RBRKT
                                 ; alternatives (i.e. A OR B OR ...)
                               / LSBRKT VALUE COLON VALUE RSBRKT )
                                 ; range
```

```
INEQUAL                 = LWSP (">" / "<" / "#" ) LWSP
LSBRKT                  = LWSP "[" LWSP
RSBRKT                  = LWSP "]" LWSP

localDescriptor         = LocalToken LBRKT octetString RBRKT

remoteDescriptor        = RemoteToken LBRKT octetString RBRKT

eventBufferDescriptor= EventBufferToken LBRKT eventSpec
                         *( COMMA eventSpec ) RBRKT
eventSpec               = pkgdName [ LBRKT eventSpecParameter
                            *(COMMA eventSpecParameter) RBRKT ]
eventSpecParameter   = (eventStream / eventOther)

eventBufferControl      = BufferToken EQUAL ( "OFF" / LockStepToken
)

terminationStateDescriptor = TerminationStateToken LBRKT
            terminationStateParm *( COMMA terminationStateParm )
RBRKT

; at-most-once per item
terminationStateParm =(propertyParm / serviceStates /
eventBufferControl )

serviceStates           = ServiceStatesToken EQUAL ( TestToken /
                          OutOfSvcToken / InSvcToken )

muxDescriptor           = MuxToken EQUAL MuxType  terminationIDList

MuxType                 = ( H221Token / H223Token / H226Token /
                        V76Token / extensionParameter )

StreamID                = UINT16
pkgdName                = (PackageName SLASH ItemID) ;specific item
                      / (PackageName SLASH "*") ;all events in package
                      / ("*" SLASH "*") ; all events supported by the
MG
PackageName             = NAME
ItemID                  = NAME
```

```
eventsDescriptor        = EventsToken EQUAL RequestID LBRKT
                         requestedEvent *( COMMA requestedEvent )
RBRKT

requestedEvent          = pkgdName [ LBRKT eventParameter
                         *( COMMA eventParameter ) RBRKT ]

; at-most-once each of KeepActiveToken , eventDM and eventStream
;at most one of either embedWithSig or embedNoSig but not both
;KeepActiveToken and embedWithSig must not both be present
```

```
    eventParameter          = ( embedWithSig / embedNoSig /
KeepActiveToken

                          /eventDM / eventStream / eventOther )

    embedWithSig            = EmbedToken LBRKT signalsDescriptor
                              [COMMA embedFirst ] RBRKT
    embedNoSig              = EmbedToken LBRKT embedFirst RBRKT

    ; at-most-once of each
    embedFirst        = EventsToken EQUAL RequestID LBRKT
             secondRequestedEvent *(COMMA secondRequestedEvent)
RBRKT

    secondRequestedEvent = pkgdName [ LBRKT secondEventParameter
                          *( COMMA secondEventParameter ) RBRKT ]

    ; at-most-once each of embedSig , KeepActiveToken, eventDM or
    ; eventStream
    ; KeepActiveToken and embedSig must not both be present
    secondEventParameter = ( EmbedSig / KeepActiveToken / eventDM /
                             eventStream / eventOther )

    embedSig  = EmbedToken LBRKT signalsDescriptor RBRKT

    eventStream             = StreamToken EQUAL StreamID

    eventOther              = eventParameterName parmValue

    eventParameterName      = NAME

    eventDM                 = DigitMapToken ((EQUAL digitMapName ) /
                              (LBRKT digitMapValue RBRKT ))

    signalsDescriptor       = SignalsToken LBRKT [ signalParm
                              *(COMMA signalParm)] RBRKT

    signalParm              = signalList / signalRequest

    signalRequest           = signalName [ LBRKT sigParameter
                              *(COMMA sigParameter) RBRKT ]
```

```
    signalList            = SignalListToken EQUAL signalListId LBRKT
                            signalListParm *(COMMA signalListParm)
RBRKT

    signalListId          = UINT16

    ;exactly once signalType, at most once duration and every signal
    ;parameter
    signalListParm        = signalRequest
```

```
      signalName           = pkgdName
      ;at-most-once sigStream, at-most-once sigSignalType,
      ;at-most-once sigDuration, every signalParameterName at most once
      sigParameter    = sigStream / sigSignalType / sigDuration /
sigOther
                      / notifyCompletion / KeepActiveToken
      sigStream            = StreamToken EQUAL StreamID
      sigOther             = sigParameterName parmValue
      sigParameterName     = NAME
      sigSignalType        = SignalTypeToken EQUAL signalType
      signalType           = (OnOffToken / TimeOutToken / BriefToken)
      sigDuration          = DurationToken EQUAL UINT16
      notifyCompletion     = NotifyCompletionToken EQUAL (LBRKT
                      notificationReason *(COMMA notificationReason)
RBRKT

      notificationReason   = ( TimeOutToken / InterruptByEventToken
                                  / InterruptByNewSignalsDescrToken
                                  / OtherReasonToken )

      observedEventsDescriptor = ObservedEventsToken EQUAL RequestID
                      LBRKT observedEvent *(COMMA observedEvent)
RBRKT

      ;time per event, because it might be buffered
      observedEvent        = [ TimeStamp LWSP COLON] LWSP
                          pkgdName [ LBRKT observedEventParameter
                         *(COMMA observedEventParameter) RBRKT ]

      ;at-most-once eventStream, every eventParameterName at most once
      observedEventParameter = eventStream / eventOther

      RequestID            = UINT32

      modemDescriptor      = ModemToken (( EQUAL modemType) /
                           (LSBRKT modemType *(COMMA modemType)
RSBRKT))
                            [ LBRKT NAME parmValue
                           *(COMMA NAME parmValue) RBRKT ]
```

```
      ; at-most-once
      modemType            = (V32bisToken / V22bisToken / V18Token /
                             V22Token / V32Token / V34Token / V90Token
/
                          V91Token / SynchISDNToken /
extensionParameter)

      digitMapDescriptor   = DigitMapToken EQUAL
```

```
                             ( ( LBRKT digitMapValue RBRKT )
                        / (digitMapName [ LBRKT digitMapValue RBRKT ])
)
   digitMapName        = NAME
   digitMapValue       = ["T" COLON Timer COMMA] ["S" COLON Timer
COMMA]
                          ["L" COLON Timer COMMA] digitMap
   Timer               = 1*2DIGIT
   digitMap =
        digitString / LWSP "(" LWSP digitStringList LWSP ")" LWSP
   digitStringList     = digitString *( LWSP "|" LWSP digitString )
   digitString         = 1*(digitStringElement)
   digitStringElement  = digitPosition [DOT]
   digitPosition       = digitMapLetter / digitMapRange
   digitMapRange       = ("x" / LWSP "[" LWSP digitLetter LWSP "]"
LWSP)
   digitLetter         = *((DIGIT "-" DIGIT ) / digitMapLetter)
   digitMapLetter      = DIGIT   ;Basic event symbols
                   / %x41-4B / %x61-6B ; a-k, A-K
                   / "L" / "S"    ;Inter-event timers (long, short)
                   / "Z"          ;Long duration modifier

   ;at-most-once, and DigitMapToken and PackagesToken are not allowed
   ;in AuditCapabilities command

   auditItem               = ( MuxToken / ModemToken / MediaToken /
                            SignalsToken / EventBufferToken /
                            DigitMapToken / StatsToken / EventsToken /
                            ObservedEventsToken / PackagesToken )

   serviceChangeDescriptor = ServicesToken LBRKT serviceChangeParm
                             *(COMMA serviceChangeParm) RBRKT

   serviceChangeParm    = (serviceChangeMethod / serviceChangeReason
/
                            serviceChangeDelay / serviceChangeAddress /
                           serviceChangeProfile / extension / TimeStamp
/
                            serviceChangeMgcId / serviceChangeVersion )

   serviceChangeReplyDescriptor = ServicesToken LBRKT
                        servChgReplyParm *(COMMA servChgReplyParm)
RBRKT

   ;at-most-once. Version is REQUIRED on first ServiceChange response
   servChgReplyParm     = (serviceChangeAddress / serviceChangeMgcId
/
                            serviceChangeProfile / serviceChangeVersion
)
   serviceChangeMethod  = MethodToken EQUAL (FailoverToken /
```

```
                               ForcedToken / GracefulToken / RestartToken
    /
                               DisconnectedToken / HandOffToken /
                               extensionParameter)

    serviceChangeReason  = ReasonToken  EQUAL VALUE
    serviceChangeDelay   = DelayToken   EQUAL UINT32
    serviceChangeAddress = ServiceChangeAddressToken EQUAL VALUE
    serviceChangeMgcId   = MgcIdToken   EQUAL mId
    serviceChangeProfile = ProfileToken EQUAL NAME SLASH Version
    serviceChangeVersion = VersionToken EQUAL Version
    extension            = extensionParameter parmValue

    packagesDescriptor   = PackagesToken LBRKT packagesItem
                            *(COMMA packagesItem) RBRKT

    Version              = 1*2(DIGIT)
    packagesItem         = NAME "-" UINT16

    TimeStamp            = Date "T" Time ; per ISO 8601:1988
    ; Date = yyyymmdd
    Date                 = 8(DIGIT)
    ; Time = hhmmssss
    Time                 = 8(DIGIT)
    statisticsDescriptor = StatsToken LBRKT statisticsParameter
                           *(COMMA statisticsParameter ) RBRKT

    ;at-most-once per item
    statisticsParameter  = pkgdName EQUAL VALUE

    topologyDescriptor   = TopologyToken LBRKT terminationA COMMA
                            terminationB COMMA topologyDirection RBRKT
    terminationA         = TerminationID
    terminationB         = TerminationID
    topologyDirection    = BothwayToken / IsolateToken / OnewayToken

    priority             = PriorityToken EQUAL UINT16

    extensionParameter   = "X"  ("-" / "+") 1*6(ALPHA / DIGIT)

    ; octetString is used to describe SDP defined in RFC2327.
    ; Caution should be taken if CRLF in RFC2327 is used.
    ; To be safe, use EOL in this ABNF.
    ; Whenever "}" appears in SDP, it is escaped by "\", e.g., "\}"
    octetString          = *(nonEscapeChar)
    nonEscapeChar        = ( "\}" / %x01-7C / %x7E-FF )
    quotedString         = DQUOTE 1*(SafeChar / RestChar/ WSP) DQUOTE

    UINT16               = 1*5(DIGIT)  ; %x0-FFFF
```

```
    UINT32               = 1*10(DIGIT) ; %x0-FFFFFFFF
```

```
    NAME                    = ALPHA *63(ALPHA / DIGIT / "_" )
    VALUE                   = quotedString / 1*(SafeChar)
    SafeChar                = DIGIT / ALPHA / "+" / "-" / "&" /
                              "!" / "_" / "/" / "'" / "?" / "@" /
                              "^" / "`" / "~" / "*" / "$" / "\" /
                              "(" / ")" / "%" / "|" / "."

    EQUAL                   = LWSP %x3D LWSP ; "="
    COLON                   = %x3A            ; ":"
    LBRKT                   = LWSP %x7B LWSP ; "{"
    RBRKT                   = LWSP %x7D LWSP ; "}"
    COMMA                   = LWSP %x2C LWSP ; ","
    DOT                     = %x2E            ; "."
    SLASH                   = %x2F            ; "/"
    ALPHA                   = %x41-5A / %x61-7A ; A-Z / a-z
    DIGIT                   = %x30-39         ; 0-9
    DQUOTE                  = %x22            ; " (Double Quote)
    HEXDIG                  = ( DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
)
    SP                      = %x20        ; space
    HTAB                    = %x09        ; horizontal tab
    CR                      = %x0D        ; Carriage return
    LF                      = %x0A        ; linefeed
    LWSP                    = *( WSP / COMMENT / EOL )
    EOL                     = (CR [LF] / LF )
    WSP                     = SP / HTAB ; white space
    SEP                     = ( WSP / EOL / COMMENT) LWSP
    COMMENT                 = ";" *(SafeChar/ RestChar / WSP / %x22) EOL
    RestChar                = ";" / "[" / "]" / "{" / "}" / ":" / "," /
"#"
   /
                              "<" / ">" / "="


    AddToken                = ("Add"               / "A")
    AuditToken              = ("Audit"             / "AT")
    AuditCapToken           = ("AuditCapability"   / "AC")
    AuditValueToken         = ("AuditValue"        / "AV")
    AuthToken               = ("Authentication"    / "AU")
    BothwayToken            = ("Bothway"           / "BW")
    BriefToken              = ("Brief"             / "BR")
    BufferToken             = ("Buffer"            / "BF")
    CtxToken                = ("Context"           / "C")
    ContextAuditToken       = ("ContextAudit"      / "CA")
    DigitMapToken           = ("DigitMap"          / "DM")
    DisconnectedToken       = ("Disconnected"      / "DC")
    DelayToken              = ("Delay"             / "DL")
    DurationToken           = ("Duration"          / "DR")
```

```
    EmbedToken              = ("Embed"             / "EB")
    EmergencyToken          = ("Emergency"         / "EM")
    ErrorToken              = ("Error"             / "ER")
```

```
EventBufferToken          = ("EventBuffer"          / "EB")
EventsToken               = ("Events"               / "E")
FailoverToken             = ("Failover"             / "FL")
ForcedToken               = ("Forced"               / "FO")
GracefulToken             = ("Graceful"             / "GR")
H221Token                 = ("H221" )
H223Token                 = ("H223" )
H226Token                 = ("H226" )
HandOffToken              = ("HandOff"              / "HO")
ImmAckRequiredToken       = ("ImmAckRequired"       / "IA")
InactiveToken             = ("Inactive"             / "IN")
IsolateToken              = ("Isolate"              / "IS")
InSvcToken                = ("InService"            / "IV")
InterruptByEventToken     = ("IntByEvent"           / "IBE")
InterruptByNewSignalsDescrToken
                          = ("IntBySigDescr"        / "IBS")
KeepActiveToken           = ("KeepActive"           / "KA")
LocalToken                = ("Local"                / "L")
LocalControlToken         = ("LocalControl"         / "O")
LockStepToken             = ("LockStep"             / "SP")
LoopbackToken             = ("Loopback"             / "LB")
MediaToken                = ("Media"                / "M")
MegacopToken              = ("MEGACO"               / "!")
MethodToken               = ("Method"               / "MT")
MgcIdToken                = ("MgcIdToTry"           / "MG")
ModeToken                 = ("Mode"                 / "MO")
ModifyToken               = ("Modify"               / "MF")
ModemToken                = ("Modem"                / "MD")
MoveToken                 = ("Move"                 / "MV")
MTPToken                  = ("MTP")
MuxToken                  = ("Mux"                  / "MX")
NotifyToken               = ("Notify"               / "N")
NotifyCompletionToken     = ("NotifyCompletion"     / "NC")
ObservedEventsToken       = ("ObservedEvents"       / "OE")
OnewayToken               = ("Oneway"               / "OW")
OnOffToken                = ("OnOff"                / "OO")
OtherReasonToken          = ("OtherReason"          / "OR")
OutOfSvcToken             = ("OutOfService"         / "OS")
PackagesToken             = ("Packages"             / "PG")
PendingToken              = ("Pending"              / "PN")
PriorityToken             = ("Priority"             / "PR")
ProfileToken              = ("Profile"              / "PF")
ReasonToken               = ("Reason"               / "RE")
RecvonlyToken             = ("ReceiveOnly"          / "RC")
ReplyToken                = ("Reply"                / "P")
```

```
RestartToken              = ("Restart"              / "RS")
RemoteToken               = ("Remote"               / "R")
ReservedGroupToken        = ("ReservedGroup"        / "RG")
ReservedValueToken        = ("ReservedValue"        / "RV")
SendonlyToken             = ("SendOnly"             / "SO")
SendrecvToken             = ("SendReceive"          / "SR")
```

```
ServicesToken              = ("Services"              / "SV")
ServiceStatesToken         = ("ServiceStates"         / "SI")
ServiceChangeToken         = ("ServiceChange"         / "SC")
ServiceChangeAddressToken  = ("ServiceChangeAddress"  / "AD")
SignalListToken            = ("SignalList"            / "SL")
SignalsToken               = ("Signals"               / "SG")
SignalTypeToken            = ("SignalType"            / "SY")
StatsToken                 = ("Statistics"            / "SA")
StreamToken                = ("Stream"                / "ST")
SubtractToken              = ("Subtract"              / "S")
SynchISDNToken             = ("SynchISDN"             / "SN")
TerminationStateToken      = ("TerminationState"      / "TS")
TestToken                  = ("Test"                  / "TE")
TimeOutToken               = ("TimeOut"               / "TO")
TopologyToken              = ("Topology"              / "TP")
TransToken                 = ("Transaction"           / "T")
ResponseAckToken           = ("TransactionResponseAck"/ "K")
V18Token                   = ("V18")
V22Token                   = ("V22")
V22bisToken                = ("V22b")
V32Token                   = ("V32")
V32bisToken                = ("V32b")
V34Token                   = ("V34")
V76Token                   = ("V76")
V90Token                   = ("V90")
V91Token                   = ("V91")
VersionToken               = ("Version"               / "V")
```

ANNEX C TAGS FOR MEDIA STREAM PROPERTIES (NORMATIVE)

   Parameters for Local descriptors and Remote descriptors are
specified
   as tag-value pairs if binary encoding is used for the protocol.
This
   annex contains the property names (PropertyID), the tags (Property
   Tag), type of the property (Type) and the values (Value). Values
   presented in the Value field when the field contains references
shall
   be regarded as "information". The reference contains the normative
   values. If a value field does not contain a reference then the
   values in that field can be considered as "normative".

   Tags are given as hexadecimal numbers in this annex. When setting
the
   value of a property, a MGC may underspecify the value according to
   one of the mechanisms specified in section 7.1.1.

   For type "enumeration" the value is represented by the value in
   brackets, e.g., Send(0), Receive(1).

   When a type is smaller than one octet, the value shall be stored
in

the low-order bits of an octet string of size 1.

C.1 General Media Attributes

PropertyID	Tag	Type	Value
Media	1001	Enumeration	Audio(0), Video(1), Data(2)
Transmission mode	1002	Enumeration	Send(0), Receive(1), Send&Receive(2)
Number of Channels	1003	Unsigned Integer	0-255
Sampling rate	1004	Unsigned Integer	0-2^32
Bitrate	1005	Integer	(0..4294967295) Note - units of 100 bit/s
ACodec	1006	Octet String	Audio Codec Type
Reference: ITU-T Rec. Q.765 - Application transport mechanism. Non-ITU codecs are defined with the appropriate standards organisation under a defined Organizational Identitifier.			
Samplepp	1007	Unsigned Integer	Maximum samples or frames per packet: 0-65535

| Silencesupp | 1008 | BOOLEAN | Silence Suppression:
|
| | | | True/false
|
  +------------------+------+------------+------------------------
--+
  | Encrypttype | 1009 | Octet | Ref.: rec. H.245
|
  | | | String |
|
  +------------------+------+------------+------------------------
--+
  | Encryptkey | 100A | Octet | SIZE(0..65535)
|
  | | | String | Encryption key
|
  +------------------+------+------------+------------------------
--+
  | Ref.: rec. H.235
|
  +------------------+------+------------+------------------------
--+

  | Echocanc | 100B | Enumeration | Echo Canceller: Off(0),
|
  | | | | G.165(1), G168(2)
|
  +------------------+------+------------+------------------------
--+
  | Gain | 100C | Unsigned | Gain in db: 0-65535
|
  | | | Integer |
|
  +------------------+------+------------+------------------------
--+
  | Jitterbuff | 100D | Unsigned | Jitter buffer size in
ms: |
  | | | Integer | 0-65535
|
  +------------------+------+------------+------------------------
--+
  | PropDelay | 100E | Unsigned | Propagation Delay:
|
  | | | Integer | 0..65535
|
  +------------------+------+------------+------------------------
--+
  | Maximum propagation delay in milliseconds for the bearer
|
  | connection between two media gateways. The maximum delay will be
|

```
   | dependent on the bearer technology.
   |
   +------------------+------+------------+----------------------
--+
   |RTPpayload        | 100F | integer    | Payload type in RTP
   |
   +------------------+------+------------+----------------------
--+
   | Profile for Audio and Video Conferences with Minimal Control
   |
   | Ref.: RFC 1890
   |
   +-------------------------------------------------------------
--+
```

C.2 Mux Properties

```
   +------------------+------+----------+--------------------------
--+
   | PropertyID       | Tag  | Type     | Value
   |
   +------------------+------+----------+--------------------------
--+
   | H.221            | 2001 | Octet    | Ref.: rec. H.245,
   |
   |                  |      | String   |
H222LogicalChannelParameters |
   +------------------+------+----------+--------------------------
--+
   | H223             | 2002 | Octet    | Ref.: rec. H.245,
   |
   |                  |      | String   |
H223LogicalChannelParameters |
   +------------------+------+----------+--------------------------
--+
   | V76              | 2003 | Octet    | Ref.: rec. H.245,
   |
   |                  |      | String   | V76LogicalChannelParameters
   |
   +------------------+------+------------+----------------------
--+
   | H2250            | 2004 | Octet    | Ref.: rec. H.245,
   |
   |                  |      | String   |
H2250LogicalChannelParameters|
   +------------------+------+----------+--------------------------
--+
```

C.3 General bearer properties

PropertyID	Tag	Type	Value
Mediatx	3001	Enumeration	Media Transport Type:
TDM Circuit(0), ATM(1), FR(2), Ipv4(3), Ipv6(4), ...			
BIR	3002	4 OCTET	Value depends on transport
			technology
NSAP	3003	1-20 OCTETS	See NSAP:
Reference: ITU X.213 Annex A			

C.4 General ATM properties

PropertyID	Property Tag	Type	Value
AESA	4001	20 OCTETS	ATM End System Address
VPVC	4002	2 x 16 bit integer	VPCI/VCI
Reference ITU-T Recommendation Q.2931			
SC	4003	Enumeration	Service Category

```
|     Reference: ATM Forum UNI 4.0:
|
|     CBR(0), nrt-VBR1(1), nrt-VBR2(2), nrt-VBR3(3), rt-VBR1(4), rt-
|
|     VBR2(5), rt-VBR3(6), UBR1(7), UBR2(8), ABR(9).
|
  +-----------------+---------+-------------+---------------------
--+
| BCOB             | 4004    | 5 bit integer  Broadband Bearer
Class |
|     Reference: ITU Recommendation Q.2961.2
|
  +-----------------+---------+-------------+---------------------
--+
| BBTC             | 4005    | 7 bit integer| Broadband Transfer
|
|                  |         |             | Capability
|
|     Reference: ITU Recommendation Q.2961
|
  +-----------------+---------+-------------+---------------------
--+
| ATC              | 4006    | Enumeration |  I.371 ATM Traffic
|
|                  |         |             | Capability
|
|     Reference: ITU Recommendation I.371:
|
|     DBR(0), SBR1(1), SBR2(2), SBR3(3), ABT/IT(4), ABT/DT(5),
ABR(6) |
  +-----------------+---------+-------------+---------------------
--+
```

```
| STC              | 4007    | 2 bits      | Susceptibility to
|
|                  |         |             | clipping
|
|      Reference: ITU Recommendation Q.2931
|
|      00 Susceptible
|
|      01 Not-susceptible
|
  +-----------------+---------+-------------+---------------------
--+
| UPCC             | 4008    | 2 bits      | User Plane
Connection |
|                  |         |             | configuration:
|
```

```
|      |       Reference: ITU Recommendation Q.2931
|
|      |       00 Pt-to-pt,
|
|      |       01 Pt-to-mpt
|
   +-----------------+---------+-------------+---------------------
--+
   | PCR0            | 4009    | 24 bit      | Peak Cell Rate (For
|
   |                 |         | integer     | CLP=0)
|
   |       Reference: ITU Recommendation Q.2931
|
   +-----------------+---------+-------------+---------------------
--+
   | SCR0            | 400A    | 24 bit      | Sustainable Cell
Rate |
   |                 |         | integer     | (For CLP=0)
|
   |       Reference: ITU Recommendation Q.2961
|
   +-----------------+---------+-------------+---------------------
--+
   | MBS0            | 400B    | 24 bit      | Maximum Burst Size
(For|
   |                 |         | integer     | CLP=0)
|
   |       Reference: ITU Recommendation Q.2961
|
   +-----------------+---------+-------------+---------------------
--+
   | PCR1            | 400C    | 24 bit      | Peak Cell Rate (For
|
   |                 |         | integer     | CLP=0+1)
|
   |       Reference: ITU Recommendation Q.2931
|
   +-----------------+---------+-------------+---------------------
--+
   | SCR1            | 400D    | 24 bit      | Sustainable Cell
Rate |
   |                 |         | integer     | (For CLP=0+1)
|
   |       Reference: ITU Recommendation Q.2961
|
   +-----------------+---------+-------------+---------------------
--+
   | MBS1            | 400E    | 24 bit      | Maximum Burst Size
(For|
   |                 |         | integer     | CLP=0+1)
|
   |       Reference: ITU Recommendation Q.2961
|
   +-----------------+---------+-------------+---------------------
--+
   | BEI             | 400F    | Boolean     | Best Effort
Indicator |
   |       Reference: ATM Forum UNI 4.0:
|
```

```
   |          0 -- do not include BEI in ATM signalling
|
   |          1 -- include BEI in ATM signalling
|
   +-----------------+---------+-------------+--------------------
--+
   | TI              | 4010    | Boolean     | Tagging
|
   |      Reference: ITU-T Recommendation Q.2961
|
   |          0 -- tagging is not allowed
|
   |          1 -- tagging is requested
|
   +-----------------+---------+-------------+--------------------
--+
```

```
   | FD              | 4011    | Boolean     | Frame Discard
|
   |      Reference: ATM Forum UNI 4.0:
|
   |          0 -- no frame discard is allowed
|
   |          1 -- frame discard is allowed
|
   +-----------------+---------+-------------+--------------------
--+
   | A2PCDV          | 4012    | 24 bit      | Acceptable 2-point
CDV |
   |                 |         | integer     |
|
   |      Reference: ITU-T Recommendation Q.2965.2
|
   +-----------------+---------+-------------+--------------------
--+
   | C2PCDV          | 4013    | 24 bit      | Cumulative 2-point
CDV |
   |                 |         | integer     |
|
   |      Reference: ITU-T Recommendation Q.2965.2
|
   +-----------------+---------+-------------+--------------------
--+
   | APPCDV          | 4014    | 24 bit      | Acceptable P-P CDV
|
   |                 |         | integer     |
|
   |      Reference: ATM Forum UNI 4.0
|
```

```
   +-----------------+---------+-------------+---------------------
--+
   | CPPCDV          | 4015    | 24 bit      | Cumulative P-P CDV
 |
   |                 |         | integer     |
 |
   |      Reference: ATM Forum UNI 4.0
 |
   +-----------------+---------+-------------+---------------------
--+
   | ACLR            | 4016    | 8 bit integer| Acceptable Cell Loss
 |
   |                 |         |             | Ratio
 |
   |      Reference: ITU-T Recommendation Q.2965.2, ATM Forum UNI 4.0
 |
   +-----------------+---------+-------------+---------------------
--+
   | MEETD           | 4017    | 16 bit      | Maximum end-to-end
 |
   |                 |         | integer     | transit delay
 |
   |      Reference: ITU-T Recommendation Q.2965.2, ATM Forum UNI 4.0
 |
   +-----------------+---------+-------------+---------------------
--+
   | CEETD           | 4018    | 16 bit      | Cumulative end-to-
end |
   |                 |         | integer     | transit delay
 |
   |      Reference: ITU-T Recommendation Q.2965.2, ATM Forum UNI 4.0
 |
   +-----------------+---------+-------------+---------------------
--+
   | QosClass        | 4019    | Integer 0-5 | QOS Class
 |
   |      Reference: ITU-T Recommendation Q.2965.1
 |
   |          Qos Class            Meaning
 |
   |          ---------            -----------------------------
 |
   |             0                Default QoS associated with the
ATC |
   |                              as defined in ITU Rec. Q.2961.2
 |
   |             1                Stringent
 |
   |             2                Tolerant
 |
   |             3                Bi-level
 |
   |             4                Unbounded
 |
   |             5                Stringent bi-level
 |
   +-----------------+---------+-------------+---------------------
--+
```

```
  | AALType         | 401A  | 1 octet     | AAL Type         |
  |     Reference: ITU Recommendation Q.2931                |
  |     00000000      AAL for voice                         |
  |     00000001      AAL type 1                            |
  |     00000010      AAL type 2                            |
  |     00000011      AAL type 3/4                          |
  |     00000101      AAL type 5                            |
  |     00010000      user defined AAL                      |
  +-----------------+---------+-------------+----------------------+
```

C.5 Frame Relay

```
  +-----------------+------+-------------+----------------------+
  | PropertyID      | Tag  | Type        | Value                |
  +-----------------+------+-------------+----------------------+
  | DLCI            | 5001 | Unsigned    | Data link connection id |
  |                 |      | Integer     |                      |
  +-----------------+------+-------------+----------------------+
  | CID             | 5002 | Unsigned    | sub-channel id.      |
  |                 |      | Integer     |                      |
  +-----------------+------+-------------+----------------------+
  | SID/Noiselevel  | 5003 | Unsigned    | silence insertion    |
  |                 |      | Integer     | descriptor           |
  +-----------------+------+-------------+----------------------+
  | Primary Payload |5004  | Unsigned    | Primary Payload Type |
  | Type            |      | Integer     | Covers FAX and codecs |
  +-----------------+------+-------------+----------------------+
```

C.6 IP

PropertyID	Tag	Type	Value
IPv4	6001	32 BITS	Ipv4Address|Ipv4Address:
	Reference: IETF RFC791		
IPv6	6002	128 BITS	IPv6 Address:
	Reference: IETF RFC2460		
Port	6003	Unsigned Integer	0-65535
Porttype	6004	enumerated	TCP(0), UDP(1), SCTP(2)

C.7 ATM AAL2

PropertyID	Tag n	Type	Value
AESA	7001	20 OCTETS	AAL2 service endpoint address

```
|     as defined in Reference: ITU Recommendation Q.2630.1
|
|       ESEA
|
|       NSEA
|
  +-----------+-------+-----------------------+--------------------
--+
  | BIR       |See C.3| 4 OCTETS              | Served user
|
  |           |       |                       | reference
|
  +-----------+-------+-----------------------+--------------------
--+
  |    as defined in Reference: ITU Recommendation Q.2630.1 SUGR
|
  +-----------+-------+-----------------------+--------------------
--+
  | ALC       | 7002  | 12 OCTETS             | AAL2 link
|
  |           |       |                       | characteristics
|
  +-----------+-------+-----------------------+--------------------
--+
  |    as defined in Reference: ITU Recommendation Q.2630.1
|
  |     max/average CPS-SDU bitrate,
|
  |     max/average CPS-SDU size
|
  +-----------+-------+-----------------------+--------------------
--+
  | SSCS      | 7003  | I.366.2:              | Service specific
|
  |           |       | audio (8 OCTETS)      | convergence
sublayer |
  |           |       | multirate (3 OCTETS)  | information
|
  |           |       | or I.366.1:           |
|
  |           |       | SAR-assured (14 OCTETS)/|
|
  |           |       | unassured (7 OCTETS)  |
|
  +-----------+-------+-----------------------+--------------------
--+
  |   as defined in Reference: Q.2630.1 and used in I.366.1 and
I.366.2 |
  |       I.366.2: audio/multirate
|
  |       I.366.1: SAR-assured/unassured
|
  +-----------+-------+-----------------------+--------------------
--+
  | SUT       | 7004  | 1..254 octets         |Served user
transport |
  |           |       |                       | parameter
|
  +-----------+-------+-----------------------+--------------------
--+
```

| as defined in Reference: ITU Recommendation Q.2630.1
|
| TCI | 7005 |BOOLEAN | Test connection
|
| | | | indicator
|
  +----------+-------+-----------------------+------------------
--+
| as defined in Reference: ITU Recommendation Q.2630.1
|
  +----------+-------+-----------------------+------------------
--+
| Timer_CU | 7006 |32 bit integer | Timer-CU:
|
  +----------+-------+-----------------------+------------------
--+

| Milliseconds to hold partially filled cell before sending.
|
  +----------+-------+-----------------------+------------------
--+
| MaxCPSSDU | 7007 | 8 bit integer | Maximum Common Part
|
| | | | Sublayer Service
|
| | | | Data Unit
|
  +----------+-------+-----------------------+------------------
--+
| Ref.: rec. Q.2630.1
|
  +----------+-------+-----------------------+------------------
--+
| CID | 7008 | 8 bits | subchannel id, 0-
255 |
  +----------+-------+-----------------------+------------------
--+
| Ref.: rec. I.363.2
|
  +---
--+

C.8 ATM AAL1

  +---------------+---------+---------------+------------------
--+
| PropertyID | Property| Type | Value
|
| | Tag | |
|

```
   +---------------+---------+---------------+-------------------
--+
   | BIR           | See     | 29 OCTETS     | GIT (Generic
   |
   |               | Table   |               | Identifier
Transport) |
   |               | C.3     |               |
   |
   |     Ref.: Recommendation Q.2941.1
   |
   +---------------+---------+---------------+-------------------
--+
   | AAL1ST        | 8001    | 1 OCTET       | AAL1 Subtype:
   |
   |     Reference: ITU Recommendation Q.2931
   |
   |     00000000      Null
   |
   |     00000001      voiceband signal transport on 64kbit/s
   |
   |     00000010      circuit transport
   |
   |     00000100      high-quality audio signal transport
   |
   |     00000101      video signal transport
   |
   +---------------+---------+---------------+-------------------
--+
   | CBRR          | 8002    | 1 OCTET       | CBR Rate
   |
   |     Reference: ITU Recommendation Q.2931
   |
   |     00000001       64 kbit/s
   |
   |     00000100      1544 kbit/s
   |
   |     00000101      6312 kbit/s
   |
   |     00000110      32064 kbit/s
   |
   |     00000111      44736 kbit/s
   |
   |     00001000      97728 kbit/s
   |
   |     00010000      2048 kbit/s
   |
   |     00010001      8448 kbit/s
   |
   |     00010010      34368 kbit/s
   |
   |     00010011     139264 kbit/s
   |
   |     01000000     n x 64 kbit/s
   |
   |     01000001      n * 8 kbit/s
   |
   +---------------+---------+---------------+-------------------
--+
```

MULT	See		Multiplier, or n x
	Table		64k/8k/300
	C.9		
Reference: ITU Recommendation Q.2931			

SCRI	8003	1 OCTECT	Source Clock Frequency
			Recovery Method
Reference: ITU Recommendation Q.2931			
00000000 NULL			
00000001 SRTS			
00000010 ACM			

ECM	8004	1 OCTECT	Error Correction
			Method
Reference: ITU Recommendation Q.2931			
00000000 Null			
00000001 FEC-LOSS			
00000010 FEC-DELAY			

SDTB	8005	16 bit integer	Structured Data
			Transfer Blocksize
Reference: ITU Recommendation I.363.1			
Block size of SDT CBR service			

| PFCI | 8006 | 8 bit integer | Partially filled cells |
| | | | identifier |

```
   |      Reference: ITU Recommendation I.363.1
   |
   |      1-47
   |
   +--------------------------------------------------------------------+
```

C.9 Bearer Capabilities

   The table entries referencing ITU-T Recommendation Q.931 refer to the
   encoding in the bearer capability information element of Q.931, not to
   the low layer information element.

PropertyID	Property Tag	Type	Value
TMR	9001	1 OCTET	Transmission Medium Requirement (Q.763)

```
   |      Reference: ITU Recommendation Q.763
   |
   |      Bit 8 7 6 5 4 3 2 1
   |
   |      00000000 - speech
   |
   |      00000001 - spare
   |
   |      00000010 - 64 kbit/s unrestricted
   |
   |      00000011 - 3.1 kHz audio
   |
```

```
   |      00000100 - reserved for alternate speech (service 2)/64 kbit/s |
   |          unrestricted (service 1)
   |
   |      00000101 - reserved for alternate 64 kbit/s unrestricted
   |
   |          (service 1)/speech (service 2)
   |
   |      00000110 - 64 kbit/s preferred
   |
   |      00000111 - 2 x 64 kbit/s unrestricted
   |
```

| 00001000 - 384 kbit/s unrestricted
|
| 00001001 - 1536 kbit/s unrestricted
|
| 00001010 - 1920 kbit/s unrestricted
|
| 00001011 through 00001111- spare
|
| 00010000 - 3 x 64 kbit/s unrestricted
|
| 00010001 - 4 x 64 kbit/s unrestricted
|
| 00010010 - 5 x 64 kbit/s unrestricted
|
| 00010011 spare
|
| 00010100 - 7 x 64 kbit/s unrestricted
|
| 00010101 - 8 x 64 kbit/s unrestricted
|
| 00010110 - 9 x 64 kbit/s unrestricted
|
| 00010111 - 10 x 64 kbit/s unrestricted
|
| 00011000 - 11 x 64 kbit/s unrestricted
|
| 00011001 - 12 x 64 kbit/s unrestricted
|
| 00011010 - 13 x 64 kbit/s unrestricted
|
| 00011011 - 14 x 64 kbit/s unrestricted
|
| 00011100 - 15 x 64 kbit/s unrestricted
|
| 00011101 - 16 x 64 kbit/s unrestricted
|
| 00011110 - 17 x 64 kbit/s unrestricted
|
| 00011111 - 18 x 64 kbit/s unrestricted
|
| 00100000 - 19 x 64 kbit/s unrestricted
|
| 00100001 - 20 x 64 kbit/s unrestricted
|
| 00100010 - 21 x 64 kbit/s unrestricted
|
| 00100011 - 22 x 64 kbit/s unrestricted
|
| 00100100 - 23x 64 kbit/s unrestricted
|
| 00100101 - spare
|
| 00100110 - 25 x 64 kbit/s unrestricted
|
| 00100111 - 26 x 64 kbit/s unrestricted
|
| 00101000 - 27 x 64 kbit/s unrestricted
|
| 00101001 - 28 x 64 kbit/s unrestricted
|

```
   |      00101010 - 29 x 64 kbit/s unrestricted
|
   |      00101011 through 11111111 Spare
|
   +---------------+---------+---------------+--------------------
--+
   | TMRSR         | 9002    | 1 OCTET       | Transmission Medium
|
   |               |         |               |  Requirement
Subrate |
   |      0 - unspecified
|
   |      1 - 8kbit/s
|
   |      2 - 16kbit/s
|
   |      3 - 32kbit/s
|
   +---------------+---------+---------------+--------------------
--+
```

```
   | Contcheck     | 9003    | BOOLEAN       | Continuity Check
|
   |      Reference: ITU Recommendation Q.763
|
   |      0 - Not required on this circuit
|
   |      1 - Required on this circuit
|
   +---------------+---------+---------------+--------------------
--+
   | ITC           | 9004    | 5 BITS        | Information
Transfer |
   |               |         |               |  Capability
|
   |      Reference: ITU Recommendation Q.763
|
   |      Bits 5 4 3 2 1
|
   |      00000 - Speech
|
   |      01000 - Unrestricted digital information
|
   |      01001 - Restricted digital information
|
   |      10000 - 3.1 kHz audio
|
   |      10001 - Unrestricted digital information with
|
```

```
   |                tones/announcements
   |
   |      11000 - Video
   |
   |      All other values are reserved.
   |
   +---------------+---------+----------------+--------------------
--+
   | TransMode     | 9005    | 2 BITS         | Transfer Mode
   |
   |      Reference: ITU Recommendation Q.931
   |
   |      Bit 2 1
   |
   |      00 - Circuit mode
   |
   |      10 - Packet mode
   |
   +---------------+---------+----------------+--------------------
--+
   | TransRate     | 9006    | 5 BITS         | Transfer Rate
   |
   |     Reference: ITU Recommendation Q.931
   |
   |     Bit 5 4 3 2 1
   |
   |     00000 - This code shall be used for packet mode calls
   |
   |     10000 - 64 kbit/s
   |
   |     10001 - 2 x 64 kbit/s
   |
   |     10011 -384 kbit/s
   |
   |     10101 -1536 kbit/s
   |
   |     10111 -1920 kbit/s
   |
   |     11000 - Multirate (64 kbit/s base rate)
   |
   +---------------+---------+----------------+--------------------
--+
   | MULT          | 9007    | 7 BITS         | Rate Multiplier
   |
   |     Reference: ITU Recommendation Q.931
   |
   |     Any value from 2 to n (maximum number of B-channels)
   |
   +---------------+---------+----------------+--------------------
--+
```

```
  | USI            | 9008    | 5 BITS         | User Information Layer|
  |                |         |                | 1 Protocol           |
  |     Reference: ITU Recommendation Q.931                          |
  |     Bits 5 4 3 2 1                                               |
  |     00001 - CCITT standardized rate adaption V.110 and X.30.     |
  |     00010 - Recommendation G.711  u-law                          |
  |     00011 - Recommendation G.711 A-law                           |
  |     00100 - Recommendation G.721 32 kbit/s ADPCM and Recommendation |
  |             I.460.                                               |
  |     00101 - Recommendations H.221 and H.242                      |
  |     00110 - Recommendations H.223 and H.245                      |
  |     00111 - Non-ITU-T standardized rate adaption.               |
  |     01000 - ITU-T standardized rate adaption V.120.             |
  |     01001 - CCITT standardized rate adaption X.31 HDLC flag      |
  |             stuffing.                                            |
  |     All other values are reserved.                              |
  +----------------+---------+----------------+----------------------+
  | syncasync      | 9009    | BOOLEAN        | Synchronous/         |
  |                |         |                | Asynchronous         |
  |     Reference: ITU Recommendation Q.931                          |
  |     0 - Synchronous data                                         |
  |     1 - Asynchronous data                                        |
  +----------------+---------+----------------+----------------------+
  | negotiation    | 900A    | BOOLEAN        | Negotiation          |
  |     Reference: ITU Recommendation Q.931                          |
  |     0 - In-band negotiation possible                             |
```

```
|      1 - In-band negotiation not possible
|
  +---------------+---------+----------------+--------------------
--+

  +---------------+---------+----------------+--------------------
--+
  | Userrate      | 900B    | 5 BITS         | User Rate
|
  |      Reference: ITU Recommendation Q.931
|
  |      Bits 5 4 3 2 1
|
  |      00000 - Rate is indicated by E-bits specified in
Recommendation |
  |              I.460 or may be negotiated in-band
|
  |      00001 - 0.6 kbit/s Recommendations V.6 and X.1
|
  |      00010 - 1.2 kbit/s Recommendation V.6
|
  |      00011 - 2.4 kbit/s Recommendations V.6 and X.1
|
  |      00100 - 3.6 kbit/s Recommendation V.6
|
  |      00101 - 4.8 kbit/s Recommendations V.6 and X.1
|
  |      00110 - 7.2 kbit/s Recommendation V.6
|
  |      00111 - 8 kbit/s Recommendation I.460
|
  |      01000 - 9.6 kbit/s Recommendations V.6 and X.1
|
  |      01001 - 14.4 kbit/s Recommendation V.6
|
  |      01010 - 16 kbit/s Recommendation I.460
|
  |      01011 - 19.2 kbit/s Recommendation V.6
|
  |      01100 - 32 kbit/s Recommendation I.460
|
  |      01101 - 38.4 kbit/s Recommendation V.110
|
```

```
  |      01110 - 48 kbit/s Recommendations V.6 and X.1
|
  |      01111 - 56 kbit/s Recommendation V.6
|
  |      10010 - 57.6 kbit/s Recommendation V.14 extended
|
  |      10011 - 28.8 kbit/s Recommendation V.110
|
```

```
|      10100 - 24 kbit/s Recommendation V.110
|
|      10101 - 0.1345 kbit/s Recommendation X.1
|
|      10110 - 0.100 kbit/s Recommendation X.1
|
|      10111 - 0.075/1.2 kbit/s Recommendations V.6 and X.1
|
|      11000 - 1.2/0.075 kbit/s Recommendations V.6 and X.1
|
|      11001 - 0.050 kbit/s Recommendations V.6 and X.1
|
|      11010 - 0.075 kbit/s Recommendations V.6 and X.1
|
|      11011 - 0.110 kbit/s Recommendations V.6 and X.1
|
|      11100 - 0.150 kbit/s Recommendations V.6 and X.1
|
|      11101 - 0.200 kbit/s Recommendations V.6 and X.1
|
|      11110 - 0.300 kbit/s Recommendations V.6 and X.1
|
|      11111 - 12 kbit/s Recommendation V.6
|
|      All other values are reserved.
|
  +---------------+---------+----------------+--------------------
--+
  | INTRATE       | 900C    | 2 BITS         | Intermediate Rate
|
  |     Reference: ITU Recommendation Q.931
|
  |     Bit 2 1
|
  |     00 - Not used
|
  |     01 - 8 kbit/s
|
  |     10 - 16 kbit/s
|
  |     11 - 32 kbit/s
|
  +---------------+---------+----------------+--------------------
--+
  | nictx         | 900D    | BOOLEAN        | Network Independent
|
  |               |         |                | Clock (NIC) on
|
  |               |         |                | transmission
|
  |     Reference: ITU Recommendation Q.931
|
  |     0 - Not required to send data with network independent clock
|
  |     1 - Required to send data with network independent clock
|
  +---------------+---------+----------------+--------------------
--+
  | nicrx         | 900E    | BOOLEAN        | Network independent
|
```

```
  |                |          |                |  clock (NIC) on
|
  |                |          |                |  reception
|
  |      Reference: ITU Recommendation Q.931
|
  |      0 - Cannot accept data with network independent clock (i.e.
|
  |          sender does not support this optional procedure)
|
  |      1 - Can accept data with network independent clock (i.e.
sender |
  |          does support this optional procedure)
|
  +---------------+---------+----------------+--------------------
--+
  | flowconttx     | 900F     | BOOLEAN        |  Flow Control on
|
  |                |          |                |  transmission (Tx)
|
  |      Reference: ITU Recommendation Q.931
|
  |      0 - Not required to send data with flow control mechanism
|
  |      1 - Required to send data with flow control mechanism
|
  +---------------+---------+----------------+--------------------
--+
```

```
  | flowcontrx     | 9010     | BOOLEAN        |  Flow control on
|
  |                |          |                |  reception (Rx)
|
  |      Reference: ITU Recommendation Q.931
|
  |      0 - Cannot accept data with flow control mechanism (i.e.
sender |
  |          does not support this optional procedure)
|
  |      1 - Can accept data with flow control mechanism (i.e. sender
|
  |          does support this optional procedure)
|
  +---------------+---------+----------------+--------------------
--+
  | rateadapthdr   | 9011     | BOOLEAN        |  Rate adaption
|
  |                |          |                |  header/no header
|
  |      Reference: ITU Recommendation Q.931
|
```

```
   |     0 - Rate adaption header not included
|
   |     1 - Rate adaption header included
|
   +---------------+---------+----------------+--------------------
--+
   | multiframe    | 9012    | BOOLEAN        | Multiple frame
|
   |               |         |                | establishment
support|
   |               |         |                | in data link
|
   |     Reference: ITU Recommendation Q.931
|
   |     0 - Multiple frame establishment not supported. Only UI frames
|
   |         allowed.
|
   |     1 - Multiple frame establishment supported
|
   +---------------+---------+----------------+--------------------
--+
   | OPMODE        | 9013    | BOOLEAN        | Mode of operation
|
   |     Reference: ITU Recommendation Q.931
|
   |     0 Bit transparent mode of operation
|
   |     1 Protocol sensitive mode of operation
|
   +---------------+---------+----------------+--------------------
--+
   | llidnegot     | 9014    | BOOLEAN        | Logical link
|
   |               |         |                | identifier
negotiation|
   |     Reference: ITU Recommendation Q.931
|
   |     0 Default, LLI = 256 only
|
   |     1 Full protocol negotiation
|
   +---------------+---------+----------------+--------------------
--+
   | assign        | 9015    | BOOLEAN        | Assignor/assignee
|
   |     Reference: ITU Recommendation Q.931
|
   |     0 Message originator is "Default assignee"
|
   |     1 Message originator is "Assignor only"
|
   +---------------+---------+----------------+--------------------
--+
   | inbandneg     | 9016    | BOOLEAN        | In-band/out-band
|
   |               |         |                | negotiation
|
   |     Reference: ITU Recommendation Q.931
|
```

| 0- Negotiation is done with USER INFORMATION messages on a
|
| temporary signalling connection
|
| 1- Negotiation is done in-band using logical link zero
|
  +---------------+---------+---------------+--------------------
--+

  | stopbits | 9017 | 2 BITS | Number of stop bits
|
  | Reference: ITU Recommendation Q.931
|
  | Bits 2 1
|
  | 00 - Not used
|
  | 01 - 1 bit
|
  | 10 - 1.5 bits
|
  | 11 - 2 bits
|
  +---------------+---------+---------------+--------------------
--+
  | databits | 9018 | 2 BIT | Number of data bits
|
  | | | | excluding parity
Bit |
  | | | | if present
|
  | Reference: ITU Recommendation Q.931
|
  | Bit 2 1
|
  | 00 - Not used
|
  | 01 - 5 bits
|
  | 10 - 7 bits
|
  | 11 - 8 bits
|
  +---------------+---------+---------------+--------------------
--+
  | parity | 9019 | 3 BIT | Parity information
|
  | Reference: ITU Recommendation Q.931
|

```
   |     Bit 3 2 1
   |
   |     000 - Odd
   |
   |     010 - Even
   |
   |     011 - None
   |
   |     100 - Forced to 0
   |
   |     101 - Forced to 1
   |
   |     All other values are reserved.
   |
   +---------------+---------+----------------+--------------------
--+
   | duplexmode    | 901A    | BOOLEAN        | Mode duplex
   |
   |     Reference: ITU Recommendation Q.931
   |
   |     0 - Half duplex
   |
   |     1 - Full duplex
   |
   +---------------+---------+----------------+--------------------
--+
   | modem         | 901B    | 6 BIT          | Modem Type
   |
   |     Reference: ITU Recommendation Q.931
   |
   |     Bits 6 5 4 3 2 1
   |
   |     00000 through 000101 National Use
   |
   |     010001 - Recommendation V.21
   |
   |     010010 - Recommendation V.22
   |
   |     010011 - Recommendation V.22 bis
   |
   |     010100 - Recommendation V.23
   |
   |     010101 - Recommendation V.26
   |
   |     011001 - Recommendation V.26 bis
   |
   |     010111 - Recommendation V.26 ter
   |
   |     011000 - Recommendation V.27
   |
   |     011001 - Recommendation V.27 bis
   |
   |     011010 - Recommendation V.27 ter
   |
   |     011011 - Recommendation V.29
   |
```

```
   |     011101 - Recommendation V.32
|
   |     011110 - Recommendation V.34
|
   |     100000 through 101111 National Use
|
   |     110000 through 111111 User Specified
|
   +---------------+---------+---------------+--------------------
--+
   | layer2prot    | 901C     5 BIT         | User information
layer|
   |               |                        |  2 protocol
|
   |     Reference: ITU Recommendation Q.931
|
   |     Bit 5 4 3 2 1
|
   |     00010 - Recommendation Q.921/I.441 [3]
|
   |     00110 - Recommendation X.25 [5], link layer
|
   |     01100 - LAN logical link control (ISO/IEC 8802-2)
|
   |     All other values are reserved.
|
   +---------------+---------+---------------+--------------------
--+
   | layer3prot    | 901D    | 5 BIT         | User information
layer|
   |               |         |               |  3 protocol
|
   |     Reference: ITU Recommendation Q.931
|
   |     Bit 5 4 3 2 1
|
   |     00010 - Recommendation Q.931/I.451
|
   |     00110 - Recommendation X.25, packet layer
|
   |     01011 - ISO/IEC TR 9577 (Protocol identification in the
network |
   |           layer)
|
   |     All other values are reserved.
|
   +---------------+---------+---------------+--------------------
--+
   | addlayer3prot | 901E    | OCTET         | Additional User
|
   |               |         |               | Information layer 3
|
   |               |         |               | protocol
|
   |
|
```

```
   |     Bits 4321
   |     1100 1100 - Internet Protocol (RFC 791) (ISO/IEC TR 9577)
   |
   |     1100 1111 - Point-to-point Protocol (RFC 1661)
   |
   +---------------+---------+----------------+--------------------
--+
   | DialledN      | 901F    | 30 OCTETS      | Dialled Number
   |
   +---------------+---------+----------------+--------------------
--+
   | DiallingN     | 9020    | 30 OCTETS      | Dialling Number
   |
   +---------------+---------+----------------+--------------------
--+
   | ECHOCI        | 9021    | Enumeration    | Echo Control
   |
   |               |         |                | Information
   |
   |     echo canceler off (0), incoming echo canceler on (1), outgoing
   |
   |     echo canceler on (2), incoming and outgoing echo canceler on
(3)|
   +---------------+---------+----------------+--------------------
--+
   | NCI           | 9022    | 1 OCTET        | Nature of
Connection |
   |               |         |                | Indicators
   |
   |     Reference: ITU Recommendation Q.763
   |
   |     Bits 8 7 6 5 4 3 2 1
   |
   |
   |
   |     Bits 2 1 Satellite Indicator
   |
   |     0 0  no satellite circuit in the connection
   |
```

```
   |     0 1  one satellite circuit in the connection
   |
   |     1 0  two satellite circuits in the connection
   |
   |     1 1  spare
   |
   |
   |     Bits 4 3 Continuity check indicator
   |
   |     0 0  continuity check not required
   |
```

```
   |      0 1   continuity check required on this circuit
|
   |      1 0   continuity check performed on a previous circuit
|
   |      1 1   spare
|
   |
|
   |      Bits 5 Echo control device indicator
|
   |      0   outgoing echo control device not included
|
   |      1   outgoing echo control device included
|
   |
|
   |      Bits 8 7 6 Spare
|
   +---------------------------------------------------------------------
--+
```

C.10 AAL5 Properties

```
   +---------------+---------+----------------+--------------------
--+
   | PropertyID    | Property| Type           | Value
|
   |               | Tag     |                |
|
   +---------------+---------+----------------+--------------------
--+
   | FMSDU         | A001    | 32 bit integer | Forward Maximum
CPCS-|
   |               |         |                | SDU Size:
|
   |     Reference: ITU Recommendation Q.2931
|
   |     Maximum CPCS-SDU size sent in the direction from the calling
|
   |     user to the called user.
|
   +---------------+---------+----------------+--------------------
--+
   | BMSDU         | A002    | 32 bit integer | Backwards Maximum
|
   |               |         |                | CPCS-SDU Size
|
   |     Reference: ITU Recommendation Q.2931
|
   |     Maximum CPCS-SDU size sent in the direction from the called
user|
   |     to the calling user.
|
   +---------------+---------+----------------+--------------------
--+
   | SSCS          | See     | See table C.7  | See table C.7
|
   |               | table   |                |
|
   |               | C.7     |                |
|
```

```
   |     Additional values:                                             |
   |        VPI/VCI                                                      |
   +--------------------------------------------------------------------+
```

C.11 SDP Equivalents

PropertyID	Property Tag	Type	Value
SDP_V	B001	STRING	Protocol Version
SDP_O	B002	STRING	Owner/creator and session ID
SDP_S	B003	STRING	Sesson name
SDP_I	B004	STRING	Session identifier
SDP_U	B005	STRING	URI of descriptor
SDC_E	B006	STRING	email address
SDP_P	B007	STRING	phone number

```
+----------------+--------+----------------+----------------------
--+
| SDP_C          | B008   | STRING         | Connection
information|
+----------------+--------+----------------+----------------------
--+
| SDP_B          | B009   | STRING         | Bandwidth
Information |
+----------------+--------+----------------+----------------------
--+
| SDP_Z          | B00A   | STRING         | time zone adjustment
|
+----------------+--------+----------------+----------------------
--+
| SDP_K          | B00B   | STRING         | Encryption Key
|
+----------------+--------+----------------+----------------------
--+
| SDP_A          | B00C   | STRING         | Zero or more session
|
+----------------+--------+----------------+----------------------
--+
| SDP_T          | B00D   | STRING         | Active Session Time
|
+----------------+--------+----------------+----------------------
--+
| SDP_R          | B00E   | STRING         | Zero or more repeat
|
+----------------+--------+----------------+----------------------
--+
|                                                                 |
|                                                                 |
|     Reference in all cases: IETF RFC2327, "Session Description
|
|     Protocol"
|
+-----------------------------------------------------------------
--+
```

C.12 H.245

```
+----------------+--------+--------------+-----------------------
--+
```

PropertyID	Property Tag	Type	Value
OLC	C001	octet string	The value of H.245 OpenLogicalChannel structure.
OLCack	C002	octet string	The value of H.245 OpenLogicalChannelAck structure.
OLCcnf	C003	octet string	The value of H.245 OpenLogicalChannelConfirm structure.
OLCrej	C004	octet string	The value of H.245 OpenLogicalChannelReject structure.
CLC	C005	octet string	The value of H.245 CloseLogicalChannel structure.
CLCack	C006	octet string	The value of H.245 CloseLogicalChannelAck structure.
Reference in all cases: ITU-T Recommendation H.245			

ANNEX D TRANSPORT OVER IP (NORMATIVE)

D.1 Transport over IP/UDP using Application Level Framing

   Protocol messages defined in this document may be transmitted over
   UDP. When no port is provided by the peer (see section 7.2.8),
   commands should be sent to the default port number, 2944 for text-
   encoded operation or 2945 for binary-encoded operation. Responses
   must be sent to the address and port from which the corresponding
   commands were sent.

   Implementors using IP/UDP with ALF should be aware of the
   restrictions of the MTU on the maximum message size.

D.1.1 Providing At-Most-Once Functionality

   Messages, being carried over UDP, may be subject to losses. In the
   absence of a timely response, commands are repeated. Most commands
   are not idempotent. The state of the MG would become
unpredictable
   if, for example, Add commands were executed several times. The
   transmission procedures shall thus provide an "At- Most-Once"
   functionality.

   Peer protocol entities are expected to keep in memory a list of
the
   responses that they sent to recent transactions and a list of the
   transactions that are currently outstanding. The transaction
   identifier of each incoming message is compared to the transaction
   identifiers of the recent responses sent to the same MId. If a
match
   is found, the entity does not execute the transaction, but simply
   repeats the response. If no match is found, the message will be
   compared to the list of currently outstanding transactions. If a
   match is found in that list, indicating a duplicate transaction,
the
   entity does not execute the transaction (see section D.1.4 for
   procedures on sending TransactionPending).

   The procedure uses a long timer value, noted LONG-TIMER in the
   following. The timer should be set larger than the maximum
duration
   of a transaction, which should take into account the maximum
number
   of repetitions, the maximum value of the repetition timer and the
   maximum propagation delay of a packet in the network. A suggested
   value is 30 seconds.

   The copy of the responses may be destroyed either LONG-TIMER
seconds
   after the response is issued, or when the entity receives a
   confirmation that the response has been received, through the

"Response Acknowledgement parameter". For transactions that are
acknowledged through this parameter, the entity shall keep a copy
of
the transaction-id for LONG-TIMER seconds after the response is
issued, in order to detect and ignore duplicate copies of the
transaction request that could be produced by the network.

D.1.2 Transaction identifiers and three-way handshake

D.1.2.1 Transaction identifiers

Transaction identifiers are 32 bit integer numbers. A Media
Gateway
Controller may decide to use a specific number space for each of
the
MGs that they manage, or to use the same number space for all MGs
that belong to some arbitrary group. MGCs may decide to share the
load of managing a large MG between several independent processes.
These processes will share the same transaction number space.
There
are multiple possible implementations of this sharing, such as

having a centralized allocation of transaction identifiers, or
pre-
allocating non-overlapping ranges of identifiers to different
processes. The implementations shall guarantee that unique
transaction identifiers are allocated to all transactions that
originate from a logical MGC (identical mId). MGs can simply
detect
duplicate transactions by looking at the transaction identifier
and
mId only.

D.1.2.2 Three-way handshake

The TransactionResponse Acknowledgement parameter can be found in
any message. It carries a set of "confirmed transaction-id
ranges".
Entities may choose to delete the copies of the responses to
transactions whose id is included in "confirmed transaction-id
ranges" received in the transaction response messages. They should
silently discard further commands when the transaction-id falls
within these ranges.

The "confirmed transaction-id ranges" values shall not be used if
more than LONG-TIMER seconds have elapsed since the MG issued its
last response to that MGC, or when a MG resumes operation. In
this
situation, transactions should be accepted and processed, without
any test on the transaction-id.

Messages that carry the "Transaction Response Acknowledgement"

parameter may be transmitted in any order. The entity shall retain
the "confirmed transaction-id ranges" received for LONG-TIMER
seconds.

In the binary encoding, if only the firstAck is present in a
response acknowledgement (see Annex A.2), only one transaction is
acknowledged. If both firstAck and lastAck are present, then the
range of transactions from firstAck to lastAck is acknowledged.
In
the text encoding, a horizontal dash is used to indicate a range
of
transactions being acknowledged (see Annex B.2).

D.1.3 Computing retransmission timers

It is the responsibility of the requesting entity to provide
suitable time outs for all outstanding transactions, and to retry
transactions when time outs have been exceeded. Furthermore, when
repeated transactions fail to be acknowledged, it is the
responsibility of the requesting entity to seek redundant services
and/or clear existing or pending connections. Implementations
SHALL
ensure that the algorithm used to calculate retransmission timing
performs an exponentially increasing backoff of the retransmission
timeout for each retransmission or repetition after the first one.

The specification purposely avoids specifying any value for the
retransmission timers. These values are typically network
dependent.
The retransmission timers should normally estimate the timer value
by measuring the time spent between the sending of a command and
the
return of a response.

Note - One possibility is to use the algorithm implemented in TCP-
IP, which uses two variables:

* The average acknowledgement delay, AAD, estimated through an
   exponentially smoothed average of the observed delays.

* The average deviation, ADEV, estimated through an exponentially
   smoothed average of the absolute value of the difference
between
   the observed delay and the current average. The retransmission
   timer, in TCP, is set to the sum of the average delay plus N
times
   the average deviation. The maximum value of the timer should
   however be bounded for the protocol defined in this document,
in
   order to guarantee that no repeated packet would be received by

the gateways after LONG-TIMER seconds. A suggested maximum value
      is 4 seconds.

   After any retransmission, the entity SHOULD do the following:

   * It should double the estimated value of the average delay, AAD

   * It should compute a random value, uniformly distributed between
      0.5 AAD and AAD

   * It should set the retransmission timer to the sum of that random
      value and N times the average deviation.

   This procedure has two effects. Because it includes an exponentially
   increasing component, it will automatically slow down the stream of
   messages in case of congestion. Because it includes a random
   component, it will break the potential synchronization between
   notifications triggered by the same external event.

D.1.4 Provisional responses

   Executing some transactions may require a long time. Long execution
   times may interact with the timer based retransmission procedure.
   This may result either in an inordinate number of retransmissions,
   or in timer values that become too long to be efficient. Entities
   that can predict that a transaction will require a long execution
   time may send a provisional response, "Transaction Pending". They
   SHOULD send this response if they receive a repetition of a
   transaction that is still being executed.

   Entities that receive a Transaction Pending shall switch to a
   different repetition timer for repeating requests. The root
   termination has a property (ProvisionalResponseTimerValue), which
   can be set to the requested maximum number of milliseconds between
   receipt of a command and transmission of the TransactionPending
   response. Upon receipt of a final response following receipt of
   provisional responses, an immediate confirmation shall be sent, and
   normal repetition timers shall be used thereafter. An entity that
   sends a provisional response, SHALL include the immAckRequired field
   in the ensuing final response, indicating that an immediate
   confirmation is expected. Receipt of a Transaction Pending after
   receipt of a reply shall be ignored.

D.1.5 Repeating Requests, Responses and Acknowledgements

The protocol is organized as a set of transactions, each of which
is
composed request and a response, commonly referred to as an
acknowledgement. The protocol messages, being carried over UDP,
may
be subject to losses. In the absence of a timely response,
transactions are repeated. Entities are expected to keep in memory
a
list of the responses that they sent to recent transactions, i.e.
a
list of all the responses they sent over the last LONG-TIMER
seconds, and a list of the transactions that are currently being
executed.

The repetition mechanism is used to guard against three types of
possible errors:

*   transmission errors, when for example a packet is lost due to
    noise on a line or congestion in a queue;
*   component failure, when for example an interface to a entity
    becomes unavailable;
*   entity failure, when for example an entire entity become
    unavailable.

The entities should be able to derive from the past history an
estimate of the packet loss rate due to transmission errors. In a
properly configured system, this loss rate should be kept very
low,
typically less than 1%. If a Media Gateway Controller or a Media
Gateway has to repeat a message more than a few times, it is very
legitimate to assume that something else than a transmission error
is occurring. For example, given a loss rate of 1%, the
probability
that five consecutive transmission attempts fail is 1 in 100
billion, an event that should occur less than once every 10 days
for
a Media Gateway Controller that processes 1 000 transactions per
second. (Indeed, the number of repetition that is considered
excessive should be a function of the prevailing packet loss
rate.)

We should note that the "suspicion threshold", which we will call
"Max1", is normally lower than the "disconnection threshold",
which
should be set to a larger value.

A classic retransmission algorithm would simply count the number
of
successive repetitions, and conclude that the association is
broken
after retransmitting the packet an excessive number of times

(typically between 7 and 11 times.) In order to account for the
   possibility of an undetected or in-progress "failover", we modify
   the classic algorithm so that if the Media Gateway receives a
valid
   ServiceChange message announcing a failover, it will start
   transmitting outstanding commands to that new MGC. Responses to
   commands are still transmitted to the source address of the
command.

   In order to automatically adapt to network load, this document
   specifies exponentially increasing timers. If the initial timer
is
   set to 200 milliseconds, the loss of a fifth retransmission will
be
   detected after about 6 seconds. This is probably an acceptable
   waiting delay to detect a failover.   The repetitions should
   continue after that delay not only in order to perhaps overcome a
   transient connectivity problem, but also in order to allow some
more
   time for the execution of a failover - waiting a total delay of 30
   seconds is probably acceptable.

   It is, however, important that the maximum delay of
retransmissions
   be bounded. Prior to any retransmission, it is checked that the
   time elapsed since the sending of the initial datagram is no
greater
   than T-MAX. If more than T-MAX time has elapsed, the MG concludes
   that the MGC has failed, and it begins its recovery process. The
MG
   shall use a ServiceChange with ServiceChangeMethod set to
   Disconnected so that the new MGC will be aware that the MG lost
one
   or more transactions. The value T-MAX is related to the LONG-
TIMER
   value: the LONG-TIMER value is obtained by adding to T-MAX the
   maximum propagation delay in the network.

D.2 Using TCP

   Protocol messages as defined in this document may be transmitted
   over TCP. When no port is specified by the other side (see
section
   7.2.8), the commands should be sent to the default port. The
defined
   protocol has messages as the unit of transfer, while TCP is a
   stream-oriented protocol. TPKT, according to RFC1006 SHALL be
used
   to delineate messages within the TCP stream.

   In a transaction-oriented protocol, there are still ways for
   transaction requests or responses to be lost. As such, it is
   recommended that entities using TCP transport implement
application

   level timers for each request and each response, similar to those
   specified for application level framing over UDP.

D.2.1 Providing the At-Most-Once functionality

   Messages, being carried over TCP, are not subject to transport
   losses, but loss of a transaction request or its reply may
   nonetheless be noted in real implementations. In the absence of a
   timely response, commands are repeated. Most commands are not
   idempotent. The state of the MG would become unpredictable if,
for
   example, Add commands were executed several times.

   To guard against such losses, it is recommended that entities
follow
   the procedures in section D.1.1

D.2.2 Transaction identifiers and three way handshake

   For the same reasons, it is possible that transaction replies may
be
   lost even with a reliable delivery protocol such as TCP. It is
   recommended that entities follow the procedures in section
   D.1.2.2.

D.2.3 Computing retransmission timers

   With reliable delivery, the incidence of loss of a transaction
   request or reply is expected to be very low. Therefore, only
simple
   timer mechanisms are required. Exponential back-off algorithms
   should not be necessary, although they could be employed where, as
   in an MGC, the code to do so is already required, since MGCs must
   implement ALF/UDP as well as TCP.

D.2.4 Provisional responses

   As with UDP, executing some transactions may require a long time.
   Entities that can predict that a transaction will require a long
   execution time may send a provisional response, "Transaction
   Pending". They should send this response if they receive a
   repetition of a transaction that is still being executed.

   Entities that receive a Transaction Pending shall switch to a
longer
   repetition timer for that transaction.

   Entities shall retain Transactions and replies until they are
   confirmed. The basic procedure of section D.1.4 should be
followed,
   but simple timer values should be sufficient. There is no need to
   send an immediate confirmation upon receipt of a final response.

D.2.5 Ordering of commands

   TCP provides ordered delivery of transactions. No special
   procedures are required. It should be noted that ALF/UDP allows
   sending entity to modify its behavior under congestion, and in
   particular, could reorder transactions when congestion is
   encountered. TCP could not achieve the same results.

ANNEX E BASIC PACKAGES (NORMATIVE)

   This Annex contains definitions of some packages for use with the
   Megaco protocol.

E.1 Generic

   PackageID: g (0x0001)
   Version: 1
   Extends: None

   Description: Generic package for commonly encountered items.

E.1.1 Properties

   None

E.1.2 Events

   Cause

   EventID: cause (0x0001)

   Generic error event

   ObservedEvents Descriptor Parameters:

        General Cause

        ParameterID: Generalcause (0x0001)

        Description: This parameter groups the failures into six
        groups, which the MGC may act upon.

         Possible values: Enumerated,
               "NR" Normal Release (0x0001)
               "UR" Unavailable Resources (0x0002)
               "FT" Failure, Temporary (0x0003)
               "FP" Failure, Permanent (0x0004)
               "IW" Interworking Error (0x0005)
               "UN" Unsupported (0x0006)

         Failure Cause

         ParameterID: Failurecause (0x0002)

         Description: The Release Cause is the value generated by the
         Released equipment, i.e. a released network connection.
         The concerned value is defined in the appropriate bearer
         control protocol.

         Possible Values: OCTET STRING

   Signal Completion

   EventID: sc (0x0002)

   Indicates termination of one or more signals for which the
   notifyCompletion parameter was set to "ON". For further
procedural
   description, see sections 7.1.11, 7.1.17, and 7.2.7.

   ObservedEvents Descriptor parameters:

         Signal Identity

         ParameterID: SigID (0x0001)

         This parameter identifies the signals which have terminated.

         Type: list

         Possible values: a list of signals and/or sequential signal
         lists which have terminated. A signal outside of a
sequential
         signal list shall be identified using the pkgdName syntax
         without wildcarding. An individual signal inside of a
         sequential signal list shall be identified using the
sequential
         signal list syntax with the correct signal list identifier,
         enclosing the name of the specific signal which terminated in
         pkgdName syntax.

          Termination Method

          ParameterID: Meth (0x0002)

          Indicates the means by which the signal terminated.

          Type: enumeration

          Possible values:
                  "TO" (0x0001) Duration expired
                  "EV" (0x0002) Interrupted by event
                  "SD" (0x0003) Halted by new Signals Descriptor
                  "NC" (0x0004) Not completed, other cause

E.1.3 Signals

   None

E.1.4 Statistics

   None

E.2 Base Root Package

   Base Root Package
   PackageID: root (0x0002)
   Version: 1
   Extends: None

   Description: This package defines Gateway wide properties.

E.2.1 Properties

   MaxNrOfContexts

   PropertyID: maxNumberOfContexts (0x0001)

   The value of this property gives the maximum number of contexts
that
   can exist at any time. The NULL context is not included in this
   number.

   Type: Double

   Possible values: 1 and up

   Defined in: TerminationState

    MaxTerminationsPerContext

    PropertyID: maxTerminationsPerContext (0x0002)

    The maximum number of allowed terminations in a context, see
section
    6.1

    Type: Integer

    Possible Values: any integer

    Defined In: TerminationState

    normalMGExecutionTime

    PropertyId: normalMGExecutionTime (0x0003)

    Settable by the MGC to indicate the interval within which the MGC
    expects a response to any transaction from the MG (exclusive of
    network delay)

    Type: Integer

    Possible Values: any integer, represents milliseconds

    Defined in: TerminationState

    normalMGCExecutionTime

    PropertyId: normalMGCExecutionTime (0x0004)

    Settable by the MGC to indicate the interval within which the MG
    should expects a response to any transaction from the MGC
(exclusive
    of network delay)

    Type: Integer

    Possible Values: any integer, represents milliseconds

    Defined in: TerminationState

    ProvisionalResponseTimerValue

    PropertyId: ProvisionalResponseTimerValue (0x0005)

   Indicates the time within which to expect a Pending Response if a
   Transaction cannot be completed. Initially set to
   normalMGExecutionTime or normalMGCExecutionTime as appropriate,
plus
   network delay, but may be lowered.

   Type: Integer

   Possible Values: any integer, represents milliseconds

   Defined in: TerminationState

E.2.2 Events

   None

E.2.3 Signals

   None

E.2.4 Statistics

   None

E.2.5 Procedures

   None

E.3 Tone Generator Package

   PackageID: tonegen (0x0003)
   Version: 1
   Extends: None

   Description:
   This package defines signals to generate audio tones. This package
   does not specify parameter values. It is intended to be
extendable.
   Generally, tones are defined as an individual signal with a
   parameter, ind, representing "interdigit" time delay, and a tone
id
   to be used with playtones. A tone id should be kept consistent
with
   any tone generation for the same tone. MGs are expected to be
   provisioned with the characteristics of appropriate tones for the
   country in which the MG is located.

E.3.1 Properties

   None

E.3.2 Events

    None

E.3.3 Signals

    Play tone

    SignalID: pt (0x0001)

    Plays audio tone over an audio channel

    Signal Type: Brief

    Duration: Provisioned

    Additional Parameters:

         Tone id list

         ParameterID: tl (0x0001)

         Type: list of tone ids.

         List of tones to be played in sequence. The list SHALL
contain
         one or more tone ids.

         Inter signal duration

         ParameterID: ind (0x0002)

         Type: integer.

         Timeout between two consecutive tones in milliseconds

    No tone ids are specified in this package. Packages that extend
this
    package can add possible values for tone id as well as adding
    individual tone signals.

E.3.4 Statistics

    None

E.3.5 Procedures

    None

E.4 Tone Detection Package

    PackageID: tonedet (0x0004)
    Version: 1
    Extends: None

    This Package defines events for audio tone detection. Tones are
    selected by name (tone id). MGs are expected to be provisioned
with
    the characteristics of appropriate tones for the country in which
the
    MG is located.

    This package does not specify parameter values. It is intended to
be
    extendable.

E.4.1 Properties

    None

E.4.2 Events

    Start tone detected

    EventID: std, 0x0001

    Detects the start of a tone. The characteristics of positive tone
    detection is implementation dependent.

    EventsDescriptor parameters:

        Tone id list

        ParameterID: tl (0x0001)

        Type: list of tone ids

        Possible values: The only tone id defined in this package is
        "wild card" which is "*" in text encoding and 0x0000 in
binary.
        Extensions to this package would add possible values for tone
        id. If tl is "wild card", any tone id is detected.

    ObservedEventsDescriptor parameters:

        Tone id

        ParameterID: tid (0x0003)

        Type: Enumeration

        Possible values: "wildcard" as defined above is the only
value
        defined in this package. Extensions to this package would add
        additional possible values for tone id.

    End tone detected

    EventID: etd, 0x0002

    Detects the end of a tone.

    EventDescriptor parameters:

        Tone id list

        ParameterID: tl (0x0001)

        Type: enumeration or list of enumerated types

        Possible values: No possible values are specified in this
        package. Extensions to this package would add possible values
        for tone id.

    ObservedEventsDescriptor parameters:

        Tone id

        ParameterID: tid (0x0003)

        Type: Enumeration

        Possible values: "wildcard" as defined above is the only
value
        defined in this package. Extensions to this package would add
        possible values for tone id

        Duration

        ParameterId: dur (0x0002)

        Type: integer, in milliseconds

        This parameter contains the duration of the tone from first
        detection until it stopped.

    Long tone detected

    EventID: ltd, 0x0003

    Detects that a tone has been playing for at least a certain amount
    of time

    EventDescriptor parameters:

        Tone id list

        ParameterID: tl (0x0001)

        Type: enumeration or list

        Possible values: "wildcard" as defined above is the only
value
        defined in this package. Extensions to this package would add
        possible values for tone id

        Duration:

        ParameterID: dur (0x0002)

        Type: integer, duration to test against

        Possible values: any legal integer, expressed in
milliseconds.

    ObservedEventsDescriptor parameters:

        Tone id

        ParameterID: tid (0x0003)

        Possible values: No possible values are specified in this
        package. Extensions to this package would add possible values
        for tone id.

E.4.3 Signals

    None

E.4.4 Statistics

    None

E.4.5 Procedures

    None

E.5 Basic DTMF Generator Package

   PackageID: dg (0x0005)
   Version: 1
   Extends: tonegen version 1

   This package defines the basic DTMF tones as signals and extends the
   allowed values of parameter tl of playtone in tonegen.

E.5.1 Properties

   None

E.5.2 Events

   None

E.5.3 Signals

   dtmf character 0

   SignalID: d0 (0x0010)

   Generate DTMF 0 tone. The physical characteristic of DTMF 0 is
   defined in the gateway.

   Signal Type: Brief

   Duration: Provisioned

   Additional Parameters:

   None

   Additional Values:
   ----

   d0 (0x0010) is defined as a toneid for playtone.

   The other dtmf characters are specified in exactly the same way. A
   table with all signal names and signal IDs is included. Note that
   each dtmf character is defined as both a signal and a toneid, thus
   extending the basic tone generation package. Also note that dtmf
   SignalIds are different from the names used in a digit map.

```
            +------------------+------------------+
            | Signal Name      | Signal ID/tone id |
            +------------------+------------------+
            | dtmf character 0 | d0 (0x0010)      |
            | dtmf character 1 | d1 (0x0011)      |
            | dtmf character 2 | d2 (0x0012)      |
            | dtmf character 3 | d3 (0x0013)      |
            | dtmf character 4 | d4 (0x0014)      |
            | dtmf character 5 | d5 (0x0015)      |
            | dtmf character 6 | d6 (0x0016)      |
            | dtmf character 7 | d7 (0x0017)      |
            | dtmf character 8 | d8 (0x0018)      |
            | dtmf character 9 | d9 (0x0019)      |
            | dtmf character * | ds (0x0020)      |
            | dtmf character # | do (0x0021)      |
            | dtmf character A | da (0x001a)      |
            | dtmf character B | db (0x001b)      |
            | dtmf character C | dc (0x001c)      |
            | dtmf character D | dd (0x001d)      |
            +------------------+------------------+
```

E.5.4 Statistics

   None

E.5.5 Procedures

   None

E.6 DTMF detection Package

   PackageID: dd (0x0006)
   Version: 1
   Extends: tonedet version 1

   This package defines the basic DTMF tones detection. This Package
   extends the possible values of tone id in the "start tone
detected"
   "end tone detected" and "long tone detected" events.

   Additional tone id values are all tone ids described in package dg
   (basic DTMF generator package).

   The following table maps DTMF events to digit map symbols as
   described in section 7.1.14.

```
                   +------+--------------+
                   | DTMF | Event Symbol |
                   +------+--------------+
                   | d0   | "0"          |
                   | d1   | "1"          |
                   | d2   | "2"          |
                   | d3   | "3"          |
                   | d4   | "4"          |
                   | d5   | "5"          |
                   | d6   | "6"          |
                   | d7   | "7"          |
                   | d8   | "8"          |
                   | d9   | "9"          |
                   | da   | "A" or "a"   |
                   | db   | "B" or "b"   |
                   | dc   | "C" or "c"   |
                   | dd   | "D" or "d"   |
                   | ds   | "E" or "e"   |
                   | do   | "F" or "f"   |
                   +------+--------------+
```

E.6.1 Properties

   None

E.6.2 Events

   DTMF digits

   EventIds are defined with the same names as the SignalIds defined
in
   the table found in section E.5.3.

   DigitMap Completion Event

   EventID: ce, 0x0001

   Generated when a digit map completes as described in section
7.1.14.

   EventsDescriptor parameters: digit map processing is activated
only
   if a digit map parameter is present, specifying a digit map by
name
   or by value. Other parameters such as a KeepActive flag or
embedded
   Events or Signals Descriptors may be present.

ObservedEventsDescriptor parameters:

        DigitString

        ParameterID: ds (0x0001)

        Type: string of digit map symbols (possibly empty) returned
as
        a quotedString.

        Possible values: a sequence of the characters "0" through
"9",
        "A" through "F", and the long duration modifier "Z".

        Description: the portion of the current dial string as
        described in section 7.1.14 which matched part or all of an
        alternative event sequence specified in the digit map.

        Termination Method

        ParameterID: Meth (0x0003)

        Type: enumeration

        Possible values:
                "UM" (0x0001) Unambiguous match
                "PM" (0x0002) Partial match, completion by timer
                expiry or unmatched event
                "FM" (0x0003) Full match, completion by timer expiry
                or unmatched event

        Description: indicates the reason for generation of the
event.
        See the procedures in section 7.1.14.

E.6.3 Signals

    None

E.6.4 Statistics

    None

E.6.5 Procedures

    None

E.7 Call Progress Tones Generator Package

   PackageID: cg, 0x0007
   Version: 1
   Extends: tonegen version 1

   This package defines the basic call progress tones as signals and
   extends the allowed values of the tl parameter of playtone in
tonegen
    .

E.7.1 Properties

   None

E.7.2 Events

   None

E.7.3 Signals

   Dial Tone

   SignaID: dt (0x0030)

   Generate dial tone. The physical characteristic of dial tone is
   available in the gateway.

   Signal Type: Timeout

   Duration: Provisioned

   Additional Parameters:
   None

   Additional Values
   ----
   dt (0x0030) is defined as a tone id for playtone
   The other tones of this package are defined in exactly the same
way.
   A table with all signal names and signal IDs is included. Note
   that each tone is defined as both a signal and a toneid, thus
   extending the basic tone generation package.

```
              +---------------------------+-------------------+
              | Signal Name               | Signal ID/tone id |
              +---------------------------+-------------------+
              | Dial Tone                 | dt (0x0030)       |
              | Ringing Tone              | rt (0x0031)       |
              | Busy Tone                 | bt (0x0032)       |
              | Congestion Tone           | ct (0x0033)       |
              | Special Information Tone  | sit(0x0034)       |
              | Warning Tone              | wt (0x0035)       |
              | Payphone Recognition Tone | pt (0x0036)       |
              | Call Waiting Tone         | cw (0x0037)       |
              | Caller Waiting Tone       | cr (0x0038)       |
              +---------------------------+-------------------+
```

E.7.4 Statistics

   None

E.7.5 Procedures

   NOTE - The required set of tone ids corresponds to those defined
in
   Recommendation E.180/Q.35 [ITU-T Recommendation E.180/Q.35
(1998)].
   See E.180 for definition of the meanings of these tones.

E.8 Call Progress Tones Detection Package

   PackageID: cd (0x0008)
   Version: 1
   Extends: tonedet version 1

   This package defines the basic call progress detection tones. This
   Package extends the possible values of tone id in the "start tone
   detected", "end tone detected" and "long tone detected" events.

   Additional values
   ----

   tone id values are defined for start tone detected, end tone
detected
   and long tone detected with the same values as those in package cg
   (call progress tones generation package).

   The required set of tone ids corresponds to Recommendation
E.180/Q.35
   [ITU-T Recommendation E.180/Q.35 (1998)]. See Recommendation
   E.180/Q.35 for definition of the meanings of these tones.

E.8.1 Properties

E.8.2 Events

   Events are defined as in the call progress tones generator package
   (cg) for the tones listed in the table of section E.7.3

E.8.3 Signals

E.8.4 Statistics

E.8.5 Procedures

E.9 Analog Line Supervision Package

   PackageID: al, 0x0009
   Version: 1
   Extends: None

   This package defines events and signals for an analog line.

E.9.1 Properties

   None

E.9.2 Events

   onhook

   EventID: on (0x0004)

   Detects handset going on hook. Whenever an events descriptor is
   activated that requests monitoring for an on-hook event and the
line
   is already on-hook, then the MG shall behave according to the
setting
   of the "strict" parameter.

   EventDescriptor parameters

       Strict Transition

       ParameterID: strict (0x0001)

       Type: enumeration

Possible values: "exact" (0x00), "state" (0x01), "failWrong" (0x02)

"exact" means that only an actual hook state transition to on-hook is to be recognized;

"state" means that the event is to be recognized either if the hook state transition is detected or if the hook state is already on-hook;

"failWrong" means that if the hook state is already on-hook, the command fails and an error is reported.

ObservedEventsDescriptor parameters

Initial State

ParameterID: init (0x0002)

Type: Boolean

Possible values:

True means that the event was reported because the line was already on-hook when the events descriptor containing this event was activated;

False means that the event represents an actual state transition to on-hook.

offhook

EventID: of (0x0005)

Detects handset going off hook. Whenever an events descriptor is activated that requests monitoring for an off-hook event and the line is already off-hook, then the MG shall behave according to the setting of the "strict" parameter.

EventDescriptor parameters

Strict Transition

ParameterID: strict (0x0001)

Type: enumeration

Possible values: "exact" (0x00), "state" (0x01), "failWrong" (0x02)

"exact" means that only an actual hook state transition to off-
hook is to be recognized;

"state" means that the event is to be recognized either if the
hook state transition is detected or if the hook state is
already off-hook;

"failWrong" means that if the hook state is already off-hook,
the command fails and an error is reported.

ObservedEventsDescriptor parameters

Initial State

ParameterID: init (0x0002)

Type: Boolean

Possible values:

True means that the event was reported because the line was
already off-hook when the events descriptor containing this
event was activated;

False means that the event represents an actual state
transition to off-hook.

flashhook

EventID: fl, 0x0006

Detects handset flash. A flash occurs when an onhook is followed
by
an offhook between a minimum and maximum duration.

EventDescriptor parameters

Minimum duration

ParameterID: mindur (0x0004)

Type: integer in milliseconds

Default value is provisioned

```
      Maximum duration
      ----------------
      ParameterID: maxdur (0x0005)

      Type: integer in milliseconds

      Default value is provisioned

   ObservedEventsDescriptor parameters

   None

E.9.3 Signals

   ring
   ----
   SignalID: ri, 0x0002

   Applies ringing on the line

   Signal Type: TimeOut

   Duration: Provisioned

   Additional Parameters:

      Cadence
      -------
      ParameterID: cad (0x0006)

      Type: list of integers representing durations of alternating
on
      and off segments, constituting a complete ringing cycle
      starting with an on. Units in milliseconds.

      Default is fixed or provisioned.  Restricted function MGs may
      ignore cadence  values they are incapable of generating.

      Frequency
      ---------
      ParameterID: freq (0x0007)

      Type: integer in Hz

      Default is fixed or provisioned.  Restricted function MGs may
      ignore frequency  values they are incapable of generating.
```

E.9.4 Statistics

None

E.9.5 Procedures

   If the MGC sets an EventsDescriptor containing a hook state
   transition event (on-hook or off-hook) with the "strict" (0x0001)
   parameter set to "failWrong", and the hook state is already what
the
   transition implies, the execution of the command containing that
   EventsDescriptor fails. The MG SHALL include error code 540
   "Unexpected initial hook state" in its reponse.

E.9.6 Error Code

   This package defines a new error code:
        540 - Unexpected initial hook state
   The procedure for use of this code is given in section E.9.5.

E.10 Basic Continuity Package

   PackageID: ct (0x000a)
   Version: 1
   Extends: None

   This package defines events and signals for continuity test. The
   continuity test includes provision of either a loopback or
   transceiver functionality.

E.10.1 Properties

   None

E.10.2 Events

   Completion

   EventID: cmp, 0x0005

   This event detects test completion of continuity test.

   EventDescriptor parameters

   None

   ObservedEventsDescriptor parameters

        Result

        ParameterID: res (0x0008)

Type: Enumeration

            Possible values: success (0x0001), failure (0x0000)

E.10.3 Signals

    Continuity test

    SignalID: ct (0x0003)

    Initiates sending of continuity test tone on the termination to
    which it is applied.

    Signal Type: TimeOut

    Default value is provisioned

    Additional Parameters:

    None

    Respond

    SignalID: rsp (0x0004)

    The signal is used to respond to a continuity test . See section
    E.10.5 for further explanation.

    Signal Type: On/Off

    Default duration is provisioned

    Additional Parameters:

    None.

E.10.4 Statistics

    None

E.10.5 Procedures

    When a MGC wants to initiate a continuity test, it sends a command
to
    the MG containing

* a signals descriptor with the ct signal, and
       * an events descriptor containing the cmp event.

    Upon reception of a command containing the ct signal and cmp
event,
    the MG initiates the continuity test tone for the specified
    termination. If the return tone is detected and any other
required
    conditions are satisfied before the signal times out, the cmp
event
    shall be generated with the value of the result parameter equal to
    success. In all other cases, the cmp event shall be generated
with
    the value of the result parameter equal to failure.

    When a MGC wants the MG to respond to a continuity test, it sends
a
    command to the MG containing a signals descriptor with the rsp
    signal. Upon reception of a command with the rsp signal, the MG
    either applies a loopback or (for 2-wire circuits) awaits
reception
    of a continuity test tone. In the loopback case, any incoming
    information shall be reflected back as outgoing information. In
the
    2-wire case, any time the appropriate test tone is received, the
    appropriate response tone should be sent. The MGC determines when
to
    remove the rsp signal.

    When a continuity test is performed on a termination, no echo
devices
    or codecs shall be active on that termination.

    Performing voice path assurance as part of continuity testing is
    provisioned by bilateral agreement between network operators.

E.11 Network Package

    PackageID: nt (0x000b)
    Version: 1
    Extends: None

    This package defines properties of network terminations
independent
    of network type.

E.11.1 Properties

    Maximum Jitter Buffer

    PropertyID: jit (0x0007)

    This property puts a maximum size on the jitter buffer.

Type: integer in milliseconds

        Possible Values: This property is specified in milliseconds.

        Defined In: LocalControlDescriptor

        Characteristics: read/write

E.11.2 Events

    network failure

    EventID: netfail, 0x0005

    The termination generates this event upon detection of a failure
due
    to external or internal network reasons.

    EventDescriptor parameters

    None

    ObservedEventsDescriptor parameters

    cause

    ParameterID: cs (0x0001)

    Type: String

    Possible values: any text string

    This parameter may be included with the failure event to provide
    diagnostic information on the reason of failure.

    quality alert

    EventID: qualert, 0x0006

    This property allows the MG to indicate a loss of quality of the
    network connection. The MG may do this by measuring packet loss,
    interarrival jitter, propogation delay and then indicating this
using
    a percentage of quality loss.

```
    EventDescriptor parameters

         Threshold
         ---------
         ParameterId: th (0x0001)

         Type: integer

         Possible Values: threshold for percent of quality loss
         measured, calculated based on a provisioned method, that
could
         take into consideration packet loss, jitter, and delay for
         example.  Event is triggered when calculation exceeds the
         threshold.

    ObservedEventsDescriptor parameters

         Threshold
         ---------
         ParameterId: th (0x0001)

         Type: integer

         Possible Values: percent of quality loss measured, calculated
         based on a provisioned method, that could take into
         consideration packet loss, jitter, and delay for example.

E.11.3 Signals

    none

E.11.4 Statistics

    Duration
    ---------
    StatisticsID: dur (0x0001)

    Description: Provides duration of time the termination has been in
    the context.

    Type: Double, in milliseconds

    Octets Sent
    -----------
    StatisticID: os (0x0002)

    Type: double

    Possible Values: any 64 bit integer
```

```
      Octets Received
      ---------------
      StatisticID: or (0x0003)

      Type: double

      Possible Values: any 64 bit integer

E.11.5 Procedures

   none

E.12 RTP  Package

   PackageID: rtp (0x000c)
   Version: 1
   Extends: Network Package version 1

   This package is used to support packet based multimedia data
transfer
   by means of the Real-time Transport Protocol (RTP) [RFC 1889].

E.12.1 Properties

   None

E.12.2 Events

   Payload Transition
   EventID: pltrans, 0x0001
   This event detects and notifies when there is a transition of the
   RTP payload format from one format to another.

   EventDescriptor parameters

   None

   ObservedEventsDescriptor parameters

       rtppayload
       ----------
       ParameterID: rtppltype, 0x01

       Type: list of enumerated types.

       Possible values: The encoding method shall be specified by
       using one or several valid encoding names, as defined in the
       RTP AV Profile or registered with IANA.
```

```
E.12.3 Signals
```

```
   None

E.12.4 Statistics

   Packets Sent
   ------------
   StatisticID: ps (0x0004)

   Type: double

   Possible Values: any 64 bit integer

   Packets Received
   ----------------
   StatisticID: pr (0x0005)

   Type: double

   Possible Values: any 64 bit integer

   Packet Loss
   -----------
   StatisticID: pl (0x0006)

   Describes the current rate of packet loss on an RTP stream, as
   defined in IETF RFC 1889. Packet loss is expressed as percentage
   value: number of packets lost in the interval between two
reception
   reports, divided by the number of packets expected during that
   interval.

   Type: double

   Possible Values: a 32 bit whole number and a 32 bit fraction.

   Jitter
   ------
   StatisticID: jit (0x0007)

   Requests the current value of the interarrival jitter on an RTP
   stream as defined in IETF RFC 1889. Jitter measures the variation
in
   interarrival time for RTP data packets.

   Delay
   -----
   StatisticID:delay (0x0008)
```

```
   Requests the current value of packet propagation delay expressed
in
   timestamp units. Same as average latency.
```

E.12.5 Procedures

E.13 TDM Circuit Package

   PackageID: tdmc (0x000d)
   Version: 1
   Extends: Network Package version 1

   This package is used to support TDM circuit terminations.

E.13.1 Properties

   Echo Cancellation

   PropertyID: ec (0x0008)

   By default, the telephony gateways always perform echo
cancellation.
   However, it is necessary, for some calls, to turn off these
   operations.

   Type: boolean

   Possible Values:
        "on" (when the echo cancellation is requested) and
        "off" (when it is turned off.)
   The default is "on".

   Defined In: LocalControlDescriptor

   Characteristics: read/write

   Gain Control

   PropertyID: gain (0x000a)

   Gain control, or usage of of signal level adaptation and noise
level
   reduction is used to adapt the level of the signal. However, it is
   necessary, for example for modem calls, to turn off this function.

   Type: enumeration (integer)

   Possible Values:
   The gain control parameter may either be specified as "automatic"
   (0xffffffff), or as an explicit number of decibels of gain (any
other

integer value). The default is provisioned in the MG.

Defined In: LocalControlDescriptor

Characteristics: read/write

E.13.2 Events

E.13.3 Signals

E.13.4 Statistics

None

E.13.5 Procedures

None

APPENDIX A EXAMPLE CALL FLOWS (INFORMATIVE)

   All Megaco implementors must read the normative part of this document
   carefully before implementing from it. No one should use the examples

in this section as stand-alone explanations of how to create
protocol
   messages.

   The examples in this section use SDP for encoding of the Local and
   Remote stream descriptors. SDP is defined in RFC 2327. If there is
   any discrepancy between the SDP in the examples, and RFC 2327, the
   RFC should be consulted for correctness. Audio profiles used are
   those defined in RFC 1890, and others registered with IANA. For
   example, G.711 A-law is called PCMA in the SDP, and is assigned
   profile 0. G.723 is profile 4, and H263 is profile 34. See also

      http://www.isi.edu/in-notes/iana/assignments/rtp-parameters

A.1 Residential Gateway to Residential Gateway Call

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

A.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. The ROOT termination is used to indicate the entire MG
   instead of a termination within the MG.

   In this example, MG1 has the IP address 124.124.124.222, MG2 is
   125.125.125.111, and the MGC is 123.123.123.4. The default Megaco
   port is 55555 for all three.

```
1. An MG registers with an MGC using the ServiceChange command:

MG1 to MGC:
MEGACO/1 [124.124.124.222]
Transaction = 9998 {
    Context = - {
        ServiceChange = ROOT {Services {
            Method=Restart,
            ServiceChangeAddress=55555, Profile=ResGW/1}
        }

    }
}

2. The MGC sends a reply:

MGC to MG1:
MEGACO/1 [123.123.123.4]:55555
Reply = 9998 {
    Context = - {ServiceChange = ROOT {
      Services {ServiceChangeAddress=55555, Profile=ResGW/1} } }
}

3. The MGC programs a Termination in the NULL context. The
terminationId is A4444, the streamId is 1, the requestId in the
Events descriptor is 2222. The   mId is the identifier of the
sender
of this message, in this case, it is the IP address and port
[123.123.123.4]:55555. Mode for this stream is set to SendReceive.
"al" is the analog line supervision package.

MGC to MG1:
MEGACO/1 [123.123.123.4]:55555
Transaction = 9999 {
    Context = - {
        Modify = A4444 {
            Media { Stream = 1 {
                    LocalControl {
                        Mode = SendReceive,
                        tdmc/gain=2,  ; in dB,
                        tdmc/ec=on
                    },
                    Local {
v=0
c=IN IP4 $
m=audio $ RTP/AVP 0
a=fmtp:PCMU VAD=X-NNVAD ; special voice activity
                            ; detection algorithm
                    }
```

```
                }
```

```
              },
              Events = 2222 {al/of}
          }
       }
   }
```

   The dialplan script could have been loaded into the MG previously.
   Its function would be to wait for the OffHook, turn on dialtone
and
   start collecting DTMF digits. However in this example, we use the
   digit map, which is put into place after the offhook is detected
   (step 5 below).

   Note that the embedded EventsDescriptor could have been used to
   combine steps 3 and 4 with steps 8 and 9, eliminating steps 6 and
7.

   4. The MG1 accepts the Modify with this reply:

```
   MG1 to MGC:
   MEGACO/1 [124.124.124.222]:55555
   Reply = 9999 {
       Context = - {Modify = A4444}
   }
```

   5. A similar exchange happens between MG2 and the MGC, resulting
in
   an idle Termination called A5555.

A.1.2 Collecting Originator Digits and Initiating Termination

   The following builds upon the previously shown conditions. It
   illustrates the transactions from the Media Gateway Controller and
   originating Media Gateway (MG1) to get the originating Termination
   (A4444) through the stages of digit collection required to
initiate a
   connection to the terminating Media Gateway (MG2).

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

```
   MG1 to MGC:
   MEGACO/1 [124.124.124.222]:55555
   Transaction = 10000 {
       Context = - {
           Notify = A4444 {ObservedEvents =2222 {
              19990729T22000000:al/of}}
       }
   }
```

```
    7. And the Notify is acknowledged.

    MGC to MG1:
    MEGACO/1 [123.123.123.4]:55555
    Reply = 10000 {
        Context = - {Notify = A4444}
    }

    8. The MGC Modifies the termination to play dial tone, to look for
    digits according to Dialplan0 and to look for the on-hook event
now.
    MGC to MG1:

    MEGACO/1 [123.123.123.4]:55555
    Transaction = 10001 {
        Context = - {
            Modify = A4444 {
                Events = 2223 {
                    al/on, dd/ce {DigitMap=Dialplan0}
                },
                Signals {cg/dt},
                DigitMap= Dialplan0{
    (0| 00|[1-7]xxx|8xxxxxxx|Fxxxxxxx|Exx|91xxxxxxxxxx|9011x.)}
            }
        }
    }

    9. And the Modify is acknowledged.

    MG1 to MGC:
    MEGACO/1 [124.124.124.222]:55555
    Reply = 10001 {
        Context = - {Modify = A4444}
    }

    10. Next, digits are accumulated by MG1 as they are dialed by User
1.
    Dialtone is stopped upon detection of the first digit. When an
    appropriate match is made of collected digits against the
currently
    programmed Dialplan for A4444, another Notify is sent to the Media
    Gateway Controller.

    MG1 to MGC:
    MEGACO/1 [124.124.124.222]:55555
    Transaction = 10002 {
        Context = - {
            Notify = A4444 {ObservedEvents =2223 {
                19990729T22010001:dd/ce{ds="916135551212",Meth=FM}}}
        }
    }
```

```
   11. And the Notify is acknowledged.

   MGC to MG1:
   MEGACO/1 [123.123.123.4]:55555
   Reply = 10002 {
       Context = - {Notify = A4444}
   }

   12. The controller then analyses the digits and determines that a
   connection needs to be made from MG1 to MG2. Both the TDM
termination
   A4444, and an RTP termination are added to a new context in MG1.
Mode
   is ReceiveOnly since Remote descriptor values are not yet
specified.
   Preferred codecs are in the MGC's preferred order of choice.

   MGC to MG1:
   MEGACO/1 [123.123.123.4]:55555
   Transaction = 10003 {
       Context = $ {
           Add = A4444,
           Add = $ {
               Media {
                 Stream = 1 {
                     LocalControl {
                         Mode = ReceiveOnly,

                         nt/jit=40 ; in ms
                     },
                     Local {
   v=0
   c=IN IP4 $
   m=audio $ RTP/AVP 4
   a=ptime:30
   v=0
   c=IN IP4 $
   m=audio $ RTP/AVP 0
                     }
                 }
               }
           }
       }
   }

   NOTE -  The MGC states its preferred parameter values as a series
of
   sdp blocks in  Local. The MG fills in the Local Descriptor in the
   Reply.
```

```
   13. MG1 acknowledges the new Termination and fills in the Local IP
   address and UDP port. It also makes a choice for the codec based
on
   the MGC preferences in Local. MG1 sets the RTP port to 2222.
   MEGACO/1 [124.124.124.222]:55555
   Reply = 10003 {
      Context = 2000 {
         Add = A4444,
         Add=A4445{
            Media {
                Stream = 1 {
                    Local {
   v=0
   c=IN IP4 124.124.124.222
   m=audio 2222 RTP/AVP 4
   a=ptime:30
   a=recvonly
                    } ; RTP profile for G.723 is 4
                }
            }
         }
      }
   }

   14. The MGC will now associate A5555 with a new Context on MG2,
and
   establish an RTP Stream (i.e, A5556 will be assigned), SendReceive
   connection through to the originating user, User 1. The MGC also
sets
   ring on A5555.

   MGC to MG2:
   MEGACO/1 [123.123.123.4]:55555
   Transaction = 50003 {
      Context = $ {
         Add = A5555  { Media {
             Stream = 1 {
                 LocalControl {Mode = SendReceive} }},
              Events=1234{al/of},
             Signals {al/ri}
             },
         Add  = $ {Media {
             Stream = 1 {
                 LocalControl {
                    Mode = SendReceive,
                    nt/jit=40 ; in ms
                 },
                 Local {
   v=0
   c=IN IP4 $
   m=audio $ RTP/AVP 4
```

```
   a=ptime:30
                  },
                  Remote {
   v=0
   c=IN IP4 124.124.124.222
   m=audio 2222 RTP/AVP 4
   a=ptime:30
                  } ; RTP profile for G.723 is 4
                }
              }
            }
          }
   }

   15. This is acknowledged. The stream port number is different from
   the control port number. In this case it is 1111 (in the SDP).

   MG2 to MGC:
   MEGACO/1 [124.124.124.222]:55555
   Reply = 50003 {
      Context = 5000 {
        Add = A5555,
         Add = A5556{
            Media {
               Stream = 1 {
                   Local {
   v=0
   c=IN IP4 125.125.125.111
   m=audio 1111 RTP/AVP 4
   }
                } ; RTP profile for G723 is 4
              }
            }
        }
   }

   16. The above IPAddr and UDPport need to be given to MG1 now.

   MGC to MG1:
   MEGACO/1 [123.123.123.4]:55555
   Transaction = 10005 {
     Context = 2000 {
       Modify = A4444 {
         Signals {cg/rt}
       },
       Modify = A4445 {
         Media {
              Stream = 1 {
```

```
                  Remote {
   v=0
```

```
        c=IN IP4 125.125.125.111
        m=audio 1111 RTP/AVP 4
                        }
                    } ; RTP profile for G723 is 4
                }
            }
        }
    }


    MG1 to MGC:
    MEGACO/1 [124.124.124.222]:55555
    Reply = 10005 {
        Context = 2000 {Modify = A4444, Modify = A4445}
    }

    17. The two gateways are now connected and User 1 hears the
RingBack.
    The MG2 now waits until User2 picks up the receiver and then the
    two-way call is established.

    From MG2 to MGC:

    MEGACO/1 [125.125.125.111]:55555
    Transaction = 50005 {
        Context = 5000 {
            Notify = A5555 {ObservedEvents =1234 {
                19990729T22020002:al/of}}
        }
    }

    From MGC to MG2:

    MEGACO/1 [123.123.123.4]:55555
    Reply = 50005 {
        Context = - {Notify = A5555}
    }

    From MGC to MG2:

    MEGACO/1 [123.123.123.4]:55555
    Transaction = 50006 {
        Context = 5000 {
            Modify = A5555 {
                Events = 1235 {al/on},
                Signals { } ; to turn off ringing
            }
        }
```

```
    }

    From MG2 to MGC:
```

```
MEGACO/1 [125.125.125.111]:55555
Reply = 50006 {
 Context = 5000 {Modify = A4445}
}

18. Change mode on MG1 to SendReceive, and stop the ringback.

MGC to MG1:
MEGACO/1 [123.123.123.4]:55555
Transaction = 10006 {
   Context = 2000 {
      Modify = A4445 {
         Media {
            Stream = 1 {
               LocalControl {
                  Mode=SendReceive
               }
            }
         }
      },
      Modify = A4444 {
         Signals { }
      }
   }
}

from MG1 to MGC:
MEGACO/1 [124.124.124.222]:55555
Reply = 10006 {
   Context = 2000 {Modify = A4445, Modify = A4444}}

19. The MGC decides to Audit the RTP termination on MG2.

MEGACO/1 [123.123.123.4]:55555
Transaction = 50007 {
   Context = - {AuditValue = A5556{
      Audit{Media, DigitMap, Events, Signals, Packages, Statistics
}}
   }
}
```

```
20. The MG2 replies.

MEGACO/1 [125.125.125.111]:55555
Reply = 50007 {
   Context = - {
AuditValue = A5556 {
         Media {
```

```
                   TerminationState { ServiceStates = InService,
                         Buffer = OFF },
                 Stream = 1 {
                     LocalControl { Mode = SendReceive,
                        nt/jit=40 },
                     Local {
 v=0
 c=IN IP4 125.125.125.111
 m=audio 1111 RTP/AVP  4
 a=ptime:30
                     },
                      Remote {
 v=0
 c=IN IP4 124.124.124.222
 m=audio 2222 RTP/AVP  4
 a=ptime:30
                     } } },
             Events,
             Signals,
             DigitMap,
             Packages {nt-1, rtp-1},
             Statistics { rtp/ps=1200,  ; packets sent
                          nt/os=62300, ; octets sent
                          rtp/pr=700, ; packets received
                          nt/or=45100, ; octets received
                          rtp/pl=0.2,  ; % packet loss
                          rtp/jit=20,
                          rtp/delay=40 } ; avg latency
         }
       }
     }

    21. When the MGC receives an onhook signal from one of the MGs, it
    brings down the call. In this example, the user at MG2 hangs up
    first.

    From MG2 to MGC:

    MEGACO/1 [125.125.125.111]:55555
    Transaction = 50008 {
       Context = 5000 {
```

```
          Notify = A5555 {ObservedEvents =1235 {
             19990729T24020002:al/on}
          }
       }
     }

    From MGC to MG2:

    MEGACO/1 [123.123.123.4]:55555
    Reply = 50008 {
```

```
      Context = - {Notify = A5555}
   }

   22. The MGC now sends both MGs a Subtract to take down the call.
Only
   the subtracts to MG2 are shown here. Each termination has its own
set
   of statistics that it gathers. An MGC may not need to request both
to
   be returned. A5555 is a physical termination, and A5556 is an RTP
   termination.

   From MGC to MG2:

   MEGACO/1 [123.123.123.4]:55555
   Transaction = 50009 {
      Context = 5000 {
         Subtract = A5555 {Audit{Statistics}},
         Subtract = A5556 {Audit{Statistics}}
      }
   }

   From MG2 to MGC:

   MEGACO/1 [125.125.125.111]:55555
   Reply = 50009 {
      Context = 5000 {
        Subtract = A5555 {
            Statistics {
               nt/os=45123, ; Octets Sent
               nt/dur=40 ; in seconds
               }
          },
          Subtract = A5556 {
            Statistics {
               rtp/ps=1245, ; packets sent
               nt/os=62345, ; octets sent
               rtp/pr=780, ; packets received
               nt/or=45123, ; octets received
               rtp/pl=10, ;  % packets lost
               rtp/jit=27,
```

```
               rtp/delay=48 ; average latency
            }
         }
      }
   }

   23. The MGC now sets up both MG1 and MG2 to be ready to detect the
   next off-hook event. See step 1. Note that this could be the
default
   state of a termination in the null context, and if this were the
```

case, no message need be sent from the MGC to the MG. Once a
termination returns to the null context, it goes back to the
default
termination values for that termination.

Authors' Addresses

    Fernando Cuervo
    Nortel Networks
    P.O. Box 3511, Station C
    Ottawa, ON K1Y 4H7
    Canada
    E-mail: cuervo@nortelnetworks.com

    Nancy Greene
    Nortel Networks

       P.O. Box 3511, Station C
       Ottawa, ON K1Y 4H7
       Canada
       E-mail: ngreene@nortelnetworks.com

       Christian Huitema*
       Microsoft Corporation
       One Microsoft Way
       Redmond, WA 98052-6399
       USA
       E-mail: huitema@microsoft.com

       * The contribution of Christian Huitema was developed while he was
         employed by Telcordia Technologies.

       Abdallah Rayhan
       Nortel Networks
       P.O. Box 3511, Station C
       Ottawa, ON K1Y 4H7
       Canada
       E-mail: arayhan@nortelnetworks.com

       Brian Rosen
       Marconi
       1000 FORE Drive
       Warrendale, PA 15086
       USA
       E-mail: brian.rosen@marconi.com

       John Segers
       Lucent Technologies, Room HE 303
       Dept. Forward Looking Work
       P.O. Box 18, 1270 AA Huizen
       The Netherlands
       E-mail: jsegers@lucent.com

document itself may not be modified in any way, such as by removing
   the copyright notice or references to the Internet Society or other
   Internet organizations, except as needed for the purpose of
   developing Internet standards in which case the procedures for
   copyrights defined in the Internet Standards process must be
   followed, or as required to translate it into languages other than
   English.

   The limited permissions granted above are perpetual and will not be
   revoked by the Internet Society or its successors or assigns.

   This document and the information contained herein is provided on an
   "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING
   TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING
   BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
   HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
   MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement