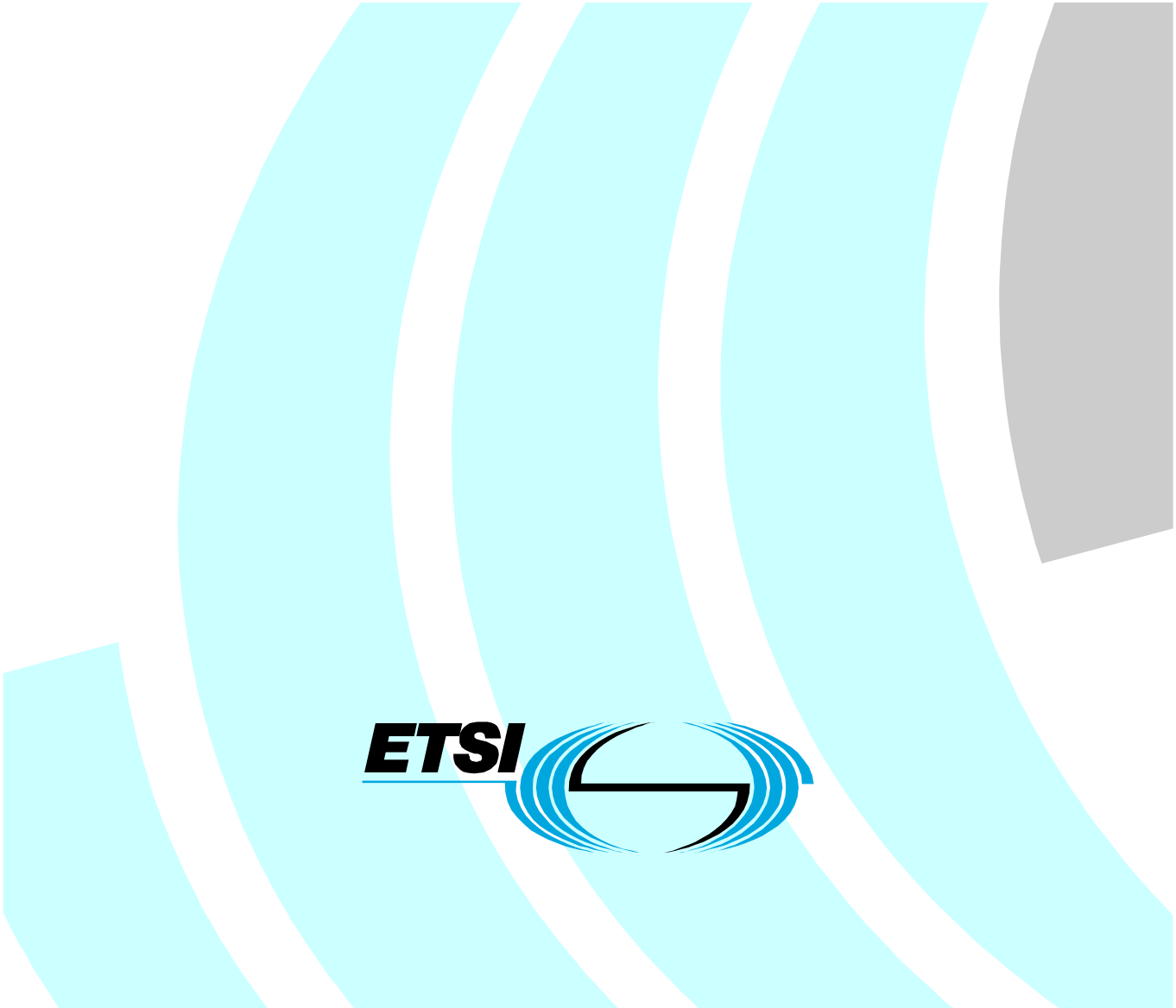
ETSI TS 101 909-13-2 V1.1.2 (2002-03)

Technical Specification

Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 13: Trunking Gateway Control Protocol; Sub-part 2: MGCP option



Reference
RTS/AT-020028-13-2

Keywords
access, broadband, cable, IP, multimedia, PSTN

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.fr

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Access and Terminals (AT).

The present document is part 13, sub-part 2 of a multi-part deliverable. Full details of the entire series can be found in part 1 [7].

Introduction

The present document specifies a profile of the Media Gateway Control Protocol (MGCP) for controlling media gateways between cable networks and the PSTN.

The present document defines a solution based on MGCP. The solution based on H.248 is defined in TS 101 909-13-1.

Where alternative solutions for the same interface are being considered, interoperability issues between the various IPCablecom system components need to be addressed.

1 Scope

The present set of documents specify IPCablecom, a set of protocols and associated element functional requirements. These have been developed to deliver Quality of Service (QoS), enhanced secure IP multimedia time critical communication services, using packetized data transmission technology to a consumer's home over a cable television Hybrid Fibre/Coaxial (HFC) data network.

NOTE 1: IPCablecom set of documents utilize a network superstructure that overlays the two-way data-ready cable television network, e.g. as specified within ES 201 488 [5] and ES 200 800 [6].

While the initial service offerings in the IPCablecom product line are anticipated to be Packet Voice and Packet Video, the long-term project vision encompasses a large family of packet-based services. This may require in the future, not only careful maintenance control, but also an extension of the present set of documents.

NOTE 2: The present set of documents aims for global acceptance and applicability. It is therefore developed in alignment with standards either already existing or under development in other regions and in International Telecommunications Union (ITU).

The present document specifies a profile of the MGCP protocol for controlling media gateways between cable networks and the PSTN.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] IETF/IETF/RFC 2327 (1998): "SDP: Session Description Protocol".

[2] ETSI TS 101 909-4: "Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 4: Network Call Signalling Protocol".

[3] ETSI TS 101 909-11: "Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 11: Security".

[4] ETSI TS 101 909-3: "Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 3: Audio Codec Requirements for the Provision of Bi-Directional Audio Service over Cable Television Networks using Cable Modems".

[5] ETSI ES 201 488: "Data-Over-Cable Service Interface Specifications Radio Frequency Interface Specification".

[6] ETSI ES 200 800: "Digital Video Broadcasting (DVB); DVB interaction channel for Cable TV distribution systems (CATV)".

[7] ETSI TS 101 909-1: "Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 1: General".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AAD	Average Acknowledgement Delay
ADEV	Average DEViation
API	Application Programming Interface
BR	BRief
CAS	Channel Associated Signalling
CCC	Call Content Connection
COT	COntinuity Test
DNS	Domain Name System
HFC	Hybrid Fibre/Coaxial
IP	Internet Protocol
ISUP	ISDN User Part
MF	Multi-Frequency
MGC	Media Gateway Controller
MGCI	Media Gateway Controller Interface
MGCP	Media Gateway Control Protocol
MIB	Management Information Base
MTA	Multimedia Terminal Adapter
MWD	Maximum Waiting Delay
NCS	Network-based Call Signalling
NTP	Network Time Protocol
OO	On/Off
OSS	Operational Support System
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RPC	Remote Procedure Call
RTCP	Real Time Control Protocol
RTO	Retransmission TimeOut
RTP	Real Time Protocol
SDP	Session Description Protocol
SNMP	Simple Network Management Protocol
SPI	Security Parameters Index
SSRC	Synchronization SouRCe
TDD	Telecom Devices for the Deaf tones
TGCP	Trunking Gateway Control Protocol
TO	Time-Out
TTL	Time To Live
UDP	User Datagram Protocol

4 Media Gateway Control Interface (MGCI)

MGCI functions provide for connection control, endpoint control, auditing, and status reporting. They each use the same system model and the same naming conventions.

4.1 Model and naming conventions

The MGCP assumes a connection model where the basic constructs are endpoints and connections. Connections are grouped in calls. One or more connections can belong to one call. Connections and calls are set up at the initiative of one or several MGCs. It should nonetheless be recognized that in none of these cases is a "connection" established within an IPCablecom network, as the term "connection" is understood within the circuit-switched PSTN. The terms "call" and "connection" in this context (and throughout the present document) are used for convenience of reference, not to indicate any actual technical or other similarity between the IPCablecom network and the PSTN.

4.1.1 Endpoint names

Endpoint names, as known as endpoint identifiers, have two components, both of which are defined to be case insensitive here:

- the domain name of the gateway managing the endpoint;
- a local endpoint name within that gateway.

Endpoint names will be of the form:

- `local-endpoint-name@domain-name`,

where domain-name is an absolute domain-name as defined in IETF/RFC 1034 (see bibliography) and includes a host portion, thus an example domain-name could be:

- `MyTrunkingGateway.cablelabs.com`.

Also, `domain-name` may be an IPv4 address in dotted decimal form represented as a text-string and surrounded by a left and a right square bracket ("[" and "]") as in "[128.96.41.1]" - please consult IETF/RFC 821 (see bibliography) for details. However, use of IP addresses is generally discouraged.

Trunking gateways have one or more endpoints (e.g. one for each trunk) associated with them, and each of the endpoints is identified by a separate local endpoint name. Just like the domain-name, the local endpoint name is case insensitive. Associated with the local endpoint name is an endpoint-type, which defines the type of the endpoint, e.g. DS-0, or an analog access line. The type can be derived from the local endpoint name. The local endpoint name is a hierarchical name, where the least specific component of the name is the leftmost term, and the most specific component is the rightmost term. More formally, the local endpoint name must adhere to the following naming rules:

- the individual terms of the local endpoint name must be separated by a single slash ("/", ASCII 2F hex);
- the individual terms are ASCII character strings composed of letters, digits or other printable characters, with the exception of characters used as delimiters in endpoint-names ("/", "@"), characters used for wildcarding ("*", "$"), and white space characters;
- wild carding is represented either by an asterisk ("*") or a dollar sign ("$") for the terms of the naming path which are to be wild-carded. Thus, if the full local endpoint name looks like:

  - `term1/term2/term3`;

  and one of the terms of the local endpoint name is wild-carded, then the local endpoint name looks like this:

  - `term1/term2/*` if `term3` is wild-carded.
    `term1/*/*` if `term2` and `term3` are wild-carded;

  In each of the examples, a dollar sign could have appeared instead of the asterisk.

- wild-carding is only allowed from the right, thus if a term is wild-carded, then all terms to the right of that term must be wild-carded as well;
- in cases where mixed dollar sign and asterisk wildcards are used, dollar-signs are only allowed from the right, thus if a term had a dollar sign wildcard, all terms to the right of that term must also contain dollar sign wildcards;
- a term represented by an asterisk is to be interpreted as: "use *all* values of this term known within the scope of the trunking gateway in question";
- a term represented by a dollar sign is to be interpreted as: "use *any one* value of this term known within the scope of the trunking gateway in question";
- each endpoint-type may specify additional detail in the naming rules for that endpoint-type, however such rules must not be in conflict with the above.

It should be noted that different endpoint-types or even different sub-terms, e.g. "lines", within the same endpoint-type will result in two different local endpoint names. Consequently, each "line" will be treated as a separate endpoint.

4.1.1.1 Trunking gateway endpoint names

Endpoints in trunking gateways will use the additional naming conventions specified in this clause.

Trunking gateways will support the following basic endpoint-type:

- `ds` A DS-0 trunk.

The basic endpoint type is expected to be provisioned with additional information about the type of signalling supported on the trunk circuit and the switching system role it provides.

- Trunk Circuit Endpoints.

In addition to the naming conventions specified above, local endpoint names for PSTN trunking gateway endpoints of type "`ds`" will adhere to the following:

- Local endpoint names will consist of a series of terms each separated by a slash ("/") that describe the physical hierarchy within the gateway:
  - `ds/<unit-type1>-<unit #>/<unit-type2>-<unit #>/…/<channel #>`.
- The first term (`ds`) identifies the endpoint naming scheme used and the basic endpoint type.
- The last term is a decimal number that indicates the *channel* number at the lowest level of the hierarchy (see note).

NOTE: Please note the use of the term "channel" as opposed to "timeslot".

- Intermediate terms between the first term (`ds`) and last term (channel number) represent intermediate levels of the hierarchy and consist of <unit-type> and <unit #> separated by a hyphen ("-") where:
  - the <unit-type> identifies the particular hierarchy level. Values of <unit-type> presently defined are: "s", "su", "oc3", "ds3", "e3", "ds2", "e2", "ds1", "e1" where "s" indicates a slot number and "su" indicates a sub-unit within a slot. Other values representing physical hierarchy levels that have not been included in this list but which follow the same basic naming rules will also be allowed;
  - the <unit #> is a decimal number which is used to reference to a particular instance of a <unit-type> at that level of the hierarchy.
- The number of levels and naming of those levels is based on the physical hierarchy within the media gateway, as illustrated by the following examples:
  - a Media Gateway that has some number of DS1 interfaces:
    - `ds/ds1-#/#`.
  - a Media Gateway that has some number of OC3 interfaces, that contain channelized DS3 and DS1 hierarchies:
    - `ds/oc3-#/ds3-#/ds1-#/#`.
  - a Media Gateway that contains some number of slots with each slot having some number of DS3 interfaces:
    - `ds/s-#/ds3-#/ds1-#/#`.
- Some endpoints may not contain all possible levels of a hierarchy, however all levels supported by a given endpoint are contained in the endpoint naming scheme. For example, a DS3 without DS1 framing could be represented by the following naming scheme:
  - `ds/s-#/ds3-#/#.`

  however, a DS3 *with* DS1 framing could not be represented by that naming scheme.

- Wildcard naming follows the conventions stated in clause 4.1.1 with the asterisk character ("*") referring to "all", and the dollar character ("$") referring to "any". A range "[N-M]" wildcarding convention representing a "range" of channels from channel N to channel M inclusive is supported as well:

  - it should be noted that the use of the "all" wildcard for the first (ds) term refers to all endpoint-types in the media gateway regardless of their type. Use of this feature is generally expected to be for administrative purposes, e.g. auditing or restart;

  - a local endpoint name may be under-specified by supplying some smaller than normal number of terms starting from the left-hand side of the endpoint name. In that case, missing terms to the right of the last term specified are assumed to be the wildcard character "*", referring to "all of", unless the terms specified contain the "any of" wildcard character, in which case missing terms to the right of the last term specified are assumed to be the "any of" wildcard character;

  - wherever use of the "all" wildcard is permitted, the range of channels "[N-M]" wildcard may be used in the last term (i.e. <channel-#>) of the local endpoint name instead. The "range" wildcard will then refer to all of the channels from N to M. The rules and restrictions that apply to the use of the "all" wildcard also apply to the use of the "range" wildcard.

The following examples illustrate the use of wild-carding:

- `ds/ds1-3/*` All channels on ds1 number 3 on the media gateway in question.
- `ds/ds1-3/$` Any channel on ds1 number 3 on the media gateway in question.
- `ds/*` All trunk-circuit endpoints on the media gateway in question.
- `*` All endpoints (regardless of endpoint-type) on the media gateway in question.
- `ds/ds1-3/[1-24]` Channels 1 to 24 on ds1 number 3 on the media gateway in question.

The above define the canonical names for endpoints in a trunking gateway. It is expected that aliasing may be supported in a future version of the present document, e.g. to support bonding of multiple DS-0 trunks for video calls, e.g. on the form "`ds/ds1-1/H0-1`".

4.1.2 Call names

Calls are identified by unique identifiers, independent of the underlying platforms or agents. Call identifiers are hexadecimal strings, which are created by the MGC. The maximum length of call identifiers is 32 characters.

At a minimum, call identifiers MUST be unique within the collection of MGCs that control the same gateways. However, the coordination of these call identifiers between MGCs is outside the scope of the present document. When an MGC builds several connections that pertain to the same call, either on the same gateway or in different gateways, these connections all will be linked to the same call through the call identifier. This identifier then can be used by accounting or management procedures, which are outside the scope of MGCP.

4.1.3 Connection names

Connection identifiers are created by the gateway when it is requested to create a connection. They identify the connection within the context of an endpoint. Connection identifiers are treated in MGCP as hexadecimal strings. The gateway MUST ensure that a proper waiting period, at least three minutes, elapses between the end of a connection that used this identifier and its use in a new connection for the same endpoint. The maximum length of a connection name is 32 characters.

4.1.4 Names of media gateway controllers and other entities

The Media Gateway Control Protocol has been designed for enhanced network reliability to allow implementation of redundant MGCs. This means that there is no fixed binding between entities and hardware platforms or network interfaces.

MGC names consist of two parts, similar to endpoint names. The local portion of the name does not exhibit any internal structure. An example MGC name is:

- `mgc1@mgc.whatever.net`.

Reliability is provided by the following precautions:

- entities such as trunking gateways or MGCs are identified by their domain name, not their network addresses. Several addresses can be associated with a domain name. If a command cannot be forwarded to one of the network addresses, implementations MUST retry the transmission using another address;

- entities may move to another platform. The association between a logical name (domain name) and the actual platform are kept in the Domain Name Service (DNS). MGCs and gateways MUST keep track of the record's time-to-live read from the DNS. They MUST query the DNS to refresh the information if the time-to-live has expired.

In addition to the indirection provided by the use of domain names and the DNS, the concept of "notified entity" is central to reliability and fail-over in MGCP. The "notified entity" for an endpoint is the MGC currently controlling that endpoint. At any point in time, an endpoint has one, and only one, "notified entity" associated with it, and when the endpoint needs to send a command to the MGC, it MUST send the command to the current "notified entity" for which endpoint(s) the command pertains. Upon startup, the "notified entity" MUST be set to a provisioned value. Most commands sent by the MGC include the ability to explicitly name the "notified entity" through the use of a "NotifiedEntity" parameter. The "notified entity" will stay the same until either a new "NotifiedEntity" parameter is received or the endpoint reboots. If the "notified entity" for an endpoint is empty or has not been set explicitly (see note), the "notified entity" will then default to the source address of the last connection handling command or notification request received for the endpoint. Auditing will thus not change the "notified entity".

NOTE: This could happen as a result of specifying an empty NotifiedEntity parameter.

Clause 4.4 contains a more detailed description of reliability and fail-over.

4.1.5 Digit maps

In MGCP, the MGC can ask the gateway to collect digits dialled by a user. This facility is typically used by analog access lines with residential gateways to collect the numbers that a user dials, or it can be used for CAS PBX interfaces. Rather than sending each digit to the MGC as the digits are detected, the MGC can provide a grammar describing how many digits should be accumulated before the MGC is notified. This grammar is known as a *digit map*.

None of the trunk types supported by the current version of the TGCP specification have a need for digit maps, and digit maps are therefore not part of the present document.

4.1.6 Events and signals

The concept of events and signals is central to MGCP. An MGC may ask to be notified about certain events occurring in an endpoint, e.g. off-hook events. An MGC also may request certain signals to be applied to an endpoint, e.g. ringback.

Events and signals are grouped in packages within which they share the same namespace, which we will refer to as event names in the following. A package is a collection of events and signals supported by a particular endpoint-type. For instance, one package may support a certain group of events and signals for ISUP trunks, and another package may support another group of events and signals for MF trunks. One or more packages may exist for a given endpoint-type, and each endpoint-type has a default package with which it is associated.

Event names consist of a package name and an event code and, since each package defines a separate namespace, the same event codes may be used in different packages. Package names and event codes are case insensitive strings of letters, digits, and hyphens, with the restriction that hyphens shall never be the first or last character in a name. Some event codes may need to be parameterized with additional data, which is accomplished by adding the parameters between a set of parentheses. The package name is separated from the event code by a slash ("/"). The package name may be excluded from the event name, in which case the default package name for the endpoint-type in question is assumed. For example, for an ISUP trunk circuit with the ISUP package (package name "IT") being the default package, the following two event names are considered equal:

- `IT/oc` Operation Complete in the ISUP package for an ISUP trunk circuit.

- `oc` Operation Complete in the ISUP package (default) for an ISUP trunk circuit.

Clause 7 defines an initial set of packages. Additional package names and event codes may be defined by and/or registered with IPCablecom. Any change to the packages defined in the present document MUST result in a change of the package name, or a change in the TGCP profile version number, or possibly both.

Each package MUST have a package definition, which MUST define the name of the package, and the definition of each event belonging to the package. The event definition MUST include the precise name of the event, i.e. the event code, a plain text definition of the event and, when appropriate, the precise definition of the corresponding signals, for example the exact frequencies of audio signals such as ringback or fax tones. Events must further specify if they are persistent (see clause 4.3.1) and if they contain auditable event-states (see clause 4.3.8.1). Signals MUST also have their type defined (On/Off, Time-Out, or Brief), and Time-Out signals MUST have a default time-out value defined - see clause 4.3.1.

In addition to IPCablecom packages, implementers MAY gain experience by defining experimental packages. The package name of experimental packages MUST begin with the two characters "x-" or "X-"; IPCablecom MUST NOT register package names that start with these two characters. A gateway that receives a command referring to an unsupported package MUST return an error (error code 518 - unsupported package).

Package names and event codes support one wildcard notation each. The wildcard character "*" (asterisk) can be used to refer to all packages supported by the endpoint in question, and the event code "all" to all events in the package in question. For example:

- `IT/all` refers to all events in the ISUP trunk package for an ISUP trunk circuit.
- `*/all` for an ISUP trunk circuit; refers to all packages and all events in those packages supported by the endpoint in question.

Consequently, the package name "*" MUST NOT be assigned to a package, and the event code "all" MUST NOT be used in any package.

Events and signals are by default detected and generated on endpoints, however some events and signals may be detected and generated on connections in addition to or instead of on an endpoint. For example, endpoints may be asked to provide a ringback tone on a connection. In order for an event or signal to be able to be detected or generated on a connection, the definition of the event/signal MUST explicitly define that the event/signal can be detected or generated on a connection.

When a signal shall be applied on a connection, the name of the connection is added to the name of the event, using an "at" sign (@) as a delimiter, as in:

- `IT/rt@0A3F58`.

The wildcard character "*" (asterisk) can be used to denote "all connections" on the affected endpoint(s). When this convention is used, the gateway will generate or detect the event on all the connections that are connected to the endpoint(s). An example of this convention is:

- `IT/ma@*`.

The wildcard character "$" (dollar sign) can be used to denote "the current connection". This convention MUST NOT be used unless the event notification request is "encapsulated" within a CreateConnection or ModifyConnection command. When the convention is used, the gateway will generate or detect the event on the connection that is currently being created or modified. An example of this convention is:

- `IT/rt@$`.

The connection id, or a wildcard replacement, can be used in conjunction with the "all packages" and "all events" conventions. For example, the notation:

- `*/all@*`,

can be used to designate all events on all connections for the affected endpoint(s).

4.2 SDP use

The MGC uses the MGCP to provide the gateways with the description of connection parameters such as IP addresses, UDP port, and RTP profiles. Except where otherwise noted or implied in the present document, SDP descriptions MUST follow the conventions delineated in the Session Description Protocol (SDP), which is now an IETF-proposed standard documented in IETF/RFC 2327 [1].

SDP allows for description of multimedia conferences. The TGCP profile will only support the setting of audio connections using the media type "audio".

4.3 Gateway control functions

This clause describes the commands of the MGCP in the form of a remote procedure call (RPC) like API, which we will refer to as the media gateway control interface (MGCI). An MGCI function is defined for each MGCP command, where the MGCI function takes and returns the same parameters as the corresponding MGCP command. The functions shown in this clause provide a high-level description of the operation of MGCP and describe an example of an RPC-like API that MAY be used for an implementation of MGCP. Although the MGCI API is merely an example API, the semantic behaviour defined by MGCI is an integral part of the Specification, and all implementations MUST conform to the semantics specified for MGCI. The actual MGCP messages exchanged, including the message formats and encodings used are defined in the protocol clause (clause 5). Trunking gateways MUST implement those exactly as specified.

The MGCI service consists of connection handling and endpoint handling commands. The following is an overview of the commands:

- The MGC can issue a NotificationRequest command to a gateway, instructing the gateway to watch for specific events such as seizure or fax tones on a specified endpoint.
- The gateway will then use the Notify command to inform the MGC when the requested events occur on the specified endpoint.
- The MGC can use the CreateConnection command to create a connection that terminates in an endpoint inside the gateway.
- The MGC can use the ModifyConnection command to change the parameters associated to a previously established connection.
- The MGC can use the DeleteConnection command to delete an existing connection. In some circumstances, the DeleteConnection command also can be used by a gateway to indicate that a connection can no longer be sustained.
- The MGC can use the AuditEndpoint and AuditConnection commands to audit the status of an "endpoint" and any connections associated with it. Network management beyond the capabilities provided by these commands are generally desirable, e.g. information about the status of the trunking gateway and each of the trunk circuits. Such capabilities are expected to be supported by the use of the Simple Network Management Protocol (SNMP) and definition of a MIB, which is outside the scope of the present document.
- The gateway can use the RestartInProgress command to notify the MGC that the endpoint, or a group of endpoints managed by the gateway, is being taken out of service or is being placed back in service.

These services allow a controller (normally the MGC) to instruct a gateway on the creation of connections that terminate in an endpoint attached to the gateway, and to be informed about events occurring at the endpoint. Currently, a trunking gateway endpoint is limited to a specific trunk circuit within a trunking gateway.

Connections are grouped into "calls". Several connections, that may or may not belong to the same call, can terminate in the same endpoint. Each connection is qualified by a "mode" parameter, which can be set to "send only" (sendonly), "receive only" (recvonly), "send/receive" (sendrecv), "inactive" (inactive), "loopback" (loopback), "continuity test" (conttest), "network loopback" (netwloop) or "network continuity test" (netwtest). The "mode" parameter determines if media packets can be sent and/or received on the connection; however, RTCP is unaffected.

Audio signals received from the endpoint will be sent on any connection for that endpoint whose mode is either "send only", or "send/receive.

Handling of the audio signals received on these connections is also determined by the mode parameters:

- Audio signals received in data packets through connections in "inactive", "loopback" or "continuity test" mode are discarded.
- Audio signals received in data packets through connections in "receive only", or "send/receive" mode are mixed together and then sent to the endpoint (see note).
- Audio signals originating from the endpoint are transmitted over all the connections whose mode is "send only", or "send/receive".
- Audio signals received in data packets through connections in "network loopback" or "network continuity test" mode will be sent back on the connection as described below.

NOTE: TGCP endpoints are currently not required to support mixing though.

The "loopback" and "continuity test" modes are used during maintenance and continuity test operations. There are two variations of COntinuity Test (COT), one specified for general use and one used in several national networks. In the first case, the test is a loopback test. The originating switch will send a tone (the go tone) on the bearer circuit and expect the terminating switch to loopback the circuit. If the originating switch sees the same tone returned (the return tone), the COT has passed.

If not, the COT has failed. In the second case, the go and return tones are different. The originating switch sends a certain go tone. The terminating switch detects the go tone, it asserts a different return tone in the backwards direction. When the originating switch detects the return tone, the COT is passed. If the originating switch does not detects the return tone within a certain period of time, the COT has failed.

If the mode is set to "loopback", the gateway is expected to return the incoming signal from the endpoint back into that same endpoint. This is the general procedure. If the mode is set to "continuity test", the gateway is informed that the other end of the circuit has initiated a continuity test procedure according to procedures specified for several national networks. The gateway will place the circuit in the transponder mode required for dual-tone continuity tests.

Furthermore, when a connection for an endpoint is in "loopback" or "continuity test" mode:

- Audio signals received on any connection for the endpoint will *not* be sent to the endpoint.
- Audio signals received on the endpoint will *not* be sent to any connection for the endpoint.

If the mode is set to "network loopback", the audio signals received from the connection will be echoed back on the same connection. The "network loopback" mode SHOULD simply operate as an RTP packet reflector.

The "network continuity test" mode is used for continuity checking across the IP network. An endpoint-type specific signal is sent to the endpoints over the IP network, and the endpoint is then supposed to echo the signal over the IP network after passing it through the gateway's internal equipment to verify proper operation. The signal MUST go through internal decoding and re-encoding prior to being passed back. For DS-0 endpoints the signal will be an audio signal, and the signal MUST NOT be passed on to a circuit connected to the endpoint, regardless of the current seizure-state of that circuit.

New and existing connections for the endpoint MUST NOT be affected by connections placed in "network loopback" or "network continuity test" mode. However, local resource constraints may limit the number of new connections that can be made.

4.3.1 NotificationRequest

The NotificationRequest command is used to request the gateway to send a notification upon the occurrence of specified events in an endpoint. For example, a notification may be requested when tones associated with fax communication are detected on the endpoint. The entity receiving this notification, usually the MGC, may then decide that a different type of encoding should be used on the connections bound to this endpoint and instruct the gateway accordingly (see note).

NOTE 1: The new instruction would be a ModifyConnection command.

```
ReturnCode
    ← NotificationRequest(EndpointId
                    [, NotifiedEntity]
                    [, RequestedEvents]
                    , RequestIdentifier
                    [, SignalRequests]
                    [, QuarantineHandling]
                    [, DetectEvents])
```

EndpointId: is the identifier for the endpoint(s) in the gateway where NotificationRequest executes. The EndpointId follows the rules for endpoint names specified in clause 4.1.1. The "any of" wildcard MUST NOT be used.

NotifiedEntity: is an optional parameter that specifies a new "notified entity" for the endpoint.

RequestIdentifier: is used to correlate this request with the notification it may trigger. It will be repeated in the corresponding Notify command.

SignalRequests: is a parameter that contains the set of signals that the gateway is asked to apply. Unless otherwise specified, signals are applied to the endpoint, however some signals can be applied to a connection. The following are examples of signals (see note):

- Continuity test;
- Setup MF OSS call.

NOTE 2: Please refer to annex A for a complete list of signals.

Signals are divided into different types depending upon their behaviour:

- `On/off (OO):` Once applied, these signals last until they are turned off. This can only happen as the result of a new SignalRequests where the signal is turned off (see later). Signals of type OO are defined to be idempotent, thus multiple requests to turn a given OO signal on (or off) are perfectly valid and MUST NOT result in any errors. Once turned on, it MUST NOT be turned off until explicitly instructed to by the MGC, or the endpoint restarts.

- `Time-out (TO):` Once applied, these signals last until they are either cancelled (by the occurrence of an event or by not being included in a subsequent [possibly empty] list of signals), or a signal-specific period of time has elapsed. A signal that times-out will generate an "operation complete" event (please see A for further definition of this event). A TO signal could be "place MF call" timing out after 16 s. If an event occurs prior to the 16 s, the signal will, by default, be stopped (see note 1). If the signal is not stopped, the signal will time-out, stop and generate an "operation complete" event, about which the MGC may or may not have requested to be notified. If the MGC has asked for the "operation complete" event to be notified, the "operation complete" event sent to the MGC will include the name(s) of the signal(s) that timed out (see note 2). Signal(s) generated on a connection will include the name of that connection. Time-out signals have a default time-out value defined for them, which may be altered by the provisioning process. Also, the time-out period may be provided as a parameter to the signal. A value of zero indicates that the time-out period is infinite. A TO signal that fails after being started, but before having generated on "operation complete" event will generate an "operation failure" event which will include the name(s) of the signal(s) that time out (see note 2).

- `Brief (BR):` The duration of these signals is so short that they stop on their own. If a signal stopping event occurs, or a new SignalRequests is applied, a currently active BR signal will not stop. However, any pending BR signals not yet applied will be cancelled.

NOTE 3: The "Keep signal(s) active" action may override this behaviour.

NOTE 4: If parameters were passed to the signal, the parameters will not be reported.

Signals are, by default, applied to endpoints. If a signal applied to an endpoint results in the generation of a media stream (audio, video, *etc.*), the media stream MUST NOT be forwarded on any connection associated with that endpoint, regardless of the mode of the connection.

EXAMPLE: If a tone is applied to an endpoint involved in an active communication, only the party using the endpoint in question will hear the tone. However, individual signals may define a different behaviour.

When a signal is applied to a connection that has received a RemoteConnectionDescriptor (see clause 4.3.3), the media stream generated by that signal will be forwarded on the connection *regardless* of the current mode of the connection. If a RemoteConnectionDescriptor has not been received, the gateway MUST return an error (error code 527 - missing RemoteConnectionDescriptor).

When a (possibly empty) list of signal(s) is supplied, this list completely replaces the current list of active time-out signals. Currently active time-out signals that are not provided in the new list MUST be stopped and the new signal(s) provided will now become active. Currently active time-out signals that are provided in the new list of signals MUST remain active without interruption, thus the timer for such time-out signals will not be affected. Consequently, there is currently no way to restart the timer for a currently active time-out signal without turning the signal off first. If the time-out signal is parameterized, the original set of parameters will remain in effect, regardless of what values are provided subsequently. A given signal MUST NOT appear more than once in a SignalRequests.

The currently defined signals can be found in clause 7.

RequestedEvents: is a list of events that the gateway is requested to detect on the endpoint. Unless otherwise specified, events are detected on the endpoint, however some events can be detected on a connection. Examples of events are:

- seizure;
- fax tones;
- operation complete;
- incoming MF call.

The currently defined events can be found in clause 7.

To each event is associated one or more **actions** that define the action that the gateway must take when the event in question occurs. The possible actions are:

- notify the event immediately, together with the accumulated list of observed events;
- accumulate the event;
- ignore the event;
- Keep Signal(s) active;
- Embedded NotificationRequest;
- Embedded ModifyConnection.

Two sets of requested events will be detected by the endpoint; persistent and non-persistent.

Persistent events are always detected on an endpoint. If a persistent event is not included in the list of RequestedEvents, and the event occurs, the event will be detected anyway, and processed like all other events, as if the persistent event had been requested with a Notify action (see note 3). Thus, informally, persistent events can be viewed as always being implicitly included in the list of RequestedEvents with an action to Notify, although no glare detection, etc., will be performed (see note 4). Persistent events are identified as such through their definition - see clause 7.

NOTE 5: Thus the RequestIdentifier will be the RequestIdentifier of the current NotificationRequest.

NOTE 6: Normally, if a request to look for, e.g. off-hook, is made, the request is only successful if the phone is not already off-hook.

Non-persistent events are those events that have to be explicitly included in the RequestedEvents list. The (possibly empty) list of requested events completely replaces the previous list of requested events. In addition to the persistent events, only the events specified in the requested events list will be detected by the endpoint. If a persistent event is included in the RequestedEvents list, the action specified will then replace the default action associated with the event for the life of the RequestedEvents list, after which the default action is restored. A given event MUST NOT appear more than once in a RequestedEvents.

More than one action can be specified for an event, although a given action can not appear more than once for a given event. The following matrix specifies the legal combinations of actions.

Table 1

	Notify	Accumulate	Ignore	Keep Signal(s) Active	Embedded Notification Request	Embedded ModifyConnection
Notify	-	-	-	√	-	√
Accumulate	-	-	-	√	√	√
Ignore	-	-	-	√	-	√
Keep Signal(s) active	√	√	√	-	√	√
Embedded NotificationRequest	-	√	-	√	-	√
Embedded ModifyConnection	√	√	√	√	√	-

If a client receives a request with an invalid action or illegal combination of actions, it MUST return an error to the MGC (error code 523-unknown or illegal combination of actions).

When multiple actions are specified, e.g. "Keep signal(s) active" and "Notify", the individual actions are assumed to occur simultaneously.

The MGC can send a NotificationRequest with an empty RequestedEvents list to the gateway. However, persistent events will still be detected and notified.

The signals being applied by the SignalRequests are synchronized with the collection of events specified or implied in the RequestedEvents parameter, except if overridden by the "Keep signal(s) active" action. The formal definition is that the generation of all "Time Out" signals MUST stop as soon as one of the requested events is detected, unless the "Keep signal(s) active" action is associated to the specified event.

If it is desired that time-out signal(s) continue when a looked-for event occurs, the "Keep Signal(s) Active" action can be used. This action has the effect of keeping all currently active time-out signal(s) active, thereby negating the default stopping of time-out signals upon the event's occurrence.

If signal(s) are desired to start when a looked-for event occurs, the "Embedded NotificationRequest" action can be used. The embedded NotificationRequest may include a new list of RequestedEvents, and a new SignalRequests. However, the "Embedded NotificationRequest" cannot include another "Embedded NotificationRequest". When the "Embedded NotificationRequest" is activated, the list of observed events and the quarantine buffer will be unaffected (see clause 4.4.3.1).

The embedded NotificationRequest action allows the MGC to set up a "mini-script" to be processed by the gateway immediately following the detection of the associated event. Any SignalRequests specified in the embedded NotificationRequest will start immediately. Considerable care must be taken to prevent discrepancies between the MGC and the gateway. However, long-term discrepancies should not occur as new SignalRequests completely replaces the old list of active time-out signals, and BR-type signals always stop on their own. Limiting the number of On/Off-type signals is encouraged. It is considered good practice for an MGC to occasionally turn on all On/Off signals that should be on, and turn off all On/Off signals that should be off.

If connection modes are desired to be changed when a looked-for event occurs, the "Embedded ModifyConnection" action can be used. The embedded ModifyConnection may include a list of connection mode changes each consisting of the mode change and the affected connection-id. The wildcard "$" can be used to denote "the current connection", however this notation MUST NOT be used outside a connection handling command - the wildcard refers to the connection in question for the connection handling command.

The embedded ModifyConnection action allows the MGC to instruct the endpoint to change the connection mode of one or more connections immediately following the detection of the associated event. Each of connection mode changes work similarly to a corresponding ModifyConnection command. When a list of connection mode changes is supplied, the connection mode changes MUST be applied one at a time in left-to-right order. When all the connection mode changes have finished, an "operation complete" event parameterized with the name of the completed action will be generated (see clause 7 for details). Should any of the connection mode changes fail, an "operation failure" event parameterized with the name of the failed action and connection mode change will be generated (see clause 7 for details) - the rest of the connection mode changes MUST NOT be attempted, and the previous successful connection mode changes in the list MUST NOT be changed either.

Finally, the Ignore action can be used to ignore an event, e.g. to prevent a persistent event from being notified. However, the synchronization between the event and an active signal will still occur by default.

Clause 4.4.3.1 contains additional details on the semantics of event detection and reporting. The reader is encouraged to study it carefully.

The specific definition of actions that are requested via these SignalRequests is outside the scope of the core TGCP specification. This definition may vary from location to location and, hence, from gateway to gateway. Consequently, the definitions are provided in event packages, which may be provided outside of the core specification. An initial list of event packages can be found in annex A.

The RequestedEvents and SignalRequests generally refer to the same events. In one case, the gateway is asked to detect the occurrence of the event and, in the other case, it is asked to generate it. There are only a few exceptions to this rule, notably the fax and modem tones, which can be detected but can not be signalled. However, we necessarily cannot expect all endpoints to detect all events. The specific events and signals that a given endpoint can detect or perform are determined by the list of event packages that are supported by that endpoint. Each package specifies a list of events and signals that can be detected or applied. A gateway that is requested to detect or to apply an event belonging to a package that is not supported by the specified endpoint MUST return an error (error code 512 or 513 - not equipped to detect event or generate signal). When the event name is not qualified by a package name, the default package name for the endpoint is assumed. If the event name is not registered in this default package, the gateway MUST return an error (error code 522 - no such event or signal).

The MGC can send a NotificationRequest whose requested signal list is empty. This has the effect of stopping all active time-out signals. It can do so, for example, when tone generation, e.g. ringback, should stop.

QuarantineHandling: is an optional parameter that specifies handling options for the quarantine buffer (see clause 4.4.3.1). It allows the MGC to specify whether quarantined events should be processed or discarded. If the parameter is absent, the quarantined events MUST be processed.

DetectEvents: is an optional parameter that specifies a minimum list of events that the gateway is requested to detect in the "notification" and "lockstep" state. The list is persistent until a new value is specified. Further explanation of this parameter may be found in clause 4.4.3.1.

ReturnCode: is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

4.3.2 Notifications

Notifications are sent via the Notify command by the gateway when an observed event is to be notified:

```
ReturnCode
    ← Notify(EndpointId
            [, NotifiedEntity]
            , RequestIdentifier
            , ObservedEvents)
```

EndpointId: is the name for the endpoint in the gateway, which is issuing the Notify command, as defined in clause 4.1.1. The identifier MUST be a fully qualified endpoint name, including the domain name of the gateway. The local part of the name MUST NOT use the wildcard convention.

NotifiedEntity: is an optional parameter that identifies the entity to which the notification is sent. This parameter is equal to the NotifiedEntity parameter of the NotificationRequest that triggered this notification. The parameter is absent if there was no such parameter in the triggering request. Regardless of the value of the "NotifiedEntity parameter, the notification MUST be sent to the current "notified entity" for the endpoint.

RequestIdentifier: is a parameter that repeats the RequestIdentifier parameter of the NotificationRequest that triggered this notification. It is used to correlate this notification with the notification request that triggered it. Persistent events will be viewed here as if they had been included in the last NotificationRequest. When no NotificationRequest has been received, the RequestIdentifier used will be zero ("0").

ObservedEvents: is a list of events that the gateway detected and accumulated, either by the "accumulate", or "notify" action. A single notification can report a list of events that will be reported in the order in which they were detected. The list can only contain persistent events and events that were requested in the RequestedEvents parameter of the triggering NotificationRequest. Events that were detected on a connection will include the name of that connection. The list will contain the events that were either accumulated (but not notified), and the final event that triggered the notification.

ReturnCode: is a parameter returned by the MGC. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

4.3.3 CreateConnection

This command is used to create a connection.

```
ReturnCode
, ConnectionId
[, SpecificEndPointId]
, LocalConnectionDescriptor,
        ← CreateConnection(CallId                                          , EndpointId
                        [, NotifiedEntity]
                        , LocalConnectionOptions
                        , Mode
                        [, RemoteConnectionDescriptor]
                        [, RequestedEvents]
                        [, RequestIdentifier]
                        [, SignalRequests]
                        [, QuarantineHandling]
                        [, DetectEvents])
```

This function is used when setting up a connection between two endpoints. A connection is defined by its attributes and the endpoints it associates. The input parameters in CreateConnection provide the data necessary to build one of the two endpoints "view" of a connection.

CallId: is a parameter that identifies the call (or session) to which this connection belongs. This parameter is, at a minimum, unique within the collection of MGCs that control the same gateways; connections that belong to the same call share the same call-id. The call-id can be used to identify calls for reporting and accounting purposes.

EndpointId: is the identifier for the endpoint in the gateway where CreateConnection executes. The EndpointId can be specified fully by assigning a non-wildcarded value to the parameter EndpointId in the function call or it can be under-specified by using the "anyone" wildcard convention. If the endpoint is under-specified, the endpoint identifier will be assigned by the gateway and its complete value returned in the **SpecificEndPointId** parameter of the response. The "all" wildcard convention MUST NOT be used.

NotifiedEntity: is an optional parameter that specifies a new "notified entity" for the endpoint.

LocalConnectionOptions: is a structure that describes the characteristics of the media data connection from the point-of-view of the gateway executing CreateConnection. It instructs the endpoint on send and receive characteristics of the media connection. The basic fields contained in LocalConnectionOptions are:

- **Encoding Method:** a list of literal names for the compression algorithm (encoding/decoding method) used to send and receive media on the connection MUST be specified with at least one value. The entries in the list are ordered by preference. The endpoint MUST choose exactly one of the codecs, and the codec SHOULD be chosen according to the preference indicated. If the endpoint receives any media on the connection encoded with a different encoding method, it MAY discard it. The endpoint MUST additionally indicate which of the remaining compression algorithms it is willing to support as alternatives - see clause 4.2 for details. A list of permissible encoding methods is specified in a separate IPCablecom document.

- **Packetization Period:** the packetization period in ms, as defined in the SDP standard (IETF/RFC 2327 [1]), MUST be specified and with exactly one value. The value only pertains to media sent. A list of permissible packetization periods is specified in a separate IPCablecom document.

- **Echo Cancellation:** whether echo cancellation should be used on the trunk side or not (see note 1). The parameter can have the value "on" (when the echo cancellation is requested) or "off" (when it is turned off). The parameter is optional. When the parameter is omitted, the trunking gateway MUST apply echo cancellation.

- **Type of Service:** specifies the class of service that will be used for sending media on the connection by encoding the 8-bit type of service value parameter of the IP header as two hexadecimal digits. The parameter is optional. When the parameter is omitted, a default value of $A0_H$ applies corresponding to an IP precedence bits setting of five.

- **Silence Suppression:** whether silence suppression should be used or not in the send direction. The parameter can have the value "on" (when silence is to be suppressed) or "off" (when silence is not to be suppressed). The parameter is optional. When the parameter is omitted, the default is not to use silence suppression.

NOTE 1: Echo cancellation on the packet side is not supported.

Additionally, the following LocalConnectionOptions fields are used to support the IPCablecom security services:

- **Secret:** the optional secret is a seed value that MUST be used to derive end-to-end encryption keys for the RTP and RTCP security services as specified in the TS 101 909-11 [3]. The secret SHOULD be encoded as clear-text if it only contains values in the ASCII character range 21_H to $7E_H$. Otherwise, the secret MUST be encoded using base64 encoding. If no value is supplied, or the parameter is omitted and security services are to be used, the endpoint MUST generate a secret on its own (see note 2). When a secret is supplied by the MGC, the secret SHOULD be used.

- **RTP ciphersuite:** a list of ciphersuites for RTP security in order of preference. The entries in the list are ordered by preference where the first ciphersuite is the preferred choice. The endpoint MUST choose exactly one of the ciphersuites. The endpoint SHOULD additionally indicate which of the remaining ciphersuites it is willing to support as alternatives (see clause 4.4.1 for details). Each ciphersuite is represented as ASCII strings consisting of two (possibly empty) substrings separated by a slash ("/"), where the first substring identifies the authentication algorithm, and the second substring identifies the encryption algorithm. A list of permissible ciphersuites are specified in the TS 101 909-11 [3].

- **RTCP ciphersuite:** a list of ciphersuites for RTCP security in order of preference. The entries in the list are ordered by preference where the first ciphersuite is the preferred choice. The endpoint MUST choose exactly one of the ciphersuites. The endpoint SHOULD additionally indicate which of the remaining ciphersuites it is willing to support as alternatives. See clause 4.2 for details. Each ciphersuite is represented as ASCII strings consisting of two (possibly empty) substrings separated by a slash ("/"), where the first substring identifies the authentication algorithm, and the second substring identifies the encryption algorithm. A list of permissible ciphersuites are specified in the TS 101 909-11 [3].

NOTE 2: This includes both generating a new secret and using a secret supplied in a RemoteConnectionDescriptor.

Furthermore, TGCP supports IPCablecom Electronic Surveillance (see PKT-SP-ESP-I01-991229). When a connection is subject to electronic surveillance, all valid media packets received on the connection and all media packets sent on the connect will be replicated and forwarded to an Electronic Surveillance Delivery Function (see note 3) after inclusion of a Call Content Connection Identifier. The replication will follow the connection mode for the connection, except for media generated by signals applies to the connection, which will be replicated regardless of the connection mode. For example, a connection in "inactive" mode will not generate any intercepted media (see note 4), whereas a connection in "sendonly" mode will only generate intercepted media in the send direction. Replicated packets will not be included in statistics for the connection. The following LocalConnectionOptions fields are used to support IPCablecom Electronic Surveillance (see PKT-SP-ESP-I01-991229 for details):

- **Call Content Connection Identifier:** the Call Content Connection (CCC) Identifier is a 32-bit value that specifies the Call Content Connection Identifier to be used for connection that are subject to electronic surveillance. It will be added to the header of intercepted voice packets.

- **Call Content Destination:** the Call Content Destination specifies an Ipv4 address followed by a colon and a UDP port number. The Call Content Destination specifies the destination IP address and port for the call content intercepted.

NOTE 3: Note that the replication occurs at the network level - see PKT-SP-ESP-I01-991229 for details.

NOTE 4: Assuming no media generating signal was applied to the connection.

The trunking gateway MUST respond with an error (error code 524 - LocalConnectionOptions inconsistency) if any of the above rules are violated. All of the above mentioned default values can be altered by the provisioning process.

RemoteConnectionDescriptor: is the connection descriptor for the remote side of a connection, on the other side of the IP network. It includes the same fields as the LocalConnectionDescriptor (not to be confused with LocalConnectionOptions), i.e. the fields that describe a session according to the SDP standard. Clause 4.2 details the supported use of SDP in the TGCP profile. This parameter may have a null value when the information for the remote end is not known. This occurs because the entity that builds a connection starts by sending a CreateConnection to one of the two gateways involved. For the first CreateConnection issued, there is no information available about the other side of the connection. This information may be provided later via a ModifyConnection call.

The TGCP profile currently assumes that the same media parameters apply to a connection in both the send and receive direction. Part of the information in the RemoteConnectionDescriptor is therefore redundant and a potential for inconsistency with the LocalConnectionOptions exists. It is however purely the responsibility of the MGC to ensure that it issues coherent commands to each endpoint to ensure that consistent media parameters are specified. If inconsistency is detected by a gateway though, the LocalConnectionOptions will simply take precedence. When codecs are changed during a communication, small periods of time may exist where the endpoints use different codes. As stated above, trunking gateways MAY discard any media received that is encoded with a different codec than what is specified in the LocalConnectionOptions for a connection.

Mode: indicates the mode of operation for this side of the connection. The options are "send only", "receive only", "send/receive", "inactive", "network loopback" or "network continuity test". The handling of these modes is specified in the beginning of clause 4.3. Some endpoints may not be capable of supporting all modes - see clause 4.2. If the command specifies a mode that the endpoint does not support, an error MUST be returned (error code 517 - unsupported mode). Also, if a connection has not yet received a RemoteConnectionDescriptor, an error MUST be returned if the connection is attempted to be placed in any of the modes "send only", or "send/receive" (error code 527 - missing RemoteConnectionDescriptor).

ConnectionId: is a parameter returned by the gateway that uniquely identifies the connection within the context of the endpoint in question.

LocalConnectionDescriptor: is a parameter returned by the gateway, which is a session description that contains information about, e.g. addresses and RTP ports for "IN" connections as defined in SDP. It is similar to the RemoteConnectionDescriptor, except that it specifies this side of the connection. Clause 4.2 details the supported use of SDP in the TGCP profile.

After receiving a "CreateConnection" command that does not include a RemoteConnectionDescriptor parameter, a gateway is in an ambiguous situation for the connection in question. Because it has exported a LocalConnectionDescriptor parameter, it potentially can receive packets on that connection. Because it has not yet received the other gateway's RemoteConnectionDescriptor parameter, it does not know whether the packets it receives have been authorized by the MGC. Thus, it must navigate between two risks, i.e. clipping some important announcements or listening to insane data. The behaviour of the gateway is determined by the value of the mode parameter (subject to security):

- if the mode was set to "receive only", the gateway MUST accept the voice signals received on the connection and transmit them through to the endpoint;
- if the mode was set to "inactive", "loopback", or "continuity test" the gateway MUST (as always) discard the voice signals received on the connection;
- if the mode was set to "network loopback" or "network continuity test" the gateway MUST perform the expected echo or response. The echoed or generated media MUST then be sent to the source of the media received.

NOTE 5: When the endpoint does not have a RemoteConnectionDescriptor for the connection, the connection can by definition not be in any of the modes "send only", or "send/receive".

The **RequestedEvents**, **RequestIdentifier**, **SignalRequests**, **QuarantineHandling**, and **DetectEvents** parameters are all optional. They can be used by the MGC to effectively include a notification request that is executed simultaneously with the creation of the connection. If one or more of these parameters is present, the RequestIdentifier MUST be one of them. Thus, the inclusion of a notification request can be recognized by the presence of a RequestIdentifier. The rest of the parameters may or may not be present. If one of the parameters is not present, it MUST be treated as if it was a normal NotificationRequest with the parameter in question being omitted. This may have the effect of cancelling signals and of stop looking for events.

As an example of use, consider an MGC that wants to place a call to an operator services system through an MF trunking gateway. The MGC could:

- ask the trunking gateway to create a connection, in order to be sure that the media gateway has resources for the call;
- ask the trunking gateway to seize an MF operator services trunk and initiate the call;
- ask the trunking gateway to notify the MGC when the call has been placed.

All of the above can be accomplished in a single CreateConnection command by including a notification request with the RequestedEvents parameters for the answer event and the SignalRequests parameter for the setup signal.

When these parameters are present, the creation of the connection and the notification request MUST be synchronized, which means that they are both either accepted or refused. In our example, the CreateConnection must be refused if the gateway does not have sufficient resources or cannot get adequate resources from the local network access. The call initiation notification request must be refused in the glare condition if the circuit is already seized. In this example, the call must not be placed if the connection cannot be established, and the connection must not be established if the circuit is already seized. An error would be returned instead (error code 401 - circuit already seized), which informs the MGC of the glare condition.

ReturnCode: is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

4.3.4 ModifyConnection

This command is used to modify the characteristics of a gateway's "view" of a connection. This "view" of the call includes both the local connection descriptor, as well as the remote connection descriptor.

```
ReturnCode
[, LocalConnectionDescriptor]
            ← ModifyConnection(CallId
                              , EndpointId
                              , ConnectionId
                              [, NotifiedEntity]
                              [, LocalConnectionOptions]
                              [, Mode]
                              [, RemoteConnectionDescriptor]
                              [, RequestedEvents]
                              [, RequestIdentifier]
                              [, SignalRequests]
                              [, QuarantineHandling]
                              [, DetectEvents])
```

The parameters used are the same as in the CreateConnection command, with the addition of a **ConnectionId** that uniquely identifies the connection within the endpoint. This parameter is returned by the CreateConnection command together with the local connection descriptor. It uniquely identifies the connection within the context of the endpoint.

The **EndpointId** MUST be a fully qualified endpoint name. The local name MUST NOT use the wildcard convention.

The ModifyConnection command can be used to affect connection parameters, subject to the same rules and constraints as specified for CreateConnection:

- Provide information on the other end of the connection through the **RemoteConnectionDescriptor**.
- Activate or deactivate the connection by changing the **mode** parameter's value. This can occur at any time during the connection, with arbitrary parameter values. An activation can, for example, be set to the "receive only" mode.
- Change the parameters of the connection through the **LocalConnectionOptions**, for example, by switching to a different coding scheme, changing the packetization period, or modifying the handling of echo cancellation.

The command will only return a **LocalConnectionDescriptor** if the local connection parameters, such as, e.g. RTP ports, etc. are modified. Thus, if, e.g. only the mode of the connection is changed, a LocalConnectionDescriptor will not be returned. If a connection parameter is omitted, e.g. mode or silence suppression, the old value of that parameter will be retained if possible. If a parameter change necessitates a change in one or more *unspecified* parameters, the gateway is free to choose suitable values for the unspecified parameters that must change (see note 1).

NOTE 1: This can for instance happen if a codec change is specified, and the old codec used silence suppression, but the new one does not support it. If, e.g. the packetization period furthermore was not specified, and the new codec supported the old packetization period, the value of this parameter would not change, as a change would not be necessary.

The RTP address information provided in the RemoteConnectionDescriptor specifies the remote RTP address of the receiver of media for the connection. This RTP address information may have been changed by the MGC (see note 2). When RTP address information is given to a trunking gateway for a connection, the trunking gateway SHOULD only accept media streams (and RTCP) from the RTP address specified as well. Any media streams received from any other addresses SHOULD be discarded. The TS 101 909-11 [3] should be consulted for additional security requirements.

NOTE 2: For instance if media needs to traverse a firewall.

The **RequestedEvents**, **RequestIdentifier**, **SignalRequests**, **QuarantineHandling**, and **DetectEvents** parameters are optional. The parameters can be used by the MGC to include a notification request that is tied to and executed simultaneously with the connection modification. If one or more of these parameters is supplied, then RequestIdentifier MUST be one of them.

When these parameters are present, the connection modification and the notification request MUST be synchronized, which means that they are both either accepted or refused.

ReturnCode: is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

4.3.5 DeleteConnection (from the Media Gateway Controller)

This command is used to terminate a connection. As a side effect, it collects statistics on the execution of the connection.

```
ReturnCode
, Connection-parameters
    ← DeleteConnection(CallId
                     , EndpointId
                     , ConnectionId
                     [, NotifiedEntity]
                     [, RequestedEvents]
                     [, RequestIdentifier]
                     [, SignalRequests]
                     [, QuarantineHandling]
                     [, DetectEvents])
```

The endpoint identifier, in this form of the DeleteConnection command, MUST be fully qualified. Wildcard conventions MUST NOT be used.

In the general case where a connection has two ends, this command has to be sent to both gateways involved in the connection. After the connection has been deleted, packet network media streams previously supported by the connection are no longer available. Any media packets received for the old connection are simply discarded and no new media packets for the stream are sent.

In response to the DeleteConnection command, the gateway returns a list of parameters that describe the status of the connection (see note). These parameters are:

- **Number of packets sent:** the total number of RTP data packets transmitted by the sender since starting transmission on the connection. The count is not reset if the sender changes its synchronization source identifier (SSRC, as defined in RTP), see example 1.

EXAMPLE 1: As a result of a Modify command: the value is zero if, e.g. the connection was always set in "receive only" mode.

- **Number of octets sent:** the total number of payload octets (i.e. not including header or padding) transmitted in RTP data packets by the sender since starting transmission on the connection. The count is not reset if the sender changes its SSRC identifier, see example 2.

EXAMPLE 2: As a result of a ModifyConnection command: the value is zero if, e.g. the connection was always set in "receive only" mode.

- **Number of packets received:** the total number of RTP data packets received by the sender since starting reception on the connection. The count includes packets received from different SSRC if the sender used several values. The value is zero if, e.g. the connection was always set in "send only" mode.

- **Number of octets received:** the total number of payload octets (i.e. not including header or padding) transmitted in RTP data packets by the sender since starting transmission on the connection. The count includes packets received from different SSRC if the sender used several values. The value is zero if, e.g. the connection was always set in "send only" mode.

- **Number of packets lost:** the total number of RTP data packets that have been lost since the beginning of reception. This number is defined to be the number of packets expected less the number of packets actually received, where the number of packets received includes any which are late or are duplicates. The count includes packets received from different SSRC if the sender used several values. Thus, packets that arrive late are not counted as lost, and the loss may be negative if there are duplicates. The count includes packets received from different SSRC if the sender used several values. The number of packets expected is defined to be the extended last sequence number received, less the initial sequence number received. The count includes packets received from different SSRC, if the sender used several values. The value is zero if, e.g. the connection was always set in "send only" mode.

- **Interarrival jitter:** an estimate of the statistical variance of the RTP data packet interarrival time measured in ms and expressed as an unsigned integer. The interarrival jitter "J" is defined to be the mean deviation (smoothed absolute value) of the difference "D" in packet spacing at the receiver compared to the sender for a pair of packets. Detailed computation algorithms are found in IETF/RFC 1889 (see bibliography). The count includes packets received from different SSRC if the sender used several values. The value is zero if, e.g. the connection was always set in "send only" mode.

- **Average transmission delay:** an estimate of the network latency, expressed in ms. This is the average value of the difference between the NTP timestamp indicated by the senders of the RTCP messages and the NTP timestamp of the receivers, measured when the messages are received. The average is obtained by summing all the estimates and then dividing by the number of RTCP messages that have been received. It should be noted that the correct calculation of this parameter relies on synchronized clocks. Trunking gateway devices MAY alternatively estimate the average transmission delay by dividing the measured roundtrip time by two.

NOTE: The values calculated will not include packets that resulted from Electronic Surveillance.

For a more detailed definition of these variables, please refer to IETF/RFC 1889 (see bibliography).

The **NotifiedEntity**, **RequestedEvents**, **RequestIdentifier**, **SignalRequests**, **QuarantineHandling**, and **DetectEvents** parameters are optional. They can be used by the MGC to transmit a notification request that is tied to and executed simultaneously with the deletion of the connection. However, if one or more of these parameters are present, RequestIdentifier MUST be one of them. For example, when a circuit is disconnected, the gateway might be instructed to delete the connection and to start looking for a seizure event. This can be accomplished in a single DeleteConnection command also by transmitting the RequestedEvents parameter for the seizure event and an empty SignalRequests parameter.

When these parameters are present, the delete connection and the notification request MUST be synchronized, which means that they are both either accepted or refused.

ReturnCode: is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

4.3.6 DeleteConnection (from the trunking gateway)

In some circumstances, a gateway may have to clear a connection, for example, because it has lost the resource associated with the connection. The gateway can terminate the connection by using a variant of the DeleteConnection command:

```
ReturnCode
    ← DeleteConnection(CallId,
                       EndpointId,
                       ConnectionId,
                       Reason-code,
                       Connection-parameters)
```

The **EndpointId**, in this form of the DeleteConnection command, MUST be fully qualified. Wildcard conventions MUST NOT be used.

The **Reason-code** is a text string starting with a numeric reason-code and optionally followed by a descriptive text string. A list of reason-codes can be found in clause 4.6.

In addition to the **CallId**, **EndpointId**, and **ConnectionId,** the trunking gateway will also send the connection's parameters, which would have been returned to the MGC in response to a DeleteConnection command from the MGC. The reason code indicates the cause of the DeleteConnection.

ReturnCode: is a parameter returned by the MGC. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

4.3.7 DeleteConnection (multiple connections from the Media Gateway Controller)

A variation of the DeleteConnection function can be used by the MGC to delete multiple connections at the same time. The command can be used to delete all connections that relate to a call for an endpoint:

```
ReturnCode
    ← DeleteConnection(CallId,
                       EndpointId)
```

The **EndpointId**, in this form of the DeleteConnection command, MUST NOT use the "any of" wildcard. All connections for the endpoint(s) with the CallId specified will be deleted. The command does not return any individual statistics or call parameters.

DeleteConnection can also be used by the MGC to delete all connections that terminate in a given endpoint:

```
ReturnCode
    ← DeleteConnection(EndpointId)
```

In this form of the DeleteConnection command, MGCs can take advantage of the hierarchical naming structure of endpoints to delete all the connections that belong to a group of endpoints. In this case, part of the "local endpoint name" component of the EndpointId can be specified using the "all" wildcarding convention, as specified in clause 4.1.1. The "any of" wildcarding convention MUST NOT be used. The command does not return any individual statistics or call parameters.

After the connection has been deleted, packet network media streams previously supported by the connection are no longer available. Any media packets received for the old connection are simply discarded and no new media packets for the stream are sent.

ReturnCode: is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

4.3.8 Auditing

The MGCP is based upon a centralized call control architecture where a MGC acts as the remote controller of client devices that provide voice communications interfaces to users and networks. In order to achieve the same or higher levels of availability as the current PSTN, some protocols have implemented mechanisms to periodically "ping" subscribers in order to minimize the time before an individual outage is detected. In this interest, an MGCP-specific auditing mechanism between the trunking gateways and the MGCs in an IPCablecom system is provided to allow the MGC to audit endpoint and connection state and to retrieve protocol-specific capabilities of an endpoint.

Two commands for auditing are defined for the trunking gateways:

- **AuditEndPoint:** Used by the MGC to determine the status of an endpoint;
- **AuditConnection:** Used by the MGC to obtain information about a connection.

Network management beyond the capabilities provided by these commands is generally desirable, e.g. information about the status of the trunking gateway as opposed to individual endpoints. Such capabilities are expected to be supported by the use of the Simple Network Management Protocol (SNMP) and by definition of a MIB for the trunking gateway, both of which are outside the scope of the present document.

4.3.8.1 AuditEndPoint

The AuditEndPoint command can be used by the MGC to find out the status of a given endpoint.

```
{ ReturnCode
    [, EndPointIdList]
    [, NumEndPoints] } |
{ ReturnCode
    [, RequestedEvents]
    [, SignalRequests]
    [, RequestIdentifier]
    [, NotifiedEntity]
    [, ConnectionIdentifiers]
    [, DetectEvents]
    [, ObservedEvents]
    [, EventStates]
    [, Capabilities] }
        ← AuditEndPoint(EndpointId
                    [, RequestedInfo] |
                    { [, SpecificEndpointID]
                        [, MaxEndpointIDs] } )
```

The **EndpointId** identifies the endpoint that is being audited. The "any of" wildcard convention MUST NOT be used.

The "all of" wildcard convention can be used to audit a group of endpoints. If this convention is used, the gateway MUST return the list of endpoint identifiers that match the wildcard in the **EndPointIdList** parameter, which is simply a list of SpecificEndpointIds - RequestedInfo MUST NOT be included in this case. **MaxEndPointIDs** is a numerical value that indicates the maximum number of EndpointIds to return. If additional endpoints exist, the **NumEndPoints** return parameter MUST be present and indicate the total number of endpoints that match the EndpointID specified. In order to retrieve the next block of EndpointIDs, the **SpecificEndPointID** is set to the value of the last endpoint returned in the previous EndpointIDList, and the command is issued.

When the wildcard convention is not used, the (possibly empty) **RequestedInfo** describes the information that is requested for the EndpointId specified - the SpecificEndpointID and MaxEndpointID parameters MUST NOT be used then. The following endpoint-specific information can then be audited with this command:

- RequestedEvents, SignalRequests, RequestIdentifier, NotifiedEntity, ConnectionIdentifiers, DetectEvents, ObservedEvents, EventStates, VersionSupported, and Capabilities.

The response will, in turn, include information about each of the items for which auditing information was requested:

- **RequestedEvents:** the current value of RequestedEvents the endpoint is using including the action associated with each event. Persistent events are included in the list.
- **SignalRequests:** a list of the Time-Out signals that are currently active, On/Off signals that are currently "on" for the endpoint (with or without parameter), and any pending Brief signals (see note 1). Time-Out signals that have timed-out, and currently playing Brief signals are not included. Parameterized signals are reported with the parameters they were applied with.
- **RequestIdentifier:** the RequestIdentifier for the last NotificationRequest received by the endpoint (includes notification request embedded in connection handling primitives). If no notification request has been received, the value zero will be returned.
- **NotifiedEntity:** the current "notified entity" for the endpoint.
- **ConnectionIdentifiers:** a comma-separated list of ConnectionIdentifiers for all connections that currently exist for the specified endpoint.
- **DetectEvents:** the current value of DetectEvents the endpoint is using. Persistent events are included in the list.
- **ObservedEvents:** the current list of observed events for the endpoint.
- **EventStates:** for events that have auditable states associated with them, the event corresponding to the state the endpoint is in, e.g. seizure if the MF trunk for the endpoint is currently seized. The definition of the individual events will state if the event in question has an auditable state associated with it.
- **VersionSupported:** a list of protocol versions supported by the endpoint.

- **Capabilities:** the capabilities for the endpoint similar to the LocalConnectionOptions parameter and including event packages and connection modes. If there is a need to specify that some parameters, such as e.g. silence suppression, are only compatible with some codecs, then the gateway will return several capability sets. If an endpoint is queried about a capability it does not understand, the endpoint MUST NOT generate an error; instead the parameter MUST be omitted from the response:
  - **Compression Algorithm:** a list of supported codecs. The rest of the parameters will apply to all codecs specified in this list.
  - **Packetization Period:** a single value or a range may be specified.
  - **Bandwidth:** a single value or a range corresponding to the range for packetization periods may be specified (assuming no silence suppression).
  - **Echo Cancellation:** whether echo cancellation is supported or not (see note 2).
  - **Silence Suppression:** whether silence suppression is supported or not.
  - **Type of Service:** whether type of service is supported or not.
  - **Event Packages:** a list of event packages supported. The first event package in the list will be the default package.
  - **Modes:** a list of supported connection modes.
  - **Security:** whether IPCablecom Security services are supported or not. If supported, the following parameters may be present as well:
    - **RTP Ciphersuites:** a list of authentication and encryption algorithms supported for RTP.
    - **RTCP Ciphersuites:** a list of authentication and encryption algorithms supported for RTCP.
    - **Electronic Surveillance:** whether IPCablecom Electronic Surveillance is supported or not.

NOTE 1: Currently, there should be no pending brief signals.

NOTE 2: Currently, all TGCP endpoints must support echo cancellation.

The MGC may then decide to use the AuditConnection command to obtain further information about the connections.

ReturnCode is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

If no info was requested and the EndpointId refers to a valid fully-specified EndpointId, the gateway simply returns a successful response (return code 200 - transaction executed normally).

It should be noted, that all of the information returned is merely a snapshot. New commands received, local activity, etc. may alter most of the above. For example the seizure state may change before the MGC receives the above information.

4.3.8.2 AuditConnection

Auditing of individual connections on an endpoint can be achieved using the AuditConnection command.

```
ReturnCode
[, CallId]
[, NotifiedEntity]
[, LocalConnectionOptions]
[, Mode]
[, RemoteConnectionDescriptor]
[, LocalConnectionDescriptor]
[, ConnectionParameters]
    ← AuditConnection(EndpointId
                    , ConnectionId
                    [, RequestedInfo])
```

The **EndpointId** identifies the endpoint that is being audited-wildcards MUST NOT be used. The (possibly empty) **RequestedInfo** describes the information that is requested for the **ConnectionId** within the EndpointId specified. The following connection info can be audited with this command:

- CallId, NotifiedEntity, LocalConnectionOptions, Mode, ConnectionParameters, RemoteConnectionDescriptor, LocalConnectionDescriptor.

The response will, in turn, include information about each of the items for which auditing info was requested:

- **CallId:** the CallId for the call to which the connection belongs.
- **NotifiedEntity:** the current "notified entity" for the endpoint.
- **LocalConnectionOptions:** the LocalConnectionOptions supplied for the connection.
- **Mode:** the current connection mode.
- **ConnectionParameters:** current connection parameters for the connection.
- **LocalConnectionDescriptor:** the LocalConnectionDescriptor that the gateway supplied for the connection.
- **RemoteConnectionDescriptor:** the RemoteConnectionDescriptor that was supplied to the gateway for the connection.

The **ReturnCode** is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

If no information was requested, and the EndpointId refers to a valid endpoint, the gateway simply checks that the connection specified exists and, if so, returns a positive response (return code 200 - transaction executed).

4.3.9 RestartinProgress

The RestartInProgress command is used by the gateway to signal that an endpoint, or a group of endpoints, is taken out of service or is being placed back in service.

```
ReturnCode
[, NotifiedEntity]
[, VersionSupported]
        ← RestartInProgress(EndpointId
                        , RestartMethod
                        [, RestartDelay])
```

The **EndpointId** identifies the endpoints that are taken in or out of service. The "all of" wildcard convention can be used to apply the command to a group of endpoints, for example, all endpoints that are attached to a specified interface, or even all endpoints that are attached to a given gateway. The "any of" wildcard convention MUST NOT be used.

The RestartMethod parameter specifies the type of restart:

- A "graceful" restart method indicates that the specified endpoint(s) will be taken out of service after the specified "restart delay". The established connections are not yet affected, but the MGC should refrain from establishing new connections, and should try to gracefully tear down any existing connections.
- A "forced" restart method indicates that the specified endpoints are taken out of service abruptly. The established connections, if any, are lost.
- A "restart" method indicates that service will be restored on the endpoints after the specified "restart delay". There are no connections that are currently established on the endpoints.
- A "disconnected" method indicates that the endpoint has become disconnected and is now trying to establish connectivity. The "restart delay" specifies the number of s the endpoint has been disconnected. Established connections are not affected.

The optional "restart delay" parameter is expressed as a number of s. If the number is absent, the delay value should be considered null. In the case of the "graceful" method, a null delay indicates that the MGC should simply wait for the natural termination of the existing connections, without establishing new connections. The restart delay is always considered null in the case of the "forced" method. A restart delay of null for the "restart" method indicates that service has already been restored. This typically will occur after gateway startup/reboot. To mitigate the effects of a gateway IP address change, the MGC MAY wish to resolve the gateway's domain name by querying the DNS regardless of the TTL of a current resource record for the restarted gateway.

Trunking gateways SHOULD send a "graceful" or "forced" RestartInProgress message as a courtesy to the MGC when they are taken out of service, e.g. by being shutdown, or taken out of service by a network management system, although the MGC cannot rely on always receiving such messages. Trunking gateways MUST send a "restart" RestartInProgress message with a null delay to their MGC when they are back in service according to the restart procedure specified in clause 4.4.3.5 - MGCs can rely on receiving this message. Also, trunking gateways MUST send a "disconnected" RestartInProgress message to their current "notified entity" according to the "disconnected" procedure specified in clause 4.4.3.6. The "restart delay" parameter MUST NOT be used with the "forced" restart method.

The RestartInProgress message will be sent to the current "notified entity" for the EndpointId in question. It is expected that a default MGC, i.e. "notified entity", has been provisioned for each endpoint so, after a reboot, the default MGC will be the "notified entity" for each endpoint. Trunking gateways MUST take full advantage of wild-carding to minimize the number of RestartInProgress messages generated when multiple endpoints in a gateway restart and the endpoints are managed by the same MGC.

ReturnCode: is a parameter returned by the MGC. It indicates the outcome of the command and consists of an integer number (see clause 4.5) optionally followed by commentary.

A **NotifiedEntity** may additionally be returned with the response from the MGC:

- If the response indicated success (return code 200 - transaction executed), the restart procedure has completed, and the NotifiedEntity returned is the new "notified entity" for the endpoint(s).
- If the response from the MGC indicated an error, the restart procedure is not yet complete, and must therefore be initiated again. If a NotifiedEntity parameter was returned, it then specifies the new "notified entity" for the endpoint(s), which must consequently be used when retrying the restart procedure.

Finally, a **VersionSupported** parameter with a list of supported versions may be returned if the response indicated version incompatibility (error code 528).

4.4 States, failover and race conditions

In order to implement proper call signalling, the MGC must keep track of the state of the endpoint, and the gateway must make sure that events are properly notified to the MGC. Special conditions may exist when the gateway or the MGC are restarted: the gateway may need to be redirected to a new MGC during "failover" procedures; Similarly, the MGC may need to take special action when the gateway is taken offline, or restarted.

4.4.1 Recaps and highlights

As mentioned in clause 4.1.4, MGCs are identified by their domain name, and each endpoint has one, and only one, "notified entity" associated with it at any given point in time. In this clause we recap and highlight the areas that are of special importance to reliability and fail-over in MGCP:

- An MGC is identified by its domain name, not its network addresses, and several network addresses can be associated with a domain name.
- An endpoint has one, and only one, MGC associated with it at any given point in time. The MGC associated with an endpoint is the current value of the "notified entity".

- The "notified entity" is initially set to a provisioned value. When commands with a NotifiedEntity parameter is received for the endpoint, including wild-carded endpoint-names, the "notified entity" is set to the value specified. If the "notified entity" for an endpoint is empty or has not been set explicitly (see note), the "notified entity" defaults to the source address of the last connection handling command or notification request received for the endpoint. In this case, the MGC will thus be identified by its network address, which SHOULD only be done on exceptional basis.

- Responses to commands are always sent to the source address of the command, regardless of the current "notified entity". When a Notify message needs to be piggy-backed with the response, the datagram is still sent to the source address of the new command received, regardless of the NotifiedEntity for any of the commands.

- When the "notified entity" refers to a domain name that resolves to multiple IP-addresses, endpoints are capable of switching between each of these addresses, however they cannot change the "notified entity" to another domain name on their own. An MGC can however instruct them to switch by providing them with a new "notified entity".

- If an MGC becomes unavailable, the endpoints managed by that MGC will eventually become "disconnected". The only way for these endpoints to become connected again is either for the failed MGC to become available again, or for another (backup) MGC to contact the affected endpoints with a new "notified entity".

- When another (backup) MGC has taken over control of a group of endpoints, it is assumed that the failed MGC will communicate and synchronize with the backup MGC in order to transfer control of the affected endpoints back to the original MGC, if so desired. Alternatively, the failed MGC could simply become the backup MGC now.

NOTE: This could for instance happen by specifying an empty NotifiedEntity parameter.

We should note that handover conflict resolution between separate MGCs is not provided - we are relying strictly on the MGCs knowing what they are doing and communicating with each other (although AuditEndpoint can be used to learn about the current "notified entity").

4.4.2 Retransmission, and detection of lost associations

The MGCP protocol is organized as a set of transactions, each of which is composed of a command and a response. The MGCP messages, being carried over UDP, may be subject to losses. In the absence of a timely response (see clause 5.5), commands are repeated. Gateways MUST keep in memory a list of the responses that they sent to recent transactions, and a list of the transactions that are currently being executed. Recent is here defined by the value Tt_{hist} that specifies the number of s that responses to old transactions must be kept for. The default value for Tt_{hist} is 30 s.

The transaction identifiers of incoming commands are first compared to the transaction identifiers of the recent responses. If a match is found, the gateway does not execute the transaction, but simply repeats the old response. If a match to a previously responded to transaction is not found, the transaction identifier of the incoming command is compared to the list of transactions that have not yet finished executing. If a match is found, the gateway does not execute the transaction, which is simply ignored - a response will be provided when the execution of the command is complete.

This repetition mechanism is used to guard against four types of possible errors:

- transmission errors, when, e.g. a packet is lost due to noise on a line or congestion in a queue;

- component failure, when, e.g. an interface for an MGC becomes unavailable;

- MGC failure, when, e.g. all interfaces for a MGC becomes unavailable;

- failover, when a new MGC is "taking over" transparently.

The elements should be able to derive from the past history an estimate of the packet loss rate. In a properly configured system, this loss rate should be very low, typically less than 1 % on average. If an MGC or a 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 uniformly distributed loss rate of 1 %, the probability that 5 consecutive transmission attempts fail is 1 in 100 billion, an event that should occur less than once every 10 days for an MGC that processes 1 000 transactions per second. (Indeed, the number of repetitions that is considered excessive should be a function of the prevailing packet loss rate.) When errors are non-uniformly distributed, the consecutive failure probability can become somewhat higher. We should note that the "suspicion threshold", which we will call "Max1", is normally lower than the "disconnection threshold", which we will call "Max2", and 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 re-transmitting 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 as follows:

- The gateway MUST always check for the presence of a new MGC. It can be noticed by:
  - receiving a command where the NotifiedEntity points to a new MGC; or
  - receiving a redirection response pointing to a new MGC.
- If a new MGC is detected, the gateway MUST direct retransmissions of any outstanding commands for the endpoint(s) redirected to that new MGC. Responses to new or old commands are still transmitted to the source address of the command.
- Prior to any retransmission, it is checked that the time elapsed since the sending of the initial datagram is no greater than Ts_{max}. If more than Ts_{max} time has elapsed, the endpoint becomes disconnected.
- If the number of retransmissions to this MGC equals "Max1", the gateway MAY actively query the name server in order to detect the possible change of MGC interfaces, regardless of the Time To Live (TTL) associated with the DNS record.
- The gateway may have learned several IP addresses for the MGC. If the number of retransmissions for this IP address is larger than "Max1" and lower than "Max2", and there are more IP addresses that have not been tried, then the gateway MUST direct the retransmissions to the remaining alternate addresses in its local list.
- If there are no more interfaces to try, and the number of retransmissions is Max2, then the gateway SHOULD contact the DNS one more time to see if any other interfaces have become available. If not, the endpoint(s) managed by this MGC are now disconnected. When an endpoint becomes disconnected, it MUST then initiate the "disconnected" procedure as specified in clause 4.4.3.6.

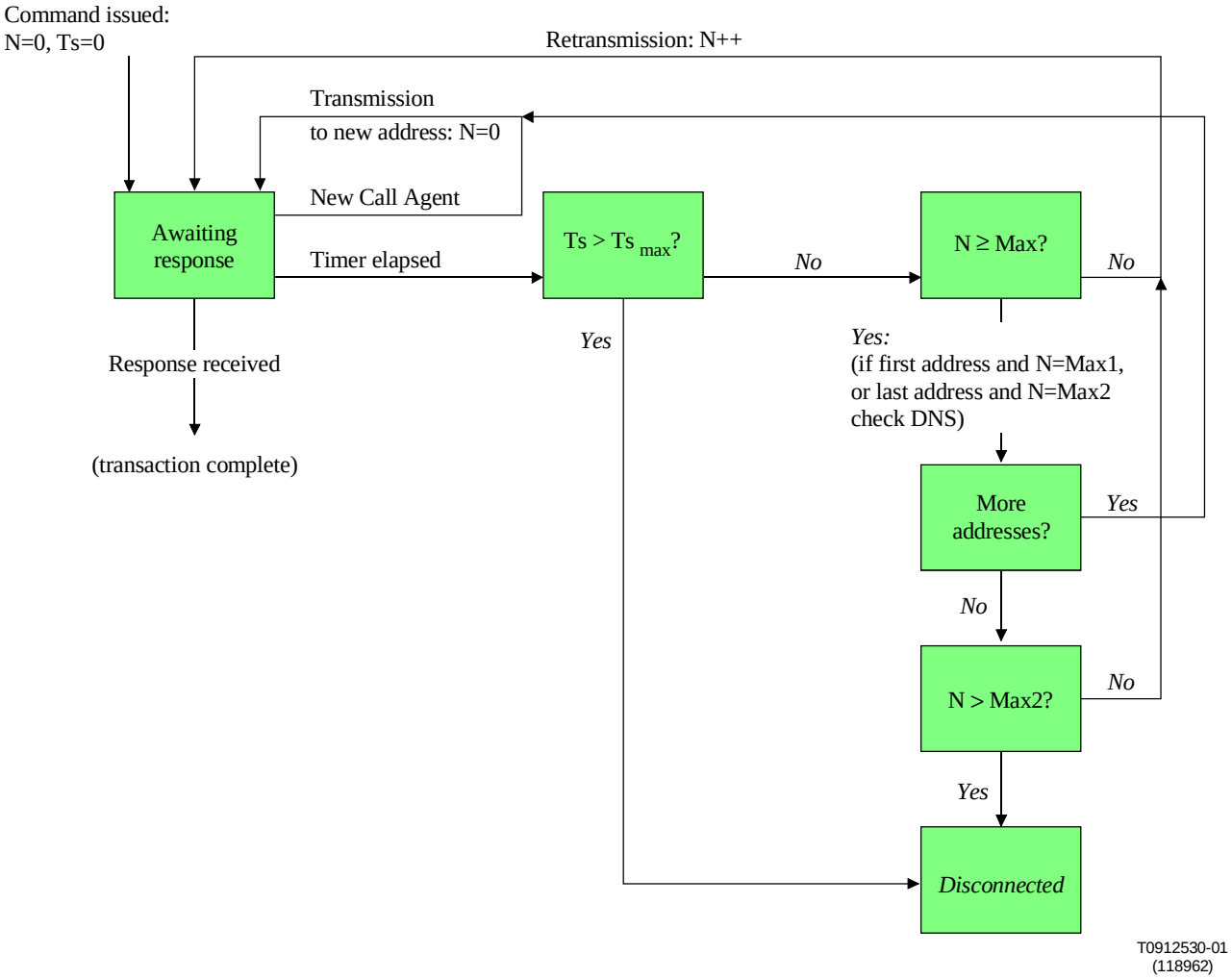


Figure 1

In order to adapt to network load automatically, MGCP specifies exponentially increasing timers (see clause 5.5.2). If the initial time-out is set to 200 ms, the loss of a fifth retransmission will be detected after about 6 s. This is probably an acceptable waiting delay to detect a failover. The retransmissions 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 s is probably acceptable.

It should be noted, that there is an intimate relationship between Ts_{max}, Tt_{hist}, and the maximum transit time, Tp_{max}. Specifically, the following relation MUST be satisfied to prevent retransmitted commands from being executed more than once:

- $Tt_{hist} \geq Ts_{max} + Tp_{max}$.

The default value for Ts_{max} is 20 s. Thus, if the assumed maximum propagation delay is 10 s, then responses to old transactions must be kept for a period of at least 30 s. The importance of having the sender and receiver agree on these values cannot be overstated.

The default value for Max1 is 5 retransmissions and the default value for Max2 is 7 retransmissions. Both of these values may be altered by the provisioning process.

Furthermore, the provisioning process MUST be able to disable one or both of the Max1 and Max2 DNS queries.

4.4.3 Race conditions

In this clause we describe how MGCP deals with race conditions.

First of all, MGCP deals with race conditions through the notion of a "quarantine list" that quarantines events and through explicit detection of desynchronization, e.g. for mismatched seizure-state due to glare for an endpoint.

Secondly, MGCP does not assume that the transport mechanism will maintain the order of commands and responses. This may cause race conditions that may be obviated through a proper behaviour of the MGC by a proper ordering of commands.

Finally, in some cases, many gateways may decide to restart operation at the same time. This may occur, for example, if an area loses power or transmission capability during an earthquake or an ice storm. When power and transmission capability are re-established, many gateways may decide to send RestartInProgress commands simultaneously, which could lead to very unstable operation if not carefully controlled.

4.4.3.1 Quarantine list

MGCP controlled gateways will receive notification requests that ask them to watch for a list of events. The protocol elements that determine the handling of these events are the "Requested Events" list, and the "Detect Events" list.

When the endpoint is initialized, the requested events list only consists of persistent events for the endpoint. After reception of a command, the gateway starts observing the endpoint for occurrences of the events mentioned in the list, including persistent events.

The events are examined as they occur. The action that follows is determined by the "action" parameter associated to the event in the list of requested events. The events that are defined as "accumulate" are accumulated in a list of observed events. This will go on until one event is encountered that triggers a Notify command which will be sent to the "notified entity".

The gateway, at this point, will transmit the Notify command and will place the endpoint in a "notification state". As long as the endpoint is in this "notification state", the events that are detected on the endpoint are stored in a "quarantine" buffer for later processing. The events are, in a sense, "quarantined". The events detected are the events specified by the union of the RequestedEvents parameter and the most recently received DetectEvents parameter or, in case no DetectEvents parameter has been received, the events that are referred to in the RequestedEvents. Persistent events are detected as well.

The endpoint exits the "notification state" when the response to the Notify command is received (see note 1). The Notify command may be retransmitted in the "notification state", as specified in clause 4.4.2.

NOTE 1: It should be noted, that the Notify action cannot be combined with an Embedded NotificationRequest.

When the endpoint exits the "notification state" it resets the list of observed events of the endpoint to a null value.

The TGCP profile mandates the use of "lockstep mode", which implies that the gateway MUST receive a new NotificationRequest command after it has sent a Notify command. Until this happens, the endpoint is in a "lockstep state", and events that occur and are to be detected are simply stored in the quarantine buffer. The events to be quarantined are the same as in the "notification state". Once the new NotificationRequest is received and executed successfully, the endpoint exits the "lockstep state".

A gateway can receive at any time a new NotificationRequest command for the endpoint which will also have the effect of taking the endpoint out of the "notification state" assuming the NotificationRequest executes successfully.

When a new NotificationRequest is received in the "notification state", the gateway shall ensure that the pending Notify is received by the MGC prior to a successful response to the new NotificationRequest. It does so by using the "piggy-backing" functionality of the protocol and placing the messages (commands and responses) to be sent in order with the oldest message first. The messages will then be sent in a single packet to the source of the new NotificationRequest, regardless of the source and "notified entity" for the old and new command. The steps involved are the following:

1) the gateway builds a message that carries in a single packet a repetition of the old outstanding Notify command and the response to the new NotificationRequest command;

2) the endpoint is then taken out of the "notification state" without waiting for the response to the Notify command;

3) a copy of the outstanding Notify command is kept until a response is received. If a time-out occurs, the Notify will be repeated, in a packet that will also carry a repetition of the response to the NotificationRequest:

   - if the packet carrying the response to the NotificationRequest is lost, the MGC will retransmit the NotificationRequest. The gateway will reply to this repetition by retransmitting in a single packet the outstanding Notify command and the response to the NotificationRequest - this datagram will be sent to the source of the NotificationRequest;

   - if the gateway has to transmit a new Notify before a response to the previous Notify is received, it constructs a packet that piggy-backs a repetition of the old Notify, a repetition of the response to the last NotificationRequest, and the new Notify - this datagram will be sent to current "notified entity".

After receiving a NotificationRequest command, the "requested events" list is replaced by the newly received parameters, and the list of "observed events" is reset to a null value. The subsequent behaviour is then conditioned by the value of the QuarantineHandling parameter. The parameter may specify that quarantined events are to be discarded, in which case all quarantined events are discarded. If the parameter specifies that the quarantined events should be processed, the gateway will start processing the list of quarantined events, using the newly received list of "requested events". When processing these events, the gateway may encounter an event, which triggers a Notify command to be sent. If that is the case, the gateway will immediately transmit a Notify command that will report all events that were accumulated in the list of "observed events" up until and including the triggering event, leaving the unprocessed events in the quarantine buffer. The endpoint then enters the "notification state" again.

The above procedure applies to all forms of notification requests, regardless of whether they are part of a connection handling command or provided as a NotificationRequest command. Connection handling commands that do not include a notification request are neither affected by nor do they affect the above procedure.

Figure 2 illustrates the procedure specified above assuming all transactions execute successfully.

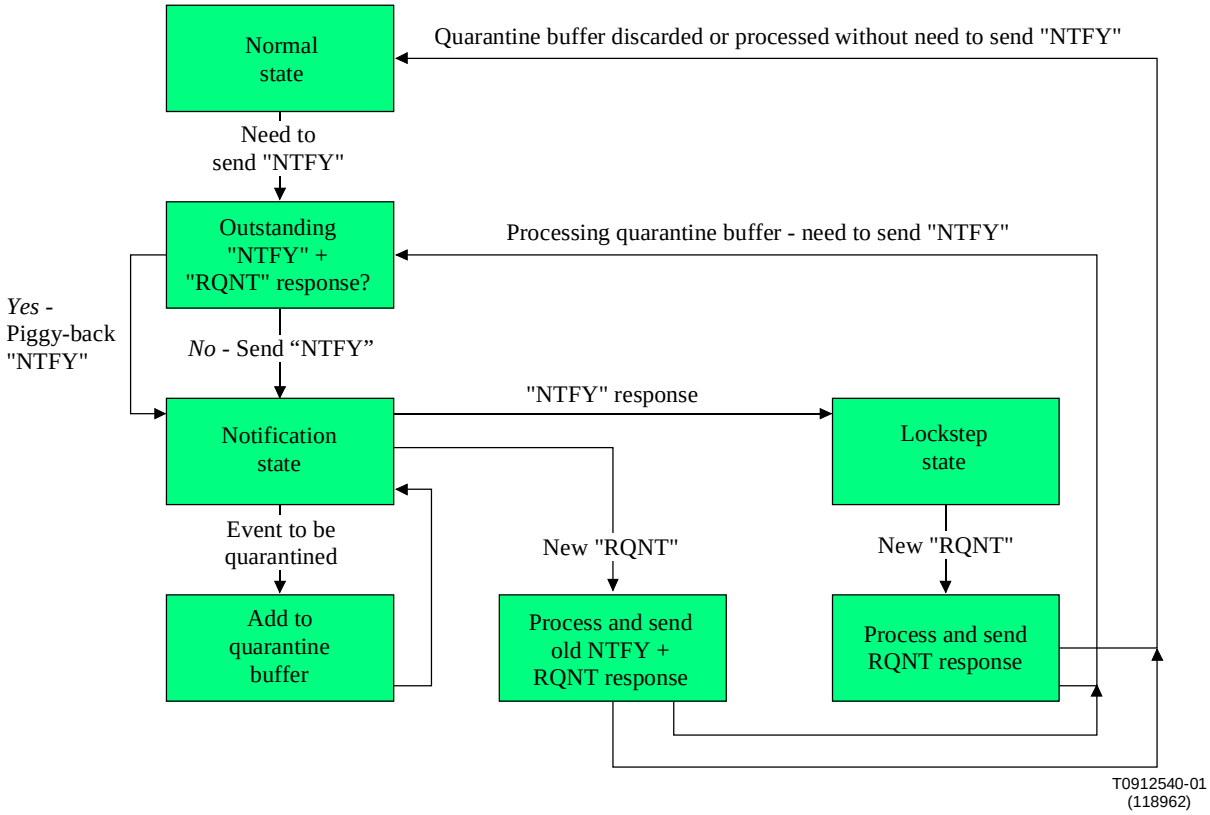


Figure 2

MGCs SHOULD provide the response to a successful Notify message and the new NotificationRequest in the same datagram using the piggy-backing mechanism (see note 2).

NOTE 2: Vendors that choose not to follow this Recommendation should examine Media Gateway Controller failure scenarios carefully.

4.4.3.2 Explicit detection

A key element of the state of several endpoints is the seizure state of a circuit. Although seizure-state changing events are persistent in TGCP, race conditions and state mismatch may still occur, for example when a circuit is seized while the MGC is in the process of requesting the gateway to look for seizure (the "glare" condition well known in telephony - this is however primarily an issue for two-way CAS trunks, which are not supported in the present document).

To avoid this race condition, the gateway MUST check the condition of the endpoint before responding to a NotificationRequest. Specifically, it MUST return an error:

1) if the gateway is requested to notify a "seizure" (see note 1) transition while the circuit is already seized (error code 401 - circuit seized);

2) if the gateway is requested to notify an "unseize" (see note 2) condition while the circuit is not seized (error code 402 - circuit not seized).

NOTE 1: For instance by requesting the "sup" event on an MF Terminating BLV/OI trunk with a call already in progress.

NOTE 2: For instance by requesting the "rel" event on an MF Operator Services trunk without any call in progress.

Additionally, individual signal definitions can specify that a signal will only operate under certain conditions, e.g. MF operator ringback may only be possible if the circuit is already seized. If such prerequisites exist for a given signal, the gateway MUST return the error specified in the signal definition if the prerequisite is not met.

It should be noted, that the condition check is performed at the time the notification request is received, where as the actual event that caused the current condition may have either been reported, or ignored earlier, or it may currently be quarantined.

The other state variables of the gateway, such as the list of requested events or list of requested signals, are entirely replaced after each successful NotificationRequest, which prevents any long term discrepancy between the MGC and the gateway.

When a NotificationRequest is unsuccessful, whether it is included in a connection-handling command or not, the gateway will simply continue as if the command had never been received. although an error is returned. As all other transactions, the NotificationRequest MUST operate as an atomic transaction, Thus any changes initiated as a result of the command MUST be reverted.

Another race condition can occur when a Notify is issued shortly before the reception by the gateway of a NotificationRequest. The RequestIdentifier is used to correlate Notify commands with NotificationRequest commands thereby enabling the MGC to determine if the Notify command was generated before or after the gateway received the new NotificationRequest.

4.4.3.3 Transactional semantics

As the potential transaction completion times increases, e.g. due to external resource reservations, a careful definition of the transactional semantics becomes increasingly important. In particular the issue of race conditions, specifically as it relates to seizure-state must be defined carefully.

An important point to consider is, that the seizure-state may in fact change between the time a transaction is initiated and the time it completes. More generally, we may say that the successful completion of a transaction depends on one or more pre-conditions where one or more of the pre-conditions may change dynamically during the execution of the transaction.

The simplest semantics for this is simply to require that all pre-conditions MUST be met from the time the transaction is initiated until the transaction completes. Thus, if any of the preconditions change during the execution of the transaction, the transaction MUST fail. Furthermore, as soon as the transaction is initiated, all new events are quarantined. When the outcome of the transaction is known, all quarantined events are then processed.

As an example, consider a transaction that includes a request for the "seizure" event. When the transaction is initiated the circuit is "not seized" and this pre-condition is therefore met. If the seizure-state changes to "seized" before the transaction completes, the pre-condition is no longer met, and the transaction therefore immediately fails. The "seizure" event will now be stored in the "quarantine" buffer which then gets processed.

4.4.3.4 Ordering of commands, and treatment of disorder

MGCP does not mandate that the underlying transport protocol guarantees the sequencing of commands sent to a gateway or an endpoint. This property tends to maximize the timeliness of actions, but it has a few drawbacks. For example:

- Notify commands may be delayed and arrive to the MGC after the transmission of a new Notification Request command.
- If a new NotificationRequest is transmitted before a response to a previous one is received, there is no guarantee that the previous one will not be received in second position.

MGCs and gateways that want to guarantee consistent operation of the endpoints can use the rules specified:

1) When a gateway handles several endpoints, commands pertaining to the different endpoints can be sent in parallel, for example following a model where each endpoint is controlled by its own process or its own thread.
2) When several connections are created on the same endpoint, commands pertaining to different connections can be sent in parallel.
3) On a given connection, there should normally be only one outstanding command (create or modify). However, a DeleteConnection command can be issued at any time. In consequence, a gateway may sometimes receive a ModifyConnection command that applies to a previously deleted connection. Such commands MUST be ignored, and an error returned (error code 515 - incorrect connection-id).
4) On a given endpoint, there should normally be only one outstanding NotificationRequest command at any time. The RequestId parameter is used to correlate Notify commands with the triggering NotificationRequest.
5) In some cases, an implicitly or explicitly wild-carded DeleteConnection command that applies to a group of endpoints can step in front of a pending CreateConnection command. The MGC should individually delete all connections whose completion was pending at the time of the global DeleteConnection command. Also, new CreateConnection commands for endpoints named by the wild-carding should not be sent until a response to the wild-carded DeleteConnection command is received.
6) When commands are embedded within each other, sequencing requirements for all commands MUST be adhered to. For example a CreateConnection command with a notification request in it must adhere to the sequencing requirements for CreateConnection and NotificationRequest at the same time.
7) AuditEndpoint and AuditConnection is not subject to any sequencing.
8) RestartInProgress must always be the first command sent by an endpoint as defined by the restart procedure (see clause 4.4.3.5). Any other command or response must be delivered after this RestartInProgress command (piggy-backing allowed).
9) When multiple messages are piggy-backed in a single packet, the messages are always processed in order.

Those of the above rules that specify gateway behaviour MUST be adhered to by trunking gateways, however the trunking gateway MUST NOT make any assumptions as to whether MGCs follow the rules or not. Consequently gateways MUST always respond to commands, regardless of whether they adhere to the above rules or not.

4.4.3.5 Fighting the restart avalanche

Let's suppose that a large number of gateways are powered on simultaneously. If they were to all initiate a RestartInProgress transaction, the MGC 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 behaviour MUST be followed:

1) When a gateway is powered on, it initiates a restart timer to a random value, uniformly distributed between 0 and a provisionable maximum waiting delay (MWD), e.g. 360 s. Care MUST be taken to avoid synchronicity of the random number generation between multiple gateways that would use the same algorithm.
2) The gateway then waits for either the end of this timer, the reception of a command from the MGC, or the detection of a local circuit activity, such as for example an seizure transition on a trunking gateway. A pre-existing seizure condition results in the generation of a seizure event.

3) When the restart timer elapses, when a command is received, or when an activity or pre-existing seizure condition is detected, the gateway initiates the restart procedure.

The restart procedure simply states that the endpoint MUST send a RestartInProgress command to the MGC informing it about the restart and furthermore guarantee that the first message (command or response) that the MGC sees from this endpoint MUST be this RestartInProgress command. The endpoint MUST take full advantage of piggy-backing in achieving this. For example, if a circuit seizure activity occurs prior to the restart timer expiring, a packet containing the RestartInProgress command, and with a piggy-backed Notify command for the seizure event will be generated. In the case where the restart timer expires without any other activity, the gateway simply sends a RestartInProgress message.

Should the gateway enter the "disconnected" state while carrying out the restart procedure, the disconnected procedure specified in clause 4.4.3.6 MUST be carried out, except that a "restart" rather than "disconnected" message is sent during the procedure.

It is expected that each endpoint in a gateway will have a provisionable MGC, i.e. "notified entity", to direct the initial restart message towards. When the collection of endpoints in a gateway is managed by more than one MGC, the above procedure must be performed for each collection of endpoints managed by a given MGC. The gateway MUST take full advantage of wild-carding to minimize the number of RestartInProgress messages generated when multiple endpoints in a gateway restart and the endpoints are managed by the same MGC.

The value of MWD is a configuration parameter that depends on the type of the gateway. The following reasoning can be used to determine the value of this delay on a gateway.

MGCs are typically dimensioned to handle the peak hour traffic load, during which, on average, 60 % of the trunks will be busy serving calls whose average duration is typically 3 minutes. The processing of a call typically involves 5 to 6 transactions between each endpoint and the MGC. This simple calculation shows that the MGC is expected to handle 5 to 6 transactions for each endpoint, every 5 minutes on average, or, to put it otherwise, about one transaction per endpoint per minute. This suggests that a reasonable value of MWD would be 2 min per endpoint. When the value of MWD is set for the gateway, the value should be inversely proportional to the number of endpoints that are being restarted. For example MWD should be set to 5 s for a gateway that handles a T1 line, or to 180 ms for a gateway that handles a T3 line.

4.4.3.6 Disconnected endpoints

In addition to the restart procedure, trunking gateways also have a "disconnected" procedure, which is initiated when an endpoint becomes "disconnected" as described in clause 4.4.2. It should here be noted, that endpoints can only become disconnected when they attempt to communicate with the MGC. The following steps are followed by an endpoint that becomes "disconnected":

1) A "disconnected" timer is initialized to a random value, uniformly distributed between 0 and a provisionable "disconnected" initial waiting delay (Td_{init}), e.g. 15 s. Care MUST be taken to avoid synchronicity of the random number generation between multiple gateways and endpoints that would use the same algorithm.

2) The gateway then waits for either the end of this timer, the reception of a command from the MGC, or the detection of a local circuit activity for the endpoint, such as for example a seizure transition.

3) When the "disconnected" timer elapses, when a command is received, or when a local circuit activity is detected, the gateway initiates the "disconnected" procedure for the endpoint. In the case of local user activity, a provisionable "disconnected" minimum waiting delay (Td_{min}) must furthermore have elapsed since the gateway became disconnected or the last time it initiated the "disconnected" procedure in order to limit the rate at which the procedure is performed.

4) If the "disconnected" procedure still left the endpoint disconnected, the "disconnected" timer is then doubled, subject to a provisionable "disconnected" maximum waiting delay (Td_{max}), e.g. 600 s, and the gateway proceeds with step 2 again.

The "disconnected" procedure is similar to the restart procedure in that it now simply states that the endpoint MUST send a RestartInProgress command to the MGC informing it that the endpoint was disconnected and furthermore guarantee that the first message (command or response) that the MGC now sees from this endpoint MUST be this RestartInProgress command. The endpoint MUST take full advantage of piggy-backing in achieving this. The MGC may then for instance decide to audit the endpoint, or simply clear all connections for the endpoint.

The present document purposely does not specify any additional behaviour for a disconnected endpoint. Vendors MAY for instance choose to provide silence, play reorder tone, or even enable a downloaded wav file to be played on affected endpoints.

The default value for Td_{init} is 15 s, the default value for Td_{min}, is 15 s, and the default value for Td_{max} is 600 s.

4.5 Return codes and error codes

All MGCP commands receive a response. The response carries a return code that indicates the status of the command. The return code is an integer number, for which three value ranges have been defined:

- value 000 indicates a response acknowledgement (see note);
- values between 100 and 199 indicate a provisional response;
- values between 200 and 299 indicate a successful completion;
- values between 400 and 499 indicate a transient error;
- values between 500 and 599 indicate a permanent error.

NOTE: Response acknowledgement is used for provisional responses (see clause 5.8).

The values that have been defined are listed in the table 2.

Table 2

Code	Meaning
000	Response acknowledgement.
100	The transaction is currently being executed. An actual completion message will follow later.
200	The requested transaction was executed normally.
250	The connection(s) was deleted.
400	The transaction could not be executed, due to a transient error.
401	The phone is already off hook or circuit already seized
402	The phone is already on hook or circuit not seized.
500	The transaction could not be executed because the endpoint is unknown.
501	The transaction could not be executed because the endpoint is not ready.
502	The transaction could not be executed because the endpoint does not have sufficient resources.
510	The transaction could not be executed because a protocol error was detected.
511	The transaction could not be executed because the command contained an unrecognized extension.
512	The transaction could not be executed because the gateway is not equipped to detect one of the requested events.
513	The transaction could not be executed because the gateway is not equipped to generate one of the requested signals.
514	The transaction could not be executed because the gateway cannot send the specified announcement.
515	The transaction refers to an incorrect connection-id (may have been already deleted).
516	The transaction refers to an unknown call-id.
517	Unsupported or invalid mode.
518	Unsupported or unknown package.
519	Endpoint does not have a digit map.
520	The transaction could not be executed because the endpoint is "restarting".
521	Endpoint redirected to another MGC.
522	No such event or signal.
523	Unknown action or illegal combination of actions.
524	Internal inconsistency in LocalConnectionOptions.
525	Unknown extension in LocalConnectionOptions.
526	Insufficient bandwidth.
527	Missing RemoteConnectionDescriptor.
528	Incompatible protocol version.
529	Internal hardware failure.
532	Unsupported value(s) in LocalConnectionOptions.
533	Response too big.

4.6 Reason-codes

Reason-codes are used by the gateway when deleting a connection to inform the MGC about the reason for deleting the connection. The reason code is an integer number and the following values have been defined, see table 3.

Table 3

Code	Meaning
900	Endpoint malfunctioning
901	Endpoint taken out of service
902	Loss of lower layer connectivity (e.g. downstream sync)

5 Media Gateway Control Protocol

The MGCP implements the media gateway control interface as a set of transactions. The transactions are composed of a command and a mandatory response. There are eight types of commands:

- CreateConnection.
- ModifyConnection.
- DeleteConnection.
- NotificationRequest.
- Notify.
- AuditEndpoint.
- AuditConnection.
- RestartInProgress.

The first four commands are sent by the MGC to a gateway. The Notify command is sent by the gateway to the MGC. The gateway can also send a DeleteConnection as defined in clause 4.3.6. The MGC can send either of the Audit commands to the gateway and, finally, the gateway can send a RestartInProgress command to the MGC.

5.1 General description

All commands are composed of a Command header, which for some commands may be followed by a session description.

All responses are composed of a Response header, which for some commands may be followed by a session description.

Headers and session descriptions are encoded as a set of text lines, separated by a carriage return and line feed character (or, optionally, a single line-feed character). The headers are separated from the session description by an empty line.

MGCP uses a transaction identifier with a value between 1 and 999 999 999 to correlate commands and responses. The transaction identifier is encoded as a component of the command header and is repeated as a component of the response header.

5.2 Command header

The command header is composed of:

- a command line identifying the requested action or verb, the transaction identifier, the endpoint towards which the action is requested, and the MGCP protocol version;
- a set of parameter lines composed of a parameter name followed by a parameter value.

Unless otherwise noted or dictated by other referenced standards, each component in the command header is case insensitive. This goes for verbs as well as parameters and values, and all comparisons MUST treat upper and lower case as well as combinations of these as being equal.

5.2.1 Command line

The command line is composed of:

- the name of the requested verb;
- the identification of the transaction;
- the name of the endpoint(s) that should execute the command (in notifications or restarts, the name of the endpoint(s) that is issuing the command);
- the protocol version.

These four items are encoded as strings of printable ASCII characters separated by white spaces, i.e. the ASCII space (0x20) or tabulation (0x09) characters. Trunking gateways SHOULD use exactly one ASCII space separator, however they MUST be able to parse messages with additional white space characters.

5.2.1.1 Requested verb coding

Requested verbs are encoded as four letter upper- and/or lower-case ASCII codes (comparisons MUST be case insensitive) as defined in table 4.

Table 4

Verb	Code
CreateConnection	CRCX
ModifyConnection	MDCX
DeleteConnection	DLCX
NotificationRequest	RQNT
Notify	NTFY
AuditEndpoint	AUEP
AuditConnection	AUCX
RestartInProgress	RSIP

New verbs may be defined in future versions of the protocol. It may be necessary, for experimental purposes, to use new verbs before they are sanctioned in a published version of this protocol. Experimental verbs should be identified by a four-letter code starting with the letter X (e.g. XPER).

A gateway that receives a command with an experimental verb it does not support MUST return an error (error code 511 - unrecognized extension).

5.2.1.2 Transaction identifiers

Transaction identifiers are used to correlate commands and responses.

A trunking gateway supports two separate transaction identifier name spaces:

- a transaction identifier name space for sending transactions; and
- a transaction identifier name space for receiving transactions.

At a minimum, transaction identifiers for commands sent to a given trunking gateway MUST be unique for the maximum lifetime of the transactions within the collection of MGCs that control that trunking gateway (see clause 5.5). Thus, regardless of the sending MGC, trunking gateways can always detect duplicate transactions by simply examining the transaction identifier. The coordination of these transaction identifiers between MGCs is outside the scope of the present document though.

Transaction identifiers for all commands sent from a given trunking gateway MUST be unique for the maximum lifetime of the transactions (see clause 5.5) regardless of which MGC the command is sent to. Thus, an MGC can always detect a duplicate transaction from a trunking gateway by the combination of the domain-name of the endpoint and the transaction identifier. The gateway in turn can always detect a duplicate response acknowledgement by looking at the transaction id(s).

The transaction identifier is encoded as a string of up to nine decimal digits. In the command lines, it immediately follows the coding of the verb.

Transaction identifiers have values between 1 and 999 999 999. An MGCP entity MUST NOT reuse a transaction identifier more quickly than three minutes after completion of the previous command in which the identifier was used.

5.2.1.3 Endpoint, Media Gateway Controller and NotifiedEntity name coding

The endpoint names and MGC names are encoded as e-mail addresses, as defined in IETF/RFC 821 (see bibliography). In these addresses, the domain name identifies the system where the endpoint is attached, while the left side identifies a specific endpoint on that system. Both components MUST be case insensitive.

Examples of such names are:

`ds/ds1-3/2@TGCP2.whatever.net`	Second circuit on the third DS1 in the trunking gateway TGCP2 in the "Whatever" network.
`MGC@mgc.whatever.net`	Media Gateway Controller for the "whatever" network.

The name of notified entities is expressed with the same syntax, with the possible addition of a port number, as in:

- `MGC@mgc.whatever.net:5234`.

In case the port number is omitted, the default MGCP port (2427) will be used. Additional detail on endpoint names can be found in clause 4.1.1.

5.2.1.4 Protocol version coding

The protocol version is coded as the keyword "MGCP" followed by a white space and the version number, which again is followed by the profile name "TGCP" and a profile version number. The version numbers are composed of a major version number, a dot, and a minor version number. The major and minor version numbers are coded as decimal numbers. The profile version number defined by the present document is 1.0.

The protocol version for the present document MUST be encoded as:

- `MGCP 1.0 TGCP 1.0`.

The "TGCP 1.0" portion signals that this is the TGCP 1.0 profile of MGCP 1.0.

An entity that receives a command with a protocol version it does not support, MUST respond with an error (error code 528 - incompatible protocol version).

5.2.2 Parameter lines

Parameter lines are composed of a parameter name, which in most cases is composed of a single upper-case character, followed by a colon, a white space, and the parameter value. Parameter names and values are still case-insensitive though. The parameters that can be present in commands are defined in table 5.

Table 5

Parameter name	Code	Parameter value
ResponseAck (see note)	K	See description.
CallId	C	Hexadecimal string, at most 32 characters.
ConnectionId	I	Hexadecimal string, at most 32 characters.
NotifiedEntity	N	An identifier, in IETF/RFC 821 format, composed of an arbitrary string and of the domain name of the requesting entity, possibly completed by a port number, as in: Call-agent@ca.whatever.net:5234
RequestIdentifier	X	See description.
LocalConnectionOptions	L	See description.
Connection Mode	M	See description.
RequestedEvents	R	See description.
SignalRequests	S	See description.
ObservedEvents	O	See description.
ConnectionParameters	P	See description.
ReasonCode	E	See description.
SpecificEndPointId	Z	An identifier, in IETF/RFC 821 format, composed of an arbitrary string, optionally followed by an "@" followed by the domain name of the trunking gateway to which this endpoint is attached.
MaxEndPointIds	ZM	Decimal string, at most 16 characters.
NumEndPoints	ZN	Decimal string, at most 16 characters.
RequestedInfo	F	See description.
QuarantineHandling	Q	See description.
DetectEvents	T	See description.
EventStates	ES	See description.
RestartMethod	RM	See description.
RestartDelay	RD	A number of s encoded as a decimal number.
Capabilities	A	See description.
VersionSupported	VS	See description.
NOTE: The ResponseAck parameter was not shown in clause 4.3 as transaction identifiers are not visible in our example API. Implementers may choose a different approach.		

The parameters are not necessarily present in all commands. Table 6 provides the association between parameters and commands. The letter M stands for mandatory, O for optional, and F for forbidden.

Table 6

Parameter name	CRCX	MDCX	DLCX	RQNT	NTFY	AUEP	AUCX	RSIP
ResponseAck (see note 1)	O	O	O	O	O	O	O	O
CallId	M	M	O	F	F	F	F	F
ConnectionId	F	M	O	F	F	F	M	F
RequestIdentifier	O	O	O	M	M	F	F	F
LocalConnectionOptions	M	O	F	F	F	F	F	F
Connection Mode	M	O	F	F	F	F	F	F
RequestedEvents	O (see note 2)	O (see note 2)	O (see note 2)	O (see note 2)	F	F	F	F
SignalRequests	O (see note 2)	O (see note 2)	O (see note 2)	O (see note 2)	F	F	F	F
NotifiedEntity	O	O	O	O	O	F	F	F
ReasonCode	F	F	O	F	F	F	F	F
ObservedEvents	F	F	F	F	M	F	F	F
Connection parameters	F	F	O	F	F	F	F	F
Specific Endpoint Id	F	F	F	F	F	O	F	F
MaxEndPointIds	F	F	F	F	F	O	F	F
NumEndPoints	F	F	F	F	F	F	F	F
RequestedInfo	F	F	F	F	F	O	O	F
QuarantineHandling	O	O	O	O	F	F	F	F
DetectEvents	O	O	O	O	F	F	F	F
EventStates	F	F	F	F	F	F	F	F
RestartMethod	F	F	F	F	F	F	F	M
RestartDelay	F	F	F	F	F	F	F	O
Capabilities	F	F	F	F	F	F	F	F
VersionSupported	F	F	F	F	F	F	F	F
RemoteConnectionDescriptor	O	O	F	F	F	F	F	F

NOTE 1: The ResponseAck parameter was not shown in clause 4.3 as transaction identifiers are not visible in our example API. Implementers may choose a different approach.

NOTE 2: The RequestedEvents and SignalRequests parameters are optional in the NotificationRequest. If these parameters are omitted, the corresponding lists will be considered empty. For the connection handling commands, this applies as well when a RequestIdentifier is included.

Trunking gateways and MGCs SHOULD always provide mandatory parameters before optional ones, however trunking gateways MUST NOT fail if the present document is not followed.

If implementers need to experiment with new parameters, for example when developing a new MGCP application, they should identify these parameters by names that begin with the string "X-" or "X+", such as for example:

- `X-FlowerOfTheDay: Daisy`.

Parameter names that start with "X+" are mandatory parameter extensions. A gateway that receives a mandatory parameter extension that it cannot understand MUST respond with an error (error code 511 - unrecognized extension).

Parameter names that start with "X-" are non critical parameter extensions. A gateway that receives a non critical parameter extension that it cannot understand can safely ignore that parameter.

It should be noted that experimental verbs are of the form X*ABC*, whereas experimental parameters are of the form X-ABC.

If a parameter line is received with a forbidden parameter, or any other formatting error, the receiving entity should respond with the most specific error code for the error in question. The least specific error code is 510 - protocol error. Commentary text can always be provided.

5.2.2.1 Response acknowledgement

The response acknowledgement parameter is used to support the three-way handshake described in clause 5.7. It contains a comma separated list of "confirmed transaction-id ranges".

NOTE: The ResponseAck parameter was not shown in clause 4.3 as transaction identifiers are not visible in our example API. Implementers may choose a different approach.

Each "confirmed transaction-id range" is composed of either one decimal number, when the range includes exactly one transaction, or two decimal numbers separated by a single hyphen, describing the lower and higher transaction identifiers included in the range.

An example of a response acknowledgement is:

```
- K: 6234-6255, 6257, 19030-19044.
```

5.2.2.2 RequestIdentifier

The request identifier correlates a Notify command with the NotificationRequest that triggered it. A RequestIdentifier is a hexadecimal string, at most 32 characters. The string "0" is reserved for reporting of persistent events in the case where no NotificationRequest has been received yet (see clause 4.3.2).

5.2.2.3 Local connection options

The local connection options describe the operational parameters that the MGCs instructs the gateway to use for a connection. These parameters are:

- The packetization period in ms, encoded as the keyword "p" followed by a colon and a decimal number.
- The literal name of the compression algorithm, encoded as the keyword "a" followed by a colon and a character string.
- The echo-cancellation parameter, encoded as the keyword "e" followed by a colon and the value "on" or "off".
- The type of service parameter, encoded as the keyword "t" followed by a colon and the value encoded as two hexadecimal digits.
- The silence suppression parameter, encoded as the keyword "s" followed by a colon and the value "on" or "off".

The LocalConnectionOptions parameters used for Security are encoded as follows:

- The secret is encoded as the keyword "sc-st" followed by a colon, a method, a colon, and the actual secret. The method is either the string "clear" if the secret is encoded in clear-text, or the string "base64" if the secret is encoded using base64.
- The RTP ciphersuite is encoded as the keyword "sc-rtp" followed by a colon and an RTP ciphersuite string as defined below. A list of values may be specified in which case the values will be separated by a semicolon.
- The RTCP ciphersuite is encoded as the keyword "sc-rtcp" followed by a colon and an RTCP ciphersuite string as defined below. A list of values may be specified in which case the values will be separated by a semicolon.

The RTP and RTCP ciphersuite strings follow the grammar:

- ciphersuite = [AuthenticationAlgorithm] "/" [EncryptionAlgorithm];
- AuthenticationAlgorithm = 1*(ALPHA/DIGIT/"-"/"_");
- EncryptionAlgorithm = 1*(ALPHA/DIGIT | "-"/"_");

where ALPHA, and DIGIT are defined in IETF/RFC 2234 (see bibliography). Whitespaces are not allowed within a ciphersuite. The following example illustrates the use of ciphersuite:

```
- 62/51.
```

The actual list of IPCablecom supported ciphersuites is provided in the TS 101 909-11 [3].

When several parameters are present, the values are separated by a comma. It is considered an error to include a parameter without a value (error code 524 - LocalConnectionOptions inconsistency).

Examples of local connection options are:

- `L: p:10, a:PCMU`;
- `L: p:10, a:PCMU, e:off, t:20, s:on`;
- `L: p:30, a:G729A, e:on, t:A0, s:off`.

The type of service hex value "20" implies an IP precedence of 1, and a type of service hex value of "A0" implies an IP precedence of 5.

This set of attributes may be extended by extension attributes. Extension attributes are composed of an attribute name, followed by a colon, and a semicolon separated list of attribute values. The attribute name MUST start with the two characters "x+", for a mandatory extension, or "x-", for a non mandatory extension. If a gateway receives a mandatory extension attribute that it does not recognize, it MUST reject the command with an error (error code 525 - Unknown extension in LocalConnectionOptions).

The LocalConnectionOptions parameters used for Electronic Surveillance are:

- The Call Content connection Identifier encoded as the keyword "es-cci" followed by a colon and a string of up to 8 hex characters corresponding to a 32 bit identifier for the Call Content connection Identifier.
- The Call Content Destination encoded as the keyword "es-ccd" followed by a colon and an IP-address encoded similarly to an IP-address for the domain name portion of an endpoint name. The IP-address is followed by a colon and up to 5 decimal characters for a UDP port number to use.

5.2.2.4 Capabilities

Capabilities inform the MGC about its capabilities when audited. The encoding of capabilities is based on the Local Connection Options encoding for the parameters that are common to both. In addition, capabilities can also contain a list of supported packages, and a list of supported modes.

The parameters used are:

- The packetization period in ms, encoded as the keyword "p" followed by a colon and a decimal number. A range may be specified as two decimal numbers separated by a hyphen.
- The literal name of the compression algorithm, encoded as the keyword "a" followed by a colon and a character string. A list of values may be specified in which case the values will be separated by a semicolon.
- The bandwidth in kilobits per second (1 kbits/s), encoded as the keyword "b" followed by a colon and a decimal number. A range may be specified as two decimal numbers separated by a hyphen.
- The echo-cancellation parameter, encoded as the keyword "e" followed by a colon and the value "on" if echo cancellation is supported, "off" otherwise.
- The type of service parameter, encoded as the keyword "t" followed by a colon and the value "0" if type of service is not supported, all other values indicated support for type of service.
- The silence suppression parameter, encoded as the keyword "s" followed by a colon and the value "on" if silence suppression is supported, "off" otherwise.
- The event packages supported by this endpoint encoded as the keyword "v" followed by a colon and then a semicolon-separated list of package names supported. The first value specified will be the default package for the endpoint.
- The connection modes supported by this endpoint encoded as the keyword "m" followed by a colon and a semicolon-separated list of connection modes supported as defined in clause 5.2.2.7.

- The keyword "sc-st" if IPCablecom Security is supported. In that case, the following keywords indicate the ciphersuites supported:
  - the keyword "sc-rtp" followed by a colon and a semi-colon separated list of RTP AuthenticationAlgorithms, a slash, and a semi-colon separated list of EncryptionAlgorithms supported;
  - the keyword "sc-rtcp" followed by a colon and a semi-colon separated list of RTCP AuthenticationAlgorithms, a slash, and a semi-colon separated list of EncryptionAlgorithms supported.

When several parameters are present, the values are separated by a comma.

Examples of capabilities are:

```
A:   a:PCMU;G729A, p:10-100, e:on, s:off, v:IT,
     m:sendonly;recvonly;sendrecv;inactive

A:   a:G729A; p:30-90, e:on, s:on, v:MT,
     m:sendonly;recvonly;sendrecv;inactive,
     sc-st, sc-rtp: 00/51;03
```

NOTE: The codecs and security algorithms are merely examples - separate IPCablecom Specifications detail the actual codecs and algorithms supported, as well as the encoding used (see TS 101 909-11 [3], TS 101 909-3 [4], TS 101 909-4 [2]).

- The keyword "es-cci" if IPCablecom Electronic Surveillance is supported.

5.2.2.5 Connection parameters

Connection parameters are encoded as a string of type and value pairs, where the type is a two-letter identifier of the parameter, and the value a decimal integer. Types are separated from values by an "=" sign. Parameters are separated from each other by a comma.

The connection parameter types are specified in table 7.

Table 7

Connection Parameter Name	Code	Connection Parameter Value
Packets sent	PS	The number of packets that were sent on the connection.
Octets sent	OS	The number of octets that were sent on the connection.
Packets received	PR	The number of packets that were received on the connection.
Octets received	OR	The number of octets that were received on the connection.
Packets lost	PL	The number of packets that were not received on the connection, as deduced from gaps in the sequence number.
Jitter	JI	The average inter-packet arrival jitter, in ms, expressed as an integer number.
Latency	LA	Average latency, in ms, expressed as an integer number.

Extension connection parameters names are composed of the string "X-" followed by a two letters extension parameter name. MGCs that receive unrecognized extensions MUST silently ignore these extensions.

An example of a connection parameter encoding is:

```
P: PS=1245, OS=62345, PR=0, OR=0, PL=0, JI=0, LA=48
```

5.2.2.6 Reason codes

Reason codes are three-digit numeric values. The reason code is optionally followed by a white space and commentary, e.g.:

```
900 Endpoint malfunctioning
```

A list of reason-codes can be found in clause 4.6.

5.2.2.7 Connection mode

The connection mode describes the connection's operation mode. The possible values are, see table 8.

Table 8

Mode	Meaning
M: sendonly	The gateway should only send packets.
M: recvonly	The gateway should only receive packets.
M: sendrecv	The gateway should send and receive packets.
M: inactive	The gateway should neither send nor receive packets.
M: loopback	The gateway should place the endpoint in Loopback mode.
M: conttest	The gateway should place the endpoint in Continuity Test mode.
M: netwloop	The gateway should place the endpoint in Network Loopback mode.
M: netwtest	The gateway should place the endpoint in Network Continuity Test mode.

5.2.2.8 Event/signal name coding

Event/signal names are composed of an optional package name, separated by a slash (/) from the name of the actual event. The event name can optionally be followed by an at sign (@) and the identifier of a connection on which the event should be observed. Event names are used in the RequestedEvents, SignalRequests, DetectEvents, ObservedEvents, and EventStates parameters. Each event is identified by an event code. These ASCII encodings are not case sensitive. Values such as "co", "Co", "CO" or "cO" should be considered equal.

The following are valid examples of event names:

`IT/co1`	Originating continuity test in the ISUP trunk package.
`co1`	Originating continuity test in the ISUP trunk package, assuming that the ISUP trunk package is the default package for the endpoint.
`IT/rt@0A3F58`	Ringback on connection "`0A3F58`"

In addition, the wildcard notation of events can be used, instead of individual names, in the RequestedEvents and DetectEvents (but not SignalRequests ObservedEvents, or EventStates):

`IT/all`	All events in the ISUP trunk package.

Finally, the star sign can be used to denote "all connections", and the dollar sign can be used to denote the "current" connection. The following are valid examples of such notations:

`IT/ma@*`	The RTP media start event on all connections for the endpoint.
`IT/rt@$`	Ringback on the current connection.

An initial set of event packages for trunking gateways can be found in clause 7.

5.2.2.9 RequestedEvents

The RequestedEvents parameter provides the list of events that have been requested. The currently defined event codes are described in annex A. Each event can be qualified by a requested action, or by a list of actions. Not all actions can be combined - please refer to clause 4.3.1 for valid combinations. The actions, when specified, are encoded as a list of keywords enclosed in parenthesis and separated by commas. The codes for the various actions are, see table 9.

Table 9

Action	Code
Notify immediately	N
Accumulate	A
Ignore	I
Keep Signal(s) active	K
Embedded NotificationRequest	E
Embedded ModifyConnection	C

When no action is specified, the default action is to notify the event. This means that, for example, "ft" and "ft(N)" are equivalent. Events that are not listed are discarded, except for persistent events.

The requested events list is encoded on a single line, with event/action groups separated by commas. An example of a RequestedEvents encodings is:

`R: oc(N), of(N)` Notify operation complete, notify operation failure.

The embedded NotificationRequest follows the format:

`E ( R( <RequestedEvents>), S( <SignalRequests>) )`

with each of R, and S being optional and possibly supplied in another order.

The embedded ModifyConnection action follows the format:

`C(M(<ConnectionMode`$_1$`>( <ConnectionID`$_1$`> )) , … ,`
`M(<ConnectionMode`$_n$`>(ConnectionID`$_n$` )))`

The following example illustrates the use of Embedded ModifyConnection:

`R: ma@23B34D(A, C(M(sendrecv($)))), oc(N), of(N)`

On media start on connection "23B34D" change the connection mode of the "current connection" to "send receive". Notify events on "operation complete" and "operation failure".

5.2.2.10 SignalRequests

The SignalRequests parameter provides the name of the signals that have been requested. The currently defined signals can be found in Bibliography. A given signal can only appear once in the list, and all signals will, by definition, be applied at the same time.

Some signals can be qualified by signal parameters. When a signal is qualified by multiple signal parameters, the signal parameters are separated by commas. Each signal parameter MUST follow the format specified below (white spaces allowed):

signal-parameter = signal-parameter-value/
signal-parameter-name "="signal-parameter-value/
signal-parameter-name "(" signal-parameter-list ")"

signal-parameter-list = signal-parameter-value 0*("," signal-parameter-value)

where signal-parameter-value may be either a string or a quoted string, i.e. a string surrounded by two double quotes. Two consecutive double-quotes in a quoted string will escape a double-quote within that quoted string. For example, `"ab""c"` will produce the string `ab"c.`

Each signal has one of the following signal-types associated with it (see clause 4.3.1):

- On/Off (OO);
- Time-out (TO);
- Brief (BR).

On/Off signals can be parameterized with a "+" to turn the signal on, or a "-" to turn the signal off. If an on/off signal is not parameterized, the signal is turned on. Both of the following will turn the "mysignal" signal on:

`mysignal(+), mysignal`

Time-out signals can be parameterized with the signal parameter "TO" and a time-out value that overrides the default time-out value. If a time-out signal is not parameterized with a time-out value the default time-out value will be used. Both of the following will apply the ringback tone signal for 6 s:

```
rt(to=6000)
rt(to(6000))
```

Individual signals may define additional signal parameters.

The signal parameters will be enclosed within parenthesis, as in the following hypothetical example:

```
S: display(10/14/17/26, "555 1212", CableLabs)
```

When several signals are requested, their codes are separated by a comma, as in:

```
S: signal1, signal2
```

5.2.2.11 ObservedEvents

The observed events parameters provide the list of events that have been observed. The event codes are the same as those used in the NotificationRequest.When an event is detected on a connection, the observed event will identify the connection the event was detected on using the "@<connection>" syntax. Examples of observed events are:

```
O: ma@A43B81
O: ft
O: IT/ft
O: IT/ft, IT/mt
```

5.2.2.12 RequestedInfo

The RequestedInfo parameter contains a comma separated list of parameter codes, as defined in the "Parameter lines" clause - clause 4.3.8 lists the parameters that can be audited. The following values are supported as well.

Table 10

RequestedInfo Parameter	Code
LocalConnectionDescriptor	LC
RemoteConnectionDescriptor	RC

For example, if one wants to audit the value of the NotifiedEntity, RequestIdentifier, RequestedEvents, SignalRequests, DetectEvents, EventStates, LocalConnectionDescriptor, and RemoteConnectionDescriptor parameters, the value of the RequestedInfo parameter will be:

```
F: N, X, R, S, T, ES, LC, RC
```

The capabilities request, for the AuditEndPoint command, is encoded by the parameter code "A", as in:

```
F: A
```

5.2.2.13 QuarantineHandling

The QuarantineHandling parameter contains the keyword "process" or "discard" to indicate the treatment of quarantined events, e.g.:

```
Q: process
```

5.2.2.14 DetectEvents

The DetectEvents parameter is encoded as a comma separated list of events, such as for example:

```
T: ft, mt
```

It should be noted, that no actions can be associated with the events.

5.2.2.15 EventStates

The EventStates parameter is encoded as a comma separated list of events, such as for example:

```
ES: MO/rlc
```

It should be noted, that no actions can be associated with the events.

5.2.2.16 RestartMethod

The RestartMethod parameter is encoded as one of the keywords "graceful", "forced", "restart", or "disconnected", as for example:

```
RM: restart
```

5.2.2.17 VersionSupported

The VersionSupported parameter is encoded as a comma separated list of versions supported, such as for example:

```
VS: MGCP 1.0, MGCP 1.0 TGCP 1.0
```

5.3 Response header formats

The response header is composed of a response line optionally followed by headers that encode the response parameters.

The response line starts with the response code, which is a three-digit numeric value. The code is followed by a white space, the transaction identifier, and optional commentary preceded by a white space, e.g.:

```
200 1201 OK
```

Table 11 summarizes the response parameters whose presence is mandatory or optional in a response header, as a function of the command that triggered the response assuming the command succeeded. The reader should still study the individual command definitions though as this table only provides summary information. The letter M stands for mandatory, O for optional and F for forbidden.

Table 11

Parameter name	CRCX	MDCX	DLCX	RQNT	NTFY	AUEP	AUCX	RSIP
ResponseAck (note 1)	O (see note 2)	O (see note 2)	O (see note 2)	O (see note 2)	O (see note 2)	O (see note 2)	O (see note 2)	O (see note 2)
CallId	F	F	F	F	F	F	O	F
ConnectionId	M	F	F	F	F	O	F	F
RequestIdentifier	F	F	F	F	F	O	F	F
LocalConnectionOptions	F	F	F	F	F	O	O	F
Connection Mode	F	F	F	F	F	F	O	F
RequestedEvents	F	F	F	F	F	O	F	F
SignalRequests	F	F	F	F	F	O	F	F
NotifiedEntity	F	F	F	F	F	O	O	O
ReasonCode	F	F	F	F	F	F	F	F
ObservedEvents	F	F	F	F	F	O	F	F
ConnectionParameters	F	F	O	F	F	F	O	F
Specific Endpoint ID	O	F	F	F	F	O	F	F
MaxEndPointIds	F	F	F	F	F	F	F	F
NumEndPoints	F	F	F	F	F	O	F	F
RequestedInfo	F	F	F	F	F	F	F	F
QuarantineHandling	F	F	F	F	F	F	F	F
DetectEvents	F	F	F	F	F	O	F	F
EventStates	F	F	F	F	F	O	F	F
RestartMethod	F	F	F	F	F	F	F	F
RestartDelay	F	F	F	F	F	F	F	F
Capabilities	F	F	F	F	F	O	F	F
VersionSupported	F	F	F	F	F	O	F	O
LocalConnection Descriptor	M	O	F	F	F	F	O	F
RemoteConnection Descriptor	F	F	F	F	F	F	O	F

NOTE 1: The ResponseAck parameter was not shown in clause 4.3 as transaction identifiers are not visible in our example API. Implementers may choose a different approach.

NOTE 2: The ResponseAck parameter MUST NOT be used with any other responses than a final response issued after a provisional response for the transaction in question. In that case, the presence of the ResponseAck parameter MUST trigger a Response Acknowledgement message - any ResponseAck values provided will be ignored.

The response parameters are described for each of the commands in the following.

5.3.1 CreateConnection

In the case of a CreateConnection message, the response line is followed by a Connection-Id parameter with a successful response (code 200). A LocalConnectionDescriptor is furthermore transmitted with a positive response. The LocalConnectionDescriptor is encoded as a "session description", as defined in clause 5.4. It is separated from the response header by an empty line, e.g.:

```
200 1204 OK
I: FDE234C8

v=0
o=- 25678 753849 IN IP4 128.96.41.1
s=-
c=IN IP4 128.96.41.1
t=0 0
m=audio 3456 RTP/AVP 96
a=rtpmap:96 G726-32/8000
```

When a provisional response has been issued previously, the final response may furthermore contain the Response Acknowledgement parameter, as in:

```
200 1204 OK
K:
I: FDE234C8

v=0
o=- 25678 753849 IN IP4 128.96.41.1
s=-
c=IN IP4 128.96.41.1
t=0 0
m=audio 3456 RTP/AVP 96
a=rtpmap:96 G726-32/8000
```

The final response is acknowledged by a Response Acknowledgement:

```
000 1204
```

5.3.2 ModifyConnection

In the case of a successful ModifyConnection message, the response line is followed by a LocalConnectionDescriptor, if the modification resulted in a modification of the session parameters (e.g. changing only the mode of a connection does not alter the session parameters). The LocalConnectionDescriptor is encoded as a "session description", as defined in clause 5.4. It is separated from the response header by an empty line.

```
200 1207 OK

v=0
o=- 25678 753849 IN IP4 128.96.41.1
s=-
c=IN IP4 128.96.41.1
t=0 0
m=audio 3456 RTP/AVP 0
```

When a provisional response has been issued previously, the final response may furthermore contain the Response Acknowledgement parameter as in:

```
526 1207 No bandwidth
K:
```

The final response is acknowledged by a Response Acknowledgement:

```
000 1207 OK
```

5.3.3 DeleteConnection

Depending on the variant of the DeleteConnection message, the response line may be followed by a Connection Parameters parameter line, as defined in clause 5.2.2.5.

```
250 1210 OK
P: PS=1245, OS=62345, PR=780, OR=45123, PL=10, JI=27,
                              LA=48
```

5.3.4 NotificationRequest

A NotificationRequest response does not include any additional response parameters.

5.3.5 Notify

A Notify response does not include any additional response parameters.

5.3.6 AuditEndpoint

In the case of an AuditEndPoint the response line may be followed by information for each of the parameters requested - each parameter will appear on a separate line. Parameters for which no value currently exists will still be provided. Each local endpoint name "expanded" by a wildcard character will appear on a separate line using the "SpecificEndPointId" parameter code, e.g.:

```
200 1200 OK
Z: ds/ds1-1/1@tgw.whatever.net
Z: ds/ds1-1/2@tgw.whatever.net
ZN: 24
```

or:

```
200 1200 OK
A: a:PCMU;G728, p:10-100, e:on, s:off, t:1, v:IT,
   m:sendonly;recvonly;sendrecv;inactive
A: a:G729A; p:30-90, e:on, s:on, t:1, v:MT,
   m:sendonly;recvonly;sendrecv;inactive
```

5.3.7 AuditConnection

In the case of an AuditConnection, the response may be followed by information for each of the parameters requested. Parameters for which no value currently exists will still be provided. Connection descriptors will always appear last and each will be preceded by an empty line, as for example:

```
200 1203 OK
C: A3C47F21456789F0
N: [128.96.41.12]
L: p:10, a:PCMU;G728
M: sendrecv
P: PS=622, OS=31172, PR=390, OR=22561, PL=5, JI=29, LA=50

v=0
o=- 4723891 7428910 IN IP4 128.96.63.25
s=-
c=IN IP4 128.96.63.25
t=0 0
m=audio 1296 RTP/AVP 96
a=rtpmap:96 G726-32/8000
```

If both a local and a remote connection descriptor are provided, the local connection descriptor will be the first of the two. If a connection descriptor is requested, but it does not exist for the connection audited, that connection descriptor will appear with the SDP protocol version field only.

5.3.8 RestartInProgress

The response to a RestartInProgress may include the name of another MGC to contact, for instance when the MGC redirects the endpoint to another MGC as in:

```
521 1204 Redirect
N: MGC-1@whatever.net
```

5.4 Session description encoding

The session description is encoded in conformance with the Session Description Protocol (SDP), however, trunking gateways may make certain simplifying assumptions about the session description as specified in the following. It should be noted, that session descriptions are case sensitive per IETF/RFC 2327 [1].

SDP usage depends on the type of session, as specified in the "media" parameter. Currently, the TGCP specification only support media of type "audio".

5.4.1 SDP audio service use

In a gateway devoted to voice-grade communications, we only have to describe sessions that use exactly one media, audio. The parameters of SDP that are relevant for the voice-grade audio application are specified below. Trunking gateway MUST support session descriptions that conform to these rules and in the following order:

1) the SDP profile presented below;

2) IETF/RFC 2327 [1].

The SDP profile provided describes the use of the session description protocol in TGCP. The general description and explanation of the individual parameters can be found in IETF/RFC 2327 [1], however below we detail what values TGCP endpoints need to provide for these fields (send) and what TGCP endpoints should do with values supplied or not supplied for these fields (receive).

Any parameter not specified below SHOULD NOT be provided by any TGCP endpoint, and if such a parameter is received, it SHOULD be ignored.

5.4.1.1 Protocol Version (v=)

v= <version>
v= 0

Send: MUST be provided in accordance with IETF/RFC 2327 [1] (i.e. v=0).

Receive: MUST be provided in accordance with IETF/RFC 2327 [1].

5.4.1.2 Origin (o=)

The origin field consists (o=) of 6 sub-fields in IETF/RFC 2327 [1]:

o= <username> <session-ID> <version> <network-type> <address-type> <address>
o= - 2987933615 2987933615 IN IP4 A3C47F2146789F0

Username:

Send: Hyphen MUST be used as username when privacy is requested. Hyphen SHOULD be used otherwise (see note).

Receive: This field SHOULD be ignored.

Session-ID:

Send: MUST be in accordance with IETF/RFC 2327 [1] for interoperability with non-IPCablecom clients.

Receive: This field SHOULD be ignored.

Version:

Send: In accordance with IETF/RFC 2327 [1].

Receive: This field SHOULD be ignored.

Network Type:

Send: Type "IN" MUST be used.

Receive: This field SHOULD be ignored.

Address Type:

Send: Type "IP4" MUST be used

Receive: This field SHOULD be ignored.

Address:

Send: MUST be in accordance with IETF/RFC 2327 [1] for interoperability with non-IPCablecom clients.

Receive: This field MUST be ignored.

NOTE: Since TGCP endpoints do not know when privacy is requested, they SHOULD always use a hyphen.

5.4.1.3 Session name (s=)

s= <session-name>
s= -

Send: Hyphen MUST be used as Session name.

Receive: This field MUST be ignored.

5.4.1.4 Session and media information (i=)

i= <session-description>

Send: For TGCP, the field MUST NOT be used.

Receive: This field MUST be ignored.

5.4.1.5 URI (u=)

u= <URI>

Send: For TGCP, the field MUST NOT be used.

Receive: This field MUST be ignored.

5.4.1.6 E-mail address and phone number (e=, p=)

e= <e-mail-address>
p= <phone-number>

Send: For TGCP, the field MUST NOT be used.

Receive: This field MUST be ignored.

5.4.1.7 Connection data (c=)

The connection data consists of 3 sub-fields:

c= <network-type> <address-type> <connection-address>

c= IN IP4 10.10.111.11

Network Type:

Send: Type "IN" MUST be used.

Receive: Type "IN" MUST be present.

Address Type:

Send: Type "IP4" MUST be used

Receive: Type "IP4" MUST be present.

Connection Address:

Send: This field MUST be filled with a unicast IP address at which the application will receive the media stream. Thus a TTL value MUST NOT be present and a "number of addresses" value MUST NOT be present. The field MUST NOT be filled with a fully-qualified domain name instead of an IP address. A non-zero address specifies both the send and receive address for the media stream(s) it covers.

Receive: A unicast IP address or a fully qualified domain name MUST be present. A non-zero address specifies both the send and receive address for the media stream(s) it covers.

5.4.1.8 Bandwidth (b=)

b= <modifier>: <bandwidth-value>
b= AS : 64

Send: Bandwidth information is optional in SDP but it SHOULD always be included (see note 1). When an rtpmap or a non well-known codec (see note 2) is used, the bandwidth information MUST be used.

Receive: Bandwidth information SHOULD be included. If a bandwidth modifier is not included, the receiver MUST assume reasonable default bandwidth values for well-known codecs.

Modifier:

Send: Type "AS" MUST be used.

Receive: Type "AS" MUST be present.

Bandwidth Value:

Send: The field MUST be filled with the Maximum Bandwidth requirement of the Media stream in kilobits per second.

Receive: The maximum bandwidth requirement of the media stream in kilobits per second MUST be present.

NOTE 1: If this field is not used, the Gate Controller might not authorize the appropriate bandwidth.

NOTE 2: A non well-known codec is a codec not defined in the codec IPCablecom Specification TS 101 909-3 [4].

5.4.1.9 Time, repeat times and time zones (t=, r=, z=)

t= <start-time> <stop-time>
t= 36124033 0
r= <repeat-interval> <active-duration> <list-of-offsets-from-start-time>
z= <adjustment-time> <offset>

Send: Time MUST be present; start time MAY be zero, but SHOULD be the current time, and stop time SHOULD be zero. Repeat Times, and Time Zones SHOULD NOT be used, if they are used it should be in accordance with IETF/RFC 2327 [1].

Receive: If any of these fields are present, they SHOULD be ignored.

5.4.1.10 Encryption keys

k= <method>
k= <method>: <encryption-keys>

Security services for IPCablecom are defined by the TS 101 909-11 [3]. The security services specified for RTP and RTCP do not comply with those of IETF/RFC 1889, IETF/RFC 1890 (see bibliography), and IETF/RFC 2327 [1]. In the interest of interoperability with non-IPCablecom devices, the "k" parameter will therefore not be used to convey security parameters.

Send: MUST NOT be used.

Receive: This field SHOULD be ignored.

5.4.1.11 Attributes (a=)

a= <attribute>: <value>
a= rtpmap: <payload type> <encoding name>/<clock rate>
[/<encoding parameters>]
a= rtpmap : 0 PCMU / 8000
a= X-pc-codecs: <alternative 1> <alternative 2> …
a= X-pc-secret: <method>:<encryption key>
a = X-pc-csuites-rtp: <alternative 1> <alternative 2> …
a = X-pc-csuites-rtcp: <alternative 1> <alternative 2> …
a = X-pc-spi-rtcp: <value>
a = X-pc-bridge: <number-ports>

a= <attribute>
a= recvonly
a= sendrecv
a= sendonly
a= ptime

Send: One or more of the "a" attribute lines specified below MAY be included. An attribute line not specified below SHOULD NOT be used.

Receive: One or more of the "a" attribute lines specified below MAY be included and MUST be acted upon accordingly. "a" attribute lines not specified below may be present but MUST be ignored.

rtpmap:

Send: When used, the field MUST be used in accordance with IETF/RFC 2327 [1]. It MAY be used for well-known as well as well as non well-known codecs. The encoding names used are provided in a separate IPCablecom Specification (see TS 101 909-3 [4] and TS 101 909-11 [3]).

Receive: The field MUST be used in accordance with IETF/RFC 2327 [1].

X-pc-codecs:

Send: The field contains a list of alternative codecs that the endpoint is capable of using for this connection. The list is ordered by decreasing degree of preference, i.e. the most preferred alternative codec is the first one in the list. A codec is encoded similarly to "encoding name" in rtpmap.

Receive: Conveys a list of codecs that the remote endpoint is capable of using for this connection. The codecs MUST NOT be used until signaled through a media (m=) line.

X-pc-secret:

Send: The field contains an end-to-end secret to be used for RTP and RTCP security. The secret is encoded similarly to the encryption key (k=) parameter of IETF/RFC 2327 [1] with the following constraints:

- The encryption key MUST NOT contain a ciphersuite, only a passphrase.
- The <method> specifying the encoding of the pass-phrase MUST be either "clear" or "base64" as defined in IETF/RFC 2045, except for the maximum line length which is not specified here. The method "clear" MUST NOT be used if the secret contains any characters that are prohibited in SDP.

Receive: Conveys the end-to-end secret to be used for RTP and RTCP security.

X-pc-csuites-rtp/X-pc-csuites-rtcp:

Send: The field contains a list of ciphersuites that the endpoint is capable of using for this connection (respectively RTP and RTCP). The first ciphersuite listed is what the endpoint is currently expecting to use. Any remaining ciphersuites in the list represent alternatives ordered by decreasing degree of preference, i.e. the most preferred alternative ciphersuite is the second one in the list. A ciphersuite is encoded as specified below:

- ciphersuite = [AuthenticationAlgorithm] "/" [EncryptionAlgorithm];
- AuthenticationAlgorithm = 1*(ALPHA/DIGIT/"-"/"_");
- EncryptionAlgorithm = 1*(ALPHA/DIGIT/"-"/"_");

where ALPHA, and DIGIT are defined in IETF/RFC 2234. Whitespaces are not allowed within a ciphersuite. The following example illustrates the use of ciphersuite:

- 62/51.

The actual list of ciphersuites is provided in the IPCablecom Specification TS 101 909-11 [3].

Receive: Conveys a list of ciphersuites that the remote endpoint is capable of using for this connection. Any other ciphersuite than the first in the list cannot be used until signaled through a new ciphersuite line with the desired ciphersuite listed first.

X-pc-spi-rtcp:

Send: The field contains the IPSEC Security Parameter Index (SPI) to be used when sending RTCP packets to the endpoint for the media stream in question. The SPI is a 32-bit identifier encoded as a string of up to 8 hex characters. The field MUST be supplied when RTCP security is used.

Receive: Conveys the IPSEC SPI to be used when sending RTCP packets over IPSEC. The field MUST be present when RTCP security is used.

X-pc-bridge:

Send: TGCP endpoints MUST NOT use this attribute.

Receive: TGCP endpoints MUST ignore this attribute if received.

recvonly:

Send: The field MUST be used in accordance with IETF/RFC 2543 (see bibliography).

Receive: The field MUST be used in accordance with IETF/RFC 2543 (see bibliography).

sendrecv:

Send: The field MUST be used in accordance with IETF/RFC 2543 (see bibliography).

Receive: The field MUST be used in accordance with IETF/RFC 2543 (see bibliography).

sendonly:

Send: The field MUST be used in accordance with IETF/RFC 2543 (see bibliography), except that the IP address and port number MUST NOT be zeroed.

Receive: The field MUST be used in accordance with IETF/RFC 2543 (see bibliography).

ptime:

Send: The ptime SHOULD always be provided and when used it MUST be used in accordance with IETF/RFC 2327 [1]. When an rtpmap or non well-known codec is used, the ptime MUST be provided.

Receive: The field MUST be used in accordance with IETF/RFC 2327 [1]. When "ptime" is present, the MTA MUST use the ptime in the calculation of QoS reservations. If "ptime" is not present, the MTA MUST assume reasonable default values for well-known codecs.

5.4.1.12 Media announcements (m=)

Media announcements (m=) consists of 3 sub-fields:

M= <media> <port> <transport> <format>
M= audio 3456 RTP/AVP 0

Media:

Send: The "audio" media type MUST be used.

Receive: The type received MUST be "audio".

Port:

Send: MUST be filled in accordance with IETF/RFC 2327 [1]. The port specified is the receive port, regardless of whether the stream is unidirectional or bidirectional. The sending port may be different.

Receive: MUST be used in accordance with IETF/RFC 2327 [1]. The port specified is the receive port. The sending port may be different.

Transport:

Send: The transport protocol "RTP/AVP" MUST be used.

Receive: The transport protocol MUST be "RTP/AVP".

Media Formats:

Send: Appropriate media type as defined in IETF/RFC 2327 [1] MUST be used.

Receive: In accordance with IETF/RFC 2327 [1].

5.5 Transmission over UDP

5.5.1 Reliable message delivery

MGCP messages are transmitted over UDP. Commands are sent to one of the IP addresses defined in the Domain Name System (DNS) for the specified endpoint or MGC. The responses are sent back to the source address of the command. However, it should be noted that the response may, in fact, come from another IP address than the one to which the command was sent.

When no port is provisioned for the endpoint (see note), the commands should be sent to the default MGCP port, 2427.

NOTE: Each endpoint may be provisioned with a separate MGC address and port.

MGCP messages, carried over UDP, may be subject to losses. In the absence of a timely response, commands are repeated. MGCP entities are expected to keep, in memory, a list of the responses sent to recent transactions, i.e. a list of all the responses sent over the last Tt_{hist} s, as well as a list of the transactions that are being executed currently. Transaction identifiers of incoming commands are compared to transaction identifiers of the recent responses. If a match is found, the MGCP entity does not execute the transaction, but simply repeats the response. If no match is found, the MGCP entity examines the list of currently executing transactions. If a match is found, the MGCP entity will not execute the transaction, which is simply ignored.

It is the responsibility of the requesting entity to provide suitable time-outs for all outstanding commands and to retry commands when time-outs have been exceeded. A retransmission strategy is specified in clause 5.5.2.

Furthermore, when repeated commands fail to get a response, the destination entity is assumed to be unavailable. It is the responsibility of the requesting entity to seek redundant services and/or clear existing or pending connections as specified in clause 4.4.

5.5.2 Retransmission strategy

The present document avoids specifying any static values for the retransmission timers since these values are typically network-dependent. Normally, the retransmission timers should estimate the timer by measuring the time spent between sending a command and the return of a response. Trunking gateways MUST at a minimum implement a retransmission strategy using exponential back-off with configurable initial and maximum retransmission timer values.

Trunking gateways SHOULD use the algorithm implemented in TCP-IP, which uses two variables (see e.g. TCP/IP Illustrated, Volume 1, The Protocols, see Bibliography).

The average response 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, RTO, in TCP, is set to the sum of the average delay plus N times the average deviation, where N is a constant.

After any retransmission, the MGCP 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 (RTO) to the minimum of:
  - the sum of that random value and N times the average deviation;
  - RTO_{max}, where the default value for RTO_{max} is 4 s.

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 subject to the needs of real-time communication. Because it includes a random component, it will break the potential synchronization between notifications triggered by the same external event.

The initial value used for the retransmission timer is 200 ms by default and the maximum value for the retransmission timer is 4 s by default. These default values may be altered by the provisioning process.

5.6 Piggy-backing

There are cases when an MGC will want to send several messages at the same time to one or more endpoints in a gateway and vice versa. When several messages have to be sent in the same UDP packets, they are separated by a line of text that contain a single dot, as in for example:

```
200 2005 OK
.
DLCX 1244 ds/ds1-2/2@tgw.whatever.net MGCP 1.0 TGCP 1.0
C: A3C47F21456789F0
I: FDE234C8
```

The piggy-backed messages MUST be processed as if they had been received in separate datagrams, however if a message (command or response) needs to be retransmitted, the entire datagram MUST be retransmitted, not just the missing message. The individual messages in the datagram MUST be processed in order starting with the first message.

Errors encountered in a message that was piggybacked MUST NOT affect any of the other messages received in that packet - each message is processed on its own.

5.7 Transaction identifiers and three ways handshake

Transaction identifiers are integer numbers in the range from 1 to 999 999 999. MGCs may decide to use a specific number space for each of the gateways that they manage, or to use the same number space for all gateways that belong to some arbitrary group. MGCs may decide to share the load of managing a large gateway 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 MUST guarantee that unique transaction identifiers are allocated to all transactions that originate from any MGC sent to a particular gateway within a period of Tt_{hist} s. Gateways can simply detect duplicate transactions by looking at the transaction identifier only.

The Response Acknowledgement parameter can be found in any command. It carries a set of "confirmed transaction-id ranges" for final responses received - provisional responses MUST NOT be confirmed.

MGCP gateways may choose to delete the copies of the responses to transactions whose id is included in "confirmed transaction-id ranges" received in a message, however the fact that the transaction was executed MUST still be retained for Tt_{hist} s. Also, when a Response Acknowledgement message is received (see note), the response that is being acknowledged by it can be deleted. Gateways should silently discard further commands from that MGC when the transaction-id falls within these ranges, and the response was issued less than Tt_{hist} s ago.

NOTE: As opposed to a command with a Response Acknowledgement parameter.

Let $term_{new}$ and $term_{old}$ be the endpoint-name in respectively a new command, cmd_{new}, and some old command. cmd_{old}. The transaction-ids to be confirmed in cmd_{new} SHOULD then be determined as follows:

1) If $term_{new}$ does not contain any wildcards:

   a) Unconfirmed responses to old commands where $term_{old}$ equals $term_{new}$.

   b) Optionally, one or more unconfirmed responses where $term_{old}$ contained the "any-of" wildcard, and the endpoint-name returned in the response was $term_{new}$.

   c) Optionally, one or more unconfirmed responses where $term_{old}$ contained the "all" wildcard, and $term_{new}$ is covered by the wildcard in $term_{old}$.

   d) Optionally, one or more unconfirmed responses where termold contained the "any-of" wildcard, no endpoint-name was returned, and termnew is covered by the wildcard in termold.

2) If $term_{new}$ contains the "all" wildcard:

   a) Optionally, one or more unconfirmed responses where $term_{old}$ contained the "all" wildcard, and $term_{new}$ is covered by the wildcard in $term_{old}$.

3) If $term_{new}$ contains the "any of" wildcard:

   a) Optionally, one or more unconfirmed responses where $term_{old}$ contained the "all" wildcard, and $term_{new}$ is covered by the wildcard in $term_{old}$ if the "any of" wildcard in $term_{new}$ was replaced with the "all" wildcard.

A given response SHOULD NOT be confirmed in two separate messages.

The following examples illustrate the use of these rules:

- If $term_{new}$ is "ds/ds1-2/1" and $term_{old}$ is "ds/ds1-2/1" then the old response can be confirmed per rule 1a.
- If $term_{new}$ is "ds/ds1-1/3" and $term_{old}$ is "*" then the old response can be confirmed per rule 1c.
- If $term_{new}$ is "ds/ds1-2/*" and $term_{old}$ is "*" then the old response can be confirmed per rule 2a.
- If $term_{new}$ is "ds/ds1-2/$" and $term_{old}$ is "ds/ds1-2/*" then the old response can be confirmed per rule 3a.

The "confirmed transaction-id ranges" values SHOULD NOT be used if more than Tt_{hist} s have elapsed since the gateway issued its last response to that MGC, or when a gateway resumes operation. In this situation, commands should be accepted and processed, without any test on the transaction-id.

Also, a response SHOULD NOT be confirmed if the response was received more than Tt_{hist} s ago.

Messages that confirm responses may be transmitted and received in disorder. The gateway shall retain the union of the confirmed transaction-ids received in recent commands.

5.8 Provisional responses

In some cases, transaction completion times may be significantly longer than otherwise. TGCP uses UDP as the transport protocol and reliability is achieved by selective time-out based retransmissions where the time-out is based on an estimate of the sum of the network roundtrip time and transaction completion time. Significant variance in the transaction completion time is therefore problematic when rapid message loss detection without excessive overhead is desired.

In order to overcome this problem, a provisional response MUST therefore be issued if, and only if, the transaction completion time exceeds some small period of time. The provisional response acknowledges the receipt of the command although the outcome of the command may not yet be known, e.g. due to a pending resource reservation. As a guideline, a transaction that requires external communication to complete, e.g. network resource reservation, should issue a provisional response. Furthermore, if a duplicate CreateConnection or ModifyConnection command is received, and the transaction has not yet finished executing, a provisional response MUST then be sent back.

Pure transactional semantics would imply, that provisional responses should not return any other information than the fact that the transaction is currently executing, however an optimistic approach allowing some information to be returned enables a reduction in the delay that would otherwise be incurred in the system.

Provisional responses MUST only be sent in response to a CreateConnection or ModifyConnection command. In order to reduce the delay in the system, a connection identifier and session description MUST be included in the provisional response to the CreateConnection command. If a session description will be returned by the ModifyConnection command, the session description MUST be included in the provisional response here as well. If the transaction completes successfully, the information returned in the provisional response MUST be repeated in the final response. It is considered a protocol error not to repeat this information or to change any of the previously supplied information in a successful response. If the transaction fails, an error code is returned - the information returned previously is no longer valid.

A currently executing CreateConnection or ModifyConnection transaction MUST be cancelled if a DeleteConnection command for the endpoint is received. In that case, a response for the cancelled transaction SHOULD still be returned automatically, and a response for the cancelled transaction MUST be returned if a retransmission of the cancelled transaction is detected.

When a provisional response is received, the time-out period for the transaction in question MUST be set to a significantly higher value for this transaction ($Tt_{longtran}$). The purpose of this timer is primarily to detect endpoint failure. The default value of $Tt_{longtran}$ is 5 s, however the provisioning process may alter this.

When the transaction finishes execution, the final response is sent and the by now obsolete provisional response is deleted. In order to ensure rapid detection of a lost final response, final responses issued after provisional responses for a transaction MUST be acknowledged. The endpoint MUST therefore include an empty "ResponseAck" parameter in those, and only those, final responses. The presence of the "ResponseAck" parameter in the final response will trigger a "Response Acknowledgement" response to be sent back to the endpoint. The "Response Acknowledgement" response will include the transaction-id of the response it acknowledges in the response header. Receipt of this "Response Acknowledgement" response is subject to the same time-out and retransmission strategies and procedures as responses to commands (see clause A.2.4), i.e. the sender of the final response will retransmit it if the "Response Acknowledgement" is not received in time. The "Response Acknowledgment " response is never acknowledged.

6 Security

If unauthorized entities could use the MGCP, they would be able to set up unauthorized calls or interfere with authorized calls. Security is not provided as an integral part of MGCP. Instead MGCP assumes the existence of a lower layer providing the actual security.

Security requirements and solutions for TGCP are provided in the IPCablecom specification TS 101 909-11 [3], which should be consulted for further information.

7 Event packages

This clause defines an initial set of event packages for the various types of endpoints currently defined by IPCablecom for trunking gateways.

Each package defines a package name for the package and event codes and definitions for each of the events in the package. In the tables of events/signals for each package, there are five columns:

- **Code** The package unique event code used for the event/signal.
- **Description** A short description of the event/signal.
- **Event** A check mark appears in this column if the event can be requested by the MGC. Alternatively, one or more of the following symbols may appear:
  - "**P**" indicating that the event is persistent;
  - "**S**" indicating that the event is an event-state that may be audited;
  - "**C**" indicating that the event/signal may be detected/applied on a connection.
- **Signal** If nothing appears in this column for an event, then the event cannot be signaled on command by the MGC. Otherwise, the following symbols identify the type of event:
  - "**OO**" On/Off signal. The signal is turned on until commanded by the MGC to turn it off, and vice versa;
  - "**TO**" Time-out signal. The signal lasts for a given duration unless it is superseded by a new signal. Default time-out values are supplied. A value of zero indicates that the time-out period is infinite. The provisioning process may alter these default values;
  - "**BR**" Brief signal. The event has a short, known duration.

- **Additional info** Provides additional information about the event/signal, e.g. the default duration of TO signals.

Unless otherwise stated, all of the events/signals are detected/applied on endpoints and audio generated by them is not forwarded on any connection the endpoint may have. Audio generated by events/signals that are detected/applied on a connection will however be forwarded on the associated connection irrespective of the connection mode.

7.1 SUP trunk package

Package Name: IT.

Table 12

Code	Description	Event	Signal	Additional Info
co1	Continuity tone 1	√	TO	Time-out = 3 s
co2	Continuity tone 2	√	TO	Time-out = 3 s
ft	Fax tone	√	-	
ld	Long duration connection	C	-	
ma	Media start	C	-	
mt	Modem tone	√	-	
oc	Operation complete	√	-	
of	Operation failure	√	-	
ro	Reorder tone	-	TO	Time-out = 30 s
rt	Ringback tone	-	C, TO	Time-out = 180 s
TDD	Telecom Devices for the Deaf (TDD) Tones	√		

The definition of the individual events and signals are as follows:

Continuity Tone 1 (co1): A tone at 2 010 Hz per ITU-T Recommendation Q.724. To conform with current continuity testing practice, the event SHOULD NOT be generated until the tone has been removed. The tone is of type TO - the continuity test will only be applied for the specified period of time. The provisioning process may alter the default value.

Continuity Tone 2 (co2): A tone at 1 780 Hz per ITU-T Recommendation Q.724. To conform with current continuity testing practice, the event SHOULD NOT be generated until the tone has been removed. The tone is of type TO - the continuity test will only be applied for the specified period of time. The provisioning process may alter the default value.

The continuity tones are used when the MGC wants to initiate a continuity test. There are two types of tests, single tone and dual tone. The party originating the continuity check signals and detects the appropriate tones for the trunk in question. For instance, for a continuity test from a 4-wire to a 2-wire circuit the following messages could be used:

Originating gateway:

```
RQNT 1234 ds/ds3-1/ds1-6/17@tgw1.example.net
X: AB123FE0
S: co2
R: co1
```

Terminating gateway:

```
CRCX 1234 ds/ds1-4/7@tgw2.example.net
C: A3C47F21456789F0
L: p:10, a:PCMU
M: conttest
```

The originating gateway sends the requested signal, and looks for the return of the appropriate tone for the trunk in question. When it detects that tone and deems the continuity check to be successful, it generates the "co1" event which in the example will be notified to the MGC. If the test does not succeed prior to the time-out, an operation complete event will be generated and in this case sent to the MGC. Similarly, if an error occurs prior to the time-out, an "operation failure" event will be generated. The "oc" and "of" events will be parameterized with the name of the event/signal they report, i.e. "co1" in this case.

Fax tone (ft): The fax tone event is generated whenever a fax communication is detected - see e.g. ITU-T Recommendations T.30 or V.21 (see bibliography).

Long duration connection (ld): The "long duration connection" is detected when a connection has been established for more than a certain period of time. The default value is 1 hour, however this may be changed by the provisioning process.

The event may be detected on a connection. When no connection is specified, the event applies to all connections for the endpoint, regardless of when the connections are created.

Media start (ma): The media start event occurs on a connection when the first valid RTP media packet is received on the connection (see note 1). This event can be used to synchronize a local signal, e.g. ringback, with the arrival of media from the other party.

The event may be detected on a connection. When no connection is specified, the event applies to all connections for the endpoint, regardless of when the connections are created.

NOTE 1: When authentication and integrity security services are used, an RTP packet is not considered valid until it has passed the security checks.

Modem tones (mt): The modem tone event is generated whenever a modem communication is detected, see e.g. ITU-T Recommendation V.8 (see bibliography).

Operation complete (oc): The operation complete event is generated when the gateway was asked to apply one or several signals of type TO on the endpoint, and one or more of those signals completed without being stopped by the detection of a requested event such as "continuity tone 1". The completion report may carry as a parameter the name of the signal that came to the end of its live time, as in:

```
O: IT/oc(IT/co1)
```

When the reported signal was applied on a connection, the parameter supplied will include the name of the connection as well, as in:

```
O: IT/oc(IT/rt@0A3F58)
```

When the operation complete event is requested, it cannot be parameterized with any event parameters. When the package name is omitted, the default package name is assumed.

The operation complete event may additionally be generated as defined in the base protocol, e.g. when an embedded ModifyConnection command completes successfully, as in note 2:

```
O: IT/oc(B/C)
```

NOTE 2: Note the use of "B" here as the prefix for the parameter reported.

Operation failure (of): In general, the operation failure event may be generated when the endpoint was asked to apply one or several signals of type TO on the endpoint, and one or more of those signals failed prior to timing out. The completion report may carry as a parameter the name of the signal that failed, as in:

```
O: IT/of(IT/co2)
```

When the reported signal was applied on a connection, the parameter supplied will include the name of the connection as well, as in:

```
O: IT/of(IT/rt@0A3F58)
```

When the operation failure event is requested, event parameters can not be specified. When the package name is omitted, the default package name is assumed.

The operation failure event may additionally be generated as specified in the base protocol, e.g. when an embedded ModifyConnection command fails, as in note 2:

```
O: IT/of(B/C(M(sendrecv(AB2354))))
```

Reorder tone (ro): Reorder tone, as known as congestion tone, is specified in ITU-T Recommendation Q.35 (see bibliography).

Ring back tone (rt): Audible Ring Tone is specified in ITU-T Recommendation Q.35 (see bibliography). The definition of the tone is defined by the national characteristics of the Ringback Tone, and MAY be established via provisioning. The ringback signal can be applied to both an endpoint and a connection.

Telecomm Devices for the Deaf tones (TDD): The TDD event is generated whenever a TDD communication is detected - see e.g. ITU-T Recommendation V.18 (see bibliography).

Annex A (informative): Bibliography

Bellcore: "Bellcore Notes on the Networks", SR-2275.F.

Bellcore: "Compatibility Information for Feature Group D Switched Access Service", TR-NPL-000258, Issue 1, October 1985.

Bellcore: "Interoffice LATA Switching Systems Generic Requirements (LSSGR): Verification Connections", (25-05-0903), TR-TSY-000531, Issue 2, July 1987.

Bellcore: "LSSGR: Signalling for Analog Interfaces", GR-506-CORE, Issue 1, June 1996.

Bellcore: "LSSGR: Switching System Generic Requirements for Call Control Using the Integrated Services Digital Network User Part (ISDNUP)", GR-317-CORE, Issue 2, December 1997.

Bellcore: "OSSGR: Custom Call Handling Features (FSD 80 Series)", GR-1176-CORE, Issue 1, March 1999.

IETF/RFC 1827 (1995): "IP Encapsulating Security Payload (ESP)" (Status: Proposed Standard).

RTP Parameters: http://www.isi.edu/in-notes/iana/assignments/rtp-parameters.

"SAP - Session Announcement Protocol", Handley, M., Work in Progress.

"Simple Gateway Control Protocol 1.1", Mauricio Arango, Christian Huitema, draft-huitema-sgcp-v1-02.txt, 30 July 1998.

"IPDC Base Protocol", P. Tom Taylor, Pat R. Calhoun, Allan C. Rubens, draft-taylor-ipdc-00.txt, July 1998.

IETF/RFC 1889: "RTP: A Transport Protocol for Real-Time Applications".

IETF/RFC 1890 (1996): "RTP Profile for Audio and Video Conferences with Minimal Control".

IETF/RFC 2543 (1999): "SIP: Session Initiation Protocol".

IETF/RFC 2326 (2000): "Real Time Streaming Protocol (RTSP)".

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

ITU-T Recommendation Q.761 (1999): "Signalling system No. 7 - ISDN User Part functional description".

ITU-T Recommendation Q.762 (1999): "Signalling System No. 7 - ISDN User Part general functions of messages and signals 20 CHF".

ITU-T Recommendation H.323: "Packet-based multimedia communications systems".

ITU-T Recommendation H.225: "Call signalling protocols and media stream packetization for packet-based multimedia communication system".

ITU-T Recommendation H.245: "Control protocol for multimedia communication".

IETF/RFC 1825 (1995): "Security Architecture for the Internet Protocol".

IETF/RFC 1826 (1995): "IP Authentication Header".

IETF/RFC 2705 (1999): "Media Gateway Control Protocol (MGCP) Version 1.0".

TCP/IP Illustrated, Volume 1: "The Protocols", Stevens, W. Richard, Addison-Wesley, 1994.

ETSI TS 101 909-5: "Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 5: Dynamic Quality of Service for the Provision of Real Time Services over Cable Television Networks using Cable Modems".

ITU-T Recommendation J.83: "Digital multi-programme systems for television, sound and data services for cable distribution".

ITU-T Recommendation J.112: "Data-over-cable service interface specifications: Radio frequency interface specification".

IETF/RFC 1034: "Domain names - concepts and facilities".

IETF/RFC 821: "Simple Mail Transfer Protocol".

PKT-SP-ESP-I01-991229: "PacketCable™ Electronic Surveillance Specification".

IETF/RFC 2234: "Augmented BNF for Syntax Specifications: ABNF".

IETF/RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".

IETF/RFC 2327: "SDP: Session Description Protocol".

ITU-T Recommendation Q.724: "Telephone user part signalling procedures".

ITU-T Recommendation T.30: "Procedures for document facsimile transmission in the general switched telephone network".

ITU-T Recommendation V.21: "300 bits per second duplex modem standardized for use in the general switched telephone network".

ITU-T Recommendation V.8: "Procedures for starting sessions of data transmission over the public switched telephone network".

ITU-T Recommendation Q.35: "Technical characteristics of tones for the telephone service".

ITU-T Recommendation V.18: "Operational and interworking requirements for DCEs operating in the text telephone mode".

IETF/RFC 2705: "Media Gateway Control Protocol (MGCP) Version 1.0".

ITU-T Recommendation H.248: "Gateway control protocol".

List of ITU-T Recommendations referring to IP Cablecom:

ITU-T Recommendation J.160: "Architectural framework for the delivery of time critical services over cable television networks using cable modems".

ITU-T Recommendation J.161: "Audio codec requirements for the provision of bi-directional audio service over cable television networks using cable modems".

ITU-T Recommendation J.162: "Network call signalling protocol for the delivery of time critical services over cable television networks using cable modems".

ITU-T Recommendation J.163: "Dynamic quality of service for the provision of real time services over cable television networks using cable modems".

ITU-T Recommendation J.164: "IPCablecom event messages".

ITU-T Recommendation J.165: "IPCablecom Internet Signalling Transport Protocol".

ITU-T Recommendation J.166: "IPCablecom management information base (MIB) framework".

ITU-T Recommendation J.167: "Media terminal adapter (MTA) device provisioning requirements for the delivery of real time services over cable television networks using cable modems".

ITU-T Recommendation J.168: "IPCablecom Media Terminal Adapter (MTA) MIB requirements".

ITU-T Recommendation J.169: "IPCablecom network call signalling (NCS) MIB requirements".

ITU-T Recommendation J.170: "IPCablecom Security specification".

ITU-T Recommendation J.171: "IPCablecom Trunking Gateway Control Protocol (TGCP)".

ETSI TS 101 909-13-1: "Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 13: Trunking Gateway Control Protocol; Sub-part 1: H.248 option".

History

Document history		
V1.1.1	September 2001	Publication as TS 101 909-13
V1.1.2	March 2002	Publication