

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

J.171.1
(11/2005)

SERIES J: CABLE NETWORKS AND TRANSMISSION
OF TELEVISION, SOUND PROGRAMME AND OTHER
MULTIMEDIA SIGNALS

IPCablecom

**IPCablecom trunking gateway control
protocol (TGCP): Profile 1**

ITU-T Recommendation J.171.1

Document ID: 123456789

ITU-T Recommendation J.171.1

IPCablecom trunking gateway control protocol (TGCP): Profile 1

Summary

This Recommendation describes an IPCablecom profile of an Application Programming Interface (API) called a Media Gateway Control Interface (MGCI) and a corresponding protocol (MGCP) for controlling voice-over-IP (VoIP) PSTN gateways from external call control elements. This is one of the two profiles referenced in ITU-T Rec. J.171.0. The second profile is specified in ITU-T Rec. J.171.2.

The MGCP assumes a call control architecture where the call control "intelligence" is outside the gateways and handled by external call control elements. The IPCablecom profile as described in this Recommendation will be referred to as the IPCablecom Trunking Gateway Control Protocol (TGCP).

Source

ITU-T Recommendation J.171.1 was approved on 29 November 2005 by ITU-T Study Group 9 (2005-2008) under the ITU-T Recommendation A.8 procedure.

FOREWORD

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

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

© ITU 2006

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

CONTENTS

Page

1	Scope	1
2	References.....	1
2.1	Normative references.....	1
2.2	Informative references.....	1
3	Definitions	2
4	Abbreviations and acronyms	2
5	Convention.....	2
5.1	Background.....	3
6	Introduction	4
6.1	Relation with other IPCablecom standards	5
6.2	Relation to RFC 3435 and ABNF grammar	6
7	Media Gateway Control Interface (MGCI)	6
7.1	Model and naming conventions.....	6
7.2	SDP use	12
7.3	Gateway control functions.....	13
7.4	States, failover and race conditions	36
7.5	Return codes and error codes	51
7.6	Reason codes	52
7.7	Use of local connection options and connection descriptors.....	53
8	Media gateway control protocol	56
8.1	General description.....	56
8.2	Command header	57
8.3	Response header formats	69
8.4	Session description encoding	72
8.5	Transmission over UDP	83
8.6	Piggybacking	85
8.7	Transaction identifiers and three-way handshake	85
8.8	Provisional responses	86
9	Security	87
	Annex A – Event packages	88
	A.1 ISUP trunk package	88
	Appendix I – Mode interactions.....	92
	Appendix II – Example command encodings	94
	II.1 NotificationRequest	94
	II.2 Notify.....	94
	II.3 CreateConnection	94

	Page
II.4 ModifyConnection.....	96
II.5 DeleteConnection (from the Media Gateway Controller).....	97
II.6 DeleteConnection (from the trunking gateway).....	97
II.7 DeleteConnection (multiple connections from the Media Gateway Controller)	97
II.8 AuditEndpoint	98
II.9 AuditConnection.....	99
II.10 RestartInProgress.....	99
Appendix III – Example Call Flow.....	101
Appendix IV – Endpoint requirements	104
IV.1 Connection modes supported	104
Appendix V – Compatibility information.....	105
V.1 NCS compatibility	105
V.2 MGCP compatibility	105
Appendix VI – ABNF grammar for TGCP.....	107
Appendix VII – Electronic surveillance.....	114
VII.1 MGC	114
VII.2 MG.....	114
Appendix VIII – Example event packages	116
VIII.1 MF FGD Operator Services package	116
VIII.2 MF Terminating Protocol package.....	119
BIBLIOGRAPHY	122

ITU-T Recommendation J.171.1

IPCablecom trunking gateway control protocol (TGCP): Profile 1

1 Scope

This Recommendation describes an IPCablecom profile of an Application Programming Interface (API) called a Media Gateway Control Interface (MGCI) and a corresponding protocol (MGCP) for controlling voice-over-IP (VoIP) PSTN gateways from external call control elements. This is one of the two profiles referenced in ITU-T Rec. J.171.0. The second profile is specified in ITU-T Rec. J.171.2.

The MGCP assumes a call control architecture where the call control "intelligence" is outside the gateways and handled by external call control elements. The IPCablecom profile as described in this Recommendation will be referred to as the IPCablecom Trunking Gateway Control Protocol (TGCP).

This Recommendation is based on the IPCablecom network-based call signalling Recommendation (ITU-T Rec. J.162), and IETF RFC 2705, *Media Gateway Control Protocol (MGCP)*. This Recommendation, which defines the IPCablecom TGCP protocol, constitutes a specification that is independent of MGCP. The TGCP profile of MGCP is strictly and solely defined by the contents of this Recommendation.

2 References

2.1 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- ITU-T Recommendation J.161 (Draft, version 2), *Audio codec requirements for the provision of bidirectional audio service over cable television networks using cable modems*.
- ITU-T Recommendation J.162 (2005), *Network call signalling protocol for the delivery of time-critical services over cable television networks using cable modems*.
- ITU-T Recommendation J.170 (2005), *IPCablecom security specification*.
- IETF RFC 2327 (1998), *SDP: Session Description Protocol*.

2.2 Informative references

- ITU-T Recommendation E.180/Q.35 (1998), *Technical characteristics of tones for the telephone service*.
- ITU-T Recommendation J.163 (2005), *Dynamic quality of service for the provision of real-time services over cable television networks using cable modems*.
- ITU-T Recommendation J.171.0 (2005), *IPCablecom trunking gateway control protocol (TGCP): Profiles overview*.
- IETF RFC 1889 (1996) *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 2705 (1999), *Media Gateway Control Protocol (MGCP) Version 1.0*.
- TCP/IP Illustrated, Volume 1 (2001), *The Protocols*, Addison-Wesley, 1994.

3 Definitions

This Recommendation defines the following terms:

3.1 cable modem: The delivery of high-speed data access to customer locations using equipment built in conformance with ITU-T Recs J.83 and J.112.

3.2 IPCablecom: An ITU-T project that includes an architecture and a series of Recommendations that enable the delivery of real-time services over the cable television networks using cable modems.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

DNS	Domain Name System
IP	Internet Protocol
IPSec	Internet Protocol Security
ISUP	ISDN User Part
MGC	Media Gateway Controller
MGCP	Media Gateway Control Protocol
MIB	Management Information Base
MTA	Media Terminal Adapter
MWD	Maximum Waiting Delay
NCS	Network-based Call Signalling
NTP	Network Time Protocol
QoS	Quality of Service
RTCP	Real-Time Control Protocol
RTO	Retransmission Timeout
RTP	Real-Time Protocol
SDP	Session Description Protocol
SG	Signalling Gateway
SPI	Security Parameters Index

5 Convention

If this Recommendation is implemented, the keywords "MUST" and "SHALL" as well as "REQUIRED" are to be interpreted as indicating a mandatory aspect of this Recommendation.

The key words indicating a certain level of significance of particular requirements that are used throughout this Recommendation are summarized as below.

"MUST"	This word or the adjective "REQUIRED" means that the item is an absolute requirement of this Recommendation.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this Recommendation.
"SHOULD"	This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behaviour is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behaviour described with this label.
"MAY"	This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

5.1 Background

This Recommendation describes an IP-Cablecom profile of an application programming interface called a Media Gateway Control Interface (MGCI) and a corresponding protocol (MGCP) for controlling voice-over-IP (VoIP) PSTN Gateways from external call control elements. The MGCP assumes a call control architecture where the call control "intelligence" is outside the gateways and handled by external call control elements. The IP-Cablecom profile as described in this Recommendation will be referred to as the IP-Cablecom Trunking Gateway Control Protocol (TGCP) Profile 1.

This Recommendation is based on the IP-Cablecom Network-based Call Signaling (NCS) ITU-T Rec. J.162, the Media Gateway Control Protocol (MGCP) 1.0 IETF RFC 3435 (which was the result of a merge of the IETF draft of Simple Gateway Control Protocol and the IETF draft of IP Device Control (IPDC) family of protocols), and input generated by the IP-Cablecom PSTN Gateway focus team.

This Recommendation, which defines the IP-Cablecom TGCP Protocol, constitutes a document that is independent of MGCP in order to provide a stable reference document while meeting current time-to-market demands for such a reference. It is the intent of this Recommendation to be as closely aligned with NCS and MGCP 1.0 as possible for the IP-Cablecom environment, in order to avoid developing multiple protocols to solve the same problem. This goal has been and continues to be pursued through cooperation with the authors of the NCS and MGCP Recommendations. The TGCP profile of MGCP, however, is strictly and solely defined by the contents of this Recommendation.

This TGCP profile of MGCP, which will be referred to as the PSTN Trunking Gateway Call Signaling Protocol 1.0, TGCP 1.0, the TGCP profile, or simply TGCP in this Recommendation, has been modified from the MGCP 1.0 IETF RFC 2435 in the following ways:

- *The TGCP protocol only aims at supporting IP-Cablecom voice-over-IP PSTN Gateways.* The TGCP protocol supports voice-over-IP PSTN Gateways as defined by IP-Cablecom. Functionality present in the MGCP 1.0 protocol, which was superfluous to TGCP, has been removed.

- *The TGCP protocol contains extensions and modifications to MGCP.* IPCablecom-specific requirements are addressed in TGCP. However, the MGCP architecture, and all of the MGCP constructs relevant to PSTN Gateways, have been preserved in TGCP.
- *The TGCP protocol contains minor simplifications from MGCP 1.0.* Where several choices were available, and not necessarily needed for a PSTN Gateway in the IPCablecom environment, some simplifications have been made for trunking gateway implementations.

Although MGCP is not TGCP, and TGCP is not MGCP, the names MGCP and TGCP will be used interchangeably in this Recommendation since this Recommendation is based on MGCP. Unless otherwise stated or inferable by context, the word MGCP shall be taken to mean the TGCP profile of MGCP in this Recommendation.

TGCP is designed to meet the protocol requirements for the Media Gateway Controller to Media Gateway interface defined in the IPCablecom architecture.

6 Introduction

This Recommendation describes the TGCP profile of an application programming interface (MGCI) and a corresponding protocol (MGCP) for controlling trunking gateways from external call control elements. A trunking gateway is a network element that provides analog, emulated analog, or digital bearer and channel-associated signalling trunk circuit access to a voice-over-IP (VoIP) network.

Trunking gateways are used for interfacing to the PSTN and as such are expected to adhere to the relevant electrical, operational, and signalling standards for the trunk type they implement. The TGCP Recommendation is developed for CLECs, and in the first version of this Recommendation the trunk types that will be supported are limited to the following:

- SS7 ISUP:
 - Standard Bearer trunk.
- MF trunks on a combined Equal Access End Office/Access Tandem (EAEO/AT):
 - Operator Services trunks¹;
 - CLEC subscriber access through a LEC Access Tandem;
 - Operator access to EAEO/AT for Busy Line Verification and Barge In;
 - Emergency service trunks for access to an emergency service Tandem.

The MGCP assumes a call control architecture where the call control "intelligence" is outside the gateways and handled by external call-control elements referred to as Media Gateway Controllers. The MGCP assumes that these call-control elements, or Media Gateway Controllers (MGCs), will synchronize with each other to send coherent commands to the gateways under their control. The MGCP defined in this Recommendation does not define a mechanism for synchronizing Media Gateway Controllers, although future IPCablecom Recommendations may specify such mechanisms.

The MGCP assumes a connection model where the basic constructs are endpoints and connections. A gateway contains a collection of endpoints, which are sources, or sinks, of data and could be physical or virtual.

An example of a physical endpoint is a trunk circuit on a trunking gateway that terminates an ISUP bearer trunk to a switch. Another example is an embedded client or residential gateway that terminates residential POTS lines (to phones), although such devices are not addressed by this Recommendation.

¹ Operator Services are expected to be provided by or accessed through a LEC.

An example of a virtual endpoint is an audio source in an audio-content server. Creation of physical endpoints requires hardware installation, while creation of virtual endpoints can be accomplished by software. However, the TGCP profile of MGCP only addresses physical endpoints.

Connections are point-to-point. A point-to-point connection is an association between two endpoints with the purpose of transmitting data between these endpoints. Once this association is established for both endpoints, data transfer between these endpoints can take place. The association is established by creating the connection as two halves: one on the origination endpoint, and one on the terminating endpoint.

Media Gateway Controllers instruct the gateways to create connections between endpoints and to detect certain events, e.g., continuity test, and generate certain signals, e.g., ringback. It is strictly up to the Media Gateway Controller to specify how and when connections are made, between which endpoints they are made, as well as what events and signals are to be detected and generated on the endpoints. The gateway, thereby, becomes a simple device, without any call state, that receives general instructions from the Media Gateway Controller without any need to know about or even understand the concept of calls, call states, features, or feature interactions. When new services are introduced, customer profiles changed, etc., the changes are transparent to the gateway. The Media Gateway Controllers implement the changes and generate the appropriate new mix of instructions to the gateways for the changes made. Whenever the gateway reboots, it will come up in a clean state and simply carry out the Media Gateway Controller's instructions as they are received.

6.1 Relation with other IP-Cablecom standards

An IP-Cablecom PSTN Gateway consists of three functional components:

- Media Gateway Controller (MGC), which contains the call intelligence and terminates call signalling. This component is also referred to as a Call Agent.
- Media Gateway (MG), which terminates the bearer channels under the instruction and control of the MGC. This function is also known as a Trunking Gateway (TGW);
- Signalling Gateway (SG), which connects the call signaling to the PSTN and provides a signalling translation function.

In addition to the PSTN Gateway protocols, IP-Cablecom also includes a Network-based Call Signaling (NCS) protocol, which is a profile of MGCP. The SDP standard has a pivotal status in this architecture. Both NCS and TGCP use the Session Description Protocol (SDP) to carry session descriptions.

IP-Cablecom NCS and TGCP systems hold all of the call state in CMS and MGC Call Agents. Figure 1 below illustrates the relationships among these different components:

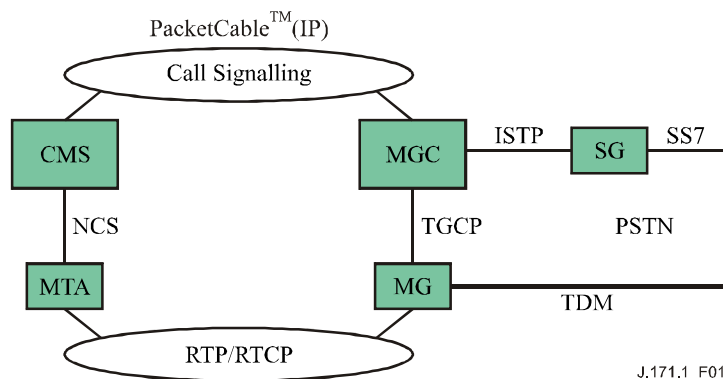


Figure 1/J.171.1 – Relationship among NCS and TGCP components

6.2 Relation to RFC 3435 and ABNF grammar

RFC 3435 includes a formal description of the MGCP protocol syntax following the "Augmented BNF for Syntax Specifications". This formal description is referenced by developers for the creation of interoperable devices. A copy of the MGCP protocol syntax, annotated and edited to indicate its applicability to IP-Cablecom Recommendations, is provided in Appendix VI. Adherence to these guidelines can improve interoperability by minimizing failures caused by different interpretations of syntax and grammar.

7 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.

7.1 Model and naming conventions

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 IP-Cablecom network, as the term "connection" is understood within the circuit-switched PSTN. The terms "call" and "connection" in this context (and throughout this Recommendation) are used for convenience of reference, not to indicate any actual technical or other similarity between the IP-Cablecom network and the PSTN.

7.1.1 Endpoint names

Endpoint names, also 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 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 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 wild-carding ("*", "\$"), 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 wild cards are used, dollar-signs are only allowed from the right: thus if a term had a dollar sign wild card, all terms to the right of that term must also contain dollar sign wild cards.
- 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". Unless specified otherwise, this refers to all endpoints configured for service, regardless of their actual service state, i.e., in-service or out-of-service.
- 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". Unless specified otherwise, this only refers to endpoints that are in-service.
- 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. Note that since the domain name portion is part of the endpoint identifier, different forms or different values referring to the same entity are not freely interchangeable. Following a restart the most recently supplied form and value MUST always be used.

7.1.1.1 Trunking gateway endpoint names

Endpoints in trunking gateways MUST support 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.

7.1.1.1.1 Trunk circuit endpoints

In addition to the naming conventions specified above, local endpoint names for PSTN trunking gateway endpoints of type "ds" MUST 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* number² at the lowest level of the hierarchy.
- 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.
- Wild-card naming follows the conventions stated in 7.1.1 with the asterisk character ("*") referring to "all", and the dollar character ("\$") referring to "any". A range "[N-M]" wild-carding 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" wild-card 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 wild-card character "*", referring to "all of", unless the terms specified contain the "any of" wild-card character, in which case missing terms to the right of the last term specified are assumed to be the "any of" wild-card character;
 - wherever use of the "all" wild-card is permitted, the range of channels "[N-M]" wild-card may be used in the last term (i.e. <channel-#>) of the local endpoint name instead. The "range" wild-card will then refer to all of the channels from N to M. The rules and

² Please note the use of the term "channel" as opposed to "time slot".

restrictions that apply to the use of the "all" wild-card also apply to the use of the "range" wild-card.

The following examples illustrate the use of wild-carding:

<code>ds/ds1-3/*</code>	All channels on ds1 number 3 on the media gateway in question.
<code>ds/ds1-3/\$</code>	Any channel on ds1 number 3 on the media gateway in question.
<code>ds/*</code>	All trunk-circuit endpoints on the media gateway in question.
<code>*</code>	All endpoints (regardless of endpoint-type) on the media gateway in question.
<code>ds/ds1-3/[1-24]</code>	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 this Recommendation, e.g., to support bonding of multiple DS-0 trunks for video calls, e.g., on the form "`ds/ds1-1/H0-1`".

7.1.2 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. Call identifiers with a maximum length of 32 characters MUST be supported.

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 MGC's is outside the scope of this Recommendation. When an MGC builds several connections that pertain to the same call, either on the same gateway or in different gateways, these connections will all 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.

7.1.3 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. Connection names with a maximum length of 32 characters MUST be supported.

7.1.4 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 System (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 failover 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 start up, 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" MUST 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 explicitly³, 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".

Clause 7.4 contains a more detailed description of reliability and failover.

7.1.5 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 Recommendation have a need for digit maps, and digit maps are therefore not part of this Recommendation.

7.1.6 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 MUST NOT 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.

³ This could happen as a result of specifying an empty NotifiedEntity parameter.

Table 1 below lists trunking gateway endpoint-types and the packages defined for them in this Recommendation. The MGC MUST support all packages listed in Table 1. The MG MUST support the "IT", "FXR", and "XRM" packages. The MG SHOULD support the "MO" and "MT" packages.

Table 1/J.171.1 – Packages Associated with endpoint-types

Endpoint-type	Package	Package name	Default package	1st release that introduced package support requirement
DS-0	ISUP trunk	IT	Yes	1.0
DS-0	MF OSS	MO	No	1.5
DS-0	MF Terminating	MT	No	1.5
DS-0	FAX	FXR	No	1.5
DS-0	VoIP Metrics	XRM	No	1.5

Additional package names and event codes may be defined by and/or registered with IPCablecom. Any change to the packages defined in this Recommendation 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 7.3.1) and if they contain auditable event-states (see 7.3.8.1). 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 7.3.1.

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 wild-card notation each. The wild-card 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

Should the connection be deleted while an event or signal is being detected or applied on it, that particular event detection or signal generation **MUST** stop. Depending on the signal type the endpoint **SHOULD** generate a failure, i.e., if the signal type is TO, the "operation failure" event will be generated since the connection associated with the signal was deleted prior to the signal timing out. Notification action associated with reporting the failure must conform to the notify operations as defined for NotificationRequest handling (clause 7.3.1).

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

IT/ma@*

However, when the event is actually observed, the gateway **MUST** include the name of the specific connection on which the event occurred. The wild-card 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 **MUST** generate or detect the event on the connection that is currently being created or modified. An example of this convention is:

IT/rt@\$

When processing a command using the "current connection" wildcard, the "\$" wildcard character **MUST** be expanded by the gateway to the value of the current connection. If a subsequent command either explicitly (e.g., by auditing) or implicitly (e.g., by persistence) refers to such an event, the expanded value **MUST** be used by the gateway. In other words, the "current connection" wildcard is expanded once, which is at the initial processing of the command in which it was explicitly included.

The connection id, or a wild-card 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). However, the use of the "all packages" and "all events" wildcards are strongly discouraged. Call Agents must be able to operate in an environment where some endpoints do not support all packages. An endpoint that receives a command referencing a package it does not support will respond with error 518 (Unsupported or Unknown Package). On receiving this error response, the Call Agent could try the command again without the package parameter, although in the case where the original command contained parameters for multiple packages, the Call Agent would have no way of knowing which specific package(s) to exclude. The Call Agent can also use the AuditEndpoint command to determine the set of packages supported by an endpoint.

7.2 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 this Recommendation, 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. In addition, all MGCs and media gateways **MUST** ignore any SDP parameters, attributes, or fields that are not understood by the call agent or gateway.

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

SDP allows for description of real-time fax using media type "image". The TGCP profile will support setting of fax connections using media type "image".

7.3 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 this Recommendation, 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 clause 8. Trunking gateways and Call Agents 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 this Recommendation.
- 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 MUST be sent on any connection whose mode is either "send only", or "send/receive ", unless the endpoint has a connection in "loopback" or "continuity test" mode. However, audio generated by applying a signal to a connection MUST be sent on the connection for all modes except "network loopback".

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 MUST be discarded.
- Audio signals received in data packets through connections in "receive only", or "send/receive" mode MUST be mixed and sent to the endpoint⁴, unless the endpoint has another connection in "loopback" or "continuity test" mode.
- Audio signals originating from the endpoint MUST be transmitted over all the connections whose mode is "send only", or "send/receive" unless the endpoint has another connection in "loopback" or "continuity test" mode.

Note that in order to detect events on a connection, the connection MUST by default be in one of the modes "receive only", "send/receive", "network loopback" or "network continuity test". The event detection only applies to the incoming audio. Connections in "send only", "inactive", "loopback", or "continuity test" mode will thus normally not detect any events, although requesting to do so is not considered an error.

The "loopback" and "continuity test" modes are used during maintenance and continuity test operations. An endpoint may have more than one connection in either "loopback" or "continuity test" mode. As long as there is one connection in that particular mode, and no other connection on the endpoint is placed in a different maintenance or test mode, the maintenance or test operation MUST continue undisturbed. There are two variations of continuity test (COT), one specified by ITU, 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 detect the return tone within a certain period of time, the COT has failed.

If the mode is set to "loopback", the gateway MUST return the incoming signal from the endpoint back into that same endpoint. This procedure will be used, typically, for testing the continuity of trunk circuits according to ITU specifications. 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 MUST *not* be sent to the endpoint;
- audio signals received on the endpoint MUST *not* be sent to any connection for the endpoint.

If the mode is set to "network loopback", the audio signals received from the connection MUST be echoed back on the same connection. The "network loopback" mode SHOULD simply operate as an RTP packet reflector. The media MUST not be forwarded to the endpoint.

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

⁴ TGCP endpoints are currently not required to support mixing however.

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.

Please refer to Appendix I for illustrations of mode interactions.

7.3.1 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 accordingly⁵.

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 **MUST** follow the rules for endpoint names specified in 7.1.1. The "any of" wildcard **MUST NOT** be used. A gateway that receives a NotificationRequest with the "any of" wildcard convention **MUST** return an error (error returned **SHOULD** be error code 500 – the transaction could not be executed because the endpoint is unknown) in response. The "all of" wildcard **MUST** be supported for NotificationRequests with each of RequestedEvents, SignalsRequest, and DetectEvents being either empty or omitted. For simplicity, some gateways may choose to not support the "all-of" wildcard for NotificationRequests where one or more of these parameters is neither empty nor omitted. Such gateways shall respond with error code 503 if they receive an "all-of" wildcarded NotificationRequest which they are unable to process for this reason.

NotifiedEntity is an optional parameter that specifies a new "notified entity" for the endpoint. When used, the entire Media Gateway Controller name **MUST** be specified which includes both the local name and domain name – even if a bracketed IP address is used for the domain name. See 7.1.1 and 7.1.4 for more information. If, however, only the domain name is provided, the MG **SHOULD** use the domain name as the Call Agent ID.

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 signals⁶:

- Continuity test;
- Set up MF OSS call.

⁵ The new instruction would be a ModifyConnection command.

⁶ 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.1 for further definition of this event). A TO signal could be "place MF call" timing out after 16 seconds. If an event occurs prior to the 16 seconds, the signal will, by default, be stopped⁷. 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 out⁸. 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 an "operation complete" event will generate an "operation failure" event which will include the name(s) of the signal(s) that time out⁸.
- **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.

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. For 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 7.3.3), the media stream generated by that signal **MUST** 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 **MUST** remain in effect, regardless of what values are provided subsequently. A given signal **MUST NOT** appear more than once in a SignalRequests. The omission of the SignalRequests parameter is interpreted as an empty SignalRequests list.

⁷ The "Keep signal(s) active" action may override this behaviour.

⁸ If parameters were passed to the signal, the parameters will not be reported.

The currently defined signals can be found in Annex A.

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 Annex A.

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 action⁹. 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 performed¹⁰. Persistent events are identified as such through their definition – see Annex A.

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. The omission of the RequestedEvents parameter is interpreted as an empty RequestedEvents list. More than one action can be specified for an event, although a given action cannot appear more than once for a given event. The matrix in Table 2 specifies the legal combinations of actions:

⁹ Thus, the RequestIdentifier will be the RequestIdentifier of the current NotificationRequest.

¹⁰ 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.

Table 2/J.171.1 – Actions associated with events

Event	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.

When a stimulus that triggers multiple requested events is detected (e.g., fax tone is the stimulus for both FXR/gwfax(start), and L/ft), the gateway **MUST** generate only one of the events (namely the most preferred event out of the multiple requested events that triggered) based on the following precedence rules:

- 1) Events included in a RequestedEvents list are prioritized from left to right, with the most preferred event listed to the left.
- 2) Persistent events that are not included in a RequestedEvents lists are less preferred than events (persistent or not) that are included in a RequestedEvents list. There is no well-defined preference order among persistent events that are not included in a RequestedEvents list.

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 7.4.3.1).

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 wild-card "\$" can be used to denote "the current connection"; however, this notation **MUST NOT** be used outside a connection handling command – the wild-card 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 Annex A 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 Annex A for details) – the rest of the connection mode changes **MUST NOT** be attempted, and the previous successful connection mode changes in the list **MUST** remain effective.

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.

NOTE – Clause 7.4.3.1 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 Recommendation. 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 Recommendation. 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 cannot 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 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 7.4.3.1), i.e., events that have been detected by the gateway before the arrival of this NotificationRequest command, but have not yet been notified to the MGC. The parameter provides a set of handling options:

- whether the quarantined events should be processed or discarded (the default is to process them);
- whether the gateway is expected to generate at most one notification (lockstep), or multiple notifications (loop), in response to this request (the default is at most one).

When the parameter is absent, the quarantined events **MUST** be processed. Support for the "lockstep" mode (via default) and "loop" mode is mandatory. An endpoint that receives a NotificationRequest with an unsupported QuarantineHandling parameter value **SHOULD** respond with error code 508 (unsupported QuarantineHandling).

Note that the quarantine-handling parameter also governs the handling of events that were detected and processed but not yet notified when the command is received.

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 7.4.3.1.

ReturnCode is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see 7.5) optionally followed by commentary.

7.3.2 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 7.1.1. 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 wild-card convention. A MGC that receives a Notification with wildcard convention **MUST** return an error (error returned **SHOULD** be error code 500 – the transaction could not be executed because the endpoint is unknown) in response.

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. Note that the MG **MAY** include only the domain name of its Notified Entity if only the domain name was received in the triggering NotificationRequest. The MGC **SHOULD** accept the value in this case. 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 (includes notification request embedded in connection handling primitives). 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 7.5) optionally followed by commentary.

7.3.3 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. At a minimum, a Trunking Gateway MUST support one connection per endpoint. 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-wild-carded value to the parameter EndpointId in the function call or it can be under-specified by using the "anyone" wild-card convention. If the endpoint is under-specified, the endpoint identifier will be assigned by the gateway and its complete value MUST be returned in the SpecificEndPointId parameter of the response only if the command is successful. In this case, the endpoint assigned MUST be in service and MUST NOT already have any connections on it. The "all" wild-card convention MUST NOT be used. A gateway that receives a CreateConnection with the "all" wildcard convention MUST return an error (error returned SHOULD be error code 500 – the transaction could not be executed because the endpoint is unknown) in response.

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 at least 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 8.4.1 for details. A list of permissible encoding methods is specified in ITU-T Rec. J.161. The literal names defined in 7.5 (Table 3)/J.161 **MUST** be used. Unknown compression algorithms **SHOULD** be ignored if they are received. See 7.7 for details on the codec selection process.

NOTE – "encoding method" includes audio, image and video encodings.

- **Packetization Period:** A single packetization period in milliseconds **MAY** be specified with exactly one decimal value. If this specifier is used, then the same packetization period **MUST** be used for all encoding methods allowed by the LocalConnectionOptions. Note that if no encoding method field is specified in the LCO, the MG **MUST NOT** choose an encoding method with a packetization period that differs from that specified here. If different packetization periods for different encoding is desired, then this field **MUST NOT** be used. The value pertains to both media sent and received. Note that only the valid packetization period in conjunction with the associated encoding method are to be used by the MG. A list of permissible packetization periods is specified in the IP-Cablecom Audio/Video Codecs Specification (J.161). This specifier **MUST NOT** be supplied in the same LCO as the Multiple Packetization Period field. An MG **MUST** return an error (error code 524 – inconsistency in LocalConnectionOptions) when it receives an LCO with both the Packetization Period and Multiple Packetization Period fields.
- **Multiple Packetization Period:** A list of packetization periods in milliseconds **MAY** be specified if, and only if, the Encoding Method field is included. When specified, the multiple packetization period in milliseconds **MUST** contain exactly one decimal value or a hyphen for each entry in the encoding method field included in the LocalConnectionOptions. This applies even if several of the encoding methods have the same value. The first entry in the list **MUST** be a decimal number. When a hyphen is used, the codec in question **MUST** use the same packetization period as one of the other entries in the list that actually contains a decimal number, and furthermore the codec **MUST NOT** consume any more bandwidth than the other entry. This can for example be used for non-voice codecs (e.g., telephone-event or comfort noise) that use the same packetization period as the voice codec with which they are being used. Successive entries in the list of packetization periods **MUST** be ordered identically to the corresponding encoding methods. The values pertain to both media sent and received. Note that the MG **MUST NOT** choose a codec with a packetization period that differs from that specified here. Note that only the valid packetization period in conjunction with the associated encoding method are to be used by the MG. A list of permissible packetization periods is specified in the IP-Cablecom Audio/Video Codecs Specification (J.161). This specifier **MUST NOT** be supplied in the same LCO as the Packetization Period field. An MG **MUST** return an error (error code 524 – inconsistency in LocalConnectionOptions) under the following conditions:
 - When it receives an LCO with both the Packetization Period and Multiple Packetization Period fields.

- When it receives an LCO where the number of codecs specified in the Encoding Method field is different from the number of elements in the Multiple Packetization Period field.
- **Echo Cancellation:** Whether echo cancellation should be used initially on the line side or not¹¹. 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 media gateway MUST apply echo cancellation initially. The media gateway SHOULD subsequently enable or disable echo cancellation in accordance with ITU-T Rec. V.8 when voiceband data is detected. For re-enabling echo cancellation, see ITU-T Rec. G.168. Following termination of voiceband data, the handling of echo cancellation MUST revert to the current value of the echo cancellation parameter. It is RECOMMENDED that echo cancellation handling is left to the gateway rather than having this parameter specified by the MGC.
- **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 0x00 (unless provisioned otherwise) MUST be used. When this parameter is present and valid, the endpoint MUST use the value provided to populate the differentiated services codepoint (DSCP) parameter in the IP header (see RFC 2474 for more information on the DSCP). The parameter value MUST be 0x00, or MUST be a multiple of four in the range of 0x01 to 0xFF (bits 6 and 7, the ECN bits, are reserved and hence must be set to "00"). An endpoint MUST return an error (error code 532 – Unsupported value(s) in LocalConnectionOptions) when it receives an invalid value. The left-most "bit" in the parameter corresponds to the most significant bit in the IP header.
- **Silence Suppression:** The telephony gateways may perform voice activity detection, and avoid sending packets during periods of silence. However, it is necessary for certain call types (e.g., modem calls) to disable silence suppression. 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 value is "off". If the value is "on", upon detecting voiceband data, the endpoint SHOULD disable silence suppression. Following termination of voiceband data, the handling of silence suppression MUST revert to the current value of the silence suppression parameter.

Additionally, the following LocalConnectionOptions fields are used to support the IPCablecom security services: (Either or both ciphersuites MAY be present):

- **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 according to the rules described in the IPCablecom Security Specification (J.170). The endpoint SHOULD additionally indicate which of the remaining ciphersuites it is willing to support as alternatives (see 8.4.1 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 is specified in ITU-T Rec. J.170. The RTP ciphersuite parameter applies to RTP media streams only. If the MGC includes LocalConnectionOptions that require use of non-RTP media only (as for example T.38 fax relay using UDPTL), the RTP ciphersuite parameter MUST NOT be included. If an LCO allows both RTP and non-RTP media and an RTP ciphersuite parameter is included, it applies to the RTP media only. In all cases, if the resulting media stream on the connection

¹¹ Echo cancellation on the packet side is not supported.

is not an RTP media stream (as for example T.38 fax relay using UDPTL), the RTP ciphersuite parameter MUST be ignored, i.e., RTP security is not being used, and RTP security parameters are not included in LocalConnectionDescriptor.

- **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 according to the rules described in the IPseccom Security Specification (J.170). The endpoint SHOULD additionally indicate which of the remaining ciphersuites it is willing to support as alternatives. See 8.4.1 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 is specified in ITU-T Rec. J.170. The RTCP ciphersuite parameter applies to RTCP for RTP media streams only. If the MGC includes LocalConnectionOptions that require use of non-RTP media only (as for example T.38 fax relay using UDPTL), the RTCP ciphersuite parameter MUST NOT be included. If an LCO allows both RTP and non-RTP media and an RTCP ciphersuite parameter is included, it applies to RTCP for the RTP media only. In all cases, if the resulting media stream on the connection is not an RTP media stream (as for example T.38 fax relay using UDPTL), the RTCP ciphersuite parameter MUST be ignored, i.e., RTCP security is not being used, and RTCP security parameters are not included in LocalConnectionDescriptor.

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.

Furthermore, TGCP supports IPseccom Electronic Surveillance. When a connection is subject to electronic surveillance, all valid media packets received on the connection and all media packets sent on the connection will be replicated and forwarded to an Electronic Surveillance Delivery Function¹² after inclusion of a Call Content Connection Identifier. The replication will follow the connection mode for the connection, except for media generated by signals applied to the connection, which will be replicated regardless of the connection mode. For example, a connection in "inactive" mode will not generate any intercepted media¹³, 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 IPseccom Electronic Surveillance:

- **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 connections 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.

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 8.4 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

¹² Note that the replication occurs at the network level – see PKT-SP-ESP-I01-991229 for details.

¹³ Assuming no media generating signal was applied to the connection.

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.

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", "loopback", "continuity test", "network loopback" or "network continuity test". The handling of these modes is specified in the beginning of 7.3. Note that signals applied to a connection do not follow the connection mode. Some endpoints may not be capable of supporting all modes – see V.1. 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", "send/receive", "network loopback", "network continuity test", or if a signal (as opposed to detecting an event) is to be applied to the connection (error code 527 – missing RemoteConnectionDescriptor is RECOMMENDED).

ConnectionId is a parameter returned by the gateway that uniquely identifies the connection within the context of the endpoint in question. The parameter MUST be included with any provisional or successful response to a CreateConnection command. The parameter MUST NOT be included when any error response is returned and the connection was not created.

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 8.4 details the supported use of SDP in the TGCP profile. The LocalConnectionDescriptor parameter MUST be included with any provisional or successful response to a CreateConnection command. The LocalConnectionDescriptor MUST NOT be included when any error response is returned and the connection was not created.

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, that 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 an embedded 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. Note that absence of the RequestedEvents and SignalRequests parameters is interpreted as an empty list only if a RequestIdentifier parameter is included.

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 set-up 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 7.5) optionally followed by commentary.

7.3.4 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 wild-card convention. A gateway that receives a ModifyConnection with a wildcard convention MUST return an error (error returned SHOULD be error code 500 – the transaction could not be executed because the endpoint is unknown) in response.

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. The **LocalConnectionDescriptor** parameter MUST NOT be included when any error response is returned and the connection was not modified. 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 change¹⁴.

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 MGC¹⁵. 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. ITU-T Rec. J.170 should be consulted for additional security requirements.

The **RequestedEvents**, **RequestIdentifier**, **SignalRequests**, **QuarantineHandling**, and **DetectEvents** parameters are optional. The parameters can be used by the MGC to include an embedded 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. Note that absence of the RequestedEvents and SignalRequests parameters is interpreted as an empty list only if a RequestIdentifier parameter is included.

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.

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

ReturnCode is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see 7.5) optionally followed by commentary.

¹⁴ 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.

¹⁵ For instance if media needs to traverse a firewall.

7.3.5 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. Wild-card conventions MUST NOT be used. A gateway that receives this form of the DeleteConnection with a wildcard convention MUST return an error (error returned SHOULD be error code 500 – the transaction could not be executed because the endpoint is unknown) in response.

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. Also after the connection has been deleted, any loopback that has been requested for the connection MUST be cancelled (unless the endpoint has another connection requesting loopback).

In response to the DeleteConnection command, the gateway returns a list of parameters that describe the status of the connection¹⁶. The connection parameters MUST only be returned if the command is successful and the connection is deleted. 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) – for example, as a result of a Modify command. The value MUST be based on the same information provided via the RTCP mechanism.
- **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 – for example, as a result of a ModifyConnection command. The value MUST be based on the same information provided via the RTCP mechanism.
- **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. All received packets MUST be counted independent of the connection mode or any type of processing error, e.g., authentication failure.
- **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. All received packets MUST be counted independent of the connection mode or any type of processing error, e.g., authentication failure.

¹⁶ The values calculated will not include packets that resulted from Electronic Surveillance.

- **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, for example, no packets were received on the connection.
- **Interarrival jitter:** An estimate of the statistical variance of the RTP data packet interarrival time measured in milliseconds 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. The count includes packets received from different SSRC if the sender used several values. The value is zero if, for example, no packets were received on the connection.
- **Average transmission delay:** An estimate of the network latency, expressed in milliseconds. 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.

For a more detailed definition of these variables, please refer to IETF RFC 1889.

The **RequestedEvents**, **RequestIdentifier**, **SignalRequests**, **QuarantineHandling**, and **DetectEvents** parameters are optional. They can be used by the MGC to transmit an embedded 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. Note that absence of the **RequestedEvents** and **SignalRequests** parameters is interpreted as an empty list only if a **RequestIdentifier** parameter is included.

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.

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

ReturnCode is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see 7.5) optionally followed by commentary.

7.3.6 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. Wild-card conventions MUST NOT be used. A MGC that receives a DeleteConnection with a wildcard convention MUST return an error (error returned SHOULD be error code 500 – the transaction could not be executed because the endpoint is unknown) in response.

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 7.6.

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 7.5) optionally followed by commentary.

7.3.7 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" wild-card. All connections for the endpoint(s) with the CallId specified will be deleted. The command does not return any individual statistics or call parameters. A gateway that receives a DeleteConnection (Multiple Connections From the Media Gateway Controller) with the "any of" wildcard convention MUST return an error (error returned SHOULD be error code 500 – the transaction could not be executed because the endpoint is unknown) in response.

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" wild-carding convention, as specified in 7.1.1. The "any of" wild-carding 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. Also after the connection has been deleted, any loopback that has been requested for the connection **MUST** be cancelled (unless the endpoint has another connection requesting loopback).

ReturnCode is a parameter returned by the gateway. It indicates the outcome of the command and consists of an integer number (see 7.5) optionally followed by commentary.

7.3.8 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 IP-Cablecom 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 this Recommendation.

7.3.8.1 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]
  [, MaxMGCPDatagram]           [, Capabilities] }
  ← AuditEndPoint(EndpointId
    [, RequestedInfo] |
    { [, SpecificEndpointID]
      [, MaxEndpointIDs] } )
```

The **EndpointId** identifies the endpoint that is being audited. The "any of" wild-card convention **MUST NOT** be used. A gateway that receives an AuditEndPoint with the "any of" wildcard convention **MUST** return an error (error returned **SHOULD** be error code 500 – the transaction could not be executed because the endpoint is unknown) in response.

The "all of" wild-card 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 wild-card 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 wild-card 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, MaxMGCPDatagram, and Capabilities.

If an endpoint is queried about a parameter it does not support, the endpoint **MUST NOT** generate an error; instead the parameter **MUST** be omitted from the response.

If an endpoint is queried about a parameter it does support, but has no value for, the endpoint **MUST NOT** generate an error; instead the parameter **MUST** be included in the response with an empty parameter value.

Only when successful, the AuditEndPoint response **MUST**, in turn, include information about each of the items for which auditing information was requested. Note that parameters that are explicitly marked "optional" below and not supported by the endpoint are omitted in the response:

- **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 signals¹⁷. 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. Note that the MG **MAY** include only the domain name of its Notified Entity if only the domain name was provided to it via the Notified Entity parameter of a TGCP message or acknowledgement. The MGC **SHOULD** accept the value in this case.
- **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.

¹⁷ Currently, there should be no pending Brief signals.

- **VersionSupported:** A list of protocol versions supported by the endpoint.
- **MaxMGCPDatagram:** The maximum size of an MGCP datagram in bytes supported by the endpoint (see 8.5.3). The value excludes any lower layer overhead. Support for this parameter is optional. The default maximum MGCP datagram size is assumed if a value is not returned.
- **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, e.g., silence suppression, are only compatible with some codecs, then the gateway will return several capability sets.
 - **Compression Algorithm** A list of supported codecs. The literal names defined in clause 7.5 (Table 3) of the IP-Cablecom Audio/Video Codecs Specification (J.161) MUST be used. Unknown compression algorithms SHOULD be ignored if they are received. 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 not¹⁸.
 - **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.
 - **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 IP-Cablecom Electronic Surveillance is supported or not.

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 7.5) 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.

¹⁸ Currently, all TGCP endpoints must support echo cancellation.