



***Society of Cable
Telecommunications
Engineers***

**ENGINEERING COMMITTEE
Data Standards Subcommittee**

AMERICAN NATIONAL STANDARD

ANSI/SCTE 165-3 2016

**IPCablecom 1.5 Part 3: Network-Based Call Signaling
Protocol**

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards and Operational Practices (hereafter called “documents”) are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability, best practices and ultimately the long term reliability of broadband communications facilities. These documents shall not in any way preclude any member or non-member of SCTE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE members.

SCTE assumes no obligations or liability whatsoever to any party who may adopt the documents. Such adopting party assumes all risks associated with adoption of these documents, and accepts full responsibility for any damage and/or claims arising from the adoption of such documents.

Attention is called to the possibility that implementation of this document may require the use of subject matter covered by patent rights. By publication of this document, no position is taken with respect to the existence or validity of any patent rights in connection therewith. If a patent holder has filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license, then details may be obtained from the standards developer. SCTE shall not be responsible for identifying patents for which a license may be required or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Patent holders who believe that they hold patents which are essential to the implementation of this document have been requested to provide information about those patents and any related licensing terms and conditions. Any such declarations made before or after publication of this document are available on the SCTE web site at <http://www.scte.org>.

All Rights Reserved

© Society of Cable Telecommunications Engineers, Inc. 2016
140 Philips Road
Exton, PA 19341

Note: DOCSIS® is a registered trademark of Cable Television Laboratories, Inc., and is used in this document with permission.

Table of Contents

1	STATUS OF THIS DOCUMENT	7
1.1	Requirements and Conventions	7
1.2	Organization of Document.....	7
2	REFERENCES	9
2.1	Normative References.....	9
2.2	Informative References.....	10
3	TERMS AND DEFINITIONS	11
4	ABBREVIATIONS AND ACRONYMS.....	15
5	SCOPE.....	22
6	INTRODUCTION	23
6.1	Relation with H.323 Standards	24
6.2	Relation with IETF Standards.....	24
6.3	Relation to RFC 3435 and ABNF Grammar.....	25
7	MEDIA GATEWAY CONTROL INTERFACE (MGCI)	26
7.1	Model and Naming Conventions	26
7.1.1	Endpoint Names.....	26
7.1.2	Call Names.....	28
7.1.3	Connection Names	28
7.1.4	Names of Call Agents and Other Entities	28
7.1.5	Digit Maps	28
7.1.6	Events and Signals.....	30
7.2	SDP Use.....	32
7.3	Gateway Control Functions	32
7.3.1	NotificationRequest.....	34
7.3.2	Notifications.....	39
7.3.3	CreateConnection	40
7.3.4	ModifyConnection.....	45
7.3.5	DeleteConnection (From the Call Agent)	47
7.3.6	DeleteConnection (From the Embedded Client).....	49
7.3.7	DeleteConnection (Multiple Connections From the Call Agent).....	49
7.3.8	Auditing	50
7.3.9	Restart in Progress	53
7.4	States, Failover, and Race Conditions	55
7.4.1	Recaps and Highlights	55
7.4.2	Retransmission and Detection of Lost Associations	55
7.4.3	Race Conditions.....	58
7.5	Return Codes and Error Codes	68
7.6	Reason Codes	69
7.7	Use of Local Connection Options and Connection Descriptors	70
7.7.1	RFC 2833 Negotiation.....	72
7.7.2	Selecting Codec Packetization Interval for T.38.....	72
7.7.3	Remote IP and Port Negotiation.....	72
7.7.4	LCO Use with FXR Package.....	74
7.7.5	LCO with FXR and VBD Being Used by CMS at Same Time	74
7.7.6	LCO Format for V.152 and RFC 2198 Redundancy Use	75
8	MEDIA GATEWAY CONTROL PROTOCOL.....	76
8.1	General Description	76

8.2	Command Header	76
8.2.1	Command Line.....	77
8.2.2	Parameter Lines.....	78
8.3	Response Header Formats	89
8.3.1	CreateConnection	91
8.3.2	ModifyConnection.....	91
8.3.3	DeleteConnection	92
8.3.4	NotificationRequest.....	92
8.3.5	Notify	92
8.3.6	AuditEndpoint.....	92
8.3.7	AuditConnection	92
8.3.8	RestartInProgress	93
8.4	Session Description Encoding	93
8.4.1	Use of SDP.....	93
8.4.2	SDP Parameters Common to both Audio (m=audio media line) and Image (m=image media line) Service Use	93
8.4.3	SDP Audio Service Use.....	97
8.4.4	SDP Image Service Use for T.38	100
8.5	Transmission Over UDP	103
8.5.1	Reliable Message Delivery	103
8.5.2	Retransmission Strategy	103
8.5.3	Maximum Datagram Size, Fragmentation, and Reassembly.....	104
8.6	Piggy-Backing	104
8.7	Transaction Identifiers and Three Ways Handshake	105
8.8	Provisional Responses	106
9	SECURITY	108
APPENDIX I	EVENT PACKAGES (INFORMATIVE).....	109
I.1	Analog Access Lines	109
I.2	Line Package.....	109
I.3	FAX Package	114
I.4	VoIP Metrics Package	145
I.5	Vbd and GPMD Packages	158
APPENDIX II	CONNECTION MODE (INFORMATIVE).....	190
APPENDIX III	DYNAMIC QUALITY-OF-SERVICE (INFORMATIVE).....	194
III.1	Introduction	194
III.2	NCS/D-QoS State Machine	195
APPENDIX IV	EXAMPLE COMMAND ENCODINGS (INFORMATIVE).....	201
IV.1	NotificationRequest	201
IV.2	Notify.....	201
IV.3	CreateConnection	201
IV.4	ModifyConnection.....	203
IV.5	DeleteConnection (From the Call Agent)	203
IV.6	DeleteConnection (From the Embedded Client).....	204
IV.7	DeleteConnection (Multiple Connections From the Call Agent).....	204
IV.8	AuditEndpoint.....	204
IV.9	AuditConnection.....	205
IV.10	RestartInProgress	206
APPENDIX V	EXAMPLE CALL FLOW (INFORMATIVE)	207
APPENDIX VI	COMPATIBILITY INFORMATION (INFORMATIVE).....	213
VI.1	MGCP Compatibility	213

APPENDIX VII ABNF GRAMMAR FOR NCS (INFORMATIVE)	214
--	------------

List of Figures

Figure 1 - Relation to H.323 Standards	24
Figure 2 - Relation to IETF Standards	25
Figure 3 - Dialed Digits – Examples	29
Figure 4 - Retransmission Algorithm	57
Figure 5 - Quarantine List Procedures	61
Figure 6 - Control of Media Streams by Connection Mode - Graphical Representation	192
Figure 7 - Graphical Representation of Connections	193
Figure 8 - Individual Media Streams on Connections	193
Figure 9 - A to B to C Interaction	193
Figure 10 - NCS/D-QoS State Diagram (1:2)	197
Figure 11 - NCS/D-QoS State Diagram (2:2)	198

List of Tables

Table 1 - Packages for Embedded Client End-point Types	31
Table 2 - Legal Combinations of Events and Actions	37
Table 3 - Default Resource Reservation Values	43
Table 4 - Return Code Definitions	68
Table 5 - Reason Code Definitions	69
Table 6 - Requested Verb Coding	77
Table 7 - Example Name Coding	78
Table 8 - Parameter Definitions	79
Table 9 - Association of Parameters with Commands	80
Table 10 - DQoS Resource Reservation Parameter Values	82
Table 11 - Connection Parameters	84
Table 12 - Connection Mode	84
Table 13 - Event Name Examples	85
Table 14 - Event Range and Wildcard Notation	85
Table 15 - "All" and "Current" Connection Notation	85
Table 16 - Requested Events Actions	86
Table 17 - RequestedInfo Parameter Values	88
Table 18 - Association of Response Header Parameters and Commands	90
Table 19 - Line Package Codes for Events and Signals	110
Table 20 - Control of Media Streams by Connection Mode	191
Table 21 - Classifiers for Resource Reservation and Commit: Remote Connection Descriptor Provided	194
Table 22 - Classifiers for Resource Reservation and Commit: Remote Connection Descriptor Not Provided	195
Table 23 - States Related to Connection Modes and DQoS Reservation Parameters	196
Table 24 - Example Call Flow	207

This page intentionally left blank.

1 STATUS OF THIS DOCUMENT

This document is considered part of the IPCablecom specification. The document is based on MGCP 1.0 [1], which is an IETF Informational RFC.

1.1 Requirements and Conventions

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"MUST"	This word or the adjective "REQUIRED" means that the item is an absolute requirement of this specification.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"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 behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior 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.

The legal/regulatory classification of IP-based voice communications provided over cable networks and otherwise, and the legal/regulatory obligations, if any, borne by providers of such voice communications, are not yet fully defined by appropriate legal and regulatory authorities. Nothing in this specification is addressed to, or intended to affect, those issues. In particular, while this document uses standard terms such as "call," "call signaling," "telephony," etc., it will be evident from this document that while an IPCablecom network performs activities analogous to these PSTN functions, the manner by which it does so differs considerably from the manner in which they are performed in the PSTN by telecommunications carriers. These differences may be significant for legal/regulatory purposes.

1.2 Organization of Document

The document is structured in the following main sections:

- The introduction presents the basic assumptions and the relation to other protocols such as H.323, RTSP, SAP, and SIP.
- The interface section presents a conceptual overview of the NCS profile, presenting the naming conventions, the usage of the session description protocol (SDP), and the procedures that compose NCS:
 - Notification Request,
 - Notify,
 - Create Connection,
 - Modify Connection,
 - Delete Connection,
 - Audit Endpoint,
 - Audit Connection, and
 - Restart In Progress.
- Each of the commands is presented as an example API with the name of the command, the parameters it can take and return, as well as the semantics of each of these. The actual encoding of the command and its

parameters is shown in the protocol description section. The section concludes with a description of how reliability and race conditions are handled.

- The protocol description section presents the actual NCS encodings of the commands and parameters, which are based on simple text formats. The transmission procedure over UDP is specified as well.
- The security section presents the security of NCS.
- The appendices contain definitions of event packages, mode interactions, example command encodings, and a complete example call flow.

2 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreement based on this standard are encouraged to investigate the possibility of applying the most recent editions of the documents listed below. References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific. For a non-specific reference, the latest version applies.

2.1 Normative References

All references are subject to revision, and users of this document are encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

- [1] IETF RFC 3435, Media Gateway Control Protocol (MGCP) Version 1.0, January 2003.
- [2] IETF RFC 1889, RTP: A Transport Protocol for Real-Time Applications, January 1996.
- [3] IETF RFC 3551, RTP Profile for Audio and Video Conferences with Minimal Control, July 2003.
- [4] IETF RFC 2327, Session Description Protocol (SDP), April 1998.
- [5] IETF RFC 2974, SAP - Session Announcement Protocol, October 2000.
- [6] IETF RFC 2543, SIP: Session Initiation Protocol, March 1999.
- [7] IETF RFC 2326, Real-time Streaming Protocol (RTSP), April 1998.
- [8] TCP/IP Illustrated, Volume 1, *The Protocols*, Stevens, W. Richard, Addison-Wesley, 1994.
- [9] RTP Parameters, <http://www.Iana.org/assignments/rtp-parameters/>.
- [10] Bellcore, Bellcore Notes on the Networks, SR-2275.
- [11] ANSI/SCTE 165-2, IPCablecom 1.5 Part 2: Audio/Video Codecs.
- [12] ANSI/SCTE 165-4, IPCablecom 1.5 Part 4: Dynamic Quality of Service.
- [13] ANSI/SCTE 165-10, IPCablecom 1.5 Part 10: Security.
- [14] IETF RFC 2821/STD0010, J. Klensin, Editor, Simple Mail Transfer Protocol, April 2001.
- [15] IETF RFC 1122/STD0003, Requirements for Internet Hosts -- Communication Layers, October 1989.
- [16] IETF STD0013, Domain Names – Implementations and Specifications, November 1987.
- [17] IETF RFC 2231, MIME Parameter Value and Encoded Word Extensions: Character Sets, November 1997.
- [18] IETF RFC 2234, Augmented BNF for Syntax Specifications: ABNF, November 1997.
- [19] IETF RFC 2833, RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals, May 2000.
- [20] GR-506, LSSGR: Signaling for Analog Interfaces, November 1996.
- [21] ITU-T Recommendation T.30, Procedures for Document Facsimile Transmission in the General Switched Telephone Network, April 1999.
- [22] ITU-T Recommendation V.8, Procedures for Starting Sessions of Data Transmission over the Public Switched Telephone Network, November 2000.
- [23] ITU-T Recommendation V.21, 300 Bits Per Second Duplex Modem Standardized for Use in the General Switched Telephone Network, November 1988.
- [24] ITU-T Recommendation V.25, Automatic Answering Equipment and General Procedures for Automatic Calling Equipment on the General Switched Telephone Network Including Procedures for Disabling of Echo Control Devices for both Manually And Automatically Established Calls, October 1996.
- [25] ITU-T Recommendation V.18, Operational and interworking requirements for DCEs operating in the text telephone mode, November 2000.

- [26] IETF RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, December 1998.
- [27] ITU-T Recommendation T.38, "Procedures for real-time Group 3 facsimile communication over IP networks", March 2002.
- [28] IETF RFC 3611, "RTP Control Protocol Extended Reports (RTCP XR)", November 2003.
- [29] IETF RFC 2198, RTP Payload for Redundant Audio Data, September 1997.
- [30] ITU-T V.152, Procedures for supporting voice-band data over IP networks, January 2005.

2.2 Informative References

The following documents may provide valuable information to the reader but are not required when complying with this standard.

- [31] ITU-T Recommendation Q.761, Functional Description of the IDSN User Part of Signalling System No. 7, (Malaga-Torremolinos, 1984; modified at Helsinki, 1993).
- [32] ITU-T Recommendation Q.762, General Function of messages and Signals of the IDSN User part of Signalling System No. 7, (Malaga-Torremolinos, 1984; modified at Helsinki, 1993).
- [33] ITU-T Recommendation H.323, Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-guaranteed Quality-of-Service, November 2000.
- [34] ITU-T Recommendation H.225, Call Signaling Protocols and Media Stream Packetization for Packet Based Multimedia Communications Systems, November 2000.
- [35] ITU-T Recommendation H.245, Line Transmission of Non-telephone Signals, July 2001.
- [36] IETF RFC 2401, Security Architecture for the Internet Protocol, November 1998.
- [37] IETF RFC 2402, IP Authentication Header, November 1998.
- [38] IETF RFC 2406, Type of Service in the Internet Protocol Suite, November 1998.
- [39] ANSI/SCTE 165-5, IPCablecom 1.5 Part 5: Media Terminal Adapter (MTA) Device Provisioning.
- [40] IETF RFC 3261, SIP: Session Initiation Protocol, June 2002.
- [41] IETF RFC 3264, An Offer/Answer Model with Session Description Protocol (SDP). June 2002.

3 TERMS AND DEFINITIONS

IPCablecom specifications use the following terms:

Access Control	Limiting the flow of information from the resources of a system only to authorized persons, programs, processes, or other system resources on a network.
Active	A service flow is said to be “active” when it is permitted to forward data packets. A service flow must first be admitted before it is active.
Admitted	A service flow is said to be “admitted” when the CMTS has reserved resources (e.g., bandwidth) for it on the DOCSIS™ network.
A-link	A-Links are SS7 links that interconnect STPs and either SSPs or SCPs. ‘A’ stands for “Access.”
Asymmetric Key	An encryption key or a decryption key used in public key cryptography, where encryption and decryption keys are always distinct.
Audio Server	An Audio Server plays informational announcements in IPCablecom network. Media announcements are needed for communications that do not complete and to provide enhanced information services to the user. The component parts of Audio Server services are Media Players and Media Player Controllers.
Authentication	The process of verifying the claimed identity of an entity to another entity.
Authenticity	The ability to ensure that the given information is without modification or forgery and was in fact produced by the entity that claims to have given the information.
Authorization	The act of giving access to a service or device if one has permission to have the access.
Cipher	An algorithm that transforms data between plaintext and ciphertext.
Ciphersuite	A set which must contain both an encryption algorithm and a message authentication algorithm (e.g., a MAC or an HMAC). In general, it may also contain a key-management algorithm, which does not apply in the context of IPCablecom.
Ciphertext	The (encrypted) message output from a cryptographic algorithm that is in a format that is unintelligible.
Cleartext	The original (unencrypted) state of a message or data. Also called plaintext.
Confidentiality	A way to ensure that information is not disclosed to anyone other than the intended parties. Information is encrypted to provide confidentiality. Also known as privacy.
Cryptanalysis	The process of recovering the plaintext of a message or the encryption key without access to the key.
Cryptographic algorithm	An algorithm used to transfer text between plaintext and ciphertext.
Decipherment	A procedure applied to ciphertext to translate it into plaintext.
Decryption	A procedure applied to ciphertext to translate it into plaintext.
Decryption key	The key in the cryptographic algorithm to translate the ciphertext to plaintext.
Digital certificate	A binding between an entity’s public key and one or more attributes relating to its identity, also known as a public key certificate.
Digital signature	A data value generated by a public-key algorithm based on the contents of a block of data and a private key, yielding an individualized cryptographic checksum.
Downstream	The direction from the headend toward the subscriber location.
Encipherment	A method used to translate plaintext into ciphertext.
Encryption	A method used to translate plaintext into ciphertext.
Encryption Key	The key used in a cryptographic algorithm to translate the plaintext to ciphertext.
Endpoint	A Terminal, Gateway or Multipoint Conference Unit (MCU).
Errored Second	Any 1-second interval containing at least one bit error.
Event Message	A message capturing a single portion of a connection.

F-link	F-Links are SS7 links that directly connect two SS7 end points, such as two SSPs. ‘F’ stands for “Fully Associated.”
Flow [DOCSIS Flow]	(a.k.a. DOCSIS-QoS “service flow”) A unidirectional sequence of packets associated with a Service ID (SID) and a QoS. Multiple multimedia streams may be carried in a single DOCSIS Flow.
Flow [IP Flow]	A unidirectional sequence of packets identified by OSI Layer 3 and Layer 4 header information. This information includes source/destination IP addresses, source/destination port numbers, protocol ID. Multiple multimedia streams may be carried in a single IP Flow.
Gateway	Devices bridging between the IPCablecom IP Voice Communication world and the PSTN. Examples are the Media Gateway, which provides the bearer circuit interfaces to the PSTN and transcodes the media stream, and the Signaling Gateway, which sends and receives circuit switched network signaling to the edge of the IPCablecom network.
H.323	An ITU-T recommendation for transmitting and controlling audio and video information. The H.323 recommendation requires the use of the ITU-T H.225 and ITU-T H.245 protocol for communication control between a “gateway” audio/video endpoint and a “gatekeeper” function.
Header	Protocol control information located at the beginning of a protocol data unit.
Integrity	A way to ensure that information is not modified except by those who are authorized to do so.
IntraLATA	Within a Local Access Transport Area.
Jitter	Variability in the delay of a stream of incoming packets making up a flow such as a voice communication.
Kerberos	A secret-key network authentication protocol that uses a choice of cryptographic algorithms for encryption and a centralized key database for authentication.
Key	A mathematical value input into the selected cryptographic algorithm.
Key Exchange	The swapping of public keys between entities to be used to encrypt communication between the entities.
Key Management	The process of distributing shared symmetric keys needed to run a security protocol.
Key Pair	An associated public and private key where the correspondence between the two are mathematically related, but it is computationally infeasible to derive the private key from the public key.
Keying Material	A set of cryptographic keys and their associated parameters, normally associated with a particular run of a security protocol.
Keyspace	The range of all possible values of the key for a particular cryptographic algorithm.
Latency	The time, expressed in quantity of symbols, taken for a signal element to pass through a device.
Link Encryption	Cryptography applied to data as it travels on data links between the network devices.
Network Layer	Layer 3 in the Open System Interconnection (OSI) architecture that provides network information that is independent from the lower layers.
Network Management	The functions related to the management of data across the network.
Network Management OSS	The functions related to the management of data link layer and physical layer resources and their stations across the data network supported by the hybrid fiber/coax system.
Nonce	A random value used only once that is sent in a communications protocol exchange to prevent replay attacks.
Non-Repudiation	The ability to prevent a sender from denying later that he or she sent a message or performed an action.

Off-Net Call	A communication connecting an IPCom subscriber out to a user on the PSTN.
On-Net Call	A communication placed by one customer to another customer entirely on the IPCom Network.
One-way Hash	A hash function that has an insignificant number of collisions upon output.
Plaintext	The original (unencrypted) state of a message or data. Also called cleartext.
Pre-shared Key	A shared secret key passed to both parties in a communication flow, using an unspecified manual or out-of-band mechanism.
Privacy	A way to ensure that information is not disclosed to any one other than the intended parties. Information is usually encrypted to provide confidentiality. Also known as confidentiality.
Private Key	The key used in public key cryptography that belongs to an individual entity and must be kept secret.
Proxy	A facility that indirectly provides some service or acts as a representative in delivering information, thereby eliminating the need for a host to support the service.
Public Key	The key used in public key cryptography that belongs to an individual entity and is distributed publicly. Other entities use this key to encrypt data to be sent to the owner of the key.
Public Key Certificate	A binding between an entity's public key and one or more attributes relating to its identity, also known as a digital certificate.
Public Key Cryptography	A procedure that uses a pair of keys, a public key and a private key, for encryption and decryption, also known as an asymmetric algorithm. A user's public key is publicly available for others to use to send a message to the owner of the key. A user's private key is kept secret and is the only key that can decrypt messages sent encrypted by the user's public key.
Root Private Key	The private signing key of the highest-level Certification Authority. It is normally used to sign public key certificates for lower-level Certification Authorities or other entities.
Root Public Key	The public key of the highest level Certification Authority, normally used to verify digital signatures generated with the corresponding root private key.
Secret Key	The cryptographic key used in a symmetric key algorithm, which results in the secrecy of the encrypted data depending solely upon keeping the key a secret, also known as a symmetric key.
Session Key	A cryptographic key intended to encrypt data for a limited period of time, typically between a pair of entities.
Signed and Sealed	An "envelope" of information which has been signed with a digital signature and sealed using encryption.
Subflow	A unidirectional flow of IP packets characterized by a single source and destination IP address and single source and destination UDP/TCP port.
Symmetric Key	The cryptographic key used in a symmetric key algorithm, which results in the secrecy of the encrypted data depending solely upon keeping the key a secret, also known as a secret key.
Systems Management	Functions in the application layer related to the management of various Open Systems Interconnection (OSI) resources and their status across all layers of the OSI architecture.
Transit Delays	The time difference between the instant at which the first bit of a Protocol Data Unit (PDU) crosses one designated boundary, and the instant at which the last bit of the same PDU crosses a second designated boundary.
Trunk	An analog or digital connection from a circuit switch that carries user media content and may carry voice signaling (M _F , R ₂ , etc.).

Tunnel Mode	An IPsec (ESP or AH) mode that is applied to an IP tunnel, where an outer IP packet header (of an intermediate destination) is added on top of the original, inner IP header. In this case, the ESP or AH transform treats the inner IP header as if it were part of the packet payload. When the packet reaches the intermediate destination, the tunnel terminates and both the outer IP packet header and the IPsec ESP or AH transform are taken out.
Upstream	The direction from the subscriber location toward the headend.
X.509 certificate	A public key certificate specification developed as part of the ITU-T X.500 standards directory.

4 ABBREVIATIONS AND ACRONYMS

IPCablecom specifications use the following abbreviations.

AAA	Authentication, Authorization and Accounting.
AES	Advanced Encryption Standard. A block cipher, used to encrypt the media traffic in IPCablecom.
AF	Assured Forwarding. This is a DiffServ Per Hop Behavior.
AH	Authentication header. An IPsec security protocol that provides message integrity for complete IP packets, including the IP header.
AMA	Automated Message Accounting. A standard form of call detail records (CDRs) developed and administered by Bellcore (now Telcordia Technologies).
ASD	Application-Specific Data. A field in some Kerberos key management messages that carries information specific to the security protocol for which the keys are being negotiated.
AT	Access Tandem.
ATM	Asynchronous Transfer Mode. A protocol for the transmission of a variety of digital signals using uniform 53-byte cells.
BAF	Bellcore AMA Format, also known as AMA.
BCID	Billing Correlation ID.
BPI+	Baseline Privacy Plus Interface Specification. The security portion of the DOCSIS 1.1 standard that runs on the MAC layer.
CA	Certification Authority. A trusted organization that accepts certificate applications from entities, authenticates applications, issues certificates and maintains status information about certificates.
CA	Call Agent. The part of the CMS that maintains the communication state, and controls the line side of the communication.
CBC	Cipher Block Chaining mode. An option in block ciphers that combine (XOR) the previous block of ciphertext with the current block of plaintext before encrypting that block of the message.
CBR	Constant Bit Rate.
CDR	Call Detail Record. A single CDR is generated at the end of each billable activity. A single billable activity may also generate multiple CDRs.
CIC	Circuit Identification Code. In ANSI SS7, a two-octet number that uniquely identifies a DSO circuit within the scope of a single SS7 Point Code.
CID	Circuit ID (Pronounced “kid”). This uniquely identifies an ISUP DS0 circuit on a Media Gateway. It is a combination of the circuit’s SS7 gateway point code and Circuit Identification Code (CIC). The SS7 DPC is associated with the Signaling Gateway that has domain over the circuit in question.
CIF	Common Intermediate Format.
CIR	Committed Information Rate.
CM	DOCSIS Cable Modem.
CMS	Cryptographic Message Syntax.
CMS	Call Management Server. Controls the audio connections. Also called a Call Agent in MGCP/SGCP terminology. This is one example of an Application Server.
CMTS	Cable Modem Termination System. The device at a cable headend which implements the DOCSIS RFI MAC protocol and connects to CMs over an HFC network.
CMSS	Call Management Server Signaling.
Codec	COder-DECoder.

COPS	Common Open Policy Service protocol. Currently an internet draft, which describes a client/server model for supporting policy control over QoS Signaling Protocols and provisioned QoS resource management.
CoS	Class of Service. The type 4 tuple of a DOCSIS configuration file.
CRCX	Create Connection.
CSR	Customer Service Representative.
DA	Directory Assistance.
DE	Default. This is a DiffServ Per Hop Behavior.
DES	Data Encryption Standard.
DF	Delivery Function.
DHCP	Dynamic Host Configuration Protocol.
DHCP-D	DHCP Default. Network Provider DHCP Server.
DNS	Domain Name Service.
DOCSIS®	Data-Over-Cable Service Interface Specifications.
DPC	Destination Point Code. In ANSI SS7, a 3-octet number which uniquely identifies an SS7 Signaling Point, either an SSP, STP, or SCP.
DQoS	Dynamic Quality-of-Service. Assigned on the fly for each communication depending on the QoS requested.
DSA	Dynamic Service Add.
DSC	Dynamic Service Change.
DSCP	DiffServ Code Point. A field in every IP packet that identifies the DiffServ Per Hop Behavior. In IP version 4, the TOS byte is redefined to be the DSCP. In IP version 6, the Traffic Class octet is used as the DSCP.
DTMF	Dual-tone Multi Frequency (tones).
EF	Expedited Forwarding. A DiffServ Per Hop Behavior.
E-MTA	Embedded MTA. A single node that contains both an MTA and a cable modem.
EO	End Office.
ESP	IPsec Encapsulating Security Payload. Protocol that provides both IP packet encryption and optional message integrity, not covering the IP packet header.
ETSI	European Telecommunications Standards Institute.
F-link	F-Links are SS7 links that directly connect two SS7 end points, such as two SSPs. 'F' stands for "Fully Associated."
FEID	Financial Entity ID.
FGD	Feature Group D signaling.
FQDN	Fully Qualified Domain Name. Refer to IETF RFC 2821 for details.
GC	Gate Controller.
GTT	Global Title Translation.
HFC	Hybrid Fiber/Coaxial. An HFC system is a broadband bi-directional shared media transmission system using fiber trunks between the headend and the fiber nodes, and coaxial distribution from the fiber nodes to the customer locations.
HMAC	Hashed Message Authentication Code. A message authentication algorithm, based on either SHA-1 or MD5 hash and defined in IETF RFC 2104.
HTTP	Hypertext Transfer Protocol. Refer to IETF RFC 1945 and RFC 2068.
IANA	Internet Assigned Numbered Authority. See www.ietf.org for details.
IC	Inter-exchange Carrier.

IETF	Internet Engineering Task Force. A body responsible, among other things, for developing standards used on the Internet. See www.ietf.org for details.
IKE	Internet Key Exchange. A key-management mechanism used to negotiate and derive keys for SAs in IPsec.
IKE–	A notation defined to refer to the use of IKE with pre-shared keys for authentication.
IKE+	A notation defined to refer to the use of IKE with X.509 certificates for authentication.
IP	Internet Protocol. An Internet network-layer protocol.
IPsec	Internet Protocol Security. A collection of Internet standards for protecting IP packets with encryption and authentication.
ISDN	Integrated Services Digital Network.
ISTP	Internet Signaling Transport Protocol.
ISUP	ISDN User Part. A protocol within the SS7 suite of protocols that is used for call signaling within an SS7 network.
ITU	International Telecommunication Union.
ITU-T	International Telecommunication Union–Telecommunication Standardization Sector.
IVR	Interactive Voice Response system.
KDC	Key Distribution Center.
LATA	Local Access and Transport Area.
LD	Long Distance.
LIDB	Line Information Database. Contains customer information required for real-time access such as calling card personal identification numbers (PINs) for real-time validation.
LLC	Logical Link Control. The Ethernet Packet header and optional 802.1P tag which may encapsulate an IP packet. A sublayer of the Data Link Layer.
LNP	Local Number Portability. Allows a customer to retain the same number when switching from one local service provider to another.
LSSGR	LATA Switching Systems Generic Requirements.
MAC	Message Authentication Code. A fixed-length data item that is sent together with a message to ensure integrity, also known as a MIC.
MAC	Media Access Control. It is a sublayer of the Data Link Layer. It normally runs directly over the physical layer.
MC	Multipoint Controller.
MCU	Multipoint Conferencing Unit.
MD5	Message Digest 5. A one-way hash algorithm that maps variable length plaintext into fixed-length (16 byte) ciphertext.
MDCP	Media Device Control Protocol. A media gateway control specification submitted to IETF by Lucent. Now called SCTP.
MDCX	Modify Connection.
MDU	Multi-Dwelling Unit. Multiple units within the same physical building. The term is usually associated with high-rise buildings.
MEGACO	Media Gateway Control IETF working group. See www.ietf.org for details.
MF	Multi-Frequency.
MG	Media Gateway. Provides the bearer circuit interfaces to the PSTN and transcodes the media stream.
MGC	Media Gateway Controller. The overall controller function of the PSTN gateway. Receives, controls and mediates call-signaling information between the IPCablecom and PSTN.
MGCP	Media Gateway Control Protocol. Protocol follow-on to SGCP. Refer to IETF 2705.
MIB	Management Information Base.

MIC	Message Integrity Code. A fixed-length data item that is sent together with a message to ensure integrity, also known as a Message Authentication Code (MAC).
MMC	Multi-Point Mixing Controller. A conferencing device for mixing media streams of multiple connections.
MSB	Most Significant Bit.
MSO	Multi-System Operator. A cable company that operates many headend locations in several cities.
MSU	Message Signal Unit.
MTA	Multimedia Terminal Adapter. Contains the interface to a physical voice device, a network interface, CODECs, and all signaling and encapsulation functions required for VoIP transport, class features signaling, and QoS signaling.
MTP	The Message Transfer Part. A set of two protocols (MTP 2, MTP 3) within the SS7 suite of protocols that are used to implement physical, data link, and network-level transport facilities within an SS7 network.
MWD	Maximum Waiting Delay.
NANP	North American Numbering Plan.
NANPNAT	North American Numbering Plan Network Address Translation.
NAT Network Layer	Network Address Translation. Layer 3 in the Open System Interconnection (OSI) architecture. This layer provides services to establish a path between open systems.
NCS	Network Call Signaling.
NPA-NXX	Numbering Plan Area (more commonly known as area code) NXX (sometimes called exchange) represents the next three numbers of a traditional phone number. The N can be any number from 2-9 and the Xs can be any number. The combination of a phone number's NPA-NXX will usually indicate the physical location of the call device. The exceptions include toll-free numbers and ported number (see LNP).
NTP	Network Time Protocol. An internet standard used for synchronizing clocks of elements distributed on an IP network.
NTSC	National Television Standards Committee. Defines the analog color television broadcast standard used today in North America.
OID	Object Identification.
OSP	Operator Service Provider.
OSS	Operations Systems Support. The back-office software used for configuration, performance, fault, accounting, and security management.
OSS-D	OSS Default. Network Provider Provisioning Server.
PAL	Phase Alternate Line. The European color television format that evolved from the American NTSC standard.
PCES	IPCablecom Electronic Surveillance.
PCM	Pulse Code Modulation. A commonly employed algorithm to digitize an analog signal (such as a human voice) into a digital bit stream using simple analog-to-digital conversion techniques.
PDU	Protocol Data Unit.
PHS	Payload Header Suppression. A DOCSIS technique for compressing the Ethernet, IP, and UDP headers of RTP packets.
PKCROSS	Public-Key Cryptography for Cross-Realm Authentication. Utilizes PKINIT for establishing the inter-realm keys and associated inter-realm policies to be applied in issuing cross-realm service tickets between realms and domains in support of Intradomain and Interdomain CMS-to-CMS signaling (CMSS).
PKCS	Public-Key Cryptography Standards. Published by RSA Data Security Inc. These Standards describe how to use public key cryptography in a reliable, secure and interoperable way.

PKI	Public-Key Infrastructure. A process for issuing public key certificates, which includes standards, Certification Authorities, communication between authorities and protocols for managing certification processes.
PKINIT	Public-Key Cryptography for Initial Authentication. The extension to the Kerberos protocol that provides a method for using public-key cryptography during initial authentication.
PSC	Payload Service Class Table, a MIB table that maps RTP payload Type to a Service Class Name.
PSFR	Provisioned Service Flow Reference. An SFR that appears in the DOCSIS configuration file.
PSTN	Public Switched Telephone Network.
QCIF	Quarter Common Intermediate Format.
QoS	Quality of Service. Guarantees network bandwidth and availability for applications.
RADIUS	Remote Authentication Dial-In User Service. An internet protocol (IETF RFC 2865 and RFC 2866) originally designed for allowing users dial-in access to the internet through remote servers. Its flexible design has allowed it to be extended well beyond its original intended use.
RAS	Registration, Admissions and Status. RAS Channel is an unreliable channel used to convey the RAS messages and bandwidth changes between two H.323 entities.
RC4	Rivest Cipher 4. A variable length stream cipher. Optionally used to encrypt the media traffic in IPCablecom.
RFC	Request for Comments. Technical policy documents approved by the IETF which are available on the World Wide Web at http://www.ietf.cnri.reston.va.us/rfc.html .
RFI	The DOCSIS Radio Frequency Interface specification.
RJ-11	Registered Jack-11. A standard 4-pin modular connector commonly used in the United States for connecting a phone unit into a wall jack.
RKS	Record Keeping Server. The device, which collects and correlates the various Event Messages.
RSA	A public-key, or asymmetric, cryptographic algorithm used to provide authentication and encryption services. RSA stands for the three inventors of the algorithm; Rivest, Shamir, Adleman.
RSA Key Pair	A public/private key pair created for use with the RSA cryptographic algorithm.
RSVP	Resource Reservation Protocol.
RTCP	Real-Time Control Protocol.
RTO	Retransmission Timeout.
RTP	Real-time Transport Protocol. A protocol for encapsulating encoded voice and video streams. Refer to IETF RFC 1889.
SA	Security Association. A one-way relationship between sender and receiver offering security services on the communication flow.
SAID	Security Association Identifier. Uniquely identifies SAs in the DOCSIS Baseline Privacy Plus Interface (BPI+) security protocol.
SCCP	Signaling Connection Control Part. A protocol within the SS7 suite of protocols that provides two functions in addition to those provided within MTP. The first function is the ability to address applications within a signaling point. The second function is Global Title Translation.
SCP	Service Control Point. A Signaling Point within the SS7 network, identifiable by a Destination Point Code that provides database services to the network.
SCTP	Stream Control Transmission Protocol.
SDP	Session Description Protocol.
SDU	Service Data Unit. Information delivered as a unit between peer service access points.
SF	Service Flow. A unidirectional flow of packets on the RF interface of a DOCSIS system.

SFID	Service Flow ID. A 32-bit integer assigned by the CMTS to each DOCSIS Service Flow defined within a DOCSIS RF MAC domain. SFIDs are considered to be in either the upstream direction (USFID) or downstream direction (DSFID). Upstream Service Flow IDs and Downstream Service Flow IDs are allocated from the same SFID number space.
SFR	Service Flow Reference. A 16-bit message element used within the DOCSIS TLV parameters of Configuration Files and Dynamic Service messages to temporarily identify a defined Service Flow. The CMTS assigns a permanent SFID to each SFR of a message.
SG	Signaling Gateway. An SG is a signaling agent that receives/sends SCN native signaling at the edge of the IP network. In particular, the SS7 SG function translates variants ISUP and TCAP in an SS7-Internet Gateway to a common version of ISUP and TCAP.
SGCP	Simple Gateway Control Protocol. Earlier draft of MGCP.
SHA – 1	Secure Hash Algorithm 1. A one-way hash algorithm.
SID	Service ID. A 14-bit number assigned by a CMTS to identify an upstream virtual circuit. Each SID separately requests and is granted the right to use upstream bandwidth.
SIP	Session Initiation Protocol. An application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants.
SIP+	Session Initiation Protocol Plus. An extension to SIP.
S-MTA	Standalone MTA. A single node that contains an MTA and a non-DOCSIS MAC (e.g., ethernet).
SNMP	Simple Network Management Protocol.
SOHO	Small Office/Home Office.
SS7	Signaling System number 7. An architecture and set of protocols for performing out-of-band call signaling with a telephone network.
SSP	Service Switching Point. SSPs are points within the SS7 network that terminate SS7 signaling links and also originate, terminate, or tandem switch calls.
STP	Signal Transfer Point. A node within an SS7 network that routes signaling messages based on their destination address. This is essentially a packet switch for SS7. It may also perform additional routing services such as Global Title Translation.
TCAP	Transaction Capabilities Application Protocol. A protocol within the SS7 stack that is used for performing remote database transactions with a Signaling Control Point.
TCP	Transmission Control Protocol.
TD	Timeout for Disconnect.
TFTP	Trivial File Transfer Protocol.
TFTP-D	Default – Trivial File Transfer Protocol.
TGS	Ticket Granting Server. A sub-system of the KDC used to grant Kerberos tickets.
TGW	Telephony Gateway.
TIPHON	Telecommunications and Internet Protocol Harmonization Over Network.
TLV	Type-Length-Value. A tuple within a DOCSIS configuration file.
TN	Telephone Number.
ToD	Time-of-Day Server.
TOS	Type of Service. An 8-bit field of every IP version 4 packet. In a DiffServ domain, the TOS byte is treated as the DiffServ Code Point, or DSCP.
TSG	Trunk Subgroup.
UDP	User Datagram Protocol. A connectionless protocol built upon Internet Protocol (IP).
VAD	Voice Activity Detection.
UDPTL	UDP Transport Layer for fax transmission based on ITU T.38 protocol
VBR	Variable Bit Rate.

VoIP Voice-over-IP.

5 SCOPE

This specification describes a profile of the Media Gateway Control Protocol (MGCP) for IPCablecom embedded clients, which we will refer to as the IPCablecom Network-based Call Signaling (NCS) protocol. MGCP is a call signaling protocol for use in a centralized call control architecture, and assumes relatively simple client devices. The call signaling protocol is one layer of the overall IPCablecom suite of specifications and relies upon companion protocol specifications to provide complete end-to-end IPCablecom functionality. The scope of NCS is currently only embedded Voice-Over-IP client devices in an IPCablecom environment and the NCS profile has therefore simplified and in some cases modified the base MGCP 1.0 protocol accordingly. Support for video will be added in a later version of this document.

This document describes an IPCablecom profile of an application programming interface (MGCI), and a corresponding protocol (MGCP) for controlling voice-over-IP (VoIP) embedded clients from external call control elements. The MGCP assumes a call control architecture where the call control "intelligence" is outside the gateways and is handled by external call control elements.

This document is based on the Media Gateway Control Protocol (MGCP) 1.0 RFC [1], 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, as well as input generated by the IPCablecom embedded client signaling team that developed this profile.

This document, which defines the IPCablecom NCS Protocol specification, 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 document to be as closely aligned with MGCP as possible for the IPCablecom 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 MGCP specification. The NCS profile of MGCP, however, is strictly and solely defined by the contents of this document.

This NCS profile of MGCP, referred to as the Network-based Call Signaling Protocol 1.0, NCS 1.0, the NCS profile, or simply NCS in this document, has been modified from the MGCP 1.0 RFC in the following ways:

- *The NCS protocol only aims at supporting IPCablecom-embedded clients:* The NCS protocol supports embedded clients as defined by IPCablecom. Functionality present in the MGCP 1.0 protocol, which was superfluous to NCS, has been removed.
- *The NCS protocol contains extensions and modifications to MGCP:* IPCablecom-specific requirements have been addressed in NCS, which has resulted in a couple of minor extensions and modifications to MGCP. However, the MGCP architecture, and all of the MGCP constructs relevant to embedded clients, are preserved in NCS.
- *The NCS protocol contains minor simplifications from MGCP 1.0:* Where several choices were available, and not necessarily needed for an embedded client in the IPCablecom environment, some simplifications have been made for embedded-client implementations.

Although MGCP is not NCS, and NCS is not MGCP, the names MGCP and NCS are used interchangeably in this document since this document is based on MGCP. Unless otherwise stated or inferred by context, the word MGCP shall be taken to mean the NCS profile of MGCP in this document.

6 INTRODUCTION

This document describes the NCS profile of an application programming interface (MGCI) and a corresponding protocol (MGCP) for controlling embedded clients from external call control elements. An embedded client is a network element that provides:

- Two or more traditional analog (RJ11) access lines to a voice-over-IP (VoIP) network.
- Optionally, one or more video lines to a VoIP network.

Embedded clients may not be confined to residential use only. For example, they may be used in a business as well. Embedded clients are used for line-side access and, as such, are expected to have line-side equipment, e.g., analog access lines for conventional telephones associated with them, as opposed to trunk gateways.

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 Call Agents. The MGCP assumes that these call-control elements, or Call Agents (CAs), will synchronize with each other to send coherent commands to the gateways under their control. The MGCP defined in this document does not define a mechanism for synchronizing Call Agents, although future IPCablecom specifications 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 an interface on a gateway that terminates an analog POTS connection to a phone, key system, PBX, etc. A gateway that terminates residential POTS lines (to phones) is called a residential gateway, an embedded client or an MTA. Embedded clients may optionally support video as well.

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 NCS 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.

Call Agents instruct the gateways to create connections between endpoints and to detect certain events, e.g., off-hook, and generate certain signals, e.g., ringing. It is strictly up to the Call Agent 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 Call Agent 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 Call Agents 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 Call Agent's instructions as they are received.

6.1 Relation with H.323 Standards

The MGCP is designed as an internal protocol within a distributed system that appears to the outside as a single VoIP gateway. This system is composed of a Call Agent, which may or may not be distributed over several computer platforms, and a set of gateways. In an H.323 configuration, this distributed gateway system may interface on one side with one or more POTS lines, and on the other side with H.323 conformant systems, as illustrated in Figure 1 below:

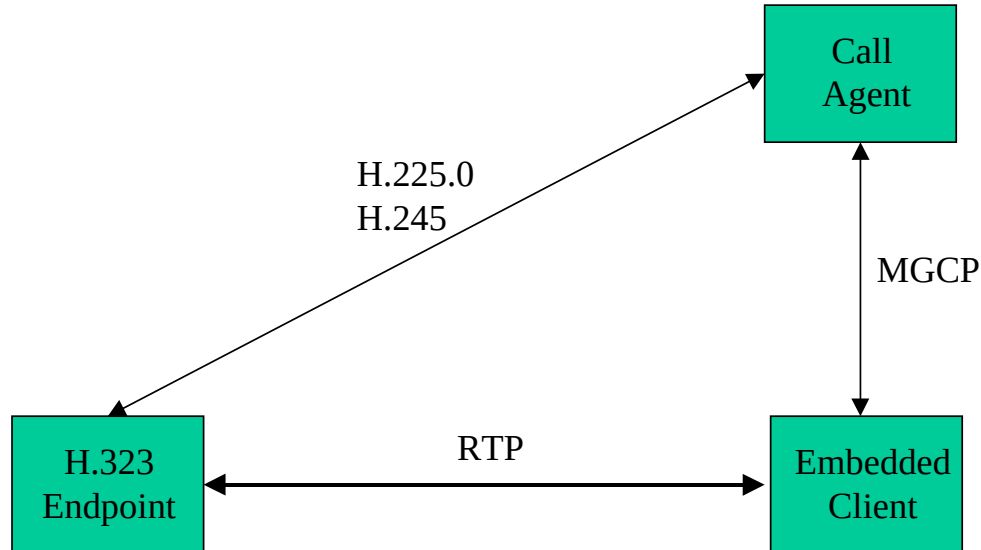


Figure 1 - Relation to H.323 Standards

In the MGCP model, the gateways focus on the audio signal translation function, while the Call Agent handles the signaling and call processing functions. As a consequence, the Call Agent implements the "signaling" layers of the H.323 standard, and presents itself as an "H.323 Gatekeeper" or as one or more "H.323 Endpoints" to the H.323 systems.

6.2 Relation with IETF Standards

While H.323 used to be the recognized standard for VoIP terminals, the IETF also has produced specifications for other types of multi-media applications. These other specifications include:

- the Session Description Protocol (SDP) [4],
- the Session Announcement Protocol (SAP) [5],
- the Session Initiation Protocol (SIP) [40],
- the Real-Time Streaming Protocol (RTSP) [7].

The latter three specifications are, in fact, alternative signaling standards that allow for the transmission of a session description to an interested party. SAP is used by multicast session managers to distribute a multicast session description to a large group of recipients. SIP is used to invite an individual user to take part in a point-to-point or unicast session. RTSP is used to interface a server that provides real-time data. In all three cases, the session description is described according to SDP; when audio is transmitted, it is transmitted through the real-time transport protocol (RTP and RTCP).

The distributed gateway systems and MGCP will enable PSTN voice communication and embedded client users to access sessions set up using SAP, SIP, or RTSP. The Call Agent provides for signaling conversion, as illustrated in Figure 2 below:

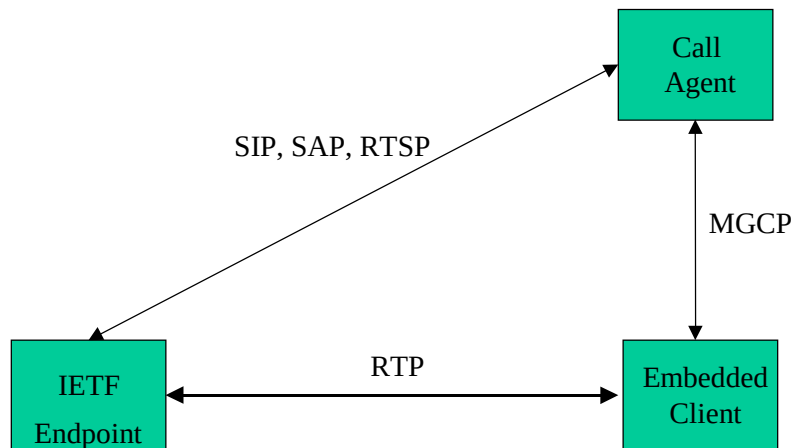


Figure 2 - Relation to IETF Standards

The SDP standard has a pivotal status in this architecture. In the following description it is used to carry session descriptions in MGCP.

6.3 Relation to RFC 3435 and ABNF Grammar

RFC 3435 (see [1]) 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 IPCablecom specifications, is provided in Appendix VII. 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

The MGCP assumes a connection model where the basic constructs are endpoints and connections. Connections are grouped in calls. One or more connections can belong to one call. Connections and calls are set up at the initiative of one or several Call Agents.

7.1.1 Endpoint Names

Endpoint names, also known as (a.k.a.), 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 [16] and includes a host portion, thus an example domain-name could be:

`MyEmbeddedClient.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 [14] for details. However, use of IP addresses is generally discouraged.

Embedded clients may have one or more endpoints (e.g., one for each RJ11 jack for black phones) 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, such as analog phone or video phone. The endpoint-type can be derived from the local endpoint name. The local endpoint name is a hierarchical name, where the least specific component of the name is the leftmost term, and the most specific component is the rightmost term. More formally, the local endpoint name MUST adhere to the following naming rules:

- The individual terms of the local endpoint name must be separated by a single slash ("/", ASCII 2F hex).
- The individual terms are ASCII character strings composed of letters, digits or other printable characters, with the exception of characters used as delimiters in endpoint-names ("/", "@"), characters used for wildcarding ("*", "\$"), and white space characters.

Wild carding is represented either by an asterisk ("*") or a dollar sign ("\$") for the terms of the naming path which are to be wild-carded. Thus, if the full local endpoint name looks like:

`term1/term2/term3`

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

<code>term1/term2/*</code>	if term3 is wild-carded.
<code>term1/*/*</code>	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 embedded client 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 embedded client 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 Embedded Client Endpoint Names

Endpoints in embedded clients MUST support the additional naming conventions specified in this section.

Embedded clients will support the following two endpoint-types:

- **Analog telephone:** The Analog Telephone is represented as an analog access line (aaln). This is basically the equivalent of an analog telephone line as known in the PSTN (see [10]).
- **Video:** The details of the video device-type will be provided in a future version of this document.

7.1.1.1.1 Analog Access Line Endpoints

In addition to the naming conventions specified above, local endpoint names for endpoints of type "analog access line" (aaln) for embedded clients MUST adhere to the following:

- Local endpoint names contain at least one and, at most, two terms.
- Term1 MUST be the term "aaln" or a wildcard character. It should be noted that the use of a wildcard character for term1 can refer to any or all endpoint-types in the embedded client regardless of their type. Use of this feature is generally expected to be for administrative purposes, e.g., auditing or restart.
- Term2 MUST be a number from one to the number of analog access lines supported by the embedded client in question. The number thus identifies a specific analog access line on the embedded client.
- If a local endpoint name is composed of only one term, that term will be term1:
 - If term1 is not a wildcard character, the wildcard character dollar sign (referring to "any one") is then assumed for term2, i.e., "aaln" is equivalent to "aaln/\$".
 - If term1 is a wildcard character, the same wildcard character is then assumed for term2, i.e., "*" and "\$" is equivalent to respectively "*/*" and "\$/\$".

Example analog access line local endpoint names could thus be:

aaln/1	The first analog access line on the embedded client in question.
aaln/2	The second analog access line on the embedded client in question.
aaln/\$	Any analog access line on the embedded client in question.
aaln/*	All analog access lines on the embedded client in question.
*	All endpoints (regardless of endpoint-type) on the embedded client in question.

The provisioning/(auto)configuration process is responsible for obtaining and providing information about how many endpoints an embedded client has, as well as the endpoint-type of each endpoint. Although they are logically different, it should be noted that the *endpoint-type* can be derived from the local portion of the endpoint name.

7.1.1.1.2 Video Endpoints

Details on video endpoints will be provided in a future version of this document.

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

At a minimum, call identifiers MUST be unique within the collection of call agents that control the same gateways. However, the coordination of these call identifiers between Call Agents is outside the scope of this document. When a Call Agent builds several connections that pertain to the same call, either on the same gateway or in different gateways, these connections may 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 Call Agents and Other Entities

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

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

`ca1@ca.whatever.net`

Reliability is provided by the following precautions:

- Entities such as embedded clients or Call Agents are identified by their domain name, not their network addresses. Several addresses can be associated with a domain name. If a command cannot be forwarded to one of the network addresses, implementations MUST retry the transmission using another address.
- Entities may move to another platform. The association between a logical name (domain name) and the actual platform are kept in the Domain Name Service (DNS). Call Agents 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 Call Agent 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 Call Agent, it MUST send the command to the current "notified entity" for which endpoint(s) the command pertains. Upon startup, the "notified entity" MUST be set to a provisioned value. Most commands sent by the Call Agent 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 explicitly¹, 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."

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

7.1.5 Digit Maps

The Call Agent can ask the gateway to collect digits dialed by the user. This facility is intended to be used for analog access lines with residential gateways to collect the numbers that a user dials; it may also be used to collect access codes, credit card numbers, and other numbers requested by call control services. Endpoints MUST support Digit Maps as defined in this section.

¹ This could happen as a result of specifying an empty NotifiedEntity parameter.

An alternative procedure involves the gateway notifying the Call Agent of the dialed digits as soon as they are dialed, a.k.a., overlap sending. However, such a procedure generates a large number of interactions. It is preferable to accumulate the dialed numbers in a buffer, and then to transmit them in a single message.

The problem with this accumulation approach, however, is that it is difficult for the gateway to predict how many numbers it needs to accumulate before transmission. For example, using the phone on our desk, we can dial the numbers illustrated in Figure 3 below:

0	Local operator
00	Long distance operator
xxxx	Local extension number
8xxxxxxx	Local number
#xxxxxxx	Shortcut to local number at other corporate sites
*xx	Star services
91xxxxxxxxxx	Long distance number
9011 + up to 15 digits	International number

Figure 3 - Dialed Digits – Examples

The solution to this problem is to load the gateway with a digit map that corresponds to the dial plan. This digit map is expressed using a syntax derived from the UNIX system command, *egrep*. For example, the dial plan described above results in the following digit map:

```
(0T| 00T|[1-7]xxx|8xxxxxxx|#xxxxxxx|*xx|91xxxxxxxxxx|9011x.T)
```

The formal syntax of the digit map is described by the following BNF notation:

```
Digit      ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
Timer      ::= "T" | "t" -- matches the detection of a timer
Letter     ::= Digit | Timer | "#" | "*" | "A" | "a" | "B" | "b" | "C" | "c" | "D" | "d"
Range      ::= "x" | "X" -- matches any digit
           | "[" Letters "]" -- matches any of the specified letters
Letters    ::= Subrange | Subrange Letters
Subrange   ::= Letter -- matches the specified letter
           | Digit "-" Digit -- matches any digit between first and last
Position   ::= Letter | Range
StringElement ::= Position -- matches an occurrence of the position
           | Position "." -- matches an arbitrary number of occurrences
           -- of the position, including 0
String     ::= StringElement | StringElement String
StringList ::= String | String "|" StringList
DigitMap   ::= String | "(" StringList ")"
```

A DigitMap, according to this syntax, is defined either by a (case insensitive) "string" or by a "list of strings" over which the gateway will attempt to find a shortest possible match. Regardless of the above syntax, a timer is currently only allowed if it appears in the last position in a string². Each string in the list is an alternate numbering scheme. A gateway that detects digits, letters, or timers will:

1. Add the event parameter code for the digit, letter, or timer, as a token to the end of the "current dial string" internal state variable.
2. Apply the "current dial string" to the digit map table, attempting a match to all expressions in the Digit Map.
3. If the result is under-qualified (partially matches at least one entry in the digit map and doesn't completely match another entry), do nothing further.

² For instance, "123T" and "123[1-2T5]" satisfy that rule, but "12T3" does not.

If the result matches an entry, or is over-qualified (i.e., no further digits could possibly produce a match), send the current dial string to the Call Agent³ and clear the "current dial string".. A match, in this specification, can be either a "perfect match," exactly matching one of the specified alternatives, or an "impossible match", which occurs when the dial string does not match any of the alternatives. Unexpected timers, for example, can cause impossible matches. Both perfect matches and impossible matches trigger notification of the accumulated digits (which may include other events).

Timer T is a digit input timer that can be used in two ways:

- When timer T is used with a digit map⁴, the timer is not started until the first digit is entered, and the timer is restarted after each new digit is entered until either a digit map match or mismatch occurs. In this case, timer T functions as an inter-digit timer.
- When timer T is used without a digit map, the timer is started immediately and simply cancelled (but not restarted) as soon as a digit is entered. In this case, timer T can be used as an inter-digit timer when overlap sending is used.

When used with a digit map, timer T takes on one of two values, T_{par} or T_{crit} . When at least one more digit is required for the digit string to match any of the patterns in the digit map, timer T takes on the value T_{par} , corresponding to partial dial timing. If a timer is all that is required to produce a match, timer T takes on the value T_{crit} corresponding to critical timing. When timer T is used without a digit map, timer T takes on the value T_{crit} . The default value for T_{par} is 16 seconds and the default value for T_{crit} is 4 seconds. The provisioning process may alter both of these.

Appendix I contains additional detail and an example on use of timer T.

The end-points MUST support at least 2048 Bytes of digit map on all of the telephony interfaces.

Digit maps can be provided to the gateway by the Call Agent, whenever the Call Agent instructs the gateway to listen for digits. Digit maps, when provided by the Call Agent, MUST be as defined in this section.

7.1.6 Events and Signals

The concept of events and signals is central to MGCP. A Call Agent may ask to be notified about certain events occurring in an endpoint, e.g., off-hook events. A Call Agent also may request certain signals to be applied to an endpoint, e.g., dial-tone.

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 analog access lines, and another package may support another group of events and signals for video lines. 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 analog access line with the line package (package name "L") being the default package, the following two event names are considered equal:

L/dl	dial-tone in the line package for an analog access line.
dl	dial-tone in the line package (default) for an analog access line.

This document defines the packages for embedded client types listed in Table 1.

The CMS MUST support the "L", "FXR" and "XRM" packages. The CMS MAY support the "VBD", "GPMD" and "FM" packages defined in Appendix I.5. The MTA MUST support the "L", "FXR" and "XRM" packages. The MTA MAY support the "VBD", "GPMD" and "FM" packages defined in Appendix I.5.

³ The list of digits may include other events as well. See Section 7.4.3.1.

⁴ Technically speaking with the "accumulate according to digit map" action.

Table 1 - Packages for Embedded Client End-point Types

Endpoint Type	Package	Package Name	Default Package	1st Release that Introduced Package Support Requirement
Analog Access Line	Line	L	Yes	1.0
Analog Access Line	FAX	FXR	No	1.5
Analog Access Line	VoIP Metrics	XRM	No	1.5
Analog Access Line	VoiceBandData	VBD	No	1.5
Analog Access Line	General Purpose Media Descriptor	GPMD	No	1.5
Analog Access Line	Media Format Parameter Package	FM	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 document **MUST** result in a change of the package name, or a change in the NCS 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 dial-tone or DTMF tones. Events must further specify if they are persistent (e.g., off-hook, see Section 7.3.1) and if they contain auditable event-states (e.g., off-hook, see Section 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 Section 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. An embedded client 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 wildcard character "*" (asterisk) can be used to refer to all packages supported by the endpoint in question, and the event code "all" to refer to all events in the package in question. For example:

- L/all refers to all events in the line package for an analog access line.
- */all for an analog access line; 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:

L/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 simply **MUST** stop. Depending on the signal, 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 Notification Request handling (Section 7.3.1).

The wildcard 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:

L/ma@*

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

L/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 wildcard replacement, can be used in conjunction with the "all packages" and "all events" conventions. For example, the notation:

/all@

can be used to designate all events on all connections for the affected endpoint(s). However, the use of the "all packages" and "all events" wildcards is 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 Call Agent 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 specification, SDP descriptions **MUST** follow the conventions delineated in the session description protocol (SDP), which is now an IETF-Proposed Standard documented in [4]. In addition, all call agents and 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 NCS profile will only support the setting of audio and video connections using the media types "audio" and "video". Currently, only "audio" connections have been specified.

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

7.3 Gateway Control Functions

This section 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 section 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 behavior defined by MGCI is an integral part of the specification, and all implementations **MUST** conform to the semantics specified for MGCI. The actual MGCP messages exchanged, including the message formats and encodings used are defined in the protocol section (Section 8). Embedded clients 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 Call Agent can issue a NotificationRequest command to a gateway, instructing the gateway to watch for specific events such as hook actions or DTMF tones on a specified endpoint.
- The gateway will then use the Notify command to inform the Call Agent when the requested events occur on the specified endpoint.

- The Call Agent can use the CreateConnection command to create a connection that terminates in an endpoint inside the gateway.
- The Call Agent can use the ModifyConnection command to change the parameters associated to a previously established connection.
- The Call Agent 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 Call Agent 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 is generally desirable, e.g., information about the status of the embedded client. 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 specification.
- The gateway can use the RestartInProgress command to notify the Call Agent 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 Call Agent) 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, an endpoint is limited to a specific analog access line within an embedded client.

- Connections are grouped into "calls". Several connections, that may or may not belong to the same call, can terminate in the same endpoint. Flow of media on each connection is controlled by a "mode" parameter, which can be set to "send only" (sendonly), "receive only" (recvonly), "send/receive" (sendrecv), "conference" (confnrc), "inactive" (inactive), "replicate" (replcate), "network loopback" (netwloop) or "network continuity test" (netwtest). The "mode" parameter determines if media packets can be sent and/or received on the connection. RTCP is independent of the connection mode; for more details see the IPCablecom Audio/Video Codecs Specification [11].

Handling of media received from the endpoint is determined by the mode parameter:

- Media originating from the endpoint will be sent on all the connections for that endpoint whose mode is either "send only", "send/receive", "conference", or "replicate".

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

- Media received in data packets through connections in "inactive", "sendonly", or "replicate" mode is discarded.
- Media received in data packets through connections in "receive only", "conference", or "send/receive" mode is mixed together and then sent to the endpoint.
- In addition to being sent to the endpoint, media received in data packets through connections in "conference" mode is replicated to all the other connections for the endpoint whose mode is "conference". The details of this forwarding, e.g., RTP translator or mixer, etc., are outside the scope of this document.
- In addition to the media received from the endpoint, media sent to the endpoint is mixed and transmitted over all the other connections for the endpoint whose mode is "replicate". This SHOULD include media generated by signals applied to the endpoint.
- Media received in data packets through connections in "network loopback" or "network continuity test" mode will be sent back on the connection as described below.

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

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

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.

The "replicate" mode MUST at a minimum support replicating the stream from the endpoint and one other connection regardless of the encoding method used for that other connection. The "replicate" connection is however only REQUIRED to support a resulting media stream in G.711 encoding⁵. Support of the "conference" mode is optional; all other connection modes MUST be supported. Please refer to Appendix II 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 Call Agent, may then decide that a different type of encoding should be used on the connections bound to this endpoint and instruct the gateway accordingly⁶.

```
ReturnCode
    ← NotificationRequest(EndpointId
        [, NotifiedEntity]
        [, RequestedEvents]
        [, RequestIdentifier]
        [, DigitMap]
        [, 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 Section 7.1.1. The "any of" wildcard MUST NOT be used. An embedded client that receives a NotificationRequest with the "any of" wildcard convention MUST return an error (the 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, DigitMap, 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 Call Agent name MUST be specified which includes both the local name and domain name - even if a bracketed IP address⁷ is used for the domain name. See Sections 7.1.1 and 7.1.4 for more information. If, however, only the domain name is provided, the MTA 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 signals⁸:

- Ringing,
- Busy tone,
- Call waiting tone,
- Off hook warning tone,
- Ringback tones on a connection.

⁵ The "replicate" connection can, e.g., be used to support "busy line verification" with minimal resource impact on the embedded client.

⁶ The new instruction would be a ModifyConnection command.

⁷ The use of an IP address in the NotifiedEntity is not permitted by the Security specification [13]. When implementing the security specification [13], an absolute domain name (including the hostname) must be used.

⁸ Please refer to Appendix I for a complete list of signals.

Signals are divided into different types depending upon their behavior:

- **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. An On/Off signal could be a visual message waiting indicator (VMWI). Once turned on, an OO signal MUST NOT be turned off until explicitly instructed to by the Call Agent; OO signals will be off following restart of the endpoint. A missing or empty SignalRequest parameter or a SignalRequest parameter that omits a particular signal is not an explicit instruction for an OO signal; it will not change the MTA state for an OO signal.
- **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 Appendix I for further definition of this event). A TO signal could be "ringback" timing out after 180 seconds. If an event occurs prior to the 180 seconds, the signal will, by default, be stopped⁹. If the signal is not stopped, the signal will time out, stop and generate an "operation complete" event, about which the Call Agent may or may not have requested to be notified. If the Call Agent has asked for the "operation complete" event to be notified, the "operation complete" event sent to the Call Agent will include the name(s) of the signal(s) that timed out¹⁰. 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 out¹⁰.
- **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. A brief tone could be a DTMF digit. If the DTMF digit "1" is currently being played, and a signal stopping event occurs, the "1" would finish playing.

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 call-waiting tone is applied to an endpoint involved in an active call, only the party using the endpoint in question will hear the call-waiting tone. However, individual signals may define a different behavior.

When a signal is applied to a connection that has received a RemoteConnectionDescriptor (see Section 7.3.3), the media stream generated by that signal MUST be forwarded on the connection for all modes except "network loopback". 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.

The currently defined signals can be found in Appendix I.

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:

- fax tones,
- modem tones,
- on-hook transition (occurring in classic telephone sets when the user hangs up the handset),

⁹ The "Keep signal(s) active" action may override this behavior.

¹⁰ If parameters were passed to the signal, the parameters will not be reported.

- off-hook transition (occurring in classic telephone sets when the user lifts the handset),
- flash hook (occurring in classic telephone sets when the user briefly presses the hook that holds the handset),
- DTMF digits (or pulse digits).

The currently defined events can be found in Appendix I.

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,
Accumulate according to Digit Map,
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 action¹¹. 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 performed¹². Persistent events are identified as such through their definition – see Appendix I.

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. For example, if "Ignore off-hook" was specified, and a new request without any off-hook instructions were received, the default "Notify off-hook" operation then would be 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.

¹¹ Thus the RequestIdentifier will be the RequestIdentifier of the current NotificationRequest.

¹² 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.

More than one action can be specified for an event, although a given action can not appear more than once for a given event. Table 2 below specifies the legal combinations of actions:

Table 2 - Legal Combinations of Events and Actions

	Notify	Accumulate	Accumulate According to Digit Map	Ignore	Keep Signal(s) Active	Embedded Notification Request	Embedded Modify Connection
Notify	-	-	-	-	✓	✓	✓
Accumulate	-	-	-	-	✓	✓	✓
Accumulate according to digit map	-	-	-	-	✓	-	✓
Ignore	-	-	-	-	✓	-	✓
Keep Signal(s) active	✓	✓	✓	✓	-	✓	✓
Embedded Notification Request	✓	✓	-	-	✓	-	✓
Embedded Modify Connection	✓	✓	✓	✓	✓	✓	-

Note: The "Embedded Notification Request" action can only be combined with the "Notify" action, if the gateway is allowed to issue more than one Notify command per Notification request.

If a client receives a request with an invalid action or illegal combination of actions, it MUST return an error to the Call Agent (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 Call Agent can send a NotificationRequest with an empty RequestedEvents list to the gateway. The Call Agent can do so, for example, to an embedded client when it does not want to collect any more DTMF digits. 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.

DigitMap is an optional parameter that allows the Call Agent to provision the endpoint with a digit map according to which digits will be accumulated when the Call Agent provides a RequestedEvents parameter with the action "accumulate according to digit map" for that endpoint. The digit map provided is persistent and, therefore, need not be provided whenever a request to "accumulate according to digit map" is made, however Call Agents can provide a digit map at any time. A digit map MUST be provided for the endpoint no later than with the first request to "accumulate according to digit map". If the gateway is requested to "accumulate according to digit map" and the gateway currently does not have a digit map for the endpoint in question, the gateway MUST return an error (error code 519 – endpoint does not have a digit map).

Each endpoint has a variable called the "current dial string" in which digits are collected for matching with the digit map, as specified in Section 7.1.5. Whenever a Notify is sent or a NotificationRequest is to be processed, the

"current dial string" is initialized to a null string. The digits to be processed may now either be detected as input, or they may be retrieved from an event input holding area known as the "quarantine buffer". Please see Section 7.4.3.1 for further details.

The signals being applied by the SignalRequests is synchronized with the collection of events specified or implied in the RequestedEvents parameter, except if overridden by the "Keep signal(s) active" action. For example, if the NotificationRequest mandates a "ringing" signal and the event request asks to look for an "off-hook" event, the ringing shall, by default, stop as soon as the gateway detects an off-hook event. If the event request did not ask to look for an "off-hook" event, the ringing would stop anyway since off-hook is a persistent event and therefore implied in the RequestedEvents parameter. 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. In the case of the action "accumulate according to digit map", the default behavior would be to stop all active time-out signals when the first digit¹³ is accumulated—it is irrelevant to this synchronization if the accumulated digit results in a match, mismatch, or partial matching to the digit map.

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, SignalRequests and a new Digit Map as well. The semantics of the embedded NotificationRequest is as if a new NotificationRequest was just received with the same NotifiedEntity, RequestIdentifier, QuarantineHandling and DetectEvents. When the "Embedded NotificationRequest" is activated, the "current dial string" will be cleared; however the list of observed events and the quarantine buffer will be unaffected (if combined with a Notify, the Notify will clear the ObservedEvents list though – see Section 7.4.3.1). Note, that the Embedded NotificationRequest action does not accumulate the triggering event, however it can be combined with the Accumulate action to achieve that. NCS implementations MUST be able to support at least one level of embedding. An embedded NotificationRequest that respects this limitation MUST NOT contain another Embedded NotificationRequest.

The embedded NotificationRequest action allows the Call Agent 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 Call Agent 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 a Call Agent to occasionally turn on all On/Off signals that should be on, and turn off all On/Off signals that should be off.

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

The embedded ModifyConnection action allows the Call Agent 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 command¹⁴. 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 Appendix I 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 Appendix I 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.

¹³ Digit as defined in digit maps, i.e., including asterisk, timer, etc.

¹⁴ Thus, if, e.g., D-QoS is used on the connection, the default D-QoS action will still be taken when the embedded ModifyConnection action is carried out.

Section 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 (e.g., the duration of and frequency of a DTMF digit) is outside the scope of the core NCS Specification. This definition may vary from location to location and, hence, from gateway to gateway. Consequently, the definitions are provided in event packages, which may be provided outside of the core specification. An initial list of event packages can be found in Appendix I.

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 a few exceptions to this rule, e.g., fax and modem tones, which can be detected but can not be signaled. 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 Call Agent 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 the handling options for events in the quarantine buffer (see Section 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 Call Agent. 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. When this parameter is absent, the events that MUST be detected in the quarantine period are those listed in the last received DetectEvents list. In addition, the gateway MUST also detect persistent events and the events specified in the RequestedEvents list, including those for which the "ignore" action is specified. Further explanation of this parameter may be found in Section 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 Section 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 Section 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 wildcard convention. A Call Agent that receives a Notification with wildcard convention MUST return an error (the 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 MTA

MAY include only the absolute domain name (including the hostname) of its NotifiedEntity if only the absolute domain name was received in the triggering NotificationRequest. The CMS 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. 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", "accumulate according to digit map", 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) or accumulated according to digit map (but no match yet), and the final event that triggered the notification or provided a final match in the digit map. It should be noted that digits are added to the list of observed events as they are accumulated, irrespective of whether they are accumulated according to the digit map or not. For example, if a user enters the digits "1234" and some event E is accumulated between the digits "3" and "4" being entered, the list of observed events would be "1, 2, 3, E, 4".

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

7.3.3 CreateConnection

This command is used to create a connection.

```

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

```

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

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

EndpointId is the identifier for the endpoint in the gateway where CreateConnection executes. The EndpointId can be specified fully by assigning a non-wildcarded value to the parameter EndpointId in the function call or it can be under-specified by using the "anyone" wildcard convention. If the endpoint is under-specified, the endpoint identifier will be assigned by the gateway and its complete value MUST be returned in the Specific EndpointId 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" wildcard convention MUST NOT be used. An embedded client that receives a CreateConnection with the "all" wildcard convention MUST return an error (the 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(s) **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. See Section 7.7 for details on the codec selection process. Note "encoding method" includes audio, image and video encodings.
 - The endpoint **MUST** additionally indicate which of the remaining compression algorithms it is willing to support as alternatives - see Section 8.4.1 for details.
 - A list of permissible encoding methods is specified in the IPCablecom Audio/Video Codecs Specification [11]. The literal names defined in Table 4 of the Audio/Video Codes Specification [11] **MUST** be used. Unknown compression algorithms **SHOULD** be ignored if they are received.
- **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 MTA **MUST NOT** choose an encoding method with a packetization period that differs from that specified here. If different packetization periods for different encoding methods 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 or methods are to be used by the MTA. A list of permissible packetization periods is specified in the IPCablecom Audio/Video Codecs Specification [11]. This specifier **MUST NOT** be supplied in the same LCO as the Multiple Packetization Period field. An MTA **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 encoding method 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 encoding method in question **MUST NOT** consume any more bandwidth than that 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 MTA **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 is to be used by the MTA. A list of permissible packetization periods is specified in the IPCablecom Audio/Video Codecs Specification [11]. This specifier **MUST NOT** be supplied in the same LCO as the Packetization Period field. An MTA **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 not¹⁵. 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 embedded client **MUST** apply echo cancellation initially. The embedded client **SHOULD** subsequently enable or disable echo cancellation in accordance with ITU-T recommendation V.8 and V.25 when voiceband data is detected. For re-enabling echo cancellation see e.g., 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 embedded client rather than having this parameter specified by the Call Agent.

¹⁵ Echo cancellation on the packet side is not supported.

- **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.¹⁶ 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 value provided MUST be used by the embedded client to populate the differentiated services codepoint (DSCP) parameter in the IP header (see [26] for more information on the DSCP). The parameter value MUST be 0x00 or MUST be a multiple of four within the range 0x01 to 0xFF (bits 6 and 7, the ECN bits, are reserved and hence must be set to "00"). An MTA 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 embedded client may perform voice activity detection, and avoid sending packets during periods of silence. However, it is necessary, for certain calls (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 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.

The following LocalConnectionOptions fields are used to support Dynamic Quality of Service (D-QoS) – please refer to Appendix III for further details:

- **D-QoS GateID:** The GateID for the gate that has been setup at the CMTS. The GateID is a 32 bit identifier encoded as a string of up to 8 hex characters. This parameter is optional in general, but mandatory when D-QoS resource reservation and/or committal is to be performed. The presence of this parameter implies that D-QoS MUST be performed for this command. The absence of this parameter indicates that D-QoS MUST NOT be performed.
- **D-QoS Resource Reservation:** Allows explicit control over whether D-QoS resource reservation and/or committal should be performed in the send and/or receive direction or not. The parameter is optional and can have one or more of the following values:

Reserve values:

"Send Reserve"	Resources are reserved in the send direction only.
"Receive Reserve"	Resources are reserved in the receive direction only.
"SendReceiveReserve"	Resources are reserved in the send and receive direction.

Commit values:

"SendCommit"	Resources are committed in the send direction only.
"ReceiveCommit"	Resources are committed in the receive direction.
"SendReceiveCommit"	Resources are committed in the send and receive direction.

The parameter is optional, and multiple values are separated by commas. When D-QoS is to be performed, and the parameter is omitted, resource reservation MUST be performed for both the send and receive direction. The resources reserved are determined by the coding parameters applied to the connection, i.e., encoding method, packetization period, silence suppression, ciphersuite, etc. External parameters, such as the use of payload header suppression, may affect the amount of resources reserved as well. Please see the IPCablecom Dynamic Quality of Service Specification [12] for details.

Receive resources can be reserved and committed without having obtained a RemoteConnectionDescriptor, whereas send resources can be reserved, but not committed, until a RemoteConnectionDescriptor is supplied. Note that, as long as a RemoteConnectionDescriptor has not been received, the resources reserved and committed must be based on the codec(s) selected locally. Once a RemoteConnectionDescriptor is received, the list of codec(s) that can actually be used for sending may contain a subset of these. The list of codecs that can be used for receiving is however unchanged until the endpoint issues a new LocalConnectionDescriptor. When D-QoS reservation is to be performed, and the parameter is omitted, resources MUST by default be committed based on the connection mode as specified in Table 3 below:

¹⁶ For RTP media streams, this type of service value applies only to the RTP media packets, not the RTCP packets.

Table 3 - Default Resource Reservation Values

Connection Mode	D-QoS
"inactive"	Do not commit
"send only", "replicate"	Commit send
"receive only"	Commit receive
"send/receive", "conference", "network loopback", "network continuity test"	Commit send and receive

If a different commit operation is desired, the appropriate commit value is supplied and will be used instead. If a commit operation is to be performed, but no reservation has been made, or an existing reservation does not fully satisfy the resources to be committed¹⁷, a reservation will be made automatically. If a reserve value is specified, but no commit value is specified, a commit operation will not be performed.

- **ResourceID:** An existing ResourceID for resources already reserved at the edge router. The use of the ResourceID allows separate reservations to reserve the same resource, however only one of the reservations can be active at a given point in time. The ResourceID is a 32-bit identifier encoded as a string of up to 8 hex characters. The parameter is optional. However, this parameter **MUST** be used for resource reservation by the embedded client if provided by the Call Agent.
- **ReserveDestination:** This optional parameter may specify an IPv4 address, optionally followed by a colon and a UDP port number, that is the destination for the resource reservation. When a UDP port number is not specified, a default value of 9 applies. The ReserveDestination is typically used when resource reservation is to be performed, and a RemoteConnectionDescriptor has not yet been provided for the connection. This enables reservations and downstream commits to be sent to the edge router when the source of a media stream is not yet known¹⁸. When a RemoteConnectionDescriptor has been provided, the parameter is ignored.

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

- **RTP ciphersuite:** A list of allowable 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 [13]. The endpoint **SHOULD** additionally indicate which of the remaining ciphersuites it is willing to support as alternatives (see Section 8.4.1 for details). Each ciphersuite is represented as ASCII strings consisting of two substrings separated by a slash ("/"), where the first substring identifies the authentication algorithm, and the second substring identifies the encryption algorithm. A list of permissible ciphersuites are specified in the IPCablecom Security Specification [13]. The RTP ciphersuite parameter applies to RTP media streams only. If the CMS 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 (local connection options) 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 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 IPCablecom Security Specification [13]. The endpoint **SHOULD** additionally indicate which of the remaining ciphersuites it is willing to support as alternatives. (See Section 8.4.1 for details). Each ciphersuite is represented as an ASCII string consisting of two 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 the IPCablecom Security Specification [13]. The RTCP ciphersuite parameter applies to RTCP for RTP media streams only. If the CMS 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

¹⁷ This is not possible for the CreateConnection command but is noted here for completeness. It is possible for the ModifyConnection command however (see Section 7.3.4).

¹⁸ Note that this will enable certain theft-of-service scenarios. See IPCablecom Dynamic Quality of Service Specification [12] for details.

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 embedded client **MUST** respond with an error (error code 524 – **LocalConnectionOptions** inconsistency) if any of the above rules are violated. All of the above mentioned default values can be altered by the provisioning process.

RemoteConnectionDescriptor is the connection descriptor for the remote side of a connection, on the other side of the IP network. It includes the same fields as the LocalConnectionDescriptor (not to be confused with LocalConnectionOptions), i.e., the fields that describe a session according to the SDP standard. Section 8.4 details the supported use of SDP in the NCS profile. This parameter may have a null value when the information for the remote end is not known. This occurs because the entity that builds a connection starts by sending a CreateConnection to one of the two gateways involved. For the first CreateConnection issued, there is no information available about the other side of the connection. This information may be provided later via a ModifyConnection call.

When codecs are changed during a call, small periods of time may exist where the endpoints use different codes. As stated above, embedded clients **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", "conference", "inactive", "replicate", "network loopback" or "network continuity test". The handling of these modes is specified in the beginning of Section 7.3. Some endpoints may not be capable of supporting all modes. 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", "replicate", or "conference", "netwloop" or "netwtest" (error code 527 – missing **RemoteConnectionDescriptor**).

ConnectionId is a parameter returned by the gateway that uniquely identifies the connection within the context of the endpoint in question. The ConnectionId **MUST** be included with any provisional or successful response to a CreateConnection command. The ConnectionId **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. Section 8.4 details the supported use of SDP in the NCS profile. The LocalConnectionDescriptor **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 Call Agent. Thus, it must navigate between two risks, i.e., clipping some important announcements or listening to insane data. The behavior 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", the gateway **MUST** (as always) discard the voice signals received on the connection.

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", "send/receive", "replicate", or "conference", "netwloop" or "netwtest".

The **RequestedEvents**, **RequestIdentifier**, **DigitMap**, **SignalRequests**, **QuarantineHandling**, and **DetectEvents** parameters are all optional. They can be used by the Call Agent to effectively include a notification request that is executed simultaneously with the creation of the connection. If one or more of these parameters is present, the RequestIdentifier **MUST** be one of them. Thus, the inclusion of a notification request can be recognized by the

presence of a RequestIdentifier. The rest of the parameters may or may not be present. If one of the parameters is not present, it **MUST** be treated as if it was a normal **NotificationRequest** with the parameter in question being omitted. This may have the effect of canceling signals and of stop looking for events. Note that if the RequestedEvents and SignalRequests parameters are omitted, then the corresponding lists are considered empty only if a RequestIdentifier parameter is included.

As an example of use, consider a Call Agent that wants to place a call to an embedded client. The Call Agent should:

- Ask the embedded client to create a connection, in order to be sure that the user can start speaking as soon as the phone goes off hook,
- Ask the embedded client to start ringing,
- Ask the embedded client to notify the Call Agent when the phone goes off-hook.

All of the above can be accomplished in a single CreateConnection command by including a notification request with the RequestedEvents parameters for the off-hook event and the SignalRequests parameter for the ringing 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 off-hook notification request must be refused in the glare condition if the user is already off-hook. In this example, the phone must not ring if the connection cannot be established, and the connection must not be established if the user is already off-hook. An error would be returned instead (error code 401 – phone off hook), which informs the Call Agent 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 Section 7.5) optionally followed by commentary.

ResourceID is a D-QoS parameter that **MAY** be returned by the gateway. Note that this parameter **MUST** be returned by the MTA when D-QoS is to be performed as instructed by the Call Agent. When a successful D-QoS resource reservation is made, the ResourceID provides a handle for the resources reserved. The ResourceID **MUST NOT** be included when any error response is returned and the connection was not created.

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]
[, ResourceID]
    ← ModifyConnection(CallId
                        , EndpointId
                        , ConnectionId
                        [, NotifiedEntity]
                        [, LocalConnectionOptions]
                        [, Mode]
                        [, RemoteConnectionDescriptor]
                        [, RequestedEvents]
                        [, RequestIdentifier]
                        [, DigitMap]
                        [, SignalRequests]
                        [, QuarantineHandling]
                        [, DetectEvents])

```

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

The **EndpointId** **MUST** be a fully qualified endpoint name. The local name **MUST NOT** use the wildcard convention. An embedded client that receives a ModifyConnection with a wildcard convention **MUST** return an

error (the 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 details of D-QoS operation were specified in the CreateConnection command and generally the same rules apply here, except as noted below:

- **D-QoS GateID** A D-QoS GateID is mandatory when D-QoS operation is required, unless D-QoS operation has previously been done for the connection in question. In the latter case, the previously supplied D-QoS GateID MUST then be used by the MTA.
- **D-QoS Resource Reservation** Allows explicit control over whether D-QoS resource reservation and/or committal should be performed in the send and/or receive direction or not. The parameter is optional and multiple values can be specified. When the parameter is omitted and D-QoS reservation is to be performed, the default is to reserve in both the send and receive direction, unless a suitable reservation for the connection has already been made (see Appendix III). In that case, a new reservation will not be made. Resources are committed the same way as for **CreateConnection**, except when changing to "inactive" mode. In that case, the committed resources MUST be lowered to zero. An existing resource reservation is still maintained though.
- **ResourceID** The parameter is optional. When supplied, it MUST be used by the embedded client for resource reservation and replaces the ResourceID kept for the connection.
- **ReserveDestination** The parameter is optional. When supplied, it replaces the ReserveDestination kept by the embedded client for the connection. If a RemoteConnectionDescriptor has been supplied for the connection, the parameter is ignored.

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 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 change¹⁹.

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 Call Agent²⁰. When RTP address information is given to an embedded client for a connection, the embedded client SHOULD only accept media streams (and RTCP) from the IP address specified as well. Any media streams received from any other addresses SHOULD be discarded. The IPCablecom Security Specification [13] should be consulted for additional security requirements.

¹⁹ 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.

²⁰ For instance if media needs to traverse a firewall.

The **RequestedEvents**, **RequestIdentifier**, **DigitMap**, **SignalRequests**, **QuarantineHandling**, and **DetectEvents** parameters are optional. The parameters can be used by the Call Agent to include a notification request that is tied to and executed simultaneously with the connection modification. If one or more of these parameters is supplied, then **RequestIdentifier** MUST be one of them. For example, when a call is accepted, the calling gateway should be instructed to place the connection in "send/receive" mode and to stop providing ringback tones. This can be accomplished in a single **ModifyConnection** command by including a notification request with the **RequestedEvents** parameters for the on-hook event, and an empty **SignalRequests** parameter, to stop the provision of ringback tones. 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 Section 7.5) optionally followed by commentary.

ResourceID is a D-QoS parameter that MUST be returned by the gateway if it performs a resource reservation and obtains a new **ResourceID** from the edge router. When a successful D-QoS resource reservation is made, the **ResourceID** provides a handle for the resources reserved. The **Resource ID** MUST NOT be included when any error response is returned and the connection was not modified.

7.3.5 DeleteConnection (From the Call Agent)

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]
    [, DigitMap]
    [, SignalRequests]
    [, QuarantineHandling]
    [, DetectEvents])

```

The endpoint identifier, in this form of the **DeleteConnection** command, MUST be fully qualified. Wildcard conventions MUST NOT be used. An embedded client that receives a Delete Connection with a wildcard convention MUST return an error (the 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. When one or more D-QoS reservations and/or committals have been made for the connection, the **DeleteConnection** command will release the resources reserved.

In response to the **DeleteConnection** command, the gateway returns a list of parameters that describe the status of the connection. 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 octets **MUST** be counted independent of the connection mode or any type of processing error, e.g., authentication failure.
- **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 [2]. The count includes packets received from different SSRC if the sender used several values. The value is zero if, e.g., 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. Embedded client 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 [2].

In addition to the parameters above, an endpoint that has received one or more RTCP sender or receiver reports from its peer **MUST** return the following parameters:

Remote Packets Sent:	The number of packets that were sent on the connection from the perspective of the remote endpoint.
Remote Octets Sent:	The number of octets that were sent on the connection from the perspective of the remote endpoint.
Remote Packets Lost:	The number of packets that were not received on the connection, as deduced from gaps in the sequence number from the perspective of the remote endpoint.
Remote Jitter:	The average inter-packet arrival jitter, in milliseconds, expressed as an integer number from the perspective of the remote endpoint.

The **RequestedEvents**, **RequestIdentifier**, **DigitMap**, **SignalRequests**, **QuarantineHandling**, and **DetectEvents** parameters are optional. They can be used by the Call Agent to transmit a notification request that is tied to and executed simultaneously with the deletion of the connection. However, if one or more of these parameters are present, **RequestIdentifier** **MUST** be one of them. For example, when a user hangs up the phone, the gateway might be instructed to delete the connection and to start looking for an off-hook event. This can be accomplished in a single **DeleteConnection** command also by transmitting the **RequestedEvents** parameter for the off-hook 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 Section 7.5) optionally followed by commentary.

7.3.6 DeleteConnection (From the Embedded Client)

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

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

The **EndpointId**, in this form of the **DeleteConnection** command, **MUST** be fully qualified. Wildcard conventions **MUST NOT** be used. A Call Agent that receives a DeleteConnection with a wildcard convention **MUST** return an error (the 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 Section 7.6.

In addition to the **CallId**, **EndpointId**, and **ConnectionId**, the embedded client will also send the connection's parameters, which would have been returned to the Call Agent in response to a DeleteConnection command from the Call Agent. The reason code indicates the cause of the DeleteConnection. When one or more D-QoS reservations and/or committals have been made for the connection, the embedded client will release the resources reserved.

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

7.3.7 DeleteConnection (Multiple Connections From the Call Agent)

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

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

The **EndpointId**, in this form of the **DeleteConnection** command, **MUST NOT** use the "any of" wildcard. All connections for the endpoint(s) with the CallId specified will be deleted. The command does not return any individual statistics or call parameters. An embedded client that receives a **DeleteConnection** (Multiple Connections From the Call Agent) with the "any of" wildcard convention **MUST** return an error (the 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 Call Agent to delete all connections that terminate in a given endpoint:

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

In this form of the **DeleteConnection** command, Call Agents can take advantage of the hierarchical naming structure of endpoints to delete all the connections that belong to a group of endpoints. In this case, part of the "local endpoint name" component of the **EndpointId** can be specified using the "all" wildcarding convention, as specified in Section 7.1.1. The "any of" wildcarding convention **MUST NOT** be used. The command does not return any individual statistics or call parameters.

After the connection has been deleted, packet network media streams previously supported by the connection are no longer available. Any media packets received for the old connection are simply discarded and no new media packets for the stream are sent. When one or more D-QoS reservations and/or committals have been made for the connection, the embedded client will release the resources reserved.

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

7.3.8 Auditing

The MGCP is based upon a centralized call control architecture where a Call Agent acts as the remote controller of client devices that provide voice 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 embedded clients and the Call Agents in an IPCablecom system is provided to allow the Call Agent to audit endpoint and connection state and to retrieve protocol-specific capabilities of an endpoint.

Two commands for auditing are defined for the embedded clients:

- **AuditEndPoint:** Used by the Call Agent to determine the status of an endpoint.
- **AuditConnection:** Used by the Call Agent 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 embedded client 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 embedded client, both of which are outside the scope of this specification.

7.3.8.1 AuditEndPoint

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

```
{ ReturnCode
  [, EndPointIdList]
  [, NumEndPoints] } |
{ ReturnCode
  [, RequestedEvents]
  [, DigitMap]
  [, SignalRequests]
  [, RequestIdentifier]
  [, NotifiedEntity]
  [, ConnectionIdentifiers]
  [, DetectEvents]
  [, ObservedEvents]
  [, EventStates]
  [, VersionSupported]
  [, ReasonCode]
  [, MaxMGCPDatagram]
  [, Capabilities] }
← AuditEndPoint(EndpointId
  [, RequestedInfo] |
  { [, SpecificEndPointID]
  [, MaxEndPointIDs] })
```

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

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