INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.225.0

(02/98)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Transmission multiplexing and synchronization

---

Call signalling protocols and media stream packetization for packet-based multimedia communication systems

ITU-T Recommendation H.225.0

(Previously CCITT Recommendation)

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Characteristics of transmission channels used for other than telephone purposes	H.10–H.19
Use of telephone-type circuits for voice-frequency telegraphy	H.20–H.29
Telephone circuits or cables used for various types of telegraph transmission or simultaneous transmission	H.30–H.39
Telephone-type circuits used for facsimile telegraphy	H.40–H.49
Characteristics of data signals	H.50–H.99
CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	**H.220–H.229**
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.399
Supplementary services for multimedia	H.450–H.499

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION H.225.0

CALL SIGNALLING PROTOCOLS AND MEDIA STREAM PACKETIZATION FOR PACKET-BASED MULTIMEDIA COMMUNICATION SYSTEMS

Summary

This Recommendation covers the technical requirements for narrow-band visual telephone services defined in H.200/AV.120-series Recommendations, in those situations where the transmission path includes one or more packet-based networks, each of which is configured and managed to provide a non-guaranteed Quality of Service (QOS) which is not equivalent to that of N-ISDN such that additional protection or recovery mechanisms beyond those mandated by Recommendation H.320 need be provided in the terminals. It is noted that Recommendation H.322 addresses the use of some other LANs which are able to provide the underlying performance not assumed by the H.323/H.225.0 Recommendations.

This Recommendation describes how audio, video, data, and control information on a packet-based network can be managed to provide conversational services in H.323 equipment.

Source

ITU-T Recommendation H.225.0 was revised by ITU-T Study Group 16 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 6th of February 1998.

FOREWORD

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

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

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

NOTE

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

INTELLECTUAL PROPERTY RIGHTS

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

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

CONTENTS

Recommendation H.225.0

CALL SIGNALLING PROTOCOLS AND MEDIA STREAM PACKETIZATION FOR PACKET-BASED MULTIMEDIA COMMUNICATION SYSTEMS

(revised in 1998)

The ITU-T,

considering

the widespread adoption of and the increasing use of Recommendation H.320 for videophony and videoconferencing services over networks conforming to the N-ISDN characteristics specified in the I-series Recommendations,

appreciating

the desirability and benefits of enabling the above services to be carried, wholly or in part, over Local Area Networks while also maintaining the capability of interworking with H.320 terminals,

and noting

the characteristics and performances of the many types of Local Area Network which are of potential interest,

recommends

that systems and equipment meeting the requirements of Recommendations H.322 or H.323 are utilized to provide these facilities.

1 Scope

This Recommendation describes the means by which audio, video, data, and control are associated, coded, and packetized for transport between H.323 equipment on a packet-based network. This includes the use of an H.323 gateway, which in turn may be connected to H.320, H.324, or H.310/H.321 terminals on N-ISDN, GSTN, or B-ISDN respectively. The equipment descriptions and procedures are described in Recommendation H.323 while this Recommendation covers protocols and message formats. Communication via an H.323 gateway to an H.322 gateway for guaranteed Quality of Service (QOS) LANs and thus to H.322 endpoints is also possible.

This Recommendation is intended to operate over a variety of different packet-based networks, including IEEE 802.3, Token Ring, etc. Thus, this Recommendation is defined as being above the Transport layer such as TCP/IP/UDP, SPX/IPX, etc. Specific profiles for particular transport protocol suites are included in Appendix IV. ***Thus, the scope of H.225.0 communication is between H.323 entities on the same packet-based network, using the same transport protocol.*** This packet-based network may be a single segment or ring, or it logically could be an enterprise data network comprising multiple packet-based networks bridged or routed to create one interconnected network. It should be emphasized that operation of H.323 terminals over the entire Internet, or even several connected packet-based networks may result in poor performance. The possible means by which quality of service might be assured on this packet-based network, or on the Internet in general is beyond the scope of this Recommendation. However, this Recommendation provides a means for the user of H.323 equipment to determine that quality problems are the result of packet-based network congestion, as well as procedures for corrective actions. It is also noted that the use of multiple

H.323 gateways connected over the public ISDN network is a straightforward method for increasing quality of service.

Recommendation H.323 and this Recommendation are intended to extend Recommendations H.320 and H.221 connections onto the non-guaranteed QOS packet-based network environment conferences. As such the primary conference model[1] is one with size in the range of a few participants to a few thousand, as opposed to large-scale broadcast operations, with strong admission control, and tight conference control.

This Recommendation makes use of (RTP/RTCP) Real-time Transport Protocol/Real-Time Transport Control Protocol for media stream packetization and synchronization for all underlying packet-based networks (see Annexes A, B, and C). Please note that the usage of RTP/RTCP as specified in this Recommendation is not tied in any way to the usage of TCP/IP/UDP. This Recommendation assumes a call model where initial signalling on a non-RTP transport address is used for call establishment and capability negotiation (see Recommendations H.323 and H.245), followed by the establishment of one or more RTP/RTCP connections. This Recommendation contains details on the usage of RTP/RTCP.

In Recommendation H.221, audio, video, data, and control are multiplexed into one or more synchronized physical SCN calls. On the packet-based network side of an H.323 call, none of these concepts apply. There is no need to carry from the SCN side the H.221 concept of a P*64 kbit/s call, e.g. 2 by 64 kbit/s, 3 by 64 kbit/s, etc. Thus, on the packet-based network side, for example, there are only single "connection" calls with a maximum rate limited to 128 kbit/s, not 2*64 kbit/s fixed rate calls. Another example has single "connection" packet-based network calls with a maximum rate limited to 384 kbit/s interworking with 6*64 kbit/s on the WAN side[2]. The primary rationale of this approach is to put complexity in the gateway rather than the terminal and to avoid extending onto the packet-based network features of H.320 that are tightly tied to ISDN unless this is necessary.

In general, H.323 terminals are not aware directly of the H.320 transfer rate while interworking through an H.323 gateway; instead, the gateway uses H.245 **FlowControlCommand** messages to limit the media rate on each logical channel in use to that allowed by the H.221 multiplex. The gateway may allow the packet-based network side video rates to substantially underrun the WAN side rates (or the reverse) though the usage of a rate reducing function and H.261 fill frames; the details of such operations are beyond the scope of Recommendation H.323 and this Recommendation. Note that the H.323 terminal is indirectly aware of the H.320 transfer rates via the video maximum bit rate fields in Recommendation H.245 and shall not transmit at rates that exceed these rates.

This Recommendation is designed so that, with an H.323 gateway, interoperability with H.320 (1990), H.320 (1993), and H.320 (1996) terminals is possible. However, some features of this Recommendation may be directed toward allowing enhanced operations with future versions of

---

[1] An optional broadcast only conference model is under consideration; of necessity the broadcast model does not provide tight admissions or conference control.

[2] Note that video and data rates on the LAN side must match the video and data rates in the SCN side H.320 multiplex; the audio and control rates are not required to match. Stated another way one would normally expect that, using H.245 flow control, the LAN/SCN gateway will force the video and data rates to fit into the H.221 SCN multiplex. However, since audio may be transcoded in the gateway often, one will frequently find that the LAN audio rate and the SCN rate do not match. Also there should be no expectation that the H.221 bit rate for control (800 bit/s) will generally match the H.245 bit rate on the LAN side. Also note that the LAN rate may under-run the SCN rate for either/both video or/and data, but it cannot exceed the maximum amount that fits into the SCN side multiplex.

Recommendation H.320. It is also possible that the quality of service on the H.320 side may vary based on the features and capabilities of the H.323 gateway (see Figure 1).

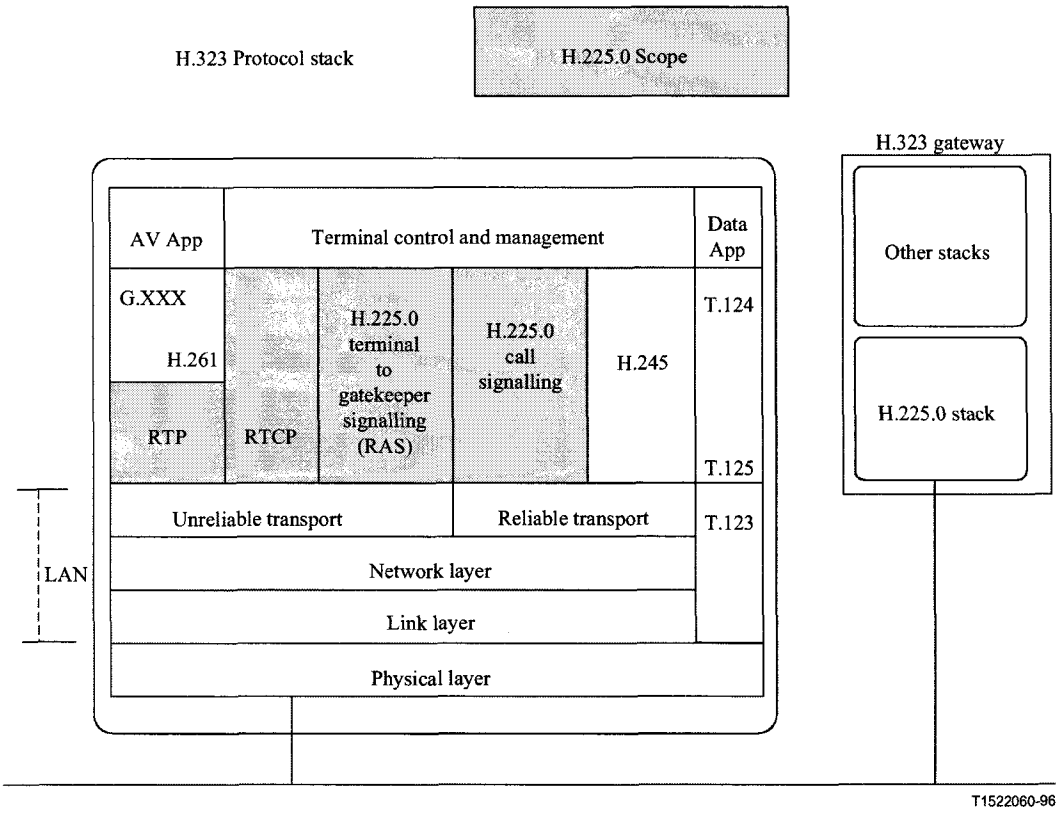


Figure 1/H.225.0 – H.225.0 Scope

The general approach of this Recommendation is to provide a means of synchronizing packets that makes use of the underlying packet-based network/transport facilities. This Recommendation does not require all media and control to be mixed into a single stream, which is then packetized. The framing mechanisms of Recommendation H.221 are not utilized for the following reasons:

- Not using H.221 allows each media to receive different error treatment as appropriate.
- H.221 is relatively sensitive to the loss of random groups of bits; packetization allows greater robustness in the packet-based network environment.
- H.245 and Q.931 can be sent over reliable links provided by the packet-based network.
- The flexibility and power of H.245 as compared to H.242.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[1] CCITT Recommendation G.711 (1988), *Pulse Code Modulation (PCM) of voice frequencies.*

[2] CCITT Recommendation G.722 (1988), *7 kHz audio-coding within 64 kbit/s.*

[3] CCITT Recommendation G.728 (1992), *Coding of speech at 16 kbit/s using Low-delay code excited linear prediction.*

[4] ITU-T Recommendation G.723.1 (1996), *Dual Rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s.*

[5] ITU-T Recommendation G.729 (1996), *Coding of speech at 8 kbits using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP).*

[6] ITU-T Recommendation H.221 (1997), *Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices.*

[7] ITU-T Recommendation H.230 (1997), *Frame synchronous control and indication signals for audiovisual systems.*

[8] ITU-T Recommendation H.233 (1995), *Confidentiality system for audiovisual services.*

[9] ITU-T Recommendation H.242 (1997), *System for establishing communication between audiovisual terminals using digital channels up to 2 Mbit/s.*

[10] ITU-T Recommendation H.243 (1997), *Procedures for establishing communication between three or more audiovisual terminals using digital channels up to 1920 kbit/s.*

[11] ITU-T Recommendation H.245 (1998), *Control protocol for multimedia communication.*

[12] ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at p × 64 kbit/s.*

[13] ITU-T Recommendation H.263 (1998), *Video coding for low bit rate communication.*

[14] ITU-T Recommendation H.320 (1997), *Narrow-band visual telephone systems and terminal equipment.*

[15] ITU-T Recommendation T.122 (1998), *Multipoint communication service – Service definition.*

[16] ITU-T Recommendation T.123 (1996), *Network specific data protocol stacks for multimedia conferencing.*

[17] ITU-T Recommendation T.125 (1998), *Multipoint communication service protocol specification.*

[18] ITU-T Recommendation H.321 (1998), *Adaptation of H.320 visual telephone terminals to B-ISDN environments.*

[19] ITU-T Recommendation H.322 (1996), *Visual telephone systems and terminal equipment for local area networks which provide a guaranteed quality of service.*

[20] ITU-T Recommendation H.324 (1998), *Terminal for low bit-rate multimedia communication.*

[21] ITU-T Recommendation H.310 (1996), *Broadband audiovisual communication systems and terminals.*

[22] ITU-T Recommendation Q.931 (1998), *ISDN user-network interface layer 3 specification for basic call control.*

[23] ITU-T Recommendation Q.932 (1998), *Digital subscriber Signalling System No. 1 – Generic procedures for the control of ISDN supplementary services..*

[24] ITU-T Recommendation X.680 (1994), *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

[25] ITU-T Recommendation X.680/Amd.1 (1995), *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation – Amendment 1: Rules of Extensibility.*

[26] ITU-T Recommendation X.681/Amd.1 (1995), *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification – Amendment 1: Rules of Extensibility.*

[27] ITU-T Recommendation X.691 (1995), *Information technology – ASN.1 encoding rules – Specification of Packed Encoding Rules (PER).*

[28] ITU-T Recommendation E.164 (1997), *The international public telecommunication numbering plan.*

[29] ISO/IEC 10646-1:1993 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.*

[30] ITU-T Recommendation Q.850 (1998), *Usage of cause and location in the digital subscriber Signalling System No. 1 and the Signalling System No. 7 ISDN user part.*

[31] ITU-T Recommendation Q.950 (1997), *Supplementary services protocols, structure, and general principles.*

[32] ITU-T Recommendation H.235 (1998), *Security and Encryption for H-series (H.323 and other H.245 based) multimedia terminals.*

3 Definitions

See definitions in Recommendation H.323. In Recommendation H.323 the term "endpoint" is used to refer to terminals, gateways, and MCUs as elements that are capable of receiving or initiating calls. In this Recommendation the term terminal is often used in a general way in descriptions of call setup, and should be understood as referring to an element that can take part in call setup, including a gateway or MCU.

4 Conventions

In this Recommendation, "shall" refers to a mandatory requirement, while "should" refers to a suggested but optional feature or procedure. The term "may" refers to an optional course of action without expressing a preference.

When a term such as "MCU" is used, an H.323 MCU is referred to. If an H.231 MCU is intended, this will be explicitly noted.

In this Recommendation kilobits/sec is abbreviated kbit/s and is measured in units of 1000. Thus, 64 kbit/s is exactly 64 000 bit.

Unless otherwise specified, the aligned variant PER encoding of ASN.1 shall be used for all ASN.1 specified in this Recommendation.

Q.931 messages are ALL CAPS; ASN.1 is in **bold.**

5 Abbreviations

This Recommendation uses the following abbreviations.

5.1 General abbreviations

BAS	Bit rate Allocation Signal
CIF	Common Intermediate Format
CRV	Call Reference Value
ECS	Encryption Control Signal
FFS	For Further Study
GOB	Group of Blocks
H-MLP	High speed Multi-Layer Protocol
HSD	High Speed Data
IA5	International Alphabet No. 5
IE	Information Element
IETF	Internet Engineering Task Force
IP	Internet Protocol
LAN	Local Area Network
LD-CELP	Low Delay – Code Excited Linear Prediction
LSB	Least Significant Bit
LSD	Low Speed Data
MB	Macro Block (see Recommendation H.261)
MBE	Multi-Byte Extension
MCC	Multipoint Command Conference
MCN	Multipoint Command Negating
MCS	Multipoint Command Symmetrical Data Transmission
MCS	Multipoint Communication Service
MCU	Multipoint Control Unit
MF	MultiFrame
MLP	Multi-Layer Protocol
MPI	Minimum Picture Interval
MSB	Most Significant Bit
NA	Not Applicable
NS	Non-Standard
NSAP	Network Service Access Point
PCM	Pulse Code Modulation
PDU	Protocol Data Unit

QCIF	Quarter Common Intermediate Format
QOS	Quality of Service
RAS	Registration, Admission and Status
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SBE	Single Byte Extension
SC	Service Channel
SCM	Selected Communications Mode
SCN	Switched Circuit Network
TCP	Transport Control Protocol
TSAP	Transport Service Access Point
UDP	User Datagram Protocol
VCF	Video Command "Freeze picture Request"
VCU	Video Command "Fast Update Request"

5.2 RAS message abbreviations

ACF	Admissions Confirm
ARJ	Admissions Reject
ARQ	Admissions Request
BCF	Bandwidth Confirm
BRJ	Bandwidth Reject
BRQ	Bandwidth Request
DCF	Disengage Confirm
DRJ	Disengage Reject
DRQ	Disengage Request
GCF	Gatekeeper Confirm
GRJ	Gatekeeper Reject
GRQ	Gatekeeper Request
IACK	Information request Acknowledgement
INAK	Information request Negative Acknowledgement
IRQ	Information Request
IRR	Information Request Response
LCF	Location Confirm
LRJ	Location Reject
LRQ	Location Request
RAC	Resource Availability Confirmation

RAI	Resource Availability Indication
RCF	Registration Confirm
RIP	Request In Progress
RRJ	Registration Reject
RRQ	Registration Request
UCF	Unregistration Confirm
URJ	Unregistration Reject
URQ	Unregistration Request

6 Packetization and synchronization mechanism

6.1 General approach

Before any calls are made, an endpoint may discover/register with a gatekeeper. If this is the case, it is desirable for the endpoint to know the vintage of the gatekeeper it is registering with. It is also desirable for the gatekeeper to know the vintage of endpoints that register with it. For these reasons, both the *discovery* and registration sequences contain an H.245 style OBJECT IDENTIFIER that allows the vintage to be determined in terms of the version of Recommendation H.323 implemented. This sequence also may contain optional non-standard message parts to allow endpoints to establish non-standard relationships. At the end of this sequence, both gatekeepers and endpoints are aware of the version numbers and the non-standard status of each other.

The version number is mandatory and non-standard information is optional in the Setup/Connect sequence described below to allow two endpoints to inform each other of their vintage and non-standard status. Note, however, that all Q.931 messages have a field for an optional non-standard message in the user-to-user information element, and that all RAS channel messages have an optional field for non-standard information. In addition, a non-standard RAS message has been defined that can be sent at any time.

The unreliable channel for registration, admissions, and status messaging is called the RAS channel. The general approach to starting a call is to send a mandatory admission request on the RAS channel[3], followed by an initial Setup message on a reliable channel transport address (this address may have been returned in the admission confirmation message, or may have been known to the calling terminal). As a result of this initial message, a call setup sequence commences based on Q.931 operations with enhancements described below. The sequence is complete when the terminal receives in the Connect message a reliable transport address on which to send H.245 control messages[4].

When messages are sent on the reliable H.225.0 call signalling channel, only one whole message shall be sent within the boundaries defined by the reliable transport; there shall be no fragmentation of H.225.0 messages across transport PDUs. (In IP implementations as outlined in Appendix IV, this PDU is defined by TPKT.)

---

[3] A terminal that is not registered with a gatekeeper is not required to send an admissions request.

[4] Note that the H.245 address may be sent in the ALERTING or CALL PROCEEDING message to shorten call setup time. Note that the H.245 channel may be opened immediately after the receipt of the H.245 address in the SETUP message.

Once the reliable H.245 control channel has been established, additional channels for audio, video, and data may be established based on the outcome of the capability exchange using H.245 logical channel procedures. Also, the nature of the packet-based network side multi-media conference (centralized vs. distributed/multicast) is negotiated on a per connection basis[5]. This negotiation is performed per media, in the sense that, for example, audio/video may be distributed, while data and control are centralized.

When messages are sent on the reliable H.245 control channel, more than one message may be sent within the boundaries defined by the reliable transport PDU as long as whole messages are sent; there shall be no fragmentation of H.245 messages across transport PDUs. (In IP implementations as outlined in Appendix IV, this PDU is defined by TPKT.)

H.225.0 terminals shall be capable of sending audio and video using RTP via unreliable channels to minimize delay. Error concealment or other recovery action may be applied to overcome lost packets; in general audio/video packets are not re-transmitted since this would result in excessive delay in the packet-based network environment[6]. It is assumed that bit errors are detected in the lower layers, and errored packets are not sent up to H.225.0. Note that audio/video and call signalling/H.245 control are never sent on the same channel, and do not share a common message structure. H.225.0 terminals shall be capable of sending and receiving audio and video on separate transport addresses using separate instances of RTP to allow for media-specific frame sequence numbers and separate quality of service treatment for each media. However, an optional mode where audio and video packets are mixed in a single frame which is sent to a single transport address is for further study.

T.120 capabilities are negotiated using H.245, and upon receipt of appropriate messages, T.120 conferences are established using the transport/packet-based network stacks of Recommendation T.123 as appropriate. T.120 shall be conveyed over the packet-based network between endpoints on another transport address. Table 1 shows the number of TSAP identifiers used for each media on a point-to-point call. It is also true that a given H.323 terminal may be able to participate in more than one conference at a time, resulting in the use of additional TSAP identifiers. All H.245 logical channels used are unidirectional except for those associated with T.120, which are bidirectional.

---

[5] The LAN side conference may be part centralized and part distributed, as decided by the MC controlling the conference. However, the terminal is not aware of this fact. Generally, of course, all terminals will see the same Selected Communications Mode (SCM) (see Recommendation H.243 for a definition).

[6] Fast Update of full frames, MBs, or GOBs may be requested via H.245 signalling.

Table 1/H.225.0 – TSAP IDs used by H.225.0 per point-to-point unicast call

Usage of TSAP Ids	Reliable or unreliable	Well known or dynamic
Audio/RTP	Unreliable	Dynamic
Audio/RTCP	Unreliable	Dynamic
Video/RTP	Unreliable	Dynamic
Video/RTCP	Unreliable	Dynamic
Call Signalling	Reliable	Well known or dynamic
H.245	Reliable	Dynamic
Data (T.120)	Reliable	Well known or dynamic
RAS	Unreliable	Well known or dynamic
NOTE – If well known TSAP identifiers are used, there can only be a single endpoint per network address. Also, in the direct call model the caller requires a well known TSAP identifier for the Call Signalling channel to start the call.		

Although the transport address for, say, audio and video, may share the same packet-based network address and differ only by TSAP identifier, some manufacturers may choose to use different packet-based network addresses for audio and video. The only requirement is that the convention of Annexes A and B should be followed in the numbering of TSAP identifiers in the RTP session[7].

Table 1 describes the basic case of point-to-point unicast operations between two terminals. To facilitate the construction of gateways, MCUs, and gatekeepers, dynamic TSAP Ids may be used instead of well known TSAP Ids. Tables 2 and 3 illustrate an example of TSAP Id usage for the gateway/MCU case, and for the gatekeeper case.

Table 2/H.225.0 – TSAP IDs used on one MCU/Gateway Port (Unicast example)

Usage of TSAP Ids	Reliable or unreliable	Well known or dynamic
Audio/RTP	Unreliable	Dynamic
Audio/RTCP	Unreliable	Dynamic
Video/RTP	Unreliable	Dynamic
Video/RTCP	Unreliable	Dynamic
Call Signalling	Reliable	Dynamic (Note)
H.245	Reliable	Dynamic
Data (T.120)	Reliable	Dynamic
RAS	Unreliable	Dynamic (Note)
NOTE – See Note 1 of Table 3.		

[7] Note that any TSAP Id can be used for the initial RTP session; the major reason to follow the RTP convention is for possible IETF RTP interoperability.

Table 3/H.225.0 – Example of TSAP IDs usage by H.225.0 gatekeeper supporting the gatekeeper mediated call model of Figure 11/H.323 for a point-to-point call

Usage of TSAP Ids	Reliable or unreliable	Well known or dynamic	Number of channels
Call Signalling	Reliable	Dynamic or well known (Note 1)	2 per call (Note 2)
H.245	Reliable	Dynamic	2 per call (Note 2)
RAS	Unreliable	Well known	1
NOTE 1 – If the well known TSAP ID is used the gatekeeper may be limited to a single endpoint per device; therefore dynamic TSAP IDs should be used. NOTE 2 – 0 for direct call model; 2 for gatekeeper mediated call model.			

Note that a reliable transport address is used for call setup for the terminal to terminal case, and also for the gatekeeper mediated case. The reliable call signalling connection shall be kept active according to the following rules:

1) For terminal-to-terminal call signalling (see Figure 9/H.323), either terminal may choose to close the reliable call signalling channel, or to leave it open.

2) For the gatekeeper mediated call signalling case (see Figure 8/H.323), the terminals shall keep the reliable port active throughout the call. However, the gatekeeper may choose to close this signalling channel, but should keep the channel open for calls that involve gateways. This will allow the end-to-end transmission of Q.931 information elements such as display information.

3) If for some reason the reliable link becomes inactive via a transport level failure or other problem, the link shall be re-opened, and the call shall not be dropped. Call state and the use of the CRV (Call Reference Value from Q.931) are not affected by the closing of the reliable link unless the H.245 channel is also closed, indicating the end of the call.

Note that more than one H.245 channel may be open at a given time, i.e. an endpoint may be in more than one call/conference at the same time. Note also that within a specific call, a terminal may have more than one channel of the same type open, e.g. two audio channels for stereo audio. The only limitation is that there shall be one and only one H.245 control channel per point-to-point call.

H.245 logical channel signalling is used to start and stop video, audio, and data protocol usage. This process calls for closing the open channel, and then re-opening with a new mode of operation. As part of the process of opening the channel, before sending the open logical channel acknowledgment, the endpoint uses the ARQ/ACF or BRQ/BCF sequence to ensure that sufficient bandwidth is available for the new channel (unless sufficient bandwidth is available from a previous ARQ/ACF or BRQ/BCF sequence). In some cases, the gateway may find that the SCN side mode change occurs more quickly than the packet-based network side mode change, resulting in the possibility of the loss of audio information. The gateway may adopt several approaches at the discretion of the manufacturer:

a) the gateway may transcode audio, thus hiding the SCN mode changes;

b) the gateway may simply throw away audio information; or

c) the gateway may operate as an H.231 MCU, thus gaining control over all SCN side mode changes.

No general rule exists concerning whether H.245 or RTP procedures (see Annexes A, B and C) take precedence; each conflict and its resolution is specifically mentioned in this Recommendation.

Note also that there is no fixed association between SSRCs and logical channels; Recommendation H.245 provides this association which may be used for audio/video synchronization.

In general, two types of conference modes of operation on the packet-based network side are possible: distributed and centralized. It is also possible that different choices may be made for different media, e.g. distributed audio/video and centralized data. Procedures for determining what sort of conference to establish are in Recommendation H.323; the messages of this Recommendation are intended to support all allowed combinations, noting that distributed control and data are for further study although supported by the H.245 capability signalling.

6.2 Use of RTP/RTCP

The H.225.0 endpoint shall be capable of using separate TSAP IDs for audio and video and the associated RTCP channels as described in Annexes A and B. Optionally, endpoints may choose to use different packet-based network addresses for audio and video, but for each packet-based network address the convention of Annexes A and B should be followed in the use of TSAP Ids. Using H.245 signalling, additional audio and video channels may be established if the terminal supports this capability.

An optional capability to use a single transport address for both audio and video is for further study.

Unless an exception is specifically mentioned here, implementations shall follow those of RTP as contained in Annex A unless modified by text in this Recommendation. Implementations shall follow the RTP profile (see Annex B) only as specifically mentioned in this Recommendation.

RTP translators and mixers are not elements of the H.323 system, and any information about them in Annexes A and B shall be regarded as informative. Note that both gateways and MCUs have some aspects of both mixers and translators, and the information in Annexes A and B may be helpful in the implementation of gateways and MCUs. However, MCUs are not mixers, and mixers are not MCUs. Note that gateways, for example, on a packet-based network-to-packet-based network call via the gateway may act as translators.

Version (V): Version 2 of RTP shall be used.

CSRC Count (CC): Use of the CSRC count in this Recommendation is optional. When not in use, the value of CC shall be zero (0). The CSRC may be used by MCUs to provide information on contributors to the audio sum when distributed audio processing is occurring. Note that there are no capabilities associated with the ability to understand the CSRC count so the MCU/MC has no way of knowing whether and how the terminal in the conference makes use of the information.

CNAME: In the simplest case of a point-to-point connection on the packet-based network, the SSRC is used to identify an audio/video source from a terminal, and the streams are associated by a CNAME as being supplied by the same endpoint as specified in Annex A.

When using RTCP, either RR or SR packets shall be sent periodically as described in Annex A. The CNAME SDES message shall be used. Other SDES messages (see Annex A) are optional, but shall not be used for conference control or conference information when either H.245 and/or T.120 control functions are in use. Information provided by Recommendations H.245 and/or T.120 shall be regarded as the correct information.

The RTCP BYE message shall not be relied on for RTP session termination. The H.323 terminal determines when a call is disconnected via the procedures of H.323. The only mandatory use of the RTCP BYE packet is for SSRC collision resolution.

The H.323 terminal, when engaged in any conference, whether point-to-point or multi-point, shall restrict the logical channel bit rate averaged over a period as defined in Recommendation H.245 to that signalled in the H.245 **FlowControlCommands**, H.245 logical channel commands, and the T.120 flow control mechanism.

When the H.323 terminal is connected to an H.323 gateway, the gateway shall use the means of Recommendations H.245 and T.120 to force the H.323 terminal to transmit at a rate less than or equal to the SCN side media rates and receive at a rate equal or higher than the SCN rate, with the following exceptions:

- Control bandwidth on the packet-based network need not match that in Recommendation H.221.
- Audio bandwidth on the packet-based network may match that in Recommendation H.221 on the WAN, but with gateway transcoding a match is not required.
- In the case where the gateway is using a rate reducer; the packet-based network side H.323 terminal shall not exceed the H.245 signalled rate, which will probably be less than the rate being sent over the WAN.

Encryption for H.323 endpoints is for further study.

6.2.1 Audio

Before considering how audio is packetized using RTP, attention must be directed toward how it is signalled via H.245, and the relationship of this signalling to RTP. In general, when the audio channel is opened, an H.245 logical channel is opened. H.245 signalling in the **AudioCapability** structure is given in terms of the maximum number of frames per packet. The frame size for this Recommendation varies with the audio coding in use.

All H.323 terminals offering audio communication shall support G.711. For all frame oriented audio codecs, receivers shall signal the maximum number of audio frames they are capable of accepting in a single audio packet. Transmitters may send any whole number of audio frames in each packet, up to the maximum stated by the receiver. Transmitters shall not split audio frames across packets, and shall send whole numbers of octets in each audio packet.

Sample based codecs, such as G.711 and G.722 shall be considered to be frame-oriented, with a frame size of eight samples. (See Annex B for additional information regarding guidelines for sample-based audio encoding.) For audio algorithms such as G.723 which use more than one size of audio frame, audio frame boundaries within each packet shall be signalled in-band to the audio channel.

For audio algorithms which use a fixed frame size (see Recommendations G.728 and G.729 for the frame size used by each) audio frame boundaries shall be implied by the ratio of packet size to audio frame size; in other words only whole audio frames shall be put in the RTP packet.

Payload Type (PT): Only ITU-T payload types such as (0)[PCMU], (8)[PCMA], (9)[G722], and (15)[G728] shall be used for ITU codecs signalled in Recommendation H.245. Dynamic payload types exchanged using H.245 signalling shall be used for any ITU-T payload types not listed in Annex B.

It is recommended that if an interruption in sequence numbers is observed, the receiver may repeat the most recent received sounds such that the amplitude of the repeated sound decays to silence; other similar procedures may be used at the discretion of the manufacturer.

Each G.711 octet shall be octet aligned in an RTP packet. The sign bit of each G.711 octet shall correspond to the most significant bit of the octet in the RTP packet (i.e. assuming the G.711 samples are handled as octets on the host machine, the sign bit shall be the most significant bit of the octet as defined by the host machine format).

When sending 48/56 kbit/s PCM toward the packet-based network, the H.323 gateway shall pad the extra 1 or 2 bits in each octet in accordance with Note 2 in Table 1b/G.711, and use the RTP values for PCMA or PCMU(8 or 0). For μ-law the padding consists of "1" in both the 7th and 8th bit. For

A-law the 7th bit shall be 0 and the 8th bit 1. In the reverse direction the H.323 gateway shall truncate 64 kbit/s G.711 on the packet-based network side to fit the G.711 rate being used in H.320. Thus, on the packet-based network side only 64 kbit/s G.711 shall be used.

When sending 48/56 kbit/s G.722 toward the packet-based network, the H.323 gateway shall pad the extra 1 or 2 bits in each octet, and use dynamic RTP payload types as signalled by Recommendation H.245 to differentiate between 64 kbit/s (which uses PT = 9) and the reduced rate cases. In the reverse direction the H.323 gateway shall truncate 64 kbit/s G.722 on the packet-based network side to fit the G.711 rate being used in H.320. Thus, on the packet-based network side only 64 kbit/s G.722 shall be used.

If possible, the H.323 terminal should make use of the silence suppression feature of RTP, especially when the conference is multicast. The H.323 terminal shall be able to receive silence compressed RTP streams. Coders may omit sending audio signals during silent periods after sending a single frame of silence, or may send silence background fill frames if such techniques are specified by the audio codec Recommendation in use.

6.2.2 Video messages

Payload Type (PT): Only ITU-T payload types such as that for Recommendation H.261 or H.263 shall be used for ITU codecs signalled in Recommendation H.245. Dynamic payload types may be used for codecs which can be signalled via H.245 and for which packetization formats have not been defined.

Marker (M): The marker bit should be set according to the procedures described in Annex A except in cases where it would increase end-to-end delay.

In order to recover from the loss of video packets, H.245 **VideoFastUpdatePicture**, **VideoFastUpdateMB**, and **VideoFastUpdateGOB** shall be supported. Use of the RTCP control packets Full Intra Request (FIR) [send me a full frame] and Negative Acknowledgment (NACK) [Send me certain packets] is optional, and signalled in H.245 capabilities.

In C.3.3 error recovery method 3) may be impractical if the NACK does not arrive within one frame time.

H.261 is packetized on the packet-based network side as per Annex C. As long as sufficiently large RTP packets are available, fragmentation on MB boundaries by the transmitter is not required. However, if the H.323 terminal fragments H.261 packets on the RTP level, this fragmentation shall occur on MB boundaries. All H.323 terminals shall be able to receive MB fragmented packets as well as GOB fragmented packets, or packets with a mix of MBs and GOBs. Note that failure to support MB fragmentation in the transmitter may result in the loss of an entire GOB, and may also lower the packet rate. RTP packets used should not exceed the size of the Maximum Transfer Unit (MTU) on a given packet-based network to maximize robustness of operation. MBs shall not be split across packets; all packets shall end on a GOB or MB boundary. The H.323 transmitter may choose to fill out a packet containing a small GOB with additional MBs, but this is not required.

To preclude the possibility of corruption in multiple pictures caused by the loss of an RTP packet, the RTP packetizer in an H.323 endpoint shall not include video from more than one picture in an RTP packet.

The SBIT is the number of most significant bits that shall be ignored in the first data octet. EBIT is the number of least significant bits that shall be ignored in the last data octet.

The RTP packetizer shall not intentionally octet align video at the start of RTP packets. In other words, if EBIT = n in an RTP packet, SBIT in the next RTP packet shall equal $8 - n$, $0 < n < 8$, and if EBIT = 0 in an RTP packet, SBIT in the next RTP packet shall equal 0. This requirement avoids

possible additional end-to-end delay caused by bit-shifting. This requirement shall apply across picture boundaries.

Annex D specifies an H.323 extension to the video packet header that contains an optional octet count. The use of this optional extension is described in Annex D.

See Appendix IV for packet-based network specific advice on video packetization.

6.2.3 Data messages

There are no special data messages or formats; T.120 is used on the packet-based network as per Recommendation T.123. Centralized vs. distributed data conferencing on the packet-based network is described in Recommendation H.323, and is negotiated via H.245.

T.120 flow control on the packet-based network is managed using packet-based network protocols when requested by H.245 **FlowControlCommand** and **maxBitRate** limits.

See Recommendation H.323 for the procedures used to connect a running T.120 conference to an H.323 conference, or to add an H.323 call to a T.120 conference.

The protocol to be used by H.224 on the packet-based network is for further study.

7 H.225.0 message definitions

This clause concerns the definition of messages for call setup, call control, and communications between terminals, gateways, gatekeepers, and MCUs.

The ASN.1 definitions for all H.225.0 messages appear in Annex H.

7.1 Use of Q.931 messages

Implementations shall follow Recommendation Q.931 as specified in this Recommendation. Terminals may also support optional Q.931 and H.450 messages. The messages shall contain all of the mandatory information elements and may contain any of the optional information elements as defined in Recommendation Q.931 as described in this Recommendation. Note that the H.225.0 endpoint may, according to Q.931, ignore all optional messages it does not support without harming interoperability, but shall respond to an unknown message with a STATUS message.

Each H.225.0 endpoint shall be able to receive and identify an incoming Q.931 or H.450 message as such. It shall be capable of processing the mandated Q.931 messages; it may be capable of processing the optional Q.931 messages. In any case, each H.225.0 endpoint shall be able to ignore messages unknown to it without disturbing operation.

Each H.225.0 endpoint shall be able to interpret and generate the information elements mandated in the following for the respective Q.931 and H.450 messages. It may interpret and generate the optional information elements as defined below as well. It also may interpret other information elements of Q.931, or other Q series or H.450 protocols. The endpoints shall be able to ignore unknown information elements contained in a Q.931 or H.450 message without disturbing operation. Procedures for receiving unrecognized "comprehension required" information elements shall apply according to 5.8.7.1/Q.931.

Intermediate systems (gateways and gatekeepers) shall follow the rules below with regard to Q.931 optional messages and information elements:

1) The gateway should and the gatekeeper shall, after appropriate modification, forward all information elements (optional or mandatory) associated with mandatory Q.931 messages either from the terminal to the gateway/terminal or in the reverse direction. This includes such information elements as user-to-user information and the display information.

2) A gateway should forward all Q.931 or H.450 optional messages and information elements in both directions. If the call signalling channel is not kept up by the gatekeeper, this is not possible.

3) As long as the Q.931 call signalling channel is up, a gatekeeper shall forward all Q.931 or H.450 optional messages and information elements in both directions after appropriate modification. If the call signalling channel is not kept up by the gatekeeper, this is not possible. Note that the gatekeeper may act as a signalling element that can provide features (such as supplementary service features) and may therefore modify, terminate, or originate Q.931 messages.

H.323 gateways may be capable of converting H.450-series of supplementary services to the corresponding ISO/IEC 11582 supplementary service and vice versa. H.323 gateways may be capable of passing on unmodified ISO/IEC 11582 supplementary service signalling through the H.323 environment within the User-to-User information element. Details are for further study.

In this version of this Recommendation, all references are to the 1993 version of Recommendation Q.931. The procedures of 3.1/Q.931 for circuit mode connection setup are followed. However, the implementor is reminded that although "bearer" is being signalled for, no actual "B-channels" of the ISDN type exist on the packet-based network side. Successful completion of the "call" results in an end-to-end reliable channel supporting H.245 messaging. Actually "bearer" setup is done using H.245. However, the use of Q.931 on the packet-based network side enables interworking with Q.931 on the WAN side as well as providing a well-tested framework for general connection oriented calling features.

In general, the symmetric procedures of Annex D/Q.931 are used. This implies that the Q.931 state machine is followed as per Annex D/Q.931 with the exception that the procedure of D.3/Q.931 (Call collisions) shall not be followed; recovery from this glare condition is left to the application layer.

Endpoints not supporting Q.931 shifted code sets shall ignore all Q.931 messages using such methods.

Table 4 shows what messages are mandatory and optional for H.323 and H.225.0 call setup using Q.931 on the packet-based network.

Table 4/H.225.0 – H.225.0 usage of Q.931/Q.932 Messages

Call establishment messages	Transmit (M, F, O, CM)[a]	Receive and act on (M, F, O[b], CM)
Alerting	M	M
Call Proceeding	O	CM[c]
Connect	M	M
Connect Acknowledge	F	F
Progress	O	O
Setup	M	M
Setup Acknowledge	O	O
Call Clearing messages		
Disconnect	F	F
Release	F	F
Release Complete	M[d]	M
Call Information Phase messages		
Resume	F	F
Resume Acknowledge	F	F
Resume Reject	F	F
Suspend	F	F
Suspend Acknowledge	F	F
Suspend Reject	F	F
User Information	O	O
Miscellaneous messages		
Congestion Control	F	F
Information	O	O
Notify	O	O
Status	M[e]	M
Status Inquiry	O	M
Q.932/H.450 messages		
Facility	M	M
Hold	F	F
Hold Acknowledge	F	F

Table 4/H.225.0 – H.225.0 usage of Q.931/Q.932 Messages *(concluded)*

Call establishment messages	Transmit (M, F, O, CM)[a)]	Receive and act on (M, F, O[b)], CM)
Hold Reject	F	F
Retrieve	F	F
Retrieve Acknowledge	F	F
Retrieve Reject	F	F

a) M Mandatory, F Forbidden, O Optional, CM Conditionally Mandatory. Something is CM if it is required once an option is supported.

b) Note that STATUS shall not be sent in response to a message listed here as "O"; the receiver shall simply ignore the message if it does not support it.

c) Terminals intended to use gateways shall receive and act on CALL PROCEEDING.

d) Release Complete is required for any situation in which the H.225.0 reliable call signalling channel is open. If this cannel is not open. H.245 session end may be used to terminate the conference.

e) The endpoint shall respond to an unknown message with a STATUS message; response to STATUS INQUIRY is also mandatory. However, an endpoint is not required to send STATUS INQUIRY. As a practical matter, the endpoint should be able to understand a STATUS message received in response to a message sent that was not known to the receiver

7.2 Common Q.931 Information Elements

7.2.1 Header Information Elements

For all Q.931 messages, there are three common fields that are mandatory in addition to the message type that are described in this subclause.

7.2.1.1 Protocol Discriminator

As defined in 4.2/Q.931.

Shall be set to 08H – this identifies the message as Q.931/I.451 user-network message (encoded following Figure 4-2/Q.931). If a gatekeeper is acting as a network to supply supplementary services, it may be appropriate to use another value. This is for further study.

7.2.1.2 Call reference

As defined in 4.3/Q.931.

A call reference value length of two octets shall be supported by any H.323 endpoint.

The call reference value is chosen at the side originating the call and has to be locally unique. For subsequent communication, the calling and the called side shall use this call reference value in all the messages belonging to this particular call.

The value is encoded following Figure 4-5/Q.931 for a two-octet call reference value. The most significant octet of the reference value is always encoded in octet No. 2.

Note that the CRV is only unique on a particular part of a call, e.g. between two terminals, or between a terminal and a gatekeeper. If a given terminal has two calls in the same conference, each shall have the same conference ID but different CRVs.

The call reference flag shall be set according to the procedures described in Recommendation Q.931.

Note that the CRV values passed in RAS messages shall conform to the structure as specified in Recommendation Q.931. Specifically, the call reference flag shall be included as the most significant bit of the CallReferenceValue. This restricts the actual CRV to the range of 0 through 32 767, inclusive.

7.2.1.3 Message type

The message type is encoded according to Figure 4-6/Q.931 using the values specified in Table 4-2/Q.931. H.225.0 specific extensions are for further study.

7.2.2 Message-specific information elements

The general encoding rules for the following information elements are defined in 4.5.1/Q.931 and Table 4-3/Q.931. These rules shall be followed. The escape mechanism (see Figure 4-8/Q.931) is optional.

7.2.2.1 Bearer capability

This information element is encoded according to Figure 4-11/Q.931 and Table 4-6/Q.931. If this information element is received in a packet-based network-to-packet-based network call, it may be ignored by the receiver. If this information element appears in a Setup message for a call-independent signalling connection as defined in Recommendation H.450.1, the coding shall follow 7.2/H.450.1. In all other cases, the following applies to the use of the various fields of this information element (the octet number references refer to Figure 4-11/Q.931):

Information transfer capability (octet No. 3)

- The extension bit (bit 8) shall be set to '1'.
- The coding standard (bits 6, 7) shall be set to '00' indicating 'ITU-T'.
- Information transfer capability (bits 0-5):
  - For calls originating from an ISDN end point the information indicated to the gateway shall be forwarded.

    NOTE – This is to allow some advance information about the nature of the connection to be forwarded to the H.323 end point, e.g. voice only vs. data vs. video; this would have an impact on the bandwidth required as well as on the ability/willingness to accept the call or not.
  - Calls that originate from an H.323 endpoint shall use this field to indicate their wish to place an audiovisual call. Therefore, the field shall be set either to 'unrestricted digital information', i. e. '01000' or to 'restricted digital information', i. e. '01001'. If a speech only call is to be placed, the H.323 terminal shall set the information transfer capability to either 'speech' (i.e. '00000') or to '3.1 kHz audio' (i.e. '10000').

Extension bit for octet No. 4 (bit 8)

- Shall be set to '0' if the information transfer rate is set to 'multirate'; shall be set to '1' otherwise.

Transfer Mode – octet No. 4 (bits 6, 7)

- Shall specify 'circuit mode', value '00'.

Information transfer rate

- Shall be encoded following Table 4-6/Q.931 except that the value '00000' (for packet mode) is not permitted unless the gateway is connected to a packet network.

Rate multiplier – octet No. 4.1

- Shall be present if information transfer rate is set to 'multirate'.
- The extension bit (bit 8) shall be set to '1'.
- The bits 1 through 7 shall indicate the bandwidth needed for the call as defined in the following (note, that in contrast to Recommendation Q.931, a value of '0000001' is allowed here).
- For a call originating from an ISDN endpoint, the gateway shall simply pass on the information that it receives from the ISDN.
- For a call incoming from an H.324 endpoint, the gateway shall set the rate multiplier to 01H.
- For a call incoming from B-ISDN, some translation from Recommendations Q.2931 to Q.931 needs to be performed. This is for further study.
- For a call originated from an H.323 endpoint, this shall be used to indicate the bandwidth to be used for this call. If the called system is another H.323 endpoint, this value may reflect the bandwidth to be used on the packet-based network but the receiving terminal is not required to follow this information. If a gateway is involved, then this value shall reflect the number of external connections to be set up. The bandwidth needed for the call is the bandwidth needed on the SCN side, and may or may not match the bandwidth allowed on the packet-based network by the ACF/BCF messages.

Layer 1 protocol – Octet No. 5

- The extension bit (bit 8) shall be set to '1'.
- Bits 6 and 7 shall indicate the layer one identifier, i. e. '01'.
- Bits 1 through 5 shall indicate the layer one protocol.
- The allowed values are G.711 (A-law '00011' and μ–Law '00010') to indicate a voice-only call and H.221 and H.242 ('00101') to indicate an H.323 videophone call.

Octets Nos. 5a, 5b, 5c, 5d shall not be present.

Layer 2 protocol identifier – Octet No. 6

- Shall not be present.

Layer 3 protocol identifier – Octet No. 7

- Shall not be present.

7.2.2.2 Call identity

The possible use of the call identity IE is for further study. This study should consider multi-stage dialing, including terminal -> gatekeeper -> terminal and terminal -> gateway -> terminal, and loose source routing.

7.2.2.3 Call state

This information element is encoded following Figure 4-13/Q.931.

Octet No. 3 Coding Standard (bits 8-7)

- Set to '00' to indicate ITU-T standardized coding.

Call State Value (octet No. 3, bits 1-6)

- Set as per Table 4-8/Q.931 but do not use the global interface state values. Values are interpreted as User State as per use of Annex D/Q.931. Note that most of the listed codes will not be generated by an H.323 terminal.

7.2.2.4 Called party number

This information element is encoded following Figure 4-14/Q.931 and Table 4-9/Q.931.

Octet No. 3 Extension (bit 8)

– Set to '1'.

Type of number (octet No. 3, bits 5-7)

– Encoded following the values and rules of Table 4-9/Q.931.

Numbering plan identification (octet No. 3, bits 1-4)

– Encoded following the values and rules of Table 4-9/Q.931. If set to '1001' (Private Numbering Plan) in a packet-based network originated call, this indicates that:

1) the E.164 address is not present in SETUP; and
2) the call will be routed via an alias address in the user-to-user information.

Number "digits"

– Any number of IA5 characters, according to the formats specified in the appropriate numbering/dialling plan.

7.2.2.5 Called party subaddress

Use as per Recommendation Q.931.

7.2.2.6 Calling party number

This information element is encoded following Figure 4-16/Q.931 and Table 4-11/Q.931.

Octet No. 3 Extension (bit 8)

– Set to '1'.

Type of number (octet No. 3, bits 5-7)

– Encoded following the values and rules of Table 4-9/Q.931.

Numbering plan identification (octet No. 3, bits 1-4)

– Encoded following the values and rules of Table 4-9/Q.931. If set to "1001" (Private Numbering Plan) in a packet-based network originated call, this indicates that:

1) the E.164 address is not present in SETUP; and
2) the call will be routed via an alias address in the user-to-user information.

Octet No. 3a

– Shall not be present.

Number "digits"

– Any number of IA5 characters, according to the formats specified in the appropriate numbering/dialling plan.

7.2.2.7 Calling party subaddress

Use as per Recommendation Q.931.

7.2.2.8 Cause

If received, the rules defined in Recommendation Q.850 apply. Note that either Cause or **RelCompReason** is mandatory for RELEASE COMPLETE; the Cause IE is optional elsewhere. The Cause IE and the ReleaseCompleteReason (a part of the Release Complete message) are mutually

exclusive. Gateways shall map from a ReleaseCompleteReason to the Cause IE when sending a Release Complete message to the circuit switched side from the packet-based network side (see Table 5). (The reverse mapping is not required as packet-based network entities are required to decode the Cause IE.)

Table 5/H.225.0 – Release Complete Reason to cause IE mapping

ReleaseCompleteReason code	Corresponding Q.931/Q.850 cause value
noBandwidth	34 – No circuit/channel available
gatekeeperResources	47 – Resource Unavailable
unreachableDestination	3 – No route to destination
destinationRejection	16 – Normal call clearing
invalidRevision	88 – Incompatible destination
noPermission	111 – Interworking, unspecified
unreachableGatekeeper	38 – Network out of order
gatewayResources	42 – Switching equipment congestion
badFormatAddress	28 – Invalid number format
adaptiveBusy	41 – Temporary Failure
inConf	17 – User busy
undefinedReason	31 – Normal, unspecified
facilityCallDeflection	16 – Normal call clearing
securityDenied	31 – Normal, unspecified
calledPartyNotRegistered	20 – Subscriber absent
callerNotRegistered	31 – Normal, unspecified

7.2.2.9 Channel identification

Use is for further study; may be used to provide feedback on multiple call attempts.

7.2.2.10 Congestion level

Shall not be used.

7.2.2.11 Date/time

Encoded following Figure 4-21/Q.931.

7.2.2.12 Display

Encoded following Figure 4-22/Q.931. The maximum length of the entire information element is 82 octets.

7.2.2.13 Extended Facility Information Element

Any Extended Facility IE that is used to indicate unmodified semantics as defined in Q.95x-series Recommendations shall be encoded following 8.2.4/Q.932. In this case, the Service ADUs shall be formed according to ROSE [uses Recommendation X.208 (Specification of ASN.1) and Recommendation X.209 (Specification of basic encoding rules for ASN.1)] as defined in Recommendation X.229.

7.2.2.14 Facility

In order to signal call redirection specific to H.323 procedures (call forwarding, redirecting a call to the MC, or forcing a call to be routed to the gatekeeper) or in case of supplementary service signalling according to Recommendation H.450, the User-to-User Information Element of the Facility is used. This particular case shall be indicated by coding a Facility IE of length zero, i.e. the Facility Information Element shall consist of exactly 2 octets as follows:

- Octet No. 1 (information element identifier) shall be set to '00011100' ('1C'H) to indicate the Facility IE.
- Octet No. 2 (information element length) shall be set to '0' to indicate that no further octets belonging to this information element follow.

In order to indicate call forwarding, the Facility IE shall be empty and the **Facility-UUIE** shall indicate in the **alternativeAddress** or the **alternativeAliasAddress** the terminal to which the call is to be redirected. In this case, the **facilityReason** shall be set to **callForwarded**.

To instruct an endpoint to call a different endpoint because the calling endpoint wishes to join a conference and the called endpoint does not have the MC, the Facility IE would be left empty as well. The **conferenceID** shall indicate the conference to join and the reason in the **Facility-UUIE** shall be **routeCallToMC**.

Also, to instruct the calling endpoint to signal the called endpoint through the called endpoint's gatekeeper, the Facility IE is left empty. The **conferenceID** in the **Facility-UUIE** shall indicate the conference to join and the reason in the **Facility-UUIE** shall be **routeCallToGatekeeper.**

Any Facility IE that is used to indicate unmodified semantics as defined in Q.95x-series Recommendations shall be encoded following 8.2.3/Q.932. In this case, the Service ADUs shall be formed according to ROSE [uses Recommendation X.208 (Specification of ASN.1) and Recommendation X.209 (Specification of basic encoding rules for ASN.1)] as defined in Recommendation X.229.

7.2.2.15 High layer compatibility

FFS.

7.2.2.16 Keypad facility

Encoded following Figure 4-24/Q.931.

7.2.2.17 Low layer compatibility

FFS.

7.2.2.18 More data

Shall not be used.

7.2.2.19 Network-specific facilities

Shall not be used.

7.2.2.20 Notification indicator

Encoded following 4.5.22/Q.931.

7.2.2.21 Progress indicator

Encoded following Figure 4-29/Q.931 and Table 4-20/Q.931.

This information element is only required for interfacing an H.323 terminal to an ISDN- and ATM-based terminal where detailed call proceeding information is available. In this case, the gateway shall forward this information to the H.323 terminal. The H.323 end system need not interpret this information element.

If this information element is generated by an H.323 terminal, the following restrictions apply:

Coding standard (octet No. 3, bit 6,7)

– Shall indicate 'ITU-T' ('00').

Location

– Following Table 4-20/Q.931.

– The values 'user' ('0000'), 'private network serving the local user' ('0001'), and 'private network serving the remote user' ('0101') are permitted.

Progress description

– Following Table 4-20/Q.931.

7.2.2.22 Repeat indicator

Shall not be used.

7.2.2.23 Restart indicator

Shall not be used.

7.2.2.24 Segmented message

Shall not be used. Note that there is no critical upper limit on the message size in Recommendation H.323 and this Recommendation.

7.2.2.25 Sending complete

Encoded following Figure 4-33/Q.931.

No restrictions apply.

7.2.2.26 Signal

Encoded following Figure 4-34/Q.931 and Table 4-24/Q.931.

No restrictions apply.

7.2.2.27 Transit network selection

Shall not be used.

7.2.2.28 User-user

Encoded following Figure 4-36/Q.931 and Table 4-26/Q.931, as modified here.

The user-user information element shall be used by all H.323 entities to convey H.323-related information. Actual user-user information to be exchanged only between the involved terminals is nested in the H323-UserInformation PDU (to which no restrictions apply).

The following restrictions apply:

Length of user-user contents

– Shall be 2 octets instead of 1 (as in Figure 4-36/Q.931).

Protocol discriminator

– Shall indicate X.208 and X.209 (ASN.1) coded user information ('00000101').

NOTE – This is taken from the 1993 revision of Recommendation Q.931 that references the earlier revisions of ASN.1. The correct references to ASN.1 are Recommendations X.680 (syntax) and X.691 (PER).

User information

– Shall contain an ASN.1 structure that – besides the H.323 relevant information – includes the actual user data as follows. The ASN.1 is encoded using the basic aligned variant of the packed encoding rules as specified in X.691. Note that the ASN.1 structure begins with **H323-UserInformation.**

– For the user-information field the rules specified in section 4.5.30/Q.931 apply.

7.3 Q.931 message details

Note that the lengths of the information elements specified in the tables below refer to messages that are generated by H.323 terminals only. The size of the User-to-User Information Element shown is understood as the size of the **user-data** structure in **H323-UserInformation** and does not include the **h323-UU-PDU**. The total size of **H323-UserInformation** is limited to 65 536 octets. Regardless of the specified sizes, messages forwarded from the SCN side may have different (larger) sizes.

Also note that an information element specified below as mandatory, optional, or forbidden refers only to whether or not H.323 terminals may originate such information elements.

7.3.1 Alerting

This message may be sent by the called user to indicate that called user alerting has been initiated. In everyday terms, the "phone is ringing."

Follow Table 3-2/Q.931 (1993 version) as modified below in Table 6.

Table 6/H.225.0 – Alerting

Information element	H.225.0 status (M/F/O)	Length in H.225.0
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Bearer capability	O	5-6
Extended facility	O	8-*
Channel identification	FFS	NA
Facility	O	8-*
Progress indicator	O	2-4
Notification indicator	O	2-*
Display	O	2-82
Signal	O	2-3
High layer compatibility	FFS	NA
User-to-User	M	2-131

The user-to-user information element contains the Alerting-UUIE defined in the H.225.0 Message Syntax. The Alerting-UUIE includes the following:

protocolIdentifier – Set to the version of H.225.0 supported.

destinationInfo – Contains an **EndpointType** to allow the caller to determine whether the call involves a gateway or not.

h245Address – This is a specific transport address on which the called endpoint or gatekeeper handling the call would like to establish H.245 signalling. This address may also be sent in Call Proceeding, Progress, or Connect.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

h245SecurityMode – An H.323 entity that receives a Setup message with the **h245SecurityCapability** set shall respond with the corresponding, acceptable **h245SecurityMode** in the CallProceeding, Alerting, Progress, or Connect.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

fastStart – Used only in the fast connect procedure, **fastStart** supports the signalling needed to open a logical channel. This uses the OpenLogicalChannel structure defined in Recommendation H.245, but the sender of this indicates the modes it prefers to receive and transmit, and the transport addresses where it expects to receive media streams.

7.3.2 Call Proceeding

This message may be sent by the called user to indicate that requested call establishment has been initiated and no more call establishment information will be accepted. See Table 7.

Table 7/H.225.0 – Call Proceeding

Information element	H.225.0 status (M/F/O)	Length in H.225.0
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Bearer capability	O	5-6
Extended facility	O	8-*
Channel identification	FFS	NA
Facility	O	8-*
Progress indicator	O	2-4
Notification indicator	O	2-*
Display	O	2-82
High layer compatibility	FFS	NA
User-to-User	M	2-131

The user-to-user information element contains the CallProceeding-UUIE defined in the H.225.0 Message Syntax. The CallProceeding-UUIE includes the following:

protocolIdentifier – Set to the version of H.225.0 supported.

destinationInfo – Contains an **EndpointType** to allow the caller to determine whether the call involves a gateway or not.

h245Address – This is a specific transport address on which the called endpoint or gatekeeper handling the call would like to establish H.245 signalling.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

h245SecurityMode – An H.323 entity that receives a Setup message with the **h245SecurityCapability** set shall respond with the corresponding, acceptable **h245SecurityMode** in the CallProceeding, Alerting, Progress, or Connect.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

fastStart – Used only in the fast connect procedure, **fastStart** supports the signalling needed to open a logical channel. This uses the OpenLogicalChannel structure defined in Recommendation H.245, but the sender of this indicates the modes it prefers to receive and transmit, and the transport addresses where it expects to receive media streams.

7.3.3 Connect

This message shall be sent by the called entity to the calling entity (gatekeeper, gateway, or calling terminal) to indicate acceptance of the call by the called entity. Follow Table 3-4/Q.931, as modified in Table 8 below.

Table 8/H.225.0 – Connect

Information element	H.225.0 status (M/F/O)	Length in H.225.0
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Bearer capability	O (Note)	5-6
Extended facility	O	8-*
Channel identification	FFS	NA
Facility	O	8-*
Progress indicator	O	2-4
Notification indicator	O	2-*
Display	O	2-82
Date/Time	O	8
High layer compatibility	FFS	NA
Low layer compatibility	FFS	NA
User-to-User	M	2-131
NOTE – Bearer capability is mandatory if the message is between a terminal and a gateway.		

The user-to-user information element contains the Connect-UUIE defined in the H.225.0 Message Syntax. The Connect-UUIE includes the following:

protocolIdentifier – Set by the called endpoint to the version of H.225 supported.

h245Address – This is a specific transport address on which the called endpoint or gatekeeper handling the call would like to establish H.245 signalling. This address shall be sent if sent earlier in Alerting, Progress, or Call Proceeding.

destinationInfo – Contains an **EndpointType** to allow the caller to determine whether the call involves a gateway or not.

conferenceID – Will contain a unique number to allow the conference to be uniquely identified from all others as received in the Setup.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

h245SecurityMode – An H.323 entity that receives a Setup message with the **h245SecurityCapability** set shall respond with the corresponding, acceptable **h245SecurityMode** in the CallProceeding, Alerting, Progress, or Connect.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

fastStart – Used only in the fast connect procedure, **fastStart** supports the signalling needed to open a logical channel. This uses the OpenLogicalChannel structure defined in Recommendation H.245, but the sender of this indicates the modes it prefers to receive and transmit, and the transport addresses where it expects to receive media streams.

7.3.4 Connect Acknowledge

Follow Table 3-5/Q.931 as modified in Table 9 below.

This message shall not be sent.

Table 9/H.225.0 – Connect Acknowledge

Information element	H.225.0 status (M/F/O)	Length in H.225.0
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Display	O	2-82
Signal	O	2-3
User-to-User	M	2-131

7.3.5 Disconnect

This message shall not be sent by an H.323 entity.

The contents and semantics of a DISCONNECT message received from the network are defined in Table 3-6/Q.931 and in 10.5 of ISO/IEC 11582.

7.3.6 User Information

This message may be sent to provide additional information. It may be used to provide information for call establishment (e.g. overlap sending) or miscellaneous call-related information. It may be used to deliver proprietary features.

This message may be sent by an H.323 entity; its processing on receipt is optional.

This message follows Table 3-7/Q.931 with the following modifications (see Table 10).

Table 10/H.225.0 – Information Message Content

Information element	H.225.0 status (M/F/O)	Length in H.225.0
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Sending complete	O	1
Display	O	2-82
Keypad facility	O	2-34
Signal	O	2-3
Called party number	O	2-35
User-to-User	M	2-131

The user-to-user information element contains the Information UUIE defined in the H.225.0 Message Syntax. The Information UUIE includes the following:

protocolIdentifier – Set to the version of H.225 supported.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

7.3.7 Progress

This message may be sent by an H.323 gateway to indicate the progress of a call in the event of interworking with SCN. This message may also be sent by an H.323 endpoint before the Connect message, depending on supplementary service interaction.

Follow Table 3-9/Q.931 and 10.10 of ISO/IEC 11582 as modified in Table 11 below.

Table 11/H.225.0 – Progress

Information element	H.225.0 status (M/F/O)	Length in H.225.0
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Bearer capability	O (Note)	5-6
Cause	O	2-32
Extended facility	O	8-*
Channel identification	FFS	NA
Facility	O	8-*
Progress indicator	O	2-4
Notification indicator	O	2-*
Display	O	2-82
High layer compatibility	FFS	NA
User-to-User	M	2-131
NOTE – The Bearer capability information element is mandatory if the message is between a terminal and a gateway.		

The user-to-user information element contains the Progress-UUIE defined in the H.225.0 Message Syntax. The Progress-UUIE includes the following:

protocolIdentifier – Set to the version of H.225.0 supported.

destinationInfo – Contains an **EndpointType** to allow the caller to determine whether the call involves a gateway or not.

h245Address – This is a specific transport address on which the called endpoint or gatekeeper handling the call would like to establish H.245 signalling. This address shall be sent if sent earlier in Call Proceeding, Alerting, or Connect.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

h245SecurityMode – An H.323 entity that receives a Setup message with the **h245SecurityCapability** set shall respond with the corresponding, acceptable **h245SecurityMode** in the CallProceeding, Alerting, Progress, or Connect.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

fastStart – Used only in the fast connect procedure, **fastStart** supports the signalling needed to open a logical channel. This uses the OpenLogicalChannel structure defined in Recommendation H.245, but the sender of this indicates the modes it prefers to receive and transmit, and the transport addresses where it expects to receive media streams.

7.3.8 Release

This message shall not be sent by an H.323 entity.

The contents and semantics of a RELEASE message received are defined in Table 3-10/Q.931 and in 10.5 of ISO/IEC 11582.

7.3.9 Release Complete

This message shall be sent by a terminal to indicate release of the call if the reliable call signalling channel is open. Afterwards, the Call Reference Value (CRV) is available for reuse.

The disconnect/release/release complete sequence is not used since the only added value is that a network-to-user information element can be appended to the release message. As this does not apply to the packet-based network environment, the single step method of sending only Release Complete is used.

Follow Table 3-11/Q.931. Table 12 modifications apply.

Table 12/H.225.0 – Release Complete

Information element	H.225.0 status (M/F/O)	Length in H.225.0
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Cause	CM (Note)	1
Facility	O	8-*
Notification indicator	O	2-*
Display	O	2-82
Signal	O	2-3
User-to-User	M	2-131
NOTE – Either the Cause IE or the **ReleaseCompleteReason** shall be present.		

If this message is sent in response to a Facility message with an empty Facility IE, the ReleaseCompleteReason shall be set to **facilityCallDeflection**.

If this message is forwarded from a SCN by a gateway, the cause value shall be set as specified in Recommendation Q.931.

The user-to-user information element contains the ReleaseComplete-UUIE defined in the H.225.0 Message Syntax. The ReleaseComplete-UUIE includes the following:

protocolIdentifier – Set to the version of H.225.0 supported.

reason – More information on why the call was released.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

7.3.10 Setup

This message shall be sent by a calling H.323 entity to indicate its desire to set up a connection to the called entity.

Follow Table 3-16/Q.931 as modified in Table 13.

Table 13/H.225.0 – Setup

Information element	H.225.0 status(M/F/O/CM)	Length in H.225.0
Protocol discriminator	M	1
Call reference	M (Note 2)	3
Message type	M	1
Sending complete	O	1
Repeat indicator	F	NA
Bearer capability	M	5-6
Extended facility	O	8-*
Channel identification	FFS	NA
Facility	O	8-*
Progress indicator	F	NA
Network specific facilities	F	NA
Notification indicator	O	2-*
Display	O	2-82
Keypad facility	O	2-34
Signal	O	2-3
Calling party number	O	2-131
Calling party subaddress	CM (Note 1)	NA
Called party number	O	2-131
Called party subaddress	CM (Note 1)	NA
Transit network selection	F	NA
Repeat indicator	F	NA
Low layer compatibility	FFS	NA
High layer compatibility	FFS	NA
User-to-User	M	2-131
NOTE 1 – Subaddresses are needed for some SCN call scenarios; they should not be used for packet-based network side only calls. NOTE 2 – If an ARQ was previously sent, the CRV used here shall be the same.		

The user-to-user information element contains the Setup-UUIE defined in the H.225.0 Message Syntax. The Setup-UUIE includes the following:

protocolIdentifier – Set to the version of H.225.0 supported.

h245Address – This is a specific transport address on which the calling endpoint or gatekeeper handling the call would like to establish the H.245 signalling. This should only be provided by the sender if it is capable of handling H.245 procedures before receiving a CONNECT on the Call Signalling channel.

sourceAddress – Contains the alias addresses for the source; the E.164 number of the source is in the Q.931 Calling Party Number IE. The primary address shall be first.

sourceInfo – Contains an **EndpointType** to allow the called party to determine whether the call involves a gateway or not.

destinationAddress – This is the address to which the endpoint wishes to be connected. The primary address shall be first. When calling an endpoint using only an E.164 address, this address shall be placed in the Q.931 Called Party Number IE. The destinationAddress, if available, shall be included in the Setup message by version 2 terminals.

destCallSignalAddress – Needed to inform the gatekeeper of the destination terminal's call signalling transport address; redundant in the direct terminal-to-terminal case. In all cases where the information is available to the sender of the Setup message, this field shall be filled in.

destExtraCallInfo – Needed to make possible additional channel calls, i.e. for a 2*64 kbit/s call on the WAN side. Shall only contain E.164 addresses and shall not contain the number of the initial channel. (See Note.)

destExtraCRV – CRVs for the additional SCN calls specified by **destExtraCallInfo.** Their use is for further study. They can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

activeMC – Indicates that the calling endpoint is under the influence of an active MC.

conferenceID – Unique conference identifier.

conferenceGoal:

create – Start a new conference.

invite – Invite a party into an existing conference.

join – Join an existing conference.

capability-negotiation – Negotiate capabilities for a later loosely-coupled conference.

callIndependentSupplementaryService – Transport of supplementary services APDUs in a non-call related manner.

callServices – Provides information on support of optional Q-series protocols to gatekeeper and called terminal.

callType – Using this value, called party's gatekeeper can attempt to determine 'real' bandwidth usage. The default value is **pointToPoint** for all calls; it should be recognized that the call type may change dynamically during the call and that the final call type may not be known when the Setup is sent.

sourceCallSignalAddress – Contains the transport address for the source; this value shall be used in the ARQ message by the receiver of the Setup. In all cases where the information is available to the sender of the Setup message, this field shall be filled in. The value of sourceCallSignalAddress shall be equal to the value that was used in the ARQ by the sender of the Setup, and shall be echoed by the endpoint receiving the Setup in its ARQ.

remoteExtensionAddress – Contains the alias address of a called endpoint in cases where this information is needed to traverse multiple Gateways. In all cases where the information is available to the sender of the Setup message, this field shall be filled in.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

h245SecurityCapability – A set of capabilities the sender can use to secure the H.245 channel.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

fastStart – Used only in the fast connect procedure, **fastStart** supports the signalling needed to open a logical channel. This uses the OpenLogicalChannel structure defined in Recommendation H.245, but the sender of this indicates the modes it prefers to receive and transmit, and the transport addresses where it expects to receive media streams.

mediaWaitForConnect – If TRUE, indicates that the recipient of the Setup message shall not transmit media until sending the Connect message.

canOverlapSend – If TRUE, indicates that the sender of Setup shall support overlap sending.

NOTE – If the **destExtraCallInfo** is present, a CRV for each call to be made may be supplied in **destExtraCRV**. These CRVs will be used to identify any response to each call launched. These procedures are for further study. If the destExtraCRV field is not present, a gateway shall aggregate all call information into a single response, with the effect that if one call fails on the SCN side, the entire call is treated as a failure.

7.3.11 Setup Acknowledge

This message may be sent by an H.323 entity. However, it may be forwarded from the network via a gateway. Processing on receipt is optional, but an entity that indicates canOverlapSend in Setup shall support Setup Acknowledge.

The contents and semantics of a SETUP ACKNOWLEDGE message received from the network are defined in Table 3-16/Q.931.

7.3.12 Status

The STATUS message shall be used to respond to an unknown call signalling message or to a STATUS INQUIRY message.

Follow Table 3-17/Q.931 with the single modification that the CRV is of length 2 octets.

7.3.13 Status Inquiry

The STATUS INQUIRY message may be used to request call status as described in 8.4.2/H.323.

Follow Table 3-18/Q.931 with the single modification that the call reference IE is of length 3 octets.

7.4 Q.932 message details

The messages defined in the following are derived from Recommendations Q.932 and H.450. Refer to Recommendations Q.932 and H.450 for further details.

7.4.1 Facility

The Facility message shall be used to provide information on where a call should be directed (FacilityReason = routeCallToMC), or for an endpoint to indicate that the incoming call must go through a gatekeeper (FacilityReason = routeCallToGatekeeper).

In order to signal call redirection specific to H.323 procedures, the User-to-User Information Element of the Facility message is used. This particular case shall be indicated by coding a Facility IE of length zero. In this case, the Facility Information Element shall consist of exactly 2 octets. An H.323 entity shall handle the empty (H.323-specific) Facility IE properly and shall be capable of skipping other Facility IEs that it does not understand.

The Facility message may be used to request or acknowledge a supplementary service according to H.450-series Recommendations. For that reason, one or more H.450 Supplementary Service APDUs shall be carried within the User-to-User information element of the Facility message. The H.450 Supplementary Service APDUs shall be coded according to clause 8/H.450.1. The Facility information element shall be contained with length zero. Note that a Facility message that carries

only H.450 Supplementary Service APDUs would not use the Facility-UUIE, but would instead use the "empty" **h323-message-body** choice.

If a Facility IE carrying semantics of Recommendation Q.932 and encoded as defined in Recommendations Q.932 and Q.95x is present, it shall consist of at least 8 octets as required by Table 7-2/Q.932. The use of Facility IEs of that type is for further study.

The Facility message may be used by an endpoint or gatekeeper to request the recipient to establish an H.245 channel between the two entities (FacilityReason = starth245).

Follow 7.1.1/Q.932 and 10.8 of ISO/IEC 11582, as modified in Table 14.

Table 14/H.225.0 – Facility

Information element	H.225.0 status(M/F/O)	Length in H.225.0
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Extended facility	O (Note)	8-*
Facility	O (Note)	2 or 8-*
Notification indicator	O	2-*
Display	O	2-82
Calling Party Number	F	NA
Called Party Number	F	NA
User-to-User	M	2-131
NOTE – If the Facility message is used for carrying Q.95x supplementary service signalling, one of either the Facility or Extended Facility information elements is required. If the Facility message is used for Supplementary Service control according to H.450.x-series Recommendations, or if the Facility message is used for the reroute to MC/GK functions, then the zero-length Facility information element is required.		

Coding of Message Type information element

The message type information element of the Facility message shall be coded "0110 0010".

The user-to-user information element contains the Facility-UUIE defined in the H.225.0 Message Syntax. The Facility-UUIE includes the following:

protocolIdentifier – Set to the version of H.225.0 supported.

alternativeAddress – This is a specific transport address to which the calling party should direct the call; if present **alternativeAliasAddress** is not needed.

alternativeAliasAddress – Contains aliases that can be used to re-direct the call; if an alias is provided, **alternativeAddress** is not needed.

conferenceID – Unique conference identifier; not needed if the **conferences** field is used.

reason – More information about the facility message.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

destExtraCallInfo – Needed to make possible additional channel calls, i.e. for a 2*64 kbit/s call on the WAN side. Shall only contain E.164 addresses and shall not contain the number of the initial channel.

remoteExtensionAddress – Contains the alias address of a called endpoint in cases where this information is needed to traverse multiple Gateways.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

conferences – One or more conferences that may be joined.

h245Address – This is a specific transport address on which the endpoint or gatekeeper sending this facility would like the recipient to establish H.245 signalling.

7.4.2 Notify

This message may be sent by an H.323 entity. Processing on receipt is optional.

The contents and semantics of a NOTIFY message are defined in 3.1.7/Q.931.

7.4.3 Other messages

The call control messages which can carry optional Facility, Extended Facility, or Notification Indicator information elements are specified in 8.3.

7.5 Q.931 timer values

Two Q.931 timers shall be supported:

- The "setup timer" T303 (see Tables 9-1/Q.931 and 9-2/Q.931) defining how long the calling endpoint shall wait for an ALERTING, CALL PROCEEDING, CONNECT, RELEASE COMPLETE or other message from the called endpoint after it has sent a SETUP message. This timeout value shall be at least 4 seconds. Note that some applications may appear in networks which have inherently longer delays (for example, compare the Internet to a local enterprise network or intranet).
- The "establishment timer" T301 (see Tables 9-1/Q.931 and 9-2/Q.931) defining after which time the calling endpoint shall stop waiting for the called endpoint to respond. This timer starts when ALERTING is received and normally terminates on CONNECT or when the caller terminates the call attempt and sends RELEASE COMPLETE. This timeout value shall be 180 seconds (3 minutes) or greater.

Note that the packet-based network side values of these timers is the same as that used in the SCN.

Other timers may be supported as part of optional H.450-series Supplementary Service Recommendations.

7.6 H.225.0 common message elements

This subclause describes ASN.1 structures that are used in more than one Registration, Admission, and Status (RAS) messages. Some may also be used in the User-to-User part of the Q.931 messages.

requestSeqNum in messages is used to keep track of multiple outstanding requests. Any associated response messages (success or failure) shall have the corresponding **requestSeqNum** returned with it. Retransmitted messages shall have the same **requestSeqNum**. **RequestSeqNum** increments by 1 modulo 65536.

The **protocolIdentifier** is included as part of discovery, registration and Setup/Connect to allow the parties involved to determine the vintage of the implementations involved.

nonStandardParameter: This parameter is optional in the discovery, registration, and Setup/Connect sequences to allow the parties involved to determine the non-standard status of the endpoints involved. A gatekeeper or gateway is not obligated to pass on nonStandardData it does not support or understand as this might interfere with its operations.

The **TransportAddress** structure is meant to capture the various transport formats and includes any transport-specific scheme in addition to the possibly local reference to a TSAP identifier.

IPv4 and IPv6 addresses shall be encoded with the most significant octet of the address being the first octet in the respective OCTET STRING, e.g. the class B IPv4 address 130.1.2.97 shall have the '130' being encoded in the first octet of the OCTET STRING, followed by the '1' and so forth.

The IPv6 address a148:2:3:4:a:b:c:d shall have the 'a1' encoded in the first octet, '48' in the second, '00' in the third, '02' in the fourth and so forth.

IPX addresses, **node, netnum**, and **port** shall be encoded with the most significant octet of each field being the first octet in the respective OCTET STRING.

Note that this structure does not use the Transport Address = "packet-based network Address plus TSAP identifier" language of Recommendation H.323. Instead, the terms common in each transport domain are used.

The **EndpointType** structure conveys information about the H.323 element at the end of the signalling link. The H.323 element would complete one or more of the **gatekeeper**, **gateway**, **mcu**, or **terminal** message elements. If the H.323 element has an MC, then the **mc** Boolean would be true.

The **GatewayInfo** structure contains a **protocol** element, which allows the gateway to indicate the protocols it supports.

dataRatesSupported indicates the data rates supported for each protocol the device supports. supportedPrefixes indicates the prefixes associated with a supported protocol, and in some cases also with the data rates.

The **DataRate** structure provides gateway protocol rate information. **channelRate** is the basic channel rate in hundreds of bits. **channelMultiplier** indicates the number of channels at the channelRate. For example, if a gateway supports a 3B call, channelMultiplier = 3 and channelRate = 640 for a 64 kbit/s channel.

The **VendorIdentifier** structure allows a vendor to identify a product. The **vendor** element allows identification in terms of country code, extension, and manufacturer code. **productId** and **versionId** are text strings that can provide product information.

The **AliasAddress** structure is meant to capture the various external address formats that reference a particular transport location on the packet-based network. When registering an E.164 address with a gatekeeper, an endpoint shall use only the digits 0-9 in the **e164** field.

The mechanisms that an endpoint uses to determine the address type is left as an implementation issue. The representation of the various number types in messages is captured in Table 15. Note that if an endpoint does not know the type or scope of an E.164 number, then it should represent this as Private Unknown when coded in Q.931 message and as an e164 AliasAddress when coded in RAS messages.

Table 15/H.225.0 – Type of number representation mapping

Type of number	Q.931 representation	H.225 RAS representation
Unknown (default and version 1 interoperability mode)	Private Numbering Plan – Unknown	e164 AliasAddress
Private unknown	Private Numbering Plan – Unknown	e164 AliasAddress
Private, subscriber/abbreviated, etc.	Private Numbering Plan, Type of number dependent on scope	privateNumber of PartyNumber AliasAddress
Public number, all kinds	ISDN/Telephony Numbering Plan	publicNumber of PartyNumber AliasAddress

The **Endpoint** structure is used to indicate back-up, redundant, or alternative information about an endpoint:

- **nonStandardData** – Carries information not defined in this Recommendation (for example, proprietary data).
- **aliasAddress** – This is a list of alias addresses, by which other endpoints may identify this endpoint.
- **callSignalAddress** – This is the call signalling transport address for this endpoint.
- **rasAddress** – This is the registration and status transport address for this endpoint.
- **endpointType** – This specifies the type of the endpoint.
- **tokens** – Tokens associated with this endpoint (i.e. endpoint described in the Endpoint structure).
- **cryptoTokens** – CryptonTokens associated with this endpoint (i.e. the endpoint described in the Endpoint structure).
- **priority** – Used when a SEQUENCE of Endpoints is presented. Endpoints with lower priority numbers are preferred over endpoints with higher priority numbers. Endpoints without priority numbers are equivalent to those with a priority of 0 (highest priority).
- **remoteExtensionAddress** – Contains the alias address of an endpoint in cases where this information is needed to traverse multiple gateways.
- **destExtraCallInfo** – Contains external addresses for multiple calls.

The **AlternateGK** structure is used to indicate a list of alternative, or back-up, gatekeepers:

- **rasAddress** – The transport address used for RAS signalling.
- **gatekeeperIdentifier** – Optionally included to identify the backup or alternative gatekeeper. If it is supplied, it shall be included in future RAS messages sent to the back-up gatekeeper.
- **needToRegister** – Set to TRUE to indicate that the endpoint must register with the alternative before sending other RAS requests.
- **priority** – Indicates the priority of the gatekeeper back-up or alternative. A lower number implies a higher priority.

The **AltGKInfo** structure is used to provide information about alternate gatekeepers:

- **alternateGatekeeper** – Sequence of prioritized alternateGatekeeper for gatekeeperIdentifer and rasAddress for client to retry the request.
- **altGKisPermanent** – TRUE if all future RAS signals should be redirected to an address from alternateGatekeeper, FALSE if only the message that caused the Reject should be redirected.

The **QseriesOptions** structure supplies information to the gatekeeper or other endpoints concerning the support provided by a terminal for optional Q-series protocols. It is used in the ARQ, SETUP, and RRQ messages.

The GloballyUniqueID and ConferenceIdentifier are meant to be globally unique identifiers (GloballyUniqueID), the use of which is described in Recommendation H.323. A GloballyUniqueID is encoded with octet zero being encoded first. A GloballyUniqueID is formed according to Table 16.

Table 16/H.225.0 – Globally unique ID formation

Field	Data type	Octet No.	Note
time_low	Unsigned 32-bit integer	0-3	The low field of the timestamp
time_mid	Unsigned 16-bit integer	4-5	The middle field of the timestamp
time_hi_and_version	Unsigned 16-bit integer	6-7	The high field of the timestamp multiplexed with the version number
clock_seq_hi_and_reserved	Unsigned 8-bit integer	8	The high field of the clock sequence multiplexed with the variant
clock_seq_low	Unsigned 8-bit integer	9	The low field of the clock sequence
node	Unsigned 48-bit integer	10-15	The spatially unique node identifier

The GloballyUniqueID consists of a record of 16 octets and must not contain padding between fields. The total size is 128 bits.

To minimize confusion about bit assignments within octets, the GloballyUniqueID record definition is defined only in terms of fields that are integral numbers of octets. The version number is multiplexed with the time stamp (*time_high*), and the variant field is multiplexed with the clock sequence (*clock_seq_high*).

The timestamp is a 60-bit value represented by Coordinated Universal Time (UTC) as a count of 100 nanosecond intervals since 00:00:00.00, 15 October 1582 (the date of Gregorian reform to the Christian calendar).

The version number is multiplexed in the 4 most significant bits of the *time_hi_and_version* field, and is set to 1 (binary 0001).

The variant field determines the layout of the GloballyUniqueID. The structure of a DCE GloballyUniqueID is fixed across different versions. Other GloballyUniqueID variants may not interoperate with a DCE GloballyUniqueID. Interoperability of GloballyUniqueIDs is defined as the applicability of operations such as string conversion, comparison, and lexical ordering across different systems. The *variant* field consists of a variable number of the MSBs of the *clock_seq_hi_and_reserved* field (see Table 17).

Table 17/H.225.0 – Contents of the DCE variant field

msb1	msb2	msb3	Description
0	-	-	Reserved, NCS backward compatibility
1	0	-	DCE variant
1	1	0	Reserved, Microsoft Corporation GUID
1	1	1	Reserved for future definition

The clock sequence is required to detect potential losses of monotonicity of the clock. The clock sequence is encoded in the 6 least significant bits of the *clock_seq_hi_and_reserved* field and in the *clock_seq_low* field.

The *node* field consists of the IEEE address, usually the host address. For systems with multiple IEEE 802 nodes, any available node address can be used. The lowest addressed octet (octet number 10) contains the global/local bit and the unicast/multicast bit, and is the first octet of the address transmitted on an 802.3 packet-based network.

The clock sequence value should be changed whenever:

- The GloballyUniqueID generator detects that the local value of UTC has gone backward; this may be due to normal functioning of the DCE Time Service.
- The GloballyUniqueID generator has lost its state of the last value of UTC used, indicating that time may have gone backward; this is typically the case on reboot.

While a node is operational, the GloballyUniqueID generator always saves the last UTC used to create a GloballyUniqueID. Each time a new GloballyUniqueID is created, the current *UTC* is compared to the saved value and if either the current value is less (the non-monotonic clock case) or the saved value was lost, then the *clock sequence* is incremented modulo 16 384, thus avoiding production of duplicate GloballyUniqueIDs.

The *clock sequence* should be initialized to a random number to minimize the correlation across systems.

A GloballyUniqueID is generated according to the following algorithm:

1) Determine the values for the UTC-based timestamp and clock sequence to be used in the GloballyUniqueID.
2) Set the *time_low* field equal to the least significant 32-bits (bits numbered 0 to 31 inclusive) of the time stamp in the same order of significance.
3) Set the *time_mid* field equal to the bits numbered 32 to 47 inclusive of the time stamp in the same order of significance.
4) Set the 12 least significant bits (bits numbered 0 to 11 inclusive) of the *time_hi_and_version* field equal to the bits numbered 48 to 59 inclusive of the time stamp in the same order of significance.
5) Set the 4 most significant bits (bits numbered 12 to 15 inclusive) of the *time_hi_and_version* field to the 4-bit version number corresponding to the GloballyUniqueID version being created, as shown in Table 17.
6) Set the *clock_seq_low* field to the 8 least significant bits (bits numbered 0 to 7 inclusive) of the *clock sequence* in the same order of significance.
7) Set the 6 least significant bits (bits numbered 0 to 5 inclusive) of the *clock_seq_hi_and_reserved* field to the 6 most significant bits (bits numbered 8 to 13 inclusive) of the *clock sequence* in the same order of significance.

8) Set the 2 most significant bits (bits numbered 6 and 7) of the *clock_seq_hi_and_reserved* to 0 and 1, respectively.

9) Set the *node* field to the 48-bit IEEE address in the same order of significance as the address.

If a system wants to generate a GloballyUniqueID but has no IEEE 802 compliant network card or other source of IEEE 802 addresses, then an alternative method should be used to generate a replacement value for the address. The ideal solution is to obtain a 47-bit cryptographic quality random number, and use it as the most significant 47 bits of the node ID, with the least significant bit of the first octet of the node ID set to 1. This bit is the unicast/multicast bit, which will never be set in IEEE 802 addresses obtained from network cards; hence, there can never be a conflict between GloballyUniqueIDs generated by machines with and without network cards.

If a system does not have a primitive to generate cryptographic quality random numbers, then in most systems there are usually a fairly large number of sources of randomness available from which one can be generated. Such sources are system specific, but often include the percent of memory in use, the size of main memory in bytes, the amount of free main memory in bytes, the size of the paging or swap file in bytes, free bytes of paging or swap file, the total size of user virtual address pace in bytes, the total available user address space bytes, the size of boot disk drive in bytes, the free disk space on boot drive in bytes, the current time, the amount of time since the system booted, the individual sizes of files in various system directories, etc.

For use in human readable text, a GloballyUniqueID string representation is specified as a sequence of fields, some of which are separated by single dashes.

Each field is treated as an integer and has its value printed as a zero-filled hexadecimal digit string with the most significant digit first. The hexadecimal values a to f inclusive are output as lower case characters, and are case insensitive on input. The sequence is the same as the GloballyUniqueID constructed type.

The formal definition of the GloballyUniqueID string representation is provided by the following extended BNF:

```
UUID                    = <time_low> <hyphen> <time_mid> <hyphen>
                          <time_high_and_version> <hyphen>
                          <clock_seq_and_reserved>
                          <clock_seq_low> <hyphen> <node>
time_low                = <hexOctet> <hexOctet> <hexOctet> <hexOctet>
time_mid                = <hexOctet> <hexOctet>
time_high_and_version   = <hexOctet> <hexOctet>
clock_seq_and_reserved  = <hexOctet>
clock_seq_low           = <hexOctet>
node                    = <hexOctet><hexOctet><hexOctet>
                          <hexOctet><hexOctet><hexOctet>
hexOctet                = <hexDigit> <hexDigit>p
hexDigit                = <digit> | <a> | <b> | <c> | <d> | <e> | <f>
digit                   = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                          "8" | "9"
hyphen                  = "-"
a                       = "a" | "A"
b                       = "b" | "B"
c                       = "c" | "C"
d                       = "d" | "D"
e                       = "e" | "E"
f                       = "f" | "F"
```

The following is an example of the string representation of a GloballyUniqueID:

f81d4fae-7dec-11d0-a765-00a0c91e6bf6

TimeToLive is a number seconds that a registration is to be considered valid.

7.7 Required Support of RAS messages

Table 18 shows the RAS messages that are supported by different endpoint types.

Table 18/H.225.0 – Status of RAS messages

RAS Message	Endpoint (Tx)	Endpoint (Rx)	Gatekeeper (Tx)	Gatekeeper (Rx)
GRQ	O			M
GCF		O	M	
GRJ		O	M	
RRQ	M			M
RCF		M	M	
RRJ		M	M	
URQ	O	M	O	M
UCF	M	O	M	O
URJ	O	O	M	O
ARQ	M			M
ACF		M	M	
ARJ		M	M	
BRQ	M	M	O	M
BCF	M (Note 1)	M	M	O
BRJ	M	M	M	O
IRQ		M	M	
IRR	M			M
IACK		O	CM	
INAK		O	CM	
DRQ	M	M	O	M
DCF	M	M	M	M
DRJ	M (Note 2)	M	M	M
LRQ	O		O	M
LCF		O	M	O
LRJ		O	M	O
NSM	O	O	O	O
XRS	M	M	M	M
RIP	CM	M	CM	M
RAI	O			M
RAC		O	M	

M Mandatory, O Optional, F Forbidden, CM Conditionally Mandatory, blank "Not Applicable".

NOTE 1 – If a gatekeeper sends a BRQ requesting a lower rate, the endpoint shall reply with BCF if the lower rate is supported, otherwise with BRJ. If a gatekeeper sends a BRQ requesting a higher rate, the endpoint may reply with BCF or BRJ.

NOTE 2 – Terminal shall not send DRJ while on a call in response to DRQ from a gatekeeper.

7.8 Terminal and Gateway Discovery messages

The GRQ message requests that any gatekeeper receiving it respond with a GCF granting it permission to register. The GRJ is a rejection of this request indicating that the requesting endpoint should seek another gatekeeper.

7.8.1 GatekeeperRequest (GRQ)

Note that one GRQ is sent per logical endpoint; thus an MCU or a Gateway might send many.

The GRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

protocolIdentifier – Identifies the H.225.0 vintage of the sending endpoint.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

rasAddress – This is the transport address that this endpoint uses for registration and status messages.

endpointType – This specifies the type(s) of the endpoint that is registering (the MC bit shall not be set by itself).

gatekeeperIdentifier – String to identify the gatekeeper from which the terminal would like to receive permission to register. A missing or null string **gatekeeperIdentifier** indicates that the terminal is interested in any available gatekeeper.

callServices – Provides information on support of optional Q-series protocols to gatekeeper and called terminal.

endpointAlias – A list of alias addresses, by which other terminals may identify this terminal.

alternateEndpoints – A sequence of prioritized endpoint alternatives for rasAddress, endpointType, or endpointAlias.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

authenticationCapability – This indicates the authentication mechanisms supported by the endpoint.

algorithmOIDs –

integrity – Indicates to the recipient which integrity mechanism is to be applied on the RAS messages.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.8.2 GatekeeperConfirm (GCF)

The GCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the GRQ.

protocolIdentifier – Identifies the vintage of the accepting gatekeeper.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

gatekeeperIdentifier – String to identify gatekeeper that is sending the GCF.

rasAddress – This is the transport address that the gatekeeper uses for registration and status messages.

alternateGatekeeper – Sequence of prioritized alternatives for gatekeeperIdentifier and rasAddress. The client should use these alternatives in the future, should a request to the gatekeeper not respond or return a reject without redirect.

authenticationMode – This indicates the authentication mechanism to be used. The gatekeeper must choose **authenticationMode** from **authenticationCapability** provided by the endpoint in GRQ.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

algorithmOID –

integrity – Indicates to the recipient which integrity mechanism is to be applied on the RAS messages.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.8.3 GatekeeperReject (GRJ)

The GRJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the GRQ.

protocolIdentifier – Identifies the vintage of the rejecting gatekeeper.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

gatekeeperIdentifier – String to identify gatekeeper that is sending the GRJ.

rejectReason – Codes for why the GRQ was rejected by this gatekeeper.

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.9 Terminal and Gateway Registration messages

The RRQ is a request from a terminal to a gatekeeper to register. If the gatekeeper responds with a RCF, the terminal shall use the responding gatekeeper for future calls. If the gatekeeper responds with a RRJ, the terminal must seek another gatekeeper to register with.

7.9.1 RegistrationRequest (RRQ)

The RRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any response associated with this specific message.

protocolIdentifier – Identifies the H.225.0 vintage of the sending endpoint.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

discoveryComplete – Set to TRUE if the requesting endpoint has preceded this message with the gatekeeper discovery procedure; set to FALSE if registering only. Note that registration may age, and the endpoint will get a failure on an RRQ or ARQ with a reason code of **discoveryRequired** or **notRegistered** respectively. This indicates that the endpoint should perform the discovery procedure (either dynamic or static) before issuing the RRQ with **discoveryComplete** set to TRUE.

callSignalAddress – This is the call signalling transport address for this endpoint. If multiple transports are supported, they must be registered all at once.

rasAddress – This is the registration and status transport address for this endpoint.

terminalType – This specifies the type(s) of the endpoint that is(are) registering; note that the MC bit shall not be set by itself; either the terminal, MCU, gateway, or gatekeeper bit shall also be set. If vendor information is provided, this information shall be identical to that in **endpointVendor**.

terminalAlias -This optional value is a list of alias addresses, by which other terminals may identify this terminal. If the **terminalAlias** is null, or an E.164 address is not present, an E.164 address may be assigned by the gatekeeper, and included in the RCF. If an email-ID is available for the endpoint, it should be registered. Note that multiple alias addresses may refer to the same transport addresses. All of the endpoint's aliases shall be included in each RRQ.

gatekeeperIdentifier – String to identify the gatekeeper that the terminal wishes to register with.

endpointVendor – Information about the endpoint vendor.

alternateEndpoints – A sequence of prioritized endpoint alternatives for callSignalAddress, rasAddress, terminalType, or terminalAlias.

timeToLive – Duration of the validity of the registration, in seconds. After this time the gatekeeper may consider the registration stale.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

keepAlive – If set to TRUE indicates that the endpoint has sent this RRQ as a "keep alive". An endpoint can send a lightweight RRQ consisting of only keepAlive, endpointIdentifier, gatekeeperIdentifier, tokens, and timeToLive. A gatekeeper in receipt of RRQ with a keepAlive field set to TRUE should ignore fields other than endpointIdentifier, gatekeeperIdentifier, tokens, and timeToLive.

endpointIdentifier – The endpointIdentifier provided by the gatekeeper during the original RCF.

willSupplyUUIEs – If set to TRUE, this indicates that the endpoint will supply Q.931 message information in IRR messages if requested by the gatekeeper.

7.9.2 RegistrationConfirm (RCF)

The RCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the RRQ.

protocolIdentifier – Identifies the vintage of the accepting gatekeeper.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

callSignalAddress – This is an array of transport addresses for H.225.0 call signalling messages; one for each transport that the gatekeeper will respond to. This address includes the TSAP identifier.

terminalAlias – This optional value is a list of alias addresses, by which other terminals may identify this terminal.

gatekeeperIdentifier – String to identify the gatekeeper that has accepted the terminals registration.

endpointIdentifier – A gatekeeper assigned terminal identity string; shall be echoed in subsequent RAS messages.

alternateGatekeeper – Sequence of prioritized alternateGatekeeper for gatekeeperIdentifer and rasAddress. The client should use these alternateGatekeeper in the future, should a request to the gatekeeper not respond or return a reject without redirect.

timeToLive – Duration of the validity of the registration, in seconds. After this time the gatekeeper may consider the registration stale.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

willRespondToIRR – True if the Gatekeeper will send an IACK or INAK message in response to an unsolicited IRR message with its **needsResponse** field set to TRUE.

preGrantedARQ – Indicates events for which the gatekeeper has pre-granted admission. This allows for faster call setup times in environments where admission is guaranteed through means other than the ARQ/ACF exchange. Note that even if these fields are set to TRUE, an endpoint can still send an ARQ to the gatekeeper for reasons such as address translation, or the endpoint does not support this modified signalling mode. If the **preGrantedARQ** sequence is not present, then ARQ signalling shall be used in all cases. The flags are:

makeCall – If the **makeCall** flag is TRUE then the gatekeeper has pre-granted permission to the endpoint to initiate calls without first sending an ARQ. If the **makeCall** flag is FALSE, the endpoint shall always send ARQ to get permission to make a call.

useGKCallSignalAddressToMakeCall – If the **makeCall** and **useGKCallSignalAddressToMakeCall** flags are both set to TRUE, then if the endpoint does not send an ARQ to the gatekeeper to make a call, the endpoint shall send all H.225 call signalling to the gatekeeper call signalling channel.

answerCall – If the **answerCall** flag is TRUE then the gatekeeper has pre-granted permission to the endpoint to answer calls without first sending an ARQ. If the **answerCall** flag is FALSE, the endpoint shall always send ARQ to get permission to answer a call.

useGKCallSignalAddressToAnswer – If the **answerCall** and **useGKCallSignalAddressToAnswer** flags are both set to true, then when an endpoint does not send an ARQ to the gatekeeper to answer a call, the endpoint shall ensure that all H.225.0 call signalling comes from the gatekeeper. If an endpoint has been instructed to use the gatekeeper when answering, but it does not know whether an incoming call has come from the gatekeeper (which may involve looking at the transport address), the endpoint shall issue ARQ irrespective of the state of the **useGKCallSignalAddressToAnswer** flag.

7.9.3 RegistrationReject (RRJ)

The RRJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the RRQ.

protocolIdentifier – Identifies the vintage of the rejecting gatekeeper.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

rejectReason – The reason for the rejection of the registration.

gatekeeperIdentifier – String to identify the gatekeeper that has rejected the terminal's registration.

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.10 Terminal/Gatekeeper Unregistration messages

7.10.1 UnregistrationRequest (URQ)

The URQ requests that the association between a terminal and a gatekeeper be broken. Note that unregister is bidirectional, i.e. a Gatekeeper can request a terminal to consider itself unregistered, and a terminal can inform a Gatekeeper that it is revoking a previous registration.

The URQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any response associated with this specific message.

callSignalAddress – This is one or more of the transport call signalling addresses for this endpoint which are to be unregistered.

endpointAlias – This optional value is a list of alias addresses, by which other terminals may identify this terminal. If this optional field is not present, all aliases are unregistered in a single message. The E.164 address, if assigned, is required. Only values listed here are unregistered; this allows, for example, an H323_ID to be unregistered while leaving the E.164 address registered.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

endpointIdentifier – Confirmation of identity; not sent by the gatekeeper.

alternateEndpoints – A sequence of prioritized endpoint alternatives for callSignalAddress or endpointAlias.

gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous URJ message. Used as a backup if the original gatekeeper did not respond or rejected the request.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

reason – Used when the gatekeeper sends the URQ to indicate why the gatekeeper considers the endpoint unregistered.

7.10.2 UnregistrationConfirm (UCF)

The UCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the URQ.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.10.3 UnregistrationReject (URJ)

The URJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the URQ.

rejectReason – The reason for the rejection of the unregistration.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.11 Terminal to Gatekeeper Admission messages

The ARQ message requests that an endpoint be allowed access to the packet-based network by the gatekeeper, which either grants the request with an ACF or denies it with an ARJ.

7.11.1 AdmissionRequest (ARQ)

The ARQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

callType – Using this value, gatekeeper can attempt to determine "real" bandwidth usage. The default value is **pointToPoint** for all calls. It should be recognized that the call type may change dynamically during the call and that the final call type may not be known when the ARQ is sent.

callModel – If **direct**, the endpoint is requesting the direct terminal to terminal call model. If **gatekeeperRouted**, the endpoint is requesting the gatekeeper mediated model. The gatekeeper is not required to comply with this request.

endpointIdentifier – This is an endpoint identifier that was assigned to the terminal by RCF.

destinationInfo – Sequence of alias addresses for the destination, such as E.164 addresses or H323_IDs. When sending the ARQ to answer a call, destinationInfo indicates the destination of the call (the answering endpoint).

destCallSignalAddress – Transport address used at the destination for call signalling.

destExtraCallInfo – Contains external addresses for multiple calls.

srcInfo – Sequence of alias addresses for the source endpoint, such as E.164 addresses or H323_IDs. When sending the ARQ to answer a call, srcInfo indicates the originator of the call.

srcCallSignalAddress – Transport address used at the source for call signalling.

bandWidth – The number of 100 bits requested for the bidirectional call. For example, a 128 kbit/s call would be signalled as a request for 256 kbit/s. The value refers only to the audio and video bit rate excluding headers and overhead.

callReferenceValue – The CRV from Q.931 for this call; only local validity. This is used by a gatekeeper to associate the ARQ with a particular call.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

callServices – Provides information on support of optional Q-series protocols to gatekeeper and called terminal.

conferenceID – Unique conference identifier.

activeMC – If TRUE, the calling party has an active MC; otherwise FALSE.

answerCall – Used to indicate to a gatekeeper that a call is incoming.

canMapAlias – If set to TRUE indicates that if the resulting ACF contains **destinationInfo**, **destExtraCallInfo** and/or **remoteExtension** fields, the endpoint can copy this information to the **destinationAddress**, **destExtraCallInfo** and **remoteExtensionAddress** fields of the SETUP message respectively. If the GK would replace addressing information from the ARQ and **canMapAlias** is FALSE, then the gatekeeper should reject the ARQ.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

srcAlternatives – A sequence of prioritized source endpoint alternatives for srcInfo, srcCallSignalAddress, or rasAddress.

destAlternatives – A sequence of prioritized destination endpoint alternatives for destinationInfo or destCallSignalAddress.

gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous ARJ message. Used as a backup if the original gatekeeper did not respond or rejected the request.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

transportQOS – An endpoint may use this to indicate its capability to reserve transport resources.

willSupplyUUIEs – If set to TRUE, this indicates that the endpoint will supply Q.931 message information in IRR messages if requested by the gatekeeper.

The TransportQOS structure includes the following:

endpointControlled – The endpoint will apply its own reservation mechanism.

gatekeeperControlled – The gatekeeper will perform resource reservation on behalf of the endpoint.

noControl – No resource reservation is needed.

NOTE – Both **destinationInfo** and **destCallSignalAddress** are not required, but at least one shall be present unless the endpoint is answering a call. There is no absolute rule over which is preferred as this may be site specific, but the E.164 address should be provided if available. It is cautioned that the best results will be obtained by considering the nature of the transport protocols in use.

7.11.2 AdmissionConfirm (ACF)

The ACF message includes the following:

requestSeqNum – This shall be the same value that was passed in the ARQ.

bandWidth – The allowed maximum bandwidth for the call; may be less than that requested.

callModel – Tells terminal whether call signalling sent on destCallSignalAddress goes to a gatekeeper or to a terminal. A value of **gatekeeperRouted** indicates that call signalling is being passed via the gatekeeper, while **direct** indicates that the endpoint-to-endpoint call mode is in use.

destCallSignalAddress – The transport address to which to send Q.931 call signalling, but may be an endpoint or gatekeeper address depending on the call model in use.

irrFrequency – The frequency, in seconds, that the endpoint shall send IRRs to the gatekeeper while on a call, including while on hold. If not present, the endpoint does not send IRRs while active on a call, and it is expected that the gatekeeper will poll the endpoint.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

destinationInfo – The address of the initial channel, used when calling through a gateway.

destExtraCallInfo – Needed to make possible additional channel calls, i.e. for a 2*64 kbit/s call on the WAN side. Shall only contain E.164 addresses and shall not contain the number of the initial channel.

destinationType – This specifies the type of the destination endpoint.

remoteExtensionAddress – Contains the alias address of a called endpoint in cases where this information is needed to traverse multiple Gateways.

alternateEndpoints – A sequence of prioritized endpoint alternatives for destCallSignalAddress or destinationInfo.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

TransportQOS – The gatekeeper may indicate to the endpoint where the responsibility lies for resource reservation. If the gatekeeper received a TransportQOS in ARQ, then it should include TransportQOS (possibly modified according to gatekeeper implementation) in ACF.

willRespondToIRR – TRUE if the Gatekeeper will send an IACK or INAK message in response to an unsolicited IRR message when the IRR's **needsResponse** field set to TRUE.

uuiesRequested – The gatekeeper may request the endpoint to notify the gatekeeper of H.225.0 call signalling messages that the endpoint sends or receives if the endpoint indicated this capability in the ARQ by setting **willSupplyUUIEs** to TRUE. **uuiesRequested** indicates the set of H.225.0 call signalling messages of which the endpoint shall notify the gatekeeper.

7.11.3 AdmissionReject (ARJ)

The ARJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the ARQ.

rejectReason – Reason the admission request was denied.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

callSignalAddress – This is the gatekeeper's call signalling address returned when the reject reason is routeCallToGatekeeper.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.12 Terminal to gatekeeper requests for changes in bandwidth

The BRQ message requests that an endpoint be granted a changed packet-based network bandwidth allocation by the gatekeeper, which either grants the request with a BCF or denies it with a BRJ.

The gatekeeper may request that an endpoint raise or lower the bandwidth in use with a BRQ. If the request is to raise the rate, the endpoint may reply with either BRJ or BCF. If the request is for a lower rate, the endpoint shall reply with a BCF if the lower rate is supported, otherwise with BRJ.

7.12.1 BandwidthRequest (BRQ)

The BRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

endpointIdentifier – This is an endpoint identifier that was assigned to the terminal by RCF.

conferenceID – ID of the call that is to have the bandwidth changed.

callReferenceValue – The CRV from Q.931 for this call; only local validity. This is used by a gatekeeper to associate the BRQ with a particular call.

callType – Using this value, gatekeeper can attempt to determine "real" bandwidth usage.

bandWidth – The NEW number of 100 bits increments requested for the call. This is an absolute value that includes only audio and video bitstreams not counting headers and overhead.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous BRJ message. Used as a back-up if the original gatekeeper did not respond or rejected the request.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

answeredCall – Set to TRUE to indicate that this party was the original destination (this party answered the call).

7.12.2 BandwidthConfirm (BCF)

The BCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the BRQ.

bandWidth – The maximum allowed at this time in increments of 100 bits.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.12.3 BandwidthReject (BRJ)

The BRJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the BRQ.

rejectReason – The reason the change was rejected by the gatekeeper.

allowedBandWidth – The maximum allowed at this time in increments of 100 bits including the current allocation.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.13 Location Request messages

The LRQ requests that a gatekeeper provide address translation. The gatekeeper responds with an LCF containing the transport address of the destination, or rejects the request with LRJ.

7.13.1 LocationRequest (LRQ)

The LRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

endpointIdentifier – This is an endpoint identifier that was assigned to the terminal by RCF.

destinationInfo – Sequence of alias addresses for the destination, such as E.164 addresses or H323_IDs.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

replyAddress – Transport address to which to send the LCF/LRQ.

sourceInfo – Indicates the sender of the LRQ. The gatekeeper can use this information to decide how to respond to the LRQ.

canMapAlias – If set to TRUE indicates that if the resulting LCF contains **destinationInfo**, **destExtraCallInfo** and/or **remoteExtension** fields, the endpoint can copy this information to the **destinationAddress**, **destExtraCallInfo** and **remoteExtensionAddress** fields of the SETUP message respectively. If the GK would replace addressing information from the LRQ and **canMapAlias** is FALSE, then the gatekeeper should reject the LRQ.

gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous LRJ message. Used as a backup if the original gatekeeper did not respond or rejected the request.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.13.2 LocationConfirm (LCF)

The LCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the LRQ.

callSignalAddress – The transport address to which to send Q.931 call signalling; uses the reliable well known or dynamic port, but may be an endpoint or gatekeeper address depending on the call model in use.

rasAddress – Registration, admissions, and status address for the located endpoint.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

destinationInfo – Sequence of alias addresses for the destination, such as E.164 addresses or H323_IDs.

destExtraCallInfo – Contains external addresses for multiple calls.

destinationType – This specifies the type of the destination endpoint.

remoteExtensionAddress – Contains the alias address of a called endpoint in cases where this information is needed to traverse multiple Gateways.

alternateEndpoints – A sequence of prioritized endpoint alternatives for callSignalAddress, rasAddress, or destinationInfo.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.13.3 LocationReject (LRJ)

The LRJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the LRQ.

rejectReason – Reason the location request was denied.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.14 Disengage messages

7.14.1 DisengageRequest (DRQ)

If sent from an endpoint to a gatekeeper, the DRQ informs the gatekeeper that an endpoint is being dropped. If sent from a gatekeeper to an endpoint, the DRQ forces a call to be dropped; such a request shall not be refused. The DRQ is not sent between endpoints directly.

Note that DRQ is not the same as **ReleaseComplete** since its purpose is to inform the gatekeeper of the termination of a call; the gatekeeper may not receive the release complete if it is not terminating the call signalling channel.

The DRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

endpointIdentifier – This is an endpoint identifier that was assigned to the terminal by RCF.

conference ID – ID of the call that is to have the bandwidth released.

callReferenceValue – The CRV from Q.931 for this call; only local validity. This is used by a gatekeeper to associate the message with a particular call.

disengageReason – The reason the change was requested by the gatekeeper or the terminal.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous DRJ message. Used as a back-up if the original gatekeeper did not respond or rejected the request.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

answeredCall – Set to TRUE to indicate that this party was the original destination (this party answered the call).

7.14.2 DisengageConfirm (DCF)

The DCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the DRQ.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.14.3 DisengageReject (DRJ)

DRJ is sent by the gatekeeper if the endpoint is unregistered.

The DRJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the DRQ.

rejectReason – The reason the request was rejected.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.15 Status Request messages

The IRQ is sent from a gatekeeper to a terminal requesting status information in the form of an IRR. The IRR may be also be sent by the terminal at an interval specified in the ACF message without the receipt of an IRQ from the gatekeeper. This message should not be confused with the Q.931 STATUS message.

When an unsolicited IRR is sent by an endpoint to a Gatekeeper of version 2 or higher, it may indicate in the **needResponse** field that it wishes the Gatekeeper to acknowledge receipt of the IRR. In this case it fills in the **requestSeqNum** field with a number other than 1. The gatekeeper returns either an IACK (positive acknowledgement) or an INAK (negative acknowledgement) message, and must return the same number in the **requestSeqNum** field.

7.15.1 InfoRequest (IRQ)

The IRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

callReferenceValue – CRV of the call that the query is about. If zero, this message is interpreted as a request for an IRR for each call the terminal is active on. If the terminal is not active on any calls, an IRR shall be sent in response to a CallReferenceValue of 0 with all appropriate fields provided.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

replyAddress – A transport address to send IRR to, perhaps not that of the gatekeeper.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

uuiesRequested – The gatekeeper may request the endpoint to notify the gatekeeper of H.225.0 call signalling messages that the endpoint sends or receives if the endpoint indicated this capability in the ARQ by setting **willSupplyUUIEs** to TRUE. **uuiesRequested** indicates the set of H.225.0 call signalling messages of which the endpoint shall notify the gatekeeper.

7.15.2 InfoRequestResponse (IRR)

The IRR message includes the following:

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

requestSeqNum – This shall contain the sequence number from the IRR or one (1) for an unsolicited report to a version 1 gatekeeper. It contains a monotonically increasing number (to be returned by the Gatekeeper in its response) if **needResponse** is TRUE.

endpointType – Provides information about the endpoint.

endpointIdentifier – Value assigned by the gatekeeper in the RCF.

rasAddress – Address for registration, admissions, etc.

callSignalAddress – Address of H.225.0 call signalling.

endpointAlias – Alias(s) for endpoint.

perCallInfo – Information about a particular call:

> **nonStandardData** – Carries information not defined in this Recommendation (for example, proprietary data).
>
> **callReferenceValue** – Q.931 CRV of that call that the response is about.

conferenceID – Unique conference identifier.

originator – If TRUE the endpoint being queried was the call originator, if FALSE the endpoint was the call destination.

audio – Information about the audio channel(s).

video – Information about the video channel(s).

data – Information about the data channel(s).

h245 – The transport address of the H.245 control channel.

callSignaling – The transport address of the H.225.0 call signalling channel.

callType – Provides information on call topology.

bandwidth – Current usage in increments of 100 bit/s; includes only audio and video excluding headers and overhead.

callModel – Indicates the endpoint's idea of which call model is in use.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

substituteConfIDs – A listing of all ConferenceIDs received in H.245 SubstituteCID messages pertaining to the original RAS **perCallInfo conferenceID**.

pdu:

h323pdu – A copy of an H.225.0 and Q.931 PDU as requested by the gatekeeper in **uuiesRequested** in either ACF or IRQ.

sent – Set to TRUE to indicate the endpoint sent the **h323pdu**; set to FALSE to indicate the endpoint received the **h323pdu**.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

needResponse – If this is set to TRUE and the gatekeeper indicated in either RCF or ACF that it will respond to unsolicited IRRs (by setting **willRespondToIRR** to TRUE), then the Gatekeeper shall reply with an IACK or INAK. If the gatekeeper had not indicated in either RCF or ACF that it will respond to unsolicited IRRs (by setting **willRespondToIRR** to FALSE), then the gatekeeper may ignore the **needResponse** BOOLEAN.

7.15.3 InfoRequestAck (IACK)

The IACK message includes the following:

requestSeqNum – This field shall contain the requestSeqNum that was in the IRR.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.15.4 InfoRequestNak (INAK)

The INAK message includes the following:

requestSeqNum – This field shall contain the requestSeqNum that was in the IRR.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

nakReason – Reason the IRR was negatively acknowledged.

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.16 Non-Standard message

The **NonStandardMessage** structure is as follows:

requestSeqNum – This is a monotonically increasing number unique to the sender.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.17 Message Not Understood

This message is sent whenever an H.323 endpoint receives a RAS message it does not understand.

RequestSeqNum – Shall be the **requestSeqNum** of the unknown message, if it can be decoded, and zero otherwise.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.18 Gateway Resource Availability messages

The Resource Availability Indication (RAI) is a notification from a gateway to a gatekeeper of its current call capacity for each H-series protocol and data rate for that protocol. The gatekeeper responds with a Resource Availability Confirmation (RAC) upon receiving a RAI to acknowledge its reception.

7.18.1 ResourcesAvailableIndicate (RAI)

The RAI message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any response associated with this specific message.

protocolIdentifier – Identifies the vintage of the sending endpoint.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

endpointIdentifier – A gatekeeper assigned endpoint identity string.

protocols – Indicates the current data rates for each protocol which can be supported given the current state of the device.

almostOutOfResources – When set to TRUE, the device is nearing or at capacity. Any action based on this field is at the manufacturer's discretion. If the device is not near or at capacity this field should be set to FALSE.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.18.2 ResourcesAvailableConfirm (RAC)

The RAC message includes the following:

requestSeqNum – This shall be the same value that was passed in the RAI.

protocolIdentifier – Identifies the vintage of the accepting gatekeeper.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

7.19 RAS timers and Request in Progress (RIP)

Table 19 shows recommended default timeout values for the response to RAS messages and subsequent retry counts if a response is not received. (These values are subject to change with further implementation experience and input.)

Table 19/H.225.0 – Recommended default timeout values

RAS message	Timeout value (sec)	Retry count
GRQ	5	2
RRQ	3	2
URQ	3	1
ARQ	3	2
BRQ	3	2
IRQ	3	1
IRR (Note)	5	2
DRQ	3	2
LRQ	5	2
NOTE – In cases where the gatekeeper is expected to reply to an unsolicited IRR with IACK or INAK, the timeout may occur if no reply to the IRR is received.		

If an entity receives a request from a version 2 (or later) entity to which a response cannot be generated within a typical retry timeout period, it can send a RIP message specifying the period (in the **delay** field) after which a response should have been generated. As soon as a response is available, the responding entity should send the response and not wait for the RIP delay to expire. If a requesting entity has not received a response by the time the RIP delay expires, it shall resend the request. The responding entity can then either send a duplicate response or another RIP message.

Figure 2 gives an example message exchange which demonstrates a number of aspects of the retry strategy.

Vendors should be aware that any retries will have an impact on the call setup time, which should be minimized. Therefore short retry times are desirable. So that remote entities can anticipate typical retry times for the purpose of deciding when to send a RIP message, entities should avoid retry periods less than 100 ms. Exponential backoff and adapting to measured round trip times is encouraged. Entities can use the measured round trip time of the RRQ/RCF registration process to modify an initially conservative estimate (of a few seconds) for this purpose. Entities may also use the registration process to exchange version numbers to ensure that the RIP based retry mechanism is not used when version 1 entities are involved in the signalling.

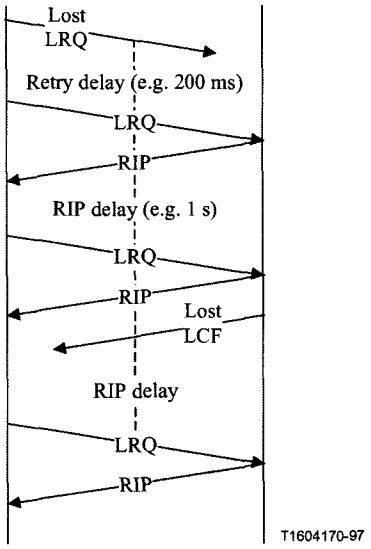


Figure 2/H.225.0 – Example Use of RIP message

The RIP message includes the following:

requestSeqNum – This is the requestSeqNum of the request which is currently being processed.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

delay – Specifies the amount of time in milliseconds that an endpoint shall wait before attempting a retry. The responding endpoint may respond before this period expires.

8 Mechanisms for maintaining QOS

8.1 General approach and assumptions

Transport QOS (Quality of Service) on a packet-based network includes such characteristics as:

- bit error rate;
- packet loss rate;
- delay.

Any transport QOS related signalling (e.g. a reservation request to a router) is done by the terminal as soon as possible, or by the gatekeeper on its behalf. The terminal may wish to make any reservations since the gatekeeper may not be logically near the terminal, or be able to make QOS related requests on behalf of the terminal. The means by which either the terminal or the gatekeeper make QOS or bandwidth reservations are beyond the scope of this Recommendation.

The Sender and Receiver Reports of RTCP shall be the means by which QOS will be assessed.

There are two types of congestion related delay that might be measured:

- Short-term increases in delay that will result in a perceptible but not annoying slowing of the frame rate.
- A general rise in delay due to packet-based network congestion over time such that a feedback based mechanism is useful.

Essentially, short-term bursts are approached by error concealment, and a longer term congestion is approached by reducing the multimedia load. The assumption is made that all packet-based network multimedia terminals are H.323 terminals, and all will attempt to reduce packet-based network usage as congestion rises rather than "steal" bandwidth from each other.

Bit errors on a packet-based network generally are either corrected at a lower layer, or result in packet loss, so they are not considered further in this clause.

Packet loss requires the receiver to be able to compensate for lost packets in a fashion that conceals errors to the maximum possible extent. For data and control, retransmission at the transport layer is used. For audio and video, retransmission is for further study.

A given level of transport QOS results in a level of user-perceived audio/video QOS that is a function in part of the effectiveness of the methods used to overcome transport QOS problems.

8.2 Use of RTCP in measuring QOS

8.2.1 Sender reports

The sender report serves three main purposes:

1) Allow synchronization of multiple RTP streams, such as audio and video.
2) Allow the receiver to know the expected data rate and packet rate.
3) Allow the receiver to measure the distance in time to the sender.

Of these three purposes, 1) is the most relevant to this Recommendation. Manufacturers may make use of the sender reports in other ways at their discretion.

The relevant field for stream synchronization is the RTP timestamp and the NTP timestamp in the sender report of RTCP. The NTP timestamp (if available) gives "wall clock" time and corresponds to the RTP timestamp which has the same units and random offset as the RTP capture timestamp in the media packets.

8.2.2 Receiver Reports

Four parts of the Receiver Reports are used in this Recommendation to measure QOS:

1) Fraction Lost.

2) The cumulative packets lost.

3) The extended highest sequence number received.

4) Interarrival jitter.

Items 2) and 3) are used to compute the number of packets lost since the previous receiver report. This can be taken as a long term measure of packet-based network congestion. See A.6.3.4 for a sample computation. If this loss rate exceeds a value set by the manufacturer, the H.225.0 terminal should reduce the media rates on the packet-based network side according to the procedures in 8.4 below. If item 1) exceeds a value set by the manufacturer, it may also be desirable to take corrective action.

If the interval between receiver reports exceeds a value set by the manufacturer, H.323 terminals should use item 1) as an indicator of serious congestion requiring media rate reduction on the packet-based network side.

Item 4) should be used as an indication of impending congestion. If interarrival jitter increases for three consecutive receiver reports, the H.323 sending terminal should take corrective action.

8.3 Audio/Video jitter procedures

Recommendation H.245 provides commands and procedures for round-trip indications using **RoundTripDelayRequest** and **RoundTripDelayResponse**. On a multipoint call the MC responds to a request from the endpoint. RTCP contains a method of calculating round-trip delays based on the Sender Report and the Receiver Report messages. Note that the quantity being measured in each case is not the same, so there is no conflict in using both methods to measure jitter.

See 6.2.5/H.323 for a discussion of how H.245 level signalling can be used to optionally reduce jitter related delays.

8.4 Audio/Video Skew Procedures

See 6.2.6/H.323 for a discussion of how H.245 level signalling is used to limit the skew between different logical channels.

8.5 Procedures for maintaining QOS

A number of methods exist for the H.323 gateway/terminal to respond to an increase in packet loss or interarrival jitter in the far-end receiver. These methods can be grouped into those that are appropriate for a rapid response to a short term problem, such as a lost or delayed packet, and those that are appropriate for a response to a longer term problem such as growing congestion on the packet-based network. Note that these methods do not seek to maintain the current quality of service, but instead to provide for an orderly degradation of service. The following priorities shall be observed such that, if present, media shall be degraded in the following order: Video, Data, Audio, Control.

Short term responses:

- Reducing the frame rate for a short period of time. This may result in the H.323 gateway sending additional H.261 fill frames in the packet-based network→WAN direction to compensate for the packet under flow.

- Reduce packet rate by switching to the optional mode where audio/video are mixed in one packet (for further study).
- Packet rate can also be reduced via the use of MB fragmentation of the video stream.

Longer term responses:

- Reducing media bit rate (e.g. switching from 384 kbit/s to 256 kbit/s). This may involve a simple instruction to the encoder in a terminal, or it may involve the use of a rate reducer function in the H.323 gateway. These changes are signalled via H.245 **FlowControl** commands, or by logical channel signalling as appropriate.
- Turning off media of lesser importance (e.g. turning off video to allow a large amount of T.120 traffic).
- Returning a busy signal (adaptive busy) to the receiver as an indication of packet-based network congestion. This may be combined with turning off a media, or even all media other than the control Transport Port. Adaptive busy is signalled via a Q.931 cause value in **Release Complete**.

It should be noted that responding to interarrival jitter in a multi-router path where a large percentage of packets arrive out of order is difficult. It may be impossible to distinguish this source of jitter from other sources, or to base error recovery strategy on measured jitter. However, packet loss is quantifiable and unambiguous.

8.6 Echo Control

Control of acoustic echo is the responsibility of the H.323 terminal. In general, given the delay involved in video/audio compression, it is assumed that all H.320, H.323 and H.324 terminals have some form of echo control (cancellation or switching).

However, when the H.323 terminal is used to call a GSTN telephone, it is typically that case that the GSTN phone does not support echo control. Thus, the user of the H.323 terminal may hear acoustic echo return from the GSTN side. This acoustic echo return can be minimized by the use of a speakerphone with echo control, or the use of a handset or ear phones. Manufacturers may add loss to the audio path when an H.323 terminal is connected to a GSTN POTS phone.

Control of hybrid (2 to 4 wire) echo is the responsibility of the H.323 gateway.

ANNEX A

RTP/RTCP

The reader should note that all references in this annex are to a bibliography, and are non-normative, with the exception of [7] to ISO/IEC 10646-1, which also appears in the references clause of this Recommendation. In some cases a reference will appear to Appendix I; such references are for informative purposes only. All details required to implement Recommendation H.323 and this Recommendation are contained in this annex and other related annexes and Recommendations or International Standards published by the ITU-T or ISO.

Readers should note that this annex is not the complete and primary specification of RTP/RTCP; please refer to Appendix I for this informative reference. This annex is intended only for usage with H.323 and this Recommendation.

Readers should also note that the terminology used in this annex differs somewhat from that used in Recommendation H.323 and this Recommendation according to Table A.1.

Table A.1/H.225.0 – Terminology correspondence

H.323 and H.225.0 term	Annex A (RTP/RTCP) term
media stream	data
transport address	transport address
packet-based network address	network address
TSAP identifier	port
Annex A	specification or document
shall	must
should	should

It should be further noted that "translators" and "mixers" are not part of the H.323 system. H.323 endpoints such as gateways and MCUs have some of the characteristics of translators and mixers, so this text has been retained as a guide to the implementor. However, support for translators and mixers is not part of H.323, and these subclauses shall be considered informative.

Finally, implementors are reminded to implement RTP only as described in this Recommendation, including Annexes A, B, and C, which contain details and clarifications relevant to H.323/H.225.0. In all cases, the text of this Recommendation shall have precedence over text in Annexes A, B, or C.

A.1 Introduction

This memorandum specifies the Real-Time Transport protocol (RTP), which provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. Those services include payload type identification, sequence numbering, timestamping and delivery monitoring. Applications typically run RTP on top of UDP to make use of its multiplexing and checksum services; both protocols contribute parts of the transport protocol functionality. However, RTP may be used with other suitable underlying network or transport protocols (see A.10, RTP over Network and Transport Protocols) RTP supports data transfer to multiple destinations using multicast distribution if provided by the underlying network.

Note that RTP itself does not provide any mechanism to ensure timely delivery or provide other quality-of-service guarantees, but relies on lower-layer services to do so. It does not guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence. The sequence numbers included in RTP allow the receiver to reconstruct the sender's packet sequence, but sequence numbers might also be used to determine the proper location of a packet, for example in video decoding, without necessarily decoding packets in sequence.

While RTP is primarily designed to satisfy the needs of multi-participant multimedia conferences, it is not limited to that particular application. Storage of continuous data, interactive distributed simulation, active badge, and control and measurement applications may also find RTP applicable.

This Recommendation defines RTP, consisting of two closely-linked parts:

- The Real-Time Transport (RTP) protocol, to carry data that has real-time properties.
- The RTP Control Protocol (RTCP), to monitor the quality of service and to convey information about the participants in an on-going session. The latter aspect of RTCP may be sufficient for "loosely controlled" sessions, i.e. where there is no explicit membership control and setup, but it is not necessarily intended to support all of an application's control communication requirements. This functionality may be fully or partially subsumed by a separate session control protocol, which is beyond the scope of this Recommendation.

RTP represents a new style of protocol following the principles of application level framing and integrated layer processing proposed by Clark and Tennenhouse [A-1]. That is, RTP is intended to be malleable to provide the information required by a particular application and will often be integrated into the application processing rather than being implemented as a separate layer. RTP is a protocol framework that is deliberately not complete. This Recommendation specifies those functions expected to be common across all the applications for which RTP would be appropriate. Unlike conventional protocols in which additional functions might be accommodated by making the protocol more general or by adding an option mechanism that would require parsing, RTP is intended to be tailored through modifications and/or additions to the headers as needed. Examples are given in A.5.3, Profile-Specific Modifications to the RTP Header.

Therefore, in addition to this Recommendation, a complete specification of RTP for a particular application will require one or more companion documents (see Annexes B and C):

- A profile specification document, which defines a set of payload type codes and their mapping to payload formats (e.g. media encodings). A profile may also define extensions or modifications to RTP that are specific to a particular class of applications. Typically an application will operate under only one profile. A profile for audio and video data may be found in Annex B.
- Payload format specification documents, which define how a particular payload, such as an audio or video encoding, is to be carried in RTP. See Annex C.

Several RTP applications, both experimental and commercial, have already been implemented from draft specifications. These applications include audio and video tools along with diagnostic tools such as traffic monitors. Users of these tools number in the thousands. However, the current Internet cannot yet support the full potential demand for real-time services. High-bandwidth services using RTP, such as video, can potentially seriously degrade the quality of service of other network services. Thus, implementors should take appropriate precautions to limit accidental bandwidth usage. Application documentation should clearly outline the limitations and possible operational impact of high-bandwidth real-time services on the Internet and other network services.

A.2 RTP use scenarios

The following subclauses describe some aspects of the use of RTP. The examples were chosen to illustrate the basic operation of applications using RTP, not to limit what RTP may be used for. In these examples, RTP is carried on top of IP and UDP, and follows the conventions established by the profile for audio and video specified in Annex B.

A.2.1 Simple multicast audio conference

A working group of the IETF meets to discuss the latest protocol draft, using the IP multicast services of the Internet for voice communications. Through some allocation mechanism the working group chair obtains a multicast group address and pair of ports. One port is used for audio data, and the other is used for control (RTCP) packets. This address and port information is distributed to the intended participants. If privacy is desired, the data and control packets may be encrypted as specified in Recommendation H.323. The audio conferencing application used by each conference participant sends audio data in small chunks of, say, 20 ms duration. Each chunk of audio data is preceded by an RTP header; RTP header and data are in turn contained in a UDP packet. The RTP header indicates what type of audio encoding (such as PCM, ADPCM or LPC) is contained in each packet so that senders can change the encoding during a conference, for example, to accommodate a new participant that is connected through a low-bandwidth link or react to indications of network congestion.

The Internet, like other packet networks, occasionally loses and reorders packets and delays them by variable amounts of time. To cope with these impairments, the RTP header contains timing information and a sequence number that allow the receivers to reconstruct the timing produced by the source, so that in this example, chunks of audio are contiguously played out the speaker every 20 ms. This timing reconstruction is performed separately for each source of RTP packets in the conference. The sequence number can also be used by the receiver to estimate how many packets are being lost.

Since members of the working group join and leave during the conference, it is useful to know who is participating at any moment and how well they are receiving the audio data. For that purpose, each instance of the audio application in the conference periodically multicasts a reception report plus the name of its user on the RTCP (control) port. The reception report indicates how well the current speaker is being received and may be used to control adaptive encodings. In addition to the user name, other identifying information may also be included subject to control bandwidth limits. A site sends the RTCP BYE packet (see A.6.5, BYE: Goodbye RTCP packet) when it leaves the conference.

A.2.2 Audio and video conference

If both audio and video media are used in a conference, they are transmitted as separate RTP sessions, RTCP packets are transmitted for each medium using two different UDP port pairs and/or multicast addresses. There is no direct coupling at the RTP level between the audio and video sessions, except that a user participating in both sessions should use the same distinguished (canonical) name in the RTCP packets for both so that the sessions can be associated.

One motivation for this separation is to allow some participants in the conference to receive only one medium if they choose. Further explanation is given in A.5.2, Multiplexing RTP Sessions. Despite the separation, synchronized playback of a source's audio and video can be achieved using timing information carried in the RTCP packets for both sessions.

A.2.3 Mixers and translators

So far, we have assumed that all sites want to receive media data in the same format. However, this may not always be appropriate. Consider the case where participants in one area are connected through a low-speed link to the majority of the conference participants who enjoy high-speed network access. Instead of forcing everyone to use a lower-bandwidth, reduced-quality audio encoding, an RTP-level relay, called a mixer, may be placed near the low-bandwidth area. This mixer resynchronizes incoming audio packets to reconstruct the constant 20 ms spacing generated by the sender, mixes these reconstructed audio streams into a single stream, translates the audio encoding to a lower bandwidth one and forwards the lower bandwidth packet stream across the low-speed link. These packets might be unicast to a single recipient or multicast on a different address to multiple recipients. The RTP header includes a means for mixers to identify the sources that contributed to a mixed packet so that correct talker indication can be provided at the receivers.

Some of the intended participants in the audio conference may be connected with high bandwidth links but might not be directly reachable via IP multicast. For example, they might be behind an application-level firewall that will not let any IP packets pass. For these sites, mixing may not be necessary, in which case another type of RTP-level relay called a translator may be used. Two translators are installed, one on either side of the firewall, with the outside one funnelling all multicast packets received through a secure connection to the translator inside the firewall. The translator inside the firewall sends them again as multicast packets to a multicast group restricted to the site's internal network.

Mixers and translators may be designed for a variety of purposes. An example is a video mixer that scales the images of individual people in separate video streams and composites them into one video

stream to simulate a group scene. Other examples of translation include the connection of a group of hosts speaking only IP/UDP to a group of hosts that understand only ST-II, or the packet-by-packet encoding translation of video streams from individual sources without resynchronization or mixing. Details of the operation of mixers and translators are given in A.7, RTP translators and mixers.

A.3 Definitions

A.3.1 RTP payload: The data transported by RTP in a packet, for example audio samples or compressed video data. The payload format and interpretation are beyond the scope of this Recommendation.

A.3.2 RTP packet: A data packet consisting of the fixed RTP header, a possibly empty list of contributing sources (see below), and the payload data. Some underlying protocols may require an encapsulation of the RTP packet to be defined. Typically one packet of the underlying protocol contains a single RTP packet, but several RTP packets may be contained if permitted by the encapsulation method (see A.10, RTP over network and transport protocols).

A.3.3 RTCP packet: A control packet consisting of a fixed header part similar to that of RTP data packets, followed by structured elements that vary depending upon the RTCP packet type. The formats are defined in A.6, RTP Control Protocol – RTCP. Typically, multiple RTCP packets are sent together as a compound RTCP packet in a single packet of the underlying protocol; this is enabled by the length field in the fixed header of each RTCP packet.

A.3.4 port: The "abstraction that transport protocols use to distinguish among multiple destinations within a given host computer. TCP/IP protocols identify ports using small positive integers" [A-2]. The Transport Selectors (TSEL) used by the OSI transport layer are equivalent to ports. RTP depends upon the lower-layer protocol to provide some mechanism such as ports to multiplex the RTP and RTCP packets of a session.

A.3.5 transport address: The combination of a network address and port that identifies a transport-level endpoint, for example an IP address and a UDP port. Packets are transmitted from a source transport address to a destination transport address.

A.3.6 RTP session: The association among a set of participants communicating with RTP. For each participant, the session is defined by a particular pair of destination transport addresses (one network address plus a port pair for RTP and RTCP). The destination transport address pair may be common for all participants, as in the case of IP multicast, or may be different for each, as in the case of individual unicast network addresses and ports. In a multimedia session, each medium is carried in a separate RTP session with its own RTCP packets. The multiple RTP sessions are distinguished by different port number pairs and/or different multicast addresses.

A.3.7 Synchronization source (SSRC): The source of a stream of RTP packets, identified by a 32-bit numeric SSRC identifier carried in the RTP header so as not to be dependent upon the network address. All packets from a synchronization source form part of the same timing and sequence number space, so a receiver groups packets by synchronization source for playback. Examples of synchronization sources include the sender of a stream of packets derived from a signal source such as a microphone or a camera, or an RTP mixer (see below). A synchronization source may change its data format, e.g. audio encoding, over time. The SSRC identifier is a randomly chosen value meant to be globally unique within a particular RTP session (see A.8, SSRC identifier allocation and use). A participant need not use the same SSRC identifier for all the RTP sessions in a multimedia session; the binding of the SSRC identifiers is provided through RTCP (see A.6.4.1, CNAME: Canonical end-point identifier SDES item). If a participant generates multiple streams in one RTP session, for example from separate video cameras, each must be identified as a different SSRC.

A.3.8 Contributing source (CSRC): A source of a stream of RTP packets that has contributed to the combined stream produced by an RTP mixer (see below). The mixer inserts a list of the SSRC identifiers of the sources that contributed to the generation of a particular packet into the RTP header of that packet. This list is called the CSRC list. An example application is audio conferencing where a mixer indicates all the talkers whose speech was combined to produce the outgoing packet, allowing the receiver to indicate the current talker, even though all the audio packets contain the same SSRC identifier (that of the mixer).

A.3.9 end system: An application that generates the content to be sent in RTP packets and/or consumes the content of received RTP packets. An end system can act as one or more synchronization sources in a particular RTP session, but typically only one.

A.3.10 mixer: An intermediate system that receives RTP packets from one or more sources, possibly changes the data format, combines the packets in some manner and then forwards a new RTP packet. Since the timing among multiple input sources will not generally be synchronized, the mixer will make timing adjustments among the streams and generate its own timing for the combined stream. Thus, all data packets originating from a mixer will be identified as having the mixer as their synchronization source.

A.3.11 translator: An intermediate system that forwards RTP packets with their synchronization source identifier intact. Examples of translators include devices that convert encodings without mixing, replicators from multicast to unicast, and application- level filters in firewalls.

A.3.12 monitor: An application that receives RTCP packets sent by participants in an RTP session, in particular the reception reports, and estimates the current quality of service for distribution monitoring, fault diagnosis and long-term statistics. The monitor function is likely to be built into the application(s) participating in the session, but may also be a separate application that does not otherwise participate and does not send or receive the RTP data packets. These are called third party monitors.

A.3.13 non-RTP means: Protocols and mechanisms that may be needed in addition to RTP to provide a usable service. In particular, for multimedia conferences, a conference control application may distribute multicast addresses and keys for encryption, negotiate the encryption algorithm to be used, and define dynamic mappings between RTP payload type values and the payload formats they represent for formats that do not have a predefined payload type value. For simple applications, electronic mail or a conference database may also be used. The specification of such protocols and mechanisms is outside the scope of this Recommendation.

A.4 Byte order, alignment and time format

All integer fields are carried in network byte order, that is, most significant byte (octet) first. This byte order is commonly known as big-endian. The transmission order is described in detail in [A-3]. Unless otherwise noted, numeric constants are in decimal (base 10).

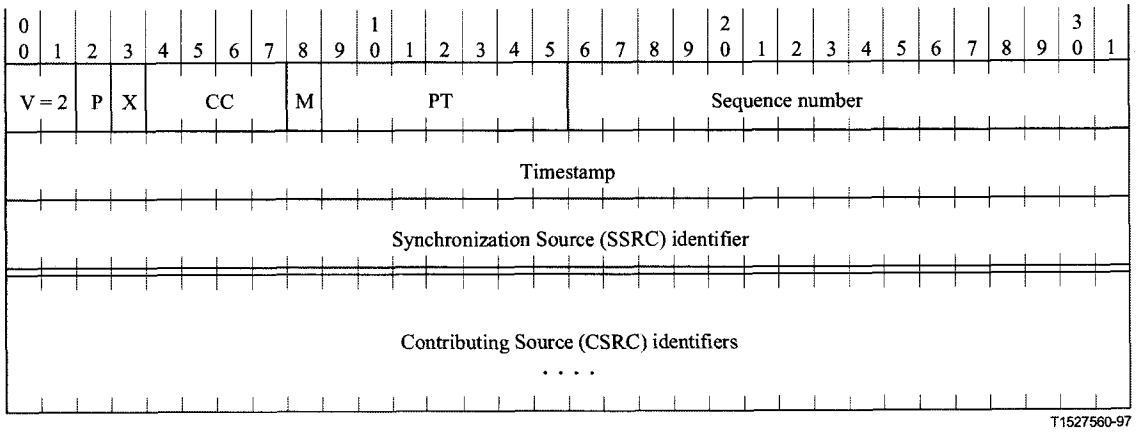
All header data is aligned to its natural length, i.e. 16-bit fields are aligned on even offsets, 32-bit fields are aligned at offsets divisible by four, etc. Octets designated as padding have the value zero.

Wallclock time (absolute time) is represented using the timestamp format of the Network Time Protocol (NTP), which is in seconds relative to 0h UTC on 1 January 1900 [A-4]. The full resolution NTP timestamp is a 64-bit unsigned fixed-point number with the integer part in the first 32 bits and the fractional part in the last 32 bits. In some fields where a more compact representation is appropriate, only the middle 32 bits are used; that is, the low 16 bits of the integer part and the high 16 bits of the fractional part. The high 16 bits of the integer part must be determined independently.

A.5 RTP data transfer protocol

A.5.1 RTP fixed header fields

The RTP header has the following format:



The first twelve octets are present in every RTP packet, while the list of CSRC identifiers is present only when inserted by a mixer. The fields have the following meaning:

version (V): 2 bits. This field identifies the version of RTP. The version defined by this specification is two (2). (The value 1 is used by the first draft version of RTP and the value 0 is used by the protocol initially implemented in the "vat" audio tool.)

padding (P): 1 bit. If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload. The last octet of the padding contains a count of how many padding octets should be ignored. Padding may be needed by some encryption algorithms with fixed block sizes or for carrying several RTP packets in a lower-layer protocol data unit.

extension (X): 1 bit. If the extension bit is set, the fixed header is followed by exactly one header extension, with a format defined in A.5.3, Profile-specific modifications to the RTP header.

CSRC count (CC): 4 bits. The CSRC count contains the number of CSRC identifiers that follow the fixed header.

marker (M): 1 bit. The interpretation of the marker is defined by a profile. It is intended to allow significant events such as frame boundaries to be marked in the packet stream. A profile may define additional marker bits or specify that there is no marker bit by changing the number of bits in the payload type field (see A.5.3, Profile-specific modifications to the RTP header).

payload type (PT): 7 bits. This field identifies the format of the RTP payload and determines its interpretation by the application. A profile specifies a default static mapping of payload type codes to payload formats. Additional payload type codes may be defined dynamically through non-RTP means (see A.3, Definitions). An initial set of default mappings for audio and video is specified in Annex B. An RTP sender emits a single RTP payload type at any given time; this field is not intended for multiplexing separate media streams (see A.5.2, Multiplexing RTP sessions).

sequence number: 16 bits. The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence. The initial value of the sequence number is random (unpredictable) to make known-plaintext attacks on encryption more difficult, even if the source itself does not encrypt, because the packets may flow

through a translator that does. Techniques for choosing unpredictable numbers are discussed in [A-5].

timestamp: 32 bits. The timestamp reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and jitter calculations (see A.6.3.1, SR: Sender Report RTCP packet). The resolution of the clock must be sufficient for the desired synchronization accuracy and for measuring packet arrival jitter (one tick per video frame is typically not sufficient). The clock frequency is dependent on the format of data carried as payload and is specified statically in the profile or payload format specification that defines the format, or may be specified dynamically for payload formats defined through non-RTP means. If RTP packets are generated periodically, the nominal sampling instant as determined from the sampling clock is to be used, not a reading of the system clock. As an example, for fixed-rate audio the timestamp clock would likely increment by one for each sampling period. If an audio application reads blocks covering 160 sampling periods from the input device, the timestamp would be increased by 160 for each such block, regardless of whether the block is transmitted in a packet or dropped as silent.

The initial value of the timestamp is random, as for the sequence number. Several consecutive RTP packets may have equal timestamps if they are (logically) generated at once, e.g. belong to the same video frame. Consecutive RTP packets may contain timestamps that are not monotonic if the data is not transmitted in the order it was sampled, as in the case of MPEG interpolated video frames. (The sequence numbers of the packets as transmitted will still be monotonic.)

SSRC: 32 bits. The SSRC field identifies the synchronization source. This identifier is chosen randomly, with the intent that no two synchronization sources within the same RTP session will have the same SSRC identifier. An example algorithm for generating a random identifier is presented in I.6. Although the probability of multiple sources choosing the same identifier is low, all RTP implementations must be prepared to detect and resolve collisions. Subclause A.8, SSRC identifier allocation and use describes the probability of collision along with a mechanism for resolving collisions and detecting RTP-level forwarding loops based on the uniqueness of the SSRC identifier. If a source changes its source transport address, it must also choose a new SSRC identifier to avoid being interpreted as a looped source.

CSRC list: 0 to 15 items, 32 bits each. The CSRC list identifies the contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field. If there are more than 15 contributing sources, only 15 may be identified. CSRC identifiers are inserted by mixers, using the SSRC identifiers of contributing sources. For example, for audio packets the SSRC identifiers of all sources that were mixed together to create a packet are listed, allowing correct talker indication at the receiver.

A.5.2 Multiplexing RTP sessions

For efficient protocol processing, the number of multiplexing points should be minimized, as described in the integrated layer processing design principle [A-1]. In RTP, multiplexing is provided by the destination transport address (network address and port number) which define an RTP session. For example, in a teleconference composed of audio and video media encoded separately, each medium should be carried in a separate RTP session with its own destination transport address. It is not intended that the audio and video be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields. Interleaving packets with different payload types but using the same SSRC would introduce several problems:

1) If one payload type were switched during a session, there would be no general means to identify which of the old values the new one replaced.

2) An SSRC is defined to identify a single timing and sequence number space. Interleaving multiple payload types would require different timing spaces if the media clock rates differ and would require different sequence number spaces to tell which payload type suffered packet loss.

3) The RTCP sender and receiver reports (see A.6.3, sender and receiver reports) can only describe one timing and sequence number space per SSRC and do not carry a payload type field.

4) An RTP mixer would not be able to combine interleaved streams of incompatible media into one stream.

5) Carrying multiple media in one RTP session precludes: the use of different network paths or network resource allocations if appropriate; reception of a subset of the media if desired, for example just audio if video would exceed the available bandwidth; and receiver implementations that use separate processes for the different media, whereas using separate RTP sessions permits either single- or multiple-process implementations.

Using a different SSRC for each medium but sending them in the same RTP session would avoid the first three problems but not the last two.

A.5.3 Profile-specific modifications to the RTP header

The existing RTP data packet header is believed to be complete for the set of functions required in common across all the application classes that RTP might support. However, in keeping with the ALF design principle, the header may be tailored through modifications or additions defined in a profile specification while still allowing profile-independent monitoring and recording tools to function:

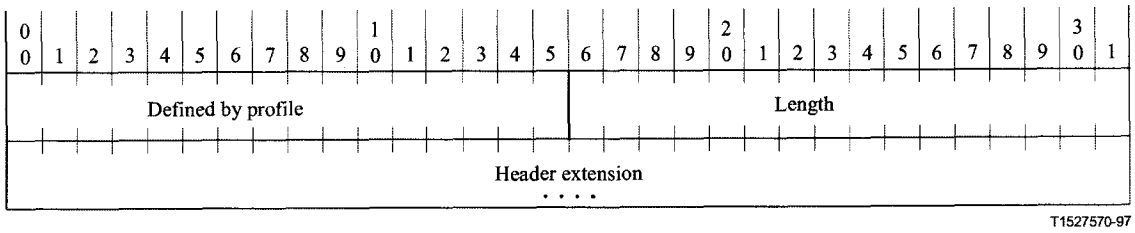
- The marker bit and payload type field carry profile-specific information, but they are allocated in the fixed header since many applications are expected to need them and might otherwise have to add another 32-bit word just to hold them. The octet containing these fields may be redefined by a profile to suit different requirements, for example with a more or fewer marker bits. If there are any marker bits, one should be located in the most significant bit of the octet since profile-independent monitors may be able to observe a correlation between packet loss patterns and the marker bit.
- Additional information that is required for a particular payload format, such as a video encoding, should be carried in the payload section of the packet. This might be in a header that is always present at the start of the payload section, or might be indicated by a reserved value in the data pattern.
- If a particular class of applications needs additional functionality independent of payload format, the profile, under which those applications operate, should define additional fixed fields to follow immediately after the SSRC field of the existing fixed header. Those applications will be able to quickly and directly access the additional fields while profile-independent monitors or recorders can still process the RTP packets by interpreting only the first twelve octets.

If it turns out that additional functionality is needed in common across all profiles, then a new version of RTP should be defined to make a permanent change to the fixed header.

A.5.3.1 RTP header extension

An extension mechanism is provided to allow individual implementations to experiment with new payload-format-independent functions that require additional information to be carried in the RTP data packet header. This mechanism is designed so that the header extension may be ignored by other interoperating implementations that have not been extended.

Note that this header extension is intended only for limited use. Most potential uses of this mechanism would be better done another way, using the methods described in the previous subclause. For example, a profile-specific extension to the fixed header is less expensive to process because it is not conditional nor in a variable location. Additional information required for a particular payload format should not use this header extension, but should be carried in the payload section of the packet:



If the X bit in the RTP header is one, a variable-length header extension is appended to the RTP header, following the CSRC list if present. The header extension contains a 16-bit length field that counts the number of 32-bit words in the extension, excluding the four-octet extension header (therefore zero is a valid length). Only a single extension may be appended to the RTP data header. To allow multiple interoperating implementations to each experiment independently with different header extensions, or to allow a particular implementation to experiment with more than one type of header extension, the first 16 bits of the header extension are left open for distinguishing identifiers or parameters. The format of these 16 bits is to be defined by the profile specification under which the implementations are operating. This RTP specification does not define any header extensions itself.

A.6 RTP Control Protocol (RTCP)

The RTP Control Protocol (RTCP) is based on the periodic transmission of control packets to all participants in the session, using the same distribution mechanism as the data packets. The underlying protocol must provide multiplexing of the data and control packets, for example using separate port numbers with UDP. RTCP performs four functions:

1) The primary function is to provide feedback on the quality of the data distribution. This is an integral part of the RTP's role as a transport protocol and is related to the flow and congestion control functions of other transport protocols. The feedback may be directly useful for control of adaptive encodings [A-6, A-7], but experiments with IP multicasting have shown that it is also critical to get feedback from the receivers to diagnose faults in the distribution. Sending reception feedback reports to all participants allows one who is observing problems to evaluate whether those problems are local or global. With a distribution mechanism like IP multicast, it is also possible for an entity such as a network service provider who is not otherwise involved in the session to receive the feedback information and act as a third-party monitor to diagnose network problems. This feedback function is performed by the RTCP sender and receiver reports, described below in A.6.3, Sender and receiver reports.

2) RTCP carries a persistent transport-level identifier for an RTP source called the canonical name or CNAME (A.6.4.1, CNAME: Canonical end-point identifer SDES item). Since the SSRC identifier may change if a conflict is discovered or a program is restarted, receivers require the CNAME to keep track of each participant. Receivers also require the CNAME to associate multiple data streams from a given participant in a set of related RTP sessions, for example to synchronize audio and video.

3) The first two functions require that all participants send RTCP packets, therefore the rate must be controlled in order for RTP to scale up to a large number of participants. By having each participant send its control packets to all the others, each can independently observe the number of participants. This number is used to calculate the rate at which the packets are sent, as explained in A.6.2, RTCP transmission interval.

4) A fourth, optional function is to convey minimal session control information, for example participant identification to be displayed in the user interface. This is most likely to be useful in "loosely controlled" sessions where participants enter and leave without membership control or parameter negotiation. RTCP serves as a convenient channel to reach all the participants, but it is not necessarily expected to support all the control communication requirements of an application. A higher-level session control protocol, which is beyond the scope of this Recommendation, may be needed.

Functions 1-3 are mandatory when RTP is used in the IP multicast environment, and are recommended for all environments. RTP application designers are advised to avoid mechanisms that can only work in unicast mode and will not scale to larger numbers.

A.6.1 RTCP packet format

This specification defines several RTCP packet types to carry a variety of control information:

SR: Sender Report, for transmission and reception statistics from participants that are active senders.

RR: Receiver Report, for reception statistics from participants that are not active senders.

SDES: Source Description items, including CNAME.

BYE: Indicates end of participation.

APP: Application specific functions.

Each RTCP packet begins with a fixed part similar to that of RTP data packets, followed by structured elements that may be of variable length according to the packet type but always end on a 32-bit boundary. The alignment requirement and a length field in the fixed part are included to make RTCP packets "stackable". Multiple RTCP packets may be concatenated without any intervening separators to form a compound RTCP packet that is sent in a single packet of the lower layer protocol, for example UDP. There is no explicit count of individual RTCP packets in the compound packet since the lower layer protocols are expected to provide an overall length to determine the end of the compound packet.

Each individual RTCP packet in the compound packet may be processed independently with no requirements upon the order or combination of packets. However, in order to perform the functions of the protocol, the following constraints are imposed:

- Reception statistics (in SR or RR) should be sent as often as bandwidth constraints will allow to maximize the resolution of the statistics, therefore each periodically transmitted compound RTCP packet should include a report packet.
- New receivers need to receive the CNAME for a source as soon as possible to identify the source and to begin associating media for purposes such as lip-sync, so each compound RTCP packet should also include the SDES CNAME.
- The number of packet types that may appear first in the compound packet should be limited to increase the number of constant bits in the first word and the probability of successfully validating RTCP packets against misaddressed RTP data packets or other unrelated packets.

Thus, all RTCP packets must be sent in a compound packet of at least two individual packets, with the following format recommended:

Encryption prefix: If and only if the compound packet is to be encrypted, it is prefixed by a random 32-bit quantity redrawn for every compound packet transmitted.

SR or RR: The first RTCP packet in the compound packet must always be a report packet to facilitate header validation as described in A.2. This is true even if no data has been sent nor received, in which case an empty RR is sent, and even if the only other RTCP packet in the compound packet is a BYE.

Additional RRs: If the number of sources for which reception statistics are being reported exceeds 31, the number that will fit into one SR or RR packet, then additional RR packets should follow the initial report packet.

SDES: An SDES packet containing a CNAME item must be included in each compound RTCP packet. Other source description items may optionally be included if required by a particular application, subject to bandwidth constraints (see A.6.2.2, Allocation of source description bandwidth).

BYE or APP: Other RTCP packet types, including those yet to be defined, may follow in any order, except that BYE should be the last packet sent with a given SSRC/CSRC. Packet types may appear more than once.

It is advisable for translators and mixers to combine individual RTCP packets from the multiple sources they are forwarding into one compound packet whenever feasible in order to amortize the packet overhead (see A.7, RTP translators and mixers). An example RTCP compound packet as might be produced by a mixer is shown in Figure A.1. If the overall length of a compound packet would exceed the Maximum Transmission Unit (MTU) of the network path, it may be segmented into multiple shorter compound packets to be transmitted in separate packets of the underlying protocol. Note that each of the compound packets must begin with an SR or RR packet.

An implementation may ignore incoming RTCP packets with types unknown to it. Additional RTCP packet types may be registered with the Internet Assigned Numbers Authority (IANA).

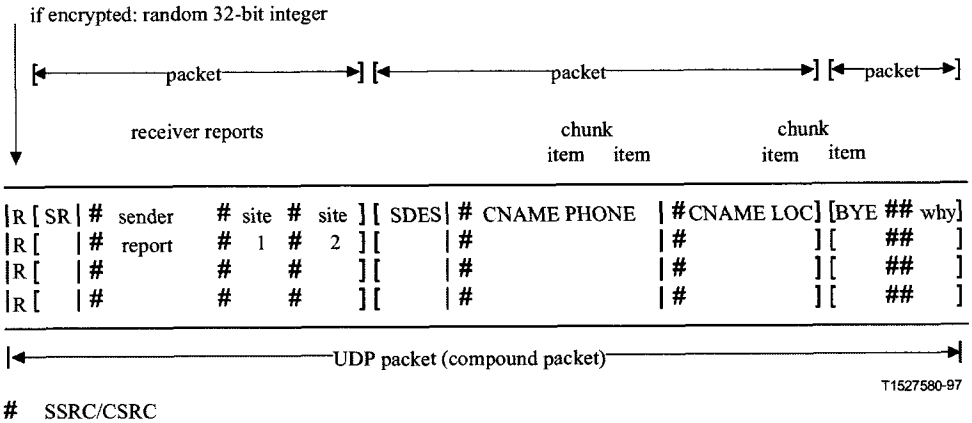


Figure A.1/H.225.0 – Example of an RTCP compound packet

A.6.2 RTCP transmission interval

RTP is designed to allow an application to scale automatically over session sizes ranging from a few participants to thousands. For example, in an audio conference the data traffic is inherently self-limiting because only one or two people will speak at a time, so with multicast distribution the data rate on any given link remains relatively constant independent of the number of participants. However, the control traffic is not self-limiting. If the reception reports from each participant were sent at a constant rate, the control traffic would grow linearly with the number of participants. Therefore, the rate must be scaled down.

For each session, it is assumed that the data traffic is subject to an aggregate limit called the "session bandwidth" to be divided among the participants. This bandwidth might be reserved and the limit enforced by the network, or it might just be a reasonable share. The session bandwidth may be chosen based on some cost or a priori knowledge of the available network bandwidth for the session. It is somewhat independent of the media encoding, but the encoding choice may be limited by the session bandwidth. The session bandwidth parameter is expected to be supplied by a session management application when it invokes a media application, but media applications may also set a default based on the single-sender data bandwidth for the encoding selected for the session. The application may also enforce bandwidth limits based on multicast scope rules or other criteria.

Bandwidth calculations for control and data traffic include lower-layer transport and network protocols (e.g. UDP and IP) since that is what the resource reservation system would need to know. The application can also be expected to know which of these protocols are in use. Link level headers are not included in the calculation since the packet will be encapsulated with different link level headers as it travels.

The control traffic should be limited to a small and known fraction of the session bandwidth: small so that the primary function of the transport protocol to carry data is not impaired; known so that the control traffic can be included in the bandwidth specification given to a resource reservation protocol, and so that each participant can independently calculate its share. It is suggested that the fraction of the session bandwidth allocated to RTCP be fixed at 5%. While the value of this and other constants in the interval calculation is not critical, all participants in the session must use the same values so the same interval will be calculated. Therefore, these constants should be fixed for a particular profile.

The algorithm described in A.7 was designed to meet the goals outlined above. It calculates the interval between sending compound RTCP packets to divide the allowed control traffic bandwidth among the participants. This allows an application to provide fast response for small sessions where, for example, identification of all participants is important, yet automatically adapt to large sessions. The algorithm incorporates the following characteristics:

- Senders are collectively allocated at least 1/4 of the control traffic bandwidth so that in sessions with a large number of receivers but a small number of senders, newly joining participants will more quickly receive the CNAME for the sending sites.
- The calculated interval between RTCP packets is required to be greater than a minimum of 5 seconds to avoid having bursts of RTCP packets exceed the allowed bandwidth when the number of participants is small and the traffic is not smoothed according to the law of large numbers.
- The interval between RTCP packets is varied randomly over the range [0.5, 1.5] times the calculated interval to avoid unintended synchronization of all participants [A-8]. The first RTCP packet sent after joining a session is also delayed by a random variation of half the minimum RTCP interval in case the application is started at multiple sites simultaneously, for example as initiated by a session announcement.

- A dynamic estimate of the average compound RTCP packet size is calculated, including all those received and sent, to automatically adapt to changes in the amount of control information carried.

This algorithm may be used for sessions in which all participants are allowed to send. In that case, the session bandwidth parameter is the product of the individual sender's bandwidth times the number of participants, and the RTCP bandwidth is 5% of that.

A.6.2.1 Maintaining the number of session members

Calculation of the RTCP packet interval depends upon an estimate of the number of sites participating in the session. New sites are added to the count when they are heard, and an entry for each is created in a table indexed by the SSRC or CSRC identifier (see A.8.2, Collision resolution and loop detection) to keep track of them. New entries may not be considered valid until multiple packets carrying the new SSRC have been received (see A.6.1). Entries may be deleted from the table when an RTCP BYE packet with the corresponding SSRC identifier is received.

A participant may mark another site inactive, or delete it if not yet valid, if no RTP or RTCP packet has been received for a small number of RTCP report intervals (5 is suggested). This provides some robustness against packet loss. All sites must calculate roughly the same value for the RTCP report interval in order for this timeout to work properly.

Once a site has been validated, then if it is later marked inactive the state for that site should still be retained and the site should continue to be counted in the total number of sites sharing RTCP bandwidth for a period long enough to span typical network partitions. This is to avoid excessive traffic, when the partition heals, due to an RTCP report interval that is too small. A timeout of 30 minutes is suggested. Note that this is still larger than 5 times the largest value to which the RTCP report interval is expected to usefully scale, about 2 to 5 minutes.

A.6.2.2 Allocation of source description bandwidth

This specification defines several source description (SDES) items in addition to the mandatory CNAME item, such as NAME (personal name) and EMAIL (email address). It also provides a means to define new application-specific RTCP packet types. Applications should exercise caution in allocating control bandwidth to this additional information because it will slow down the rate at which reception reports and CNAME are sent, thus impairing the performance of the protocol. It is recommended that no more than 20% of the RTCP bandwidth allocated to a single participant be used to carry the additional information. Furthermore, it is not intended that all SDES items should be included in every application. Those that are included should be assigned a fraction of the bandwidth according to their utility. Rather than estimate these fractions dynamically, it is recommended that the percentages be translated statically into report interval counts based on the typical length of an item.

For example, an application may be designed to send only CNAME, NAME and EMAIL and not any others. NAME might be given much higher priority than EMAIL because the NAME would be displayed continuously in the application's user interface, whereas EMAIL would be displayed only when requested. At every RTCP interval, an RR packet and an SDES packet with the CNAME item would be sent. For a small session operating at the minimum interval, that would be every 5 seconds on the average. Every third interval (15 seconds), one extra item would be included in the SDES packet. Seven out of eight times this would be the NAME item, and every eighth time (2 minutes) it would be the EMAIL item.

When multiple applications operate in concert using cross-application binding through a common CNAME for each participant, for example in a multimedia conference composed of an RTP session

for each medium, the additional SDES information might be sent in only one RTP session. The other sessions would carry only the CNAME item.

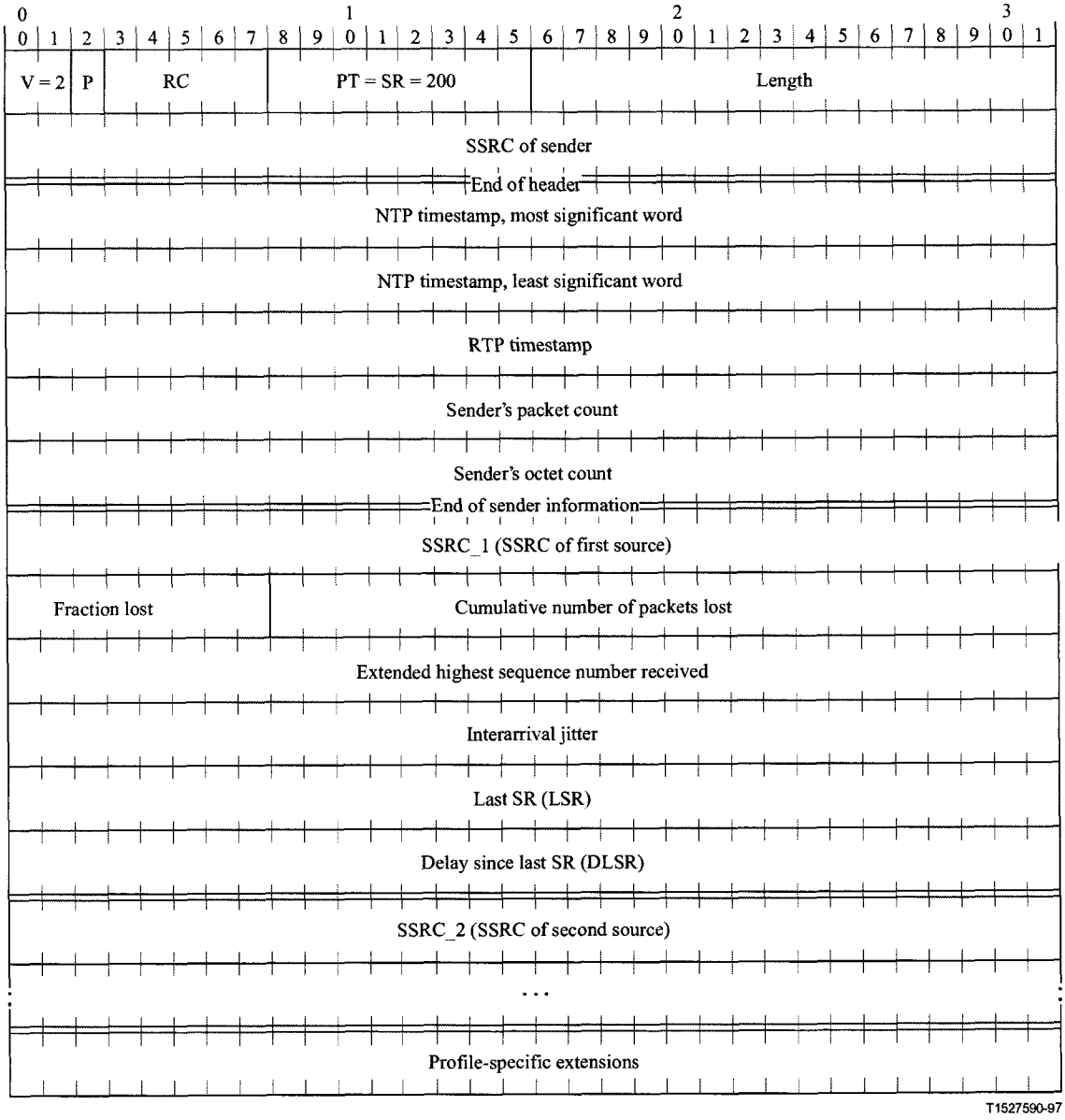
A.6.3 Sender and receiver reports

RTP receivers provide reception quality feedback using RTCP report packets which may take one of two forms depending upon whether or not the receiver is also a sender. The only difference between the Sender Report (SR) and Receiver Report (RR) forms, besides the packet type code, is that the sender report includes a 20-byte sender information section for use by active senders. The SR is issued if a site has sent any data packets during the interval since issuing the last report or the previous one, otherwise the RR is issued.

Both the SR and RR forms include zero or more reception report blocks, one for each of the synchronization sources from which this receiver has received RTP data packets since the last report. Reports are not issued for contributing sources listed in the CSRC list. Each reception report block provides statistics about the data received from the particular source indicated in that block. Since a maximum of 31 reception report blocks will fit in an SR or RR packet, additional RR packets may be stacked after the initial SR or RR packet as needed to contain the reception reports for all sources heard during the interval since the last report.

The next subclauses define the formats of the two reports, how they may be extended in a profile-specific manner if an application requires additional feedback information, and how the reports may be used. Details of reception reporting by translators and mixers are given in A.7, RTP translators and mixers.

A.6.3.1 SR: Sender Report RTCP packet



The sender report packet consists of three sections, possibly followed by a fourth profile-specific extension section if defined. The first section, the header, is 8 octets long. The fields have the following meaning:

version (V): 2 bits. Identifies the version of RTP, which is the same in RTCP packets as in RTP data packets. The version defined by this specification is two (2).

padding (P): 1 bit. If the padding bit is set, this RTCP packet contains some additional padding octets at the end which are not part of the control information. The last octet of the padding is a count of how many padding octets should be ignored. Padding may be needed by some encryption algorithms with fixed block sizes. In a compound RTCP packet, padding should only be required on the last individual packet because the compound packet is encrypted as a whole.

reception report count (RC): 5 bits. The number of reception report blocks contained in this packet. A value of zero is valid.

packet type (PT): 8 bits. Contains the constant 200 to identify this as an RTCP SR packet.

length: 16 bits. The length of this RTCP packet in 32-bit words minus one, including the header and any padding. (The offset of one makes zero a valid length and avoids a possible infinite loop in scanning a compound RTCP packet, while counting 32-bit words avoids a validity check for a multiple of 4.)

SSRC: 32 bits. The synchronization source identifier for the originator of this SR packet.

The second section, the sender information, is 20 octets long and is present in every sender report packet. It summarizes the data transmissions from this sender. The fields have the following meaning:

NTP timestamp: 64 bits. Indicates the wallclock time when this report was sent so that it may be used in combination with timestamps returned in reception reports from other receivers to measure round-trip propagation to those receivers. Receivers should expect that the measurement accuracy of the timestamp may be limited to far less than the resolution of the NTP timestamp. The measurement uncertainty of the timestamp is not indicated as it may not be known. A sender that can keep track of elapsed time but has no notion of wallclock time may use the elapsed time since joining the session instead. This is assumed to be less than 68 years, so the high bit will be zero. It is permissible to use the sampling clock to estimate elapsed wallclock time. A sender that has no notion of wallclock or elapsed time may set the NTP timestamp to zero.

RTP timestamp: 32 bits. Corresponds to the same time as the NTP timestamp (above), but in the same units and with the same random offset as the RTP timestamps in data packets. This correspondence may be used for intra- and inter-media synchronization for sources whose NTP timestamps are synchronized, and may be used by media- independent receivers to estimate the nominal RTP clock frequency. Note that in most cases this timestamp will not be equal to the RTP timestamp in any adjacent data packet. Rather, it is calculated from the corresponding NTP timestamp using the relationship between the RTP timestamp counter and real time as maintained by periodically checking the wallclock time at a sampling instant.

sender's packet count: 32 bits The total number of RTP data packets transmitted by the sender since starting transmission up until the time this SR packet was generated. The count is reset if the sender changes its SSRC identifier.

sender's octet count: 32 bits. The total number of payload octets (i.e. not including header or padding) transmitted in RTP data packets by the sender since starting transmission up until the time this SR packet was generated. The count is reset if the sender changes its SSRC identifier. This field can be used to estimate the average payload data rate.

The third section contains zero or more reception report blocks depending on the number of other sources heard by this sender since the last report. Each reception report block conveys statistics on the reception of RTP packets from a single synchronization source. Receivers do not carry over statistics when a source changes its SSRC identifier due to a collision. These statistics are:

SSRC_n (source identifier): 32 bits. The SSRC identifier of the source to which the information in this reception report block pertains.

fraction lost: 8 bits. The fraction of RTP data packets from source SSRC_n lost since the previous SR or RR packet was sent, expressed as a fixed point number with the binary point at the left edge of the field. (That is equivalent to taking the integer part after multiplying the loss fraction by 256.) This fraction is defined to be the number of packets lost divided by the number of packets expected, as defined in the next paragraph. An implementation is shown in A.6.3. If the loss is negative due to

duplicates, the fraction lost is set to zero. Note that a receiver cannot tell whether any packets were lost after the last one received, and that there will be no reception report block issued for a source if all packets from that source sent during the last reporting interval have been lost.

cumulative number of packets lost: 24 bits. The total number of RTP data packets from source SSRC_n 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 duplicates. Thus packets that arrive late are not counted as lost, and the loss may be negative if there are duplicates. The number of packets expected is defined to be the extended last sequence number received, as defined next, less the initial sequence number received. This may be calculated as shown in A.6.3.

extended highest sequence number received: 32 bits. The low 16 bits contain the highest sequence number received in an RTP data packet from source SSRC_n, and the most significant 16 bits extend that sequence number with the corresponding count of sequence number cycles, which may be maintained according to the algorithm in A.13. Note that different receivers within the same session will generate different extensions to the sequence number if their start times differ significantly.

interarrival jitter: 32 bits. An estimate of the statistical variance of the RTP data packet interarrival time, measured in timestamp units 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. As shown in the equation below, this is equivalent to the difference in the "relative transit time" for the two packets; the relative transit time is the difference between a packet's RTP timestamp and the receiver's clock at the time of arrival, measured in the same units.

If Si is the RTP timestamp from packet i, and Ri is the time of arrival in RTP timestamp units for packet i, then for two packets i and j, D may be expressed as:

$$D(i+j) = (Rj - Ri) - (Sj - Si) = (Rj - Sj) - (Ri - Si)$$

The interarrival jitter is calculated continuously as each data packet i is received from source SSRC_n, using this difference D for that packet and the previous packet i – 1 in order of arrival (not necessarily in sequence), according to the formula:

$$J = J + \frac{|D(i-1,i)| - J}{16}$$

Whenever a reception report is issued, the current value of J is sampled.

The jitter calculation is prescribed here to allow profile-independent monitors to make valid interpretations of reports coming from different implementations. This algorithm is the optimal first-order estimator and the gain parameter 1/16 gives a good noise reduction ratio while maintaining a reasonable rate of convergence [A-9]. A sample implementation is shown in A.8.

last SR timestamp (LSR): 32 bits. The middle 32 bits out of 64 in the NTP timestamp (as explained in A.4, Byte order, alignment, and time format) received as part of the most recent RTCP Sender Report (SR) packet from source SSRC_n. If no SR has been received yet, the field is set to zero.

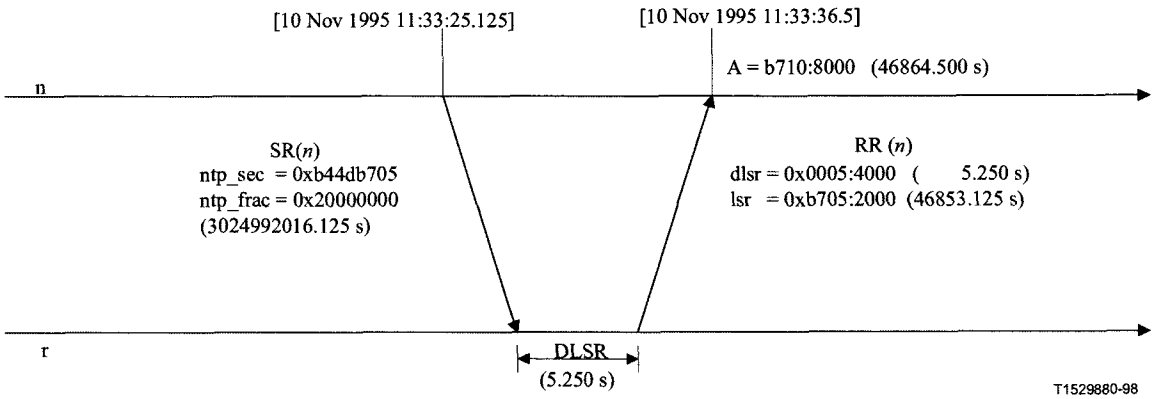
delay since last SR (DLSR): 32 bits. The delay, expressed in units of 1/65536 seconds, between receiving the last SR packet from source SSRC_n and sending this reception report block. If no SR packet has been received yet from SSRC_n, the DLSR field is set to zero.

Let SSRC_r denote the receiver issuing this receiver report. Source SSRC_n can compute the round propagation delay to SSRC_r by recording the time A when this reception report block is received. It calculates the total round-trip time A – LSR using the last SR timestamp (LSR) field, and then

subtracting this field to leave the round-trip propagation delay as (A – LSR – DLSR). This is illustrated in Figure A.2.

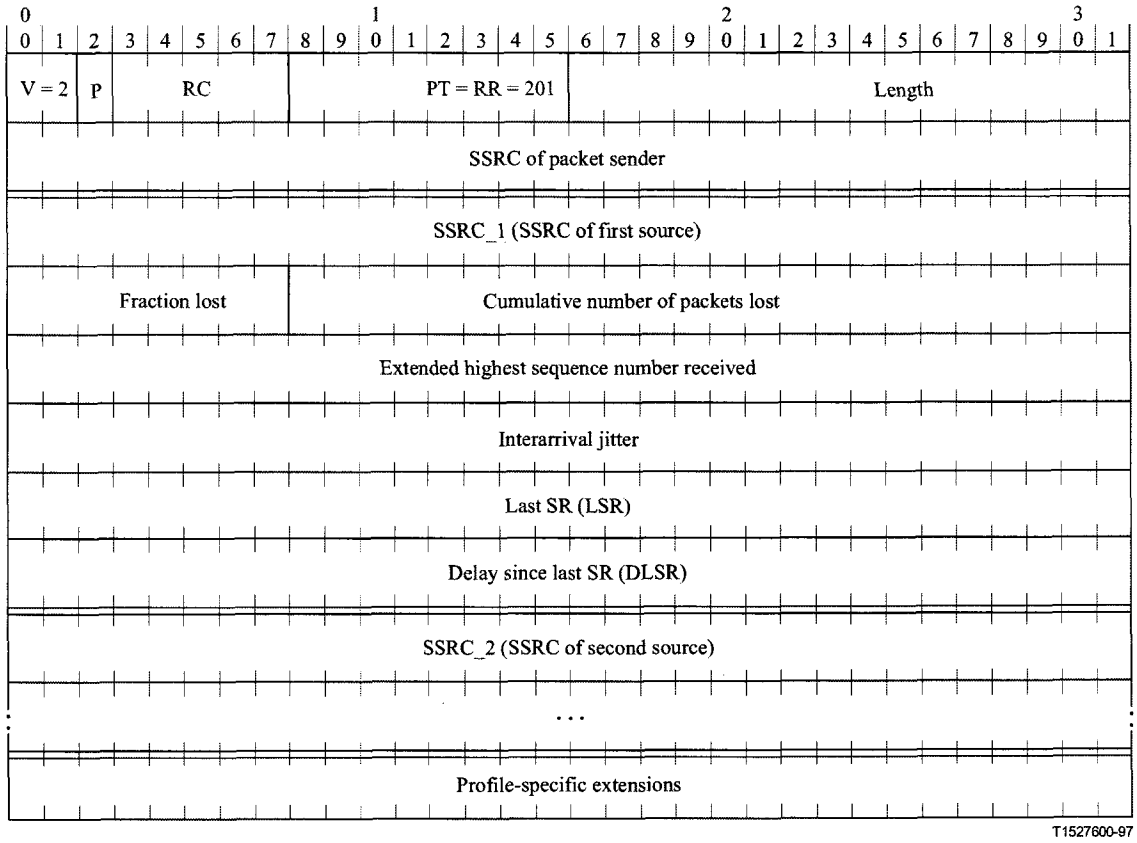
This may be used as an approximate measure of distance to cluster receivers, although some links have very asymmetric delays.

A.6.3.2 Receiver report RTCP packet



A	0xb710:8000	(46864.500 s)
– DLSR	0x0005:4000	(5.250 s)
– LSR	0xb705:2000	(46853.125 s)
= delay	0x0006:2000	(6.125 s)

Figure A.2/H.225.0 – Example for round-trip time computation



The format of the Receiver Report (RR) packet is the same as that of the SR packet except that the packet type field contains the constant 201 and the five words of sender information are omitted (these are the NTP and RTP timestamps and sender's packet and octet counts). The remaining fields have the same meaning as for the SR packet.

An empty RR packet (RC = 0) is put at the head of a compound RTCP packet when there is no data transmission or reception to report.

A.6.3.3 Extending the sender and receiver reports

A profile should define profile- or application-specific extensions to the sender report and receiver if there is additional information that should be reported regularly about the sender or receivers. This method should be used in preference to defining another RTCP packet type because it requires less overhead:

- fewer octets in the packet (no RTCP header or SSRC field);
- simpler and faster parsing because applications running under that profile would be programmed to always expect the extension fields in the directly accessible location after the reception reports.

If additional sender information is required, it should be included first in the extension for sender reports, but would not be present in receiver reports. If information about receivers is to be included, these data may be structured as an array of blocks parallel to the existing array of reception report blocks; that is, the number of blocks would be indicated by the RC field.

A.6.3.4 Analysing sender and receiver reports

It is expected that reception quality feedback will be useful not only for the sender but also for other receivers and third-party monitors. The sender may modify its transmissions based on the feedback;

receivers can determine whether problems are local, regional or global; network managers may use profile-independent monitors that receive only the RTCP packets and not the corresponding RTP data packets to evaluate the performance of their networks for multicast distribution.

Cumulative counts are used in both the sender information and receiver report blocks so that differences may be calculated between any two reports to make measurements over both short and long time periods, and to provide resilience against the loss of a report. The difference between the last two reports received can be used to estimate the recent quality of the distribution. The NTP timestamp is included so that rates may be calculated from these differences over the interval between two reports. Since that timestamp is independent of the clock rate for the data encoding, it is possible to implement encoding- and profile-independent quality monitors.

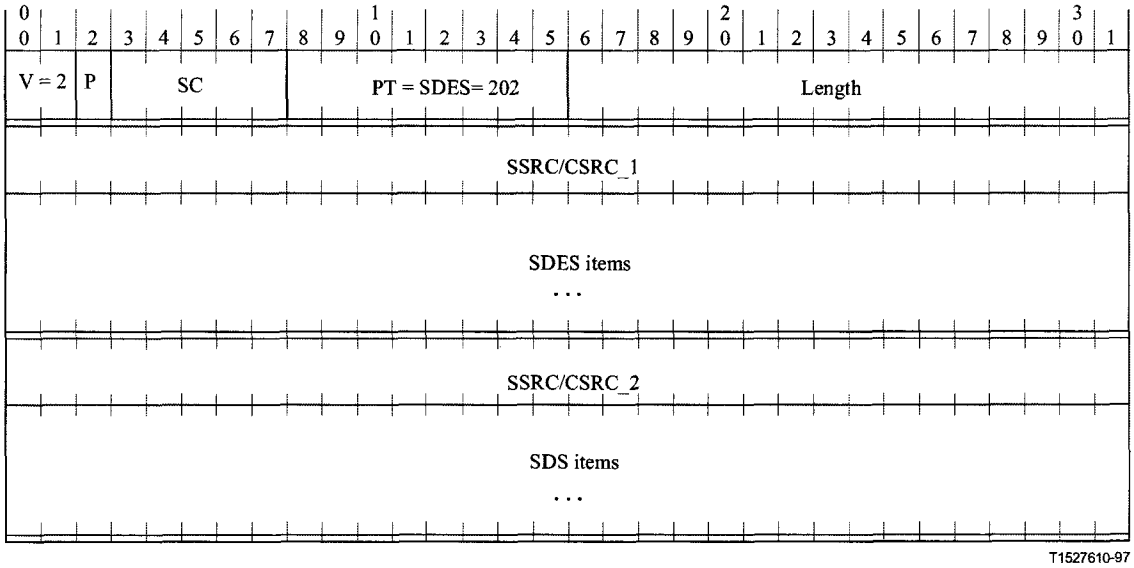
An example calculation is the packet loss rate over the interval between two reception reports. The difference in the cumulative number of packets lost gives the number lost during that interval. The difference in the extended last sequence numbers received gives the number of packets expected during the interval. The ratio of these two is the packet loss fraction over the interval. This ratio should equal the fraction lost field if the two reports are consecutive, but otherwise not. The loss rate per second can be obtained by dividing the loss fraction by the difference in NTP timestamps, expressed in seconds. The number of packets received is the number of packets expected minus the number lost. The number of packets expected may also be used to judge the statistical validity of any loss estimates. For example, 1 out of 5 packets lost has a lower significance than 200 out of 1000.

From the sender information, a third-party monitor can calculate the average payload data rate and the average packet rate over an interval without receiving the data. Taking the ratio of the two gives the average payload size. If it can be assumed that packet loss is independent of packet size, then the number of packets received by a particular receiver times the average payload size (or the corresponding packet size) gives the apparent throughput available to that receiver.

In addition to the cumulative counts which allow long-term packet loss measurements using differences between reports, the fraction lost field provides a short-term measurement from a single report. This becomes more important as the size of a session scales up enough that reception state information might not be kept for all receivers or the interval between reports becomes long enough that only one report might have been received from a particular receiver.

The interarrival jitter field provides a second short-term measure of network congestion. Packet loss tracks persistent congestion while the jitter measure tracks transient congestion. The jitter measure may indicate congestion before it leads to packet loss. Since the interarrival jitter field is only a snapshot of the jitter at the time of a report, it may be necessary to analyse a number of reports from one receiver over time or from multiple receivers, e.g. within a single network.

A.6.4 SDES: Source Description RTCP packet



The SDES packet is a three-level structure composed of a header and zero or more chunks, each of which is composed of items describing the source identified in that chunk. The items are described individually in subsequent subclauses.

version (V), padding (P), length: As described for the SR packet (see A.6.3.1, SR: Sender Report RTCP packet).

packet type (PT): 8 bits. Contains the constant 202 to identify this as an RTCP SDES packet.

source count (SC): 5 bits. The number of SSRC/CSRC chunks contained in this SDES packet. A value of zero is valid but useless.

Each chunk consists of an SSRC/CSRC identifier followed by a list of zero or more items, which carry information about the SSRC/CSRC. Each chunk starts on a 32-bit boundary. Each item consists of an 8-bit type field, an 8-bit octet count describing the length of the text (thus, not including this two-octet header), and the text itself. Note that the text can be no longer than 255 octets, but this is consistent with the need to limit RTCP bandwidth consumption.

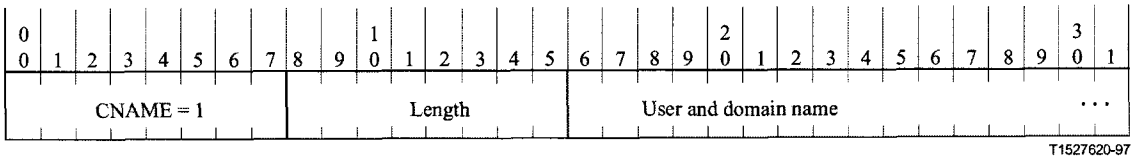
The text is encoded according to the UTF-2 encoding specified in Annex F of ISO/IEC 10646-1 [A-10]. This encoding is also known as UTF-8 or UTF-FSS. It is described in "File System Safe UCS Transformation Format (FSS_UTF)", X/Open Preliminary Specification, Document Number P316 and Unicode Technical Report No. 4. US-ASCII is a subset of this encoding and requires no additional encoding. The presence of multi-octet encodings is indicated by setting the most significant bit of a character to a value of one.

Items are contiguous, i.e. items are not individually padded to a 32-bit boundary. Text is not null terminated because some multi-octet encodings include null octets. The list of items in each chunk is terminated by one or more null octets, the first of which is interpreted as an item type of zero to denote the end of the list, and the remainder as needed to pad until the next 32-bit boundary. A chunk with zero items (four null octets) is valid but useless.

End systems send one SDES packet containing their own source identifier (the same as the SSRC in the fixed RTP header). A mixer sends one SDES packet containing a chunk for each contributing source from which it is receiving SDES information, or multiple complete SDES packets in the format above if there are more than 31 such sources (see A.2.3, Mixers and translators).

The SDES items currently defined are described in the next subclauses. Only the CNAME item is mandatory. Some items shown here may be useful only for particular profiles, but the item types are all assigned from one common space to promote shared use and to simplify profile- independent applications. Additional items may be defined in a profile by registering the type numbers with IANA.

A.6.4.1 CNAME: Canonical end-point identifier SDES item



The CNAME identifier has the following properties:

- Because the randomly allocated SSRC identifier may change if a conflict is discovered or if a program is restarted, the CNAME item is required to provide the binding from the SSRC identifier to an identifier for the source that remains constant.
- Like the SSRC identifier, the CNAME identifier should also be unique among all participants within one RTP session.
- To provide a binding across multiple media tools used by one participant in a set of related RTP sessions, the CNAME should be fixed for that participant.
- To facilitate third-party monitoring, the CNAME should be suitable for either a program or a person to locate the source.

Therefore, the CNAME should be derived algorithmically and not entered manually, when possible. To meet these requirements, the following format should be used unless a profile specifies an alternate syntax or semantics. The CNAME item should have the format "user@host", or "host" if a user name is not available as on single-user systems. For both formats, "host" is either the fully qualified domain name of the host from which the real-time data originates, formatted according to the rules specified in RFC 1034 [A-11], RFC 1035 [A-12] and 2.1/RFC 1123 [A-13]; or the standard ASCII representation of the host's numeric address on the interface used for the RTP communication. For example, the standard ASCII representation of an IP Version 4 address is "dotted decimal", also known as dotted quad. Other address types are expected to have ASCII representations that are mutually unique. The fully qualified domain name is more convenient for a human observer and may avoid the need to send a NAME item in addition, but it may be difficult or impossible to obtain reliably in some operating environments. Applications that may be run in such environments should use the ASCII representation of the address instead.

Examples are "doe@sleepy.megacorp.com" or "doe@192.0.2.89" for a multi-user system. On a system with no user name, examples would be "sleepy.megacorp.com" or "192.0.2.89".

The user name should be in a form that a program such as "finger" or "talk" could use, i.e. it typically is the login name rather than the personal name. The host name is not necessarily identical to the one in the participant's electronic mail address.

This syntax will not provide unique identifiers for each source if an application permits a user to generate multiple sources from one host. Such an application would have to rely on the SSRC to further identify the source, or the profile for that application would have to specify additional syntax for the CNAME identifier.

If each application creates its CNAME independently, the resulting CNAMEs may not be identical as would be required to provide a binding across multiple media tools belonging to one participant in

a set of related RTP sessions. If cross-media binding is required, it may be necessary for the CNAME of each tool to be externally configured with the same value by a coordination tool. Application writers should be aware that private network address assignments such as the Net-10 assignment proposed in RFC 1597 [A-14] may create network addresses that are not globally unique. This would lead to non-unique CNAMEs if hosts with private addresses and no direct IP connectivity to the public Internet have their RTP packets forwarded to the public Internet through an RTP-level translator. (See also RFC 1627 [A-15].) To handle this case, applications may provide a means to configure a unique CNAME, but the burden is on the translator to translate CNAMEs from private addresses to public addresses if necessary to keep private addresses from being exposed.

A.6.4.2 NAME: User name SDES item

See Appendix I.

A.6.4.3 EMAIL: Electronic mail address SDES item

See Appendix I.

A.6.4.4 PHONE: Phone number SDES item

See Appendix I.

A.6.4.5 LOC: Geographic user location SDES item

See Appendix I.

A.6.4.6 TOOL: Application or tool name SDES item

See Appendix I.

A.6.4.7 NOTE: Notice/status SDES item

See Appendix I.

A.6.4.8 PRIV: Private extensions SDES item

See Appendix I.

A.6.5 BYE: Goodbye RTCP packet

See Appendix I.

A.6.6 APP: Application-defined RTCP packet

See Appendix I.

A.7 RTP translators and mixers

In addition to end systems, RTP supports the notion of "translators" and "mixers", which could be considered as "intermediate systems" at the RTP level. Although this support adds some complexity to the protocol, the need for these functions has been clearly established by experiments with multicast audio and video applications in the Internet. Example uses of translators and mixers given in this subclause stem from the presence of firewalls and low bandwidth connections, both of which are likely to remain.

A.7.1 General description

An RTP translator/mixer connects two or more transport-level "clouds". Typically, each cloud is defined by a common network and transport protocol (e.g. IP/UDP), multicast address or pair of unicast addresses, and transport level destination port. (Network-level protocol translators, such as IP

version 4 to IP version 6, may be present within a cloud invisibly to RTP.) One system may serve as a translator or mixer for a number of RTP sessions, but each is considered a logically separate entity.

In order to avoid creating a loop when a translator or mixer is installed, the following rules must be observed:

- Each of the clouds connected by translators and mixers participating in one RTP session either must be distinct from all the others in at least one of these parameters (protocol, address, port), or must be isolated at the network level from the others.
- A derivative of the first rule is that there must not be multiple translators or mixers connected in parallel unless by some arrangement they partition the set of sources to be forwarded.

Similarly, all RTP end systems that can communicate through one or more RTP translators or mixers share the same SSRC space, that is, the SSRC identifiers must be unique among all these end systems. Subclause A.8.2, Collision resolution and loop detection describes the collision resolution algorithm by which SSRC identifiers are kept unique and loops are detected.

There may be many varieties of translators and mixers designed for different purposes and applications. Some examples are to add or remove encryption, change the encoding of the data or the underlying protocols, or replicate between a multicast address and one or more unicast addresses. The distinction between translators and mixers is that a translator passes through the data streams from different sources separately, whereas a mixer combines them to form one new stream:

Translator: Forwards RTP packets with their SSRC identifier intact; this makes it possible for receivers to identify individual sources even though packets from all the sources pass through the same translator and carry the translator's network source address. Some kinds of translators will pass through the data untouched, but others may change the encoding of the data and thus the RTP data payload type and timestamp. If multiple data packets are re-encoded into one, or vice versa, a translator must assign new sequence numbers to the outgoing packets. Losses in the incoming packet stream may induce corresponding gaps in the outgoing sequence numbers. Receivers cannot detect the presence of a translator unless they know by some other means what payload type or transport address was used by the original source.

Mixer: Receives streams of RTP data packets from one or more sources, possibly changes the data format, combines the streams in some manner and then forwards the combined stream. Since the timing among multiple input sources will not generally be synchronized, the mixer will make timing adjustments among the streams and generate its own timing for the combined stream, so it is the synchronization source. Thus, all data packets forwarded by a mixer will be marked with the mixer's own SSRC identifier. In order to preserve the identity of the original sources contributing to the mixed packet, the mixer should insert their SSRC identifiers into the CSRC identifier list following the fixed RTP header of the packet. A mixer that is also itself a contributing source for some packet should explicitly include its own SSRC identifier in the CSRC list for that packet.

For some applications, it may be acceptable for a mixer not to identify sources in the CSRC list. However, this introduces the danger that loops involving those sources could not be detected.

The advantage of a mixer over a translator for applications like audio is that the output bandwidth is limited to that of one source even when multiple sources are active on the input side. This may be important for low-bandwidth links. The disadvantage is that receivers on the output side don't have any control over which sources are passed through or muted, unless some mechanism is implemented for remote control of the mixer. The regeneration of synchronization information by mixers also means that receivers cannot do intermedia synchronization of the original streams. A multimedia mixer could do it.

A collection of mixers and translators is shown in Figure A.3 to illustrate their effect on SSRC and CSRC identifiers. In the figure, end systems are shown as rectangles (named E), translators as diamonds (named T) and mixers as ovals (named M). The notation "M1:48 (1, 17)" designates a packet originating a mixer M1, identified with M1's (random) SSRC value of 48 and two CSRC identifiers, 1 and 17, copied from the SSRC identifiers of packets from E1 and E2.

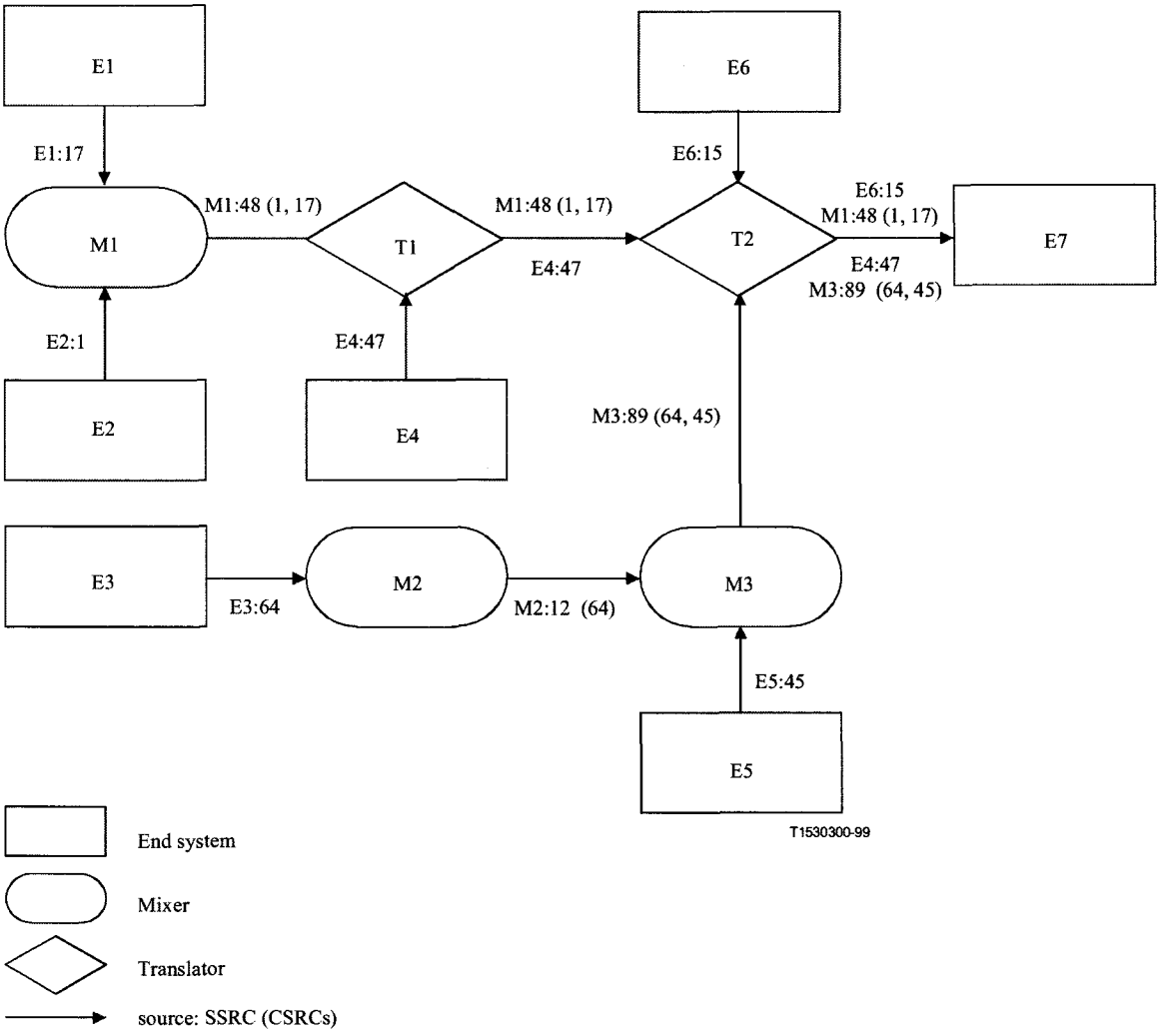


Figure A.3/H.225.0 – Sample RTP network with end systems, mixers and translators

A.7.2 RTCP processing in translators

In addition to forwarding data packets, perhaps modified, translators and mixers must also process RTCP packets. In many cases, they will take apart the compound RTCP packets received from end systems to aggregate SDES information and to modify the SR or RR packets. Retransmission of this information may be triggered by the packet arrival or by the RTCP interval timer of the translator or mixer itself.

A translator that does not modify the data packets, for example one that just replicates between a multicast address and a unicast address, may simply forward RTCP packets unmodified as well. A translator that transforms the payload in some way must make corresponding transformations in the

SR and RR information so that it still reflects the characteristics of the data and the reception quality. These translators must not simply forward RTCP packets. In general, a translator should not aggregate SR and RR packets from different sources into one packet since that would reduce the accuracy of the propagation delay measurements based on the LSR and DLSR fields.

SR sender information: A translator does not generate its own sender information, but forwards the SR packets received from one cloud to the others. The SSRC is left intact but the sender information must be modified if required by the translation. If a translator changes the data encoding, it must change the "sender's byte count" field. If it also combines several data packets into one output packet, it must change the "sender's packet count" field. If it changes the timestamp frequency, it must change the "RTP timestamp" field in the SR packet.

SR/RR reception report blocks: A translator forwards reception reports received from one cloud to the others. Note that these flow in the direction opposite to the data. The SSRC is left intact. If a translator combines several data packets into one output packet, and therefore changes the sequence numbers, it must make the inverse manipulation for the packet loss fields and the "extended last sequence number" field. This may be complex. In the extreme case, there may be no meaningful way to translate the reception reports, so the translator may pass on no reception report at all or a synthetic report based on its own reception. The general rule is to do what makes sense for a particular translation.

A translator does not require an SSRC identifier of its own, but may choose to allocate one for the purpose of sending reports about what it has received. These would be sent to all the connected clouds, each corresponding to the translation of the data stream as sent to that cloud, since reception reports are normally multicast to all participants.

SDES: Translators typically forward without change the SDES information they receive from one cloud to the others, but may, for example, decide to filter non-CNAME SDES information if bandwidth is limited. The CNAMEs must be forwarded to allow SSRC identifier collision detection to work. A translator that generates its own RR packets must send SDES CNAME information about itself to the same clouds to which it sends those RR packets.

BYE: Translators forward BYE packets unchanged. Translators with their own SSRC should generate BYE packets with that SSRC identifier if they are about to cease forwarding packets.

APP: Translators forward APP packets unchanged.

A.7.3 RTCP Processing in mixers

Since a mixer generates a new data stream of its own, it does not pass through SR or RR packets at all and instead generates new information for both sides.

SR sender information: A mixer does not pass through sender information from the sources it mixes because the characteristics of the source streams are lost in the mix. As a synchronization source, the mixer generates its own SR packets with sender information about the mixed data stream and sends them in the same direction as the mixed stream.

SR/RR reception report blocks: A mixer generates its own reception reports for sources in each cloud and sends them out only to the same cloud. It does not send these reception reports to the other clouds and does not forward reception reports from one cloud to the others because the sources would not be SSRCs there (only CSRCs).

SDES: Mixers typically forward without change the SDES information they receive from one cloud to the others, but may, for example, decide to filter non-CNAME SDES information if bandwidth is limited. The CNAMEs must be forwarded to allow SSRC identifier collision detection to work. (An identifier in a CSRC list generated by a mixer might collide with an SSRC identifier generated by an

end system.) A mixer must send SDES CNAME information about itself to the same clouds that it sends SR or RR packets.

Since mixers do not forward SR or RR packets, they will typically be extracting SDES packets from a compound RTCP packet. To minimize overhead, chunks from the SDES packets may be aggregated into a single SDES packet which is then stacked on an SR or RR packet originating from the mixer. The RTCP packet rate may be different on each side of the mixer.

A mixer that does not insert CSRC identifiers may also refrain from forwarding SDES CNAMEs. In this case, the SSRC identifier spaces in the two clouds are independent. As mentioned earlier, this mode of operation creates a danger that loops can't be detected.

BYE: Mixers need to forward BYE packets. They should generate BYE packets with their own SSRC identifiers if they are about to cease forwarding packets.

APP: The treatment of APP packets by mixers is application-specific.

A.7.4 Cascaded mixers

An RTP session may involve a collection of mixers and translators as shown in Figure A.3. If two mixers are cascaded, such as M2 and M3 in the figure, packets received by a mixer may already have been mixed and may include a CSRC list with multiple identifiers. The second mixer should build the CSRC list for the outgoing packet using the CSRC identifiers from already-mixed input packets and the SSRC identifiers from unmixed input packets. This is shown in the output arc from mixer M3 labelled M3:89 (64, 45) in the figure. As in the case of mixers that are not cascaded, if the resulting CSRC list has more than 15 identifiers, the remainder cannot be included.

A.8 SSRC identifier allocation and use

The SSRC identifier carried in the RTP header and in various fields of RTCP packets is a random 32-bit number that is required to be globally unique within an RTP session. It is crucial that the number be chosen with care in order that participants on the same network or starting at the same time are not likely to choose the same number.

It is not sufficient to use the local network address (such as an IPv4 address) for the identifier because the address may not be unique. Since RTP translators and mixers enable interoperation among multiple networks with different address spaces, the allocation patterns for addresses within two spaces might result in a much higher rate of collision than would occur with random allocation.

Multiple sources running on one host would also conflict.

It is also not sufficient to obtain an SSRC identifier simply by calling random () without carefully initializing the state. An example of how to generate a random identifier is presented in A.8.

A.8.1 Probability of Collision

Since the identifiers are chosen randomly, it is possible that two or more sources will choose the same number. Collision occurs with the highest probability when all sources are started simultaneously, for example when triggered automatically by some session management event. If N is the number of sources and L the length of the identifier (here, 32 bits), the probability that two sources independently pick the same value can be approximated for large N [20] as $1 - \exp^{-\frac{N^2}{2^{(L+1)}}}$. For N = 1000, the probability is roughly 10^{-4}.

The typical collision probability is much lower than the worst-case above. When one new source joins an RTP session in which all the other sources already have unique identifiers, the probability of collision is just the fraction of numbers used out of the space. Again, if N is the number of sources

and L the length of the identifier, the probability of collision is $\frac{N}{2^L}$. For N = 1000, the probability is roughly $2 \cdot 10^{-7}$. The probability of collision is further reduced by the opportunity for a new source to receive packets from other participants before sending its first packet (either data or control). If the new source keeps track of the other participants (by SSRC identifier), then before transmitting its first packet, the new source can verify that its identifier does not conflict with any that have been received, or else choose again.

A.8.2 Collision resolution and loop detection

Although the probability of SSRC identifier collision is low, all RTP implementations must be prepared to detect collisions and take the appropriate actions to resolve them. If a source discovers at any time that another source is using the same SSRC identifier as its own, it must send an RTCP BYE packet for the old identifier and choose another random one. If a receiver discovers that two other sources are colliding, it may keep the packets from one and discard the packets from the other when this can be detected by different source transport addresses or CNAMEs. The two sources are expected to resolve the collision so that the situation does not last.

Because the random identifiers are kept globally unique for each RTP session, they can also be used to detect loops that may be introduced by mixers or translators. A loop causes duplication of data and control information, either unmodified or possibly mixed, as in the following examples:

- A translator may incorrectly forward a packet to the same multicast group from which it has received the packet, either directly or through a chain of translators. In that case, the same packet appears several times, originating from different network sources.
- Two translators incorrectly set up in parallel, i.e. with the same multicast groups on both sides, would both forward packets from one multicast group to the other. Unidirectional translators would produce two copies; bidirectional translators would form a loop.
- A mixer can close a loop by sending to the same transport destination upon which it receives packets, either directly or through another mixer or translator. In this case a source might show up both as an SSRC on a data packet and a CSRC in a mixed data packet.

A source may discover that its own packets are being looped, or that packets from another source are being looped (a third-party loop). Both loops and collisions in the random selection of a source identifier result in packets arriving with the same SSRC identifier but a different source transport address, which may be that of the end system originating the packet or an intermediate system. Consequently, if a source changes its source transport address, it must also choose a new SSRC identifier to avoid being interpreted as a looped source. Loops or collisions occurring on the far side of a translator or mixer cannot be detected using the source transport address if all copies of the packets go through the translator or mixer, however collisions may still be detected when chunks from two RTCP SDES packets contain the same SSRC identifier but different CNAMEs.

To detect and resolve these conflicts, an RTP implementation must include an algorithm similar to the one described below. It ignores packets from a new source or loop that collide with an established source. It resolves collisions with the participant's own SSRC identifier by sending an RTCP BYE for the old identifier and choosing a new one. However, when the collision was induced by a loop of the participant's own packets, the algorithm will choose a new identifier only once and thereafter ignore packets from the looping source transport address. This is required to avoid a flood of BYE packets.

This algorithm depends upon the source transport address being the same for both RTP and RTCP packets from a source. The algorithm would require modifications to support applications that don't meet this constraint.

This algorithm requires keeping a table indexed by source identifiers and containing the source transport address from which the identifier was (first) received, along with other state for that source. Each SSRC or CSRC identifier received in a data or control packet is looked up in this table in order to process that data or control information. For control packets, each element with its own SSRC, for example an SDES chunk, requires a separate look-up. (The SSRC in a reception report block is an exception.) If the SSRC or CSRC is not found, a new entry is created. These table entries are removed when an RTCP BYE packet is received with the corresponding SSRC, or after no packets have arrived for a relatively long time (see A.6.2.1, Maintaining the number of session members).

In order to track loops of the participant's own data packets, it is also necessary to keep a separate list of source transport addresses (not identifiers) that have been found to be conflicting. Note that this should be a short list, usually empty. Each element in this list stores the source address plus the time when the most recent conflicting packet was received. An element may be removed from the list when no conflicting packet has arrived from that source for a time on the order of 10 RTCP report intervals (see A.6.2, RTCP transmission interval).

For the algorithm as shown, it is assumed that the participant's own source identifier and state are included in the source identifier table. The algorithm could be restructured to first make a separate comparison against the participant's own source identifier.

IF the SSRC or CSRC identifier is not found in the source identifier table:

THEN create a new entry storing the source transport address and the SSRC or CSRC along with other state.

CONTINUE with normal processing.

(Identifier is found in the table.)

IF the source transport address from the packet matches the one saved in the table entry for this identifier:

THEN CONTINUE with normal processing.

(An identifier collision or a loop is indicated.)

IF the source identifier is not the participant's own:

THEN IF the source identifier is from an RTCP SDES chunk containing a CNAME item that differs from the CNAME in the table entry:

THEN (optionally) count a third-party collision.

ELSE (optionally) count a third-party loop.

ABORT processing of data packet or control element.

(A collision or loop of the participant's own data.)

IF the source transport address is found in the list of conflicting addresses:

THEN IF the source identifier is not from an RTCP SDES chunk containing a CNAME item OR if that CNAME is the participant's own:

THEN (optionally) count occurrence of own traffic looped. mark current time in conflicting address list entry.

ABORT processing of data packet or control element.

Log occurrence of a collision.

Create a new entry in the conflicting address list and mark current time.

Send an RTCP BYE packet with the old SSRC identifier.

Choose a new identifier.

Create a new entry in the source identifier table with the old SSRC plus the source transport address from the packet being processed.

CONTINUE with normal processing.

In this algorithm, packets from a newly conflicting source address will be ignored and packets from the original source will be kept. (If the original source was through a mixer and later the same source is received directly, the receiver may be well advised to switch unless other sources in the mix would be lost.) If no packets arrive from the original source for an extended period, the table entry will be timed out and the new source will be able to take over. This might occur if the original source detects the collision and moves to a new source identifier, but in the usual case an RTCP BYE packet will be received from the original source to delete the state without having to wait for a timeout.

When a new SSRC identifier is chosen due to a collision, the candidate identifier should first be looked up in the source identifier table to see if it was already in use by some other source. If so, another candidate should be generated and the process repeated.

A loop of data packets to a multicast destination can cause severe network flooding. All mixers and translators are required to implement a loop detection algorithm like the one here so that they can break loops. This should limit the excess traffic to no more than one duplicate copy of the original traffic, which may allow the session to continue so that the cause of the loop can be found and fixed. However, in extreme cases where a mixer or translator does not properly break the loop and high traffic levels result, it may be necessary for end systems to cease transmitting data or control packets entirely. This decision may depend upon the application. An error condition should be indicated as appropriate. Transmission might be attempted again periodically after a long, random time (on the order of minutes).

A.9 Security

See Appendix I for an informative look at some Internet security methods. H.323 privacy and key exchange methods are described in Recommendation H.323.

A.10 RTP over network and transport protocols

This subclause describes issues specific to carrying RTP packets within particular network and transport protocols. The following rules apply unless superseded by protocol-specific definitions outside this specification.

RTP relies on the underlying protocol(s) to provide demultiplexing of RTP data and RTCP control streams. For UDP and similar protocols, RTP uses an even port number and the corresponding RTCP stream uses the next higher (odd) port number. If an application is supplied with an odd number for use as the RTP port, it should replace this number with the next lower (even) number.

RTP data packets contain no length field or other delineation, therefore RTP relies on the underlying protocol(s) to provide a length indication. The maximum length of RTP packets is limited only by the underlying protocols.

If RTP packets are to be carried in an underlying protocol that provides the abstraction of a continuous octet stream rather than messages (packets), an encapsulation of the RTP packets must be defined to provide a framing mechanism. Framing is also needed if the underlying protocol may contain padding so that the extent of the RTP payload cannot be determined. The framing mechanism is not defined here.

A profile may specify a framing method to be used even when RTP is carried in protocols that do provide framing in order to allow carrying several RTP packets in one lower-layer protocol data unit,

such as a UDP packet. Carrying several RTP packets in one network or transport packet reduces header overhead and may simplify synchronization between different streams.

A.11 Summary of protocol constants

This subclause contains a summary listing of the constants defined in this specification.

The RTP Payload Type (PT) constants are defined in profiles rather than in this Recommendation. However, the octet of the RTP header which contains the marker bit(s) and payload type must avoid the reserved values 200 and 201 (decimal) to distinguish RTP packets from the RTCP SR and RR packet types for the header validation procedure described in A.6.3. For the standard definition of one marker bit and a 7-bit payload type field as shown in this specification, this restriction means that payload types 72 and 73 are reserved.

A.11.1 RTCP packet types

Abbreviations	Name	Value
SR	sender report	200
RR	receiver report	201
SDES	source description	202
BYE	goodbye	203
APP	application-defined	204

These type values were chosen in the range 200-204 for improved header validity checking of RTCP packets compared to RTP packets or other unrelated packets. When the RTCP packet type field is compared to the corresponding octet of the RTP header, this range corresponds to the marker bit being 1 (which it usually is not in data packets) and to the high bit of the standard payload type field being 1 (since the static payload types are typically defined in the low half). This range was also chosen to be some distance numerically from 0 and 255 since all-zeros and all-ones are common data patterns.

Since all compound RTCP packets must begin with SR or RR, these codes were chosen as an even/odd pair to allow the RTCP validity check to test the maximum number of bits with mask and value.

Other constants are assigned by IANA. Experimenters are encouraged to register the numbers they need for experiments, and then unregister those which prove to be unneeded.

A.11.2 SDES types

Abbreviations	Name	Value
END	end of SDES list	0
CNAME	canonical name	1
NAME	user name	2
EMAIL	user's electronic mail address	3
PHONE	user's phone number	4
LOC	geographic user location	5
TOOL	name of application or tool	6
NOTE	notice about the source	7
PRIV	private extensions	8

Other constants are assigned by IANA. Experimenters are encouraged to register the numbers they need for experiments, and then unregister those which prove to be unneeded.

A.12 RTP profiles and payload format specifications

A complete specification of RTP for a particular application will require one or more companion documents of two types described here: profiles, and payload format specifications.

RTP may be used for a variety of applications with somewhat differing requirements. The flexibility to adapt to those requirements is provided by allowing multiple choices in the main protocol specification, then selecting the appropriate choices or defining extensions for a particular environment and class of applications in a separate profile document. Typically, an application will operate under only one profile so there is no explicit indication of which profile is in use. A profile for audio and video applications may be found in Annex B.

The second type of companion document is a payload format specification, which defines how a particular kind of payload data, such as H.261 encoded video, should be carried in RTP. These documents are typically titled "RTP Payload Format for XYZ Audio/Video Encoding". Payload formats may be useful under multiple profiles and may therefore be defined independently of any particular profile. The profile documents are then responsible for assigning a default mapping of that format to a payload type value if needed. See Annex C for this information.

Within this specification, the following items have been identified for possible definition within a profile, but this list is not meant to be exhaustive:

RTP data header: The octet in the RTP data header that contains the marker bit and payload type field may be redefined by a profile to suit different requirements, for example with more or fewer marker bits (see A.5.3, Profile-specific modifications to the RTP header).

Payload types: Assuming that a payload type field is included, the profile will usually define a set of payload formats (e.g. media encodings) and a default static mapping of those formats to payload type values. Some of the payload formats may be defined by reference to separate payload format specifications. For each payload type defined, the profile must specify the RTP timestamp clock rate to be used (see A.5.1, RTP Fixed Header Fields).

RTP data header additions: Additional fields may be appended to the fixed RTP data header if some additional functionality is required across the profile's class of applications independent of payload type (see A.5.3, Profile-specific modifications to the RTP header).

RTP data header extensions: The contents of the first 16 bits of the RTP data header extension structure must be defined if use of that mechanism is to be allowed under the profile for implementation-specific extensions (see A.5.3, Profile-specific modifications to the RTP header).

RTCP packet types: New application-class-specific RTCP packet types may be defined and registered with IANA.

RTCP report interval: A profile should specify that the values suggested in A.6.2, RTCP Transmission Interval for the constants employed in the calculation of the RTCP report interval will be used. Those are the RTCP fraction of session bandwidth, the minimum report interval, and the bandwidth split between senders and receivers. A profile may specify alternate values if they have been demonstrated to work in a scalable manner.

SR/RR extension: An extension section may be defined for the RTCP SR and RR packets if there is additional information that should be reported regularly about the sender or receivers (see A.6.3.3, Extending the sender and receiver reports).

SDES use: The profile may specify the relative priorities for RTCP SDES items to be transmitted or excluded entirely (see A.6.2.2, Allocation of source description bandwidth); an alternate syntax or semantics for the CNAME item (see A.6.4.1, CNAME: Canonical end-point identifier SDES item); the format of the LOC item (see A.6.4.5, LOC: Geographic user location SDES item); the semantics and use of the NOTE item (see A.6.4.7, NOTE – Notice/status SDES item); or new SDES item types to be registered with IANA.

Security: A profile may specify which security services and algorithms should be offered by applications, and may provide guidance as to their appropriate use (see A.9, Security).

String-to-key mapping: A profile may specify how a user-provided password or pass phrase is mapped into an encryption key.

Underlying protocol: Use of a particular underlying network or transport layer protocol to carry RTP packets may be required.

Transport mapping: A mapping of RTP and RTCP to transport-level addresses, e.g. UDP ports, other than the standard mapping defined in Annex B, may be specified.

Encapsulation: An encapsulation of RTP packets may be defined to allow multiple RTP data packets to be carried in one lower-layer packet or to provide framing over underlying protocols that do not already do so (see A.10, RTP over Network and Transport Protocols).

A.13 Algorithms

This subclause can be found as Appendix I. All such sample implementations are non-normative and hence are not included here.

A.14 Bibliography

Note that the material in this bibliography is informative, and is not required to implement this annex.

[A-1] CLARK (D.D.) and TENNENHOUSE (D.L.): Architectural considerations for a new generation of protocols, in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Philadelphia, Pennsylvania), pp. 200-208, *IEEE*, September 1990. Computer Communications Review, Vol. 20 (4), September 1990.

[A-2] COMER (D.E.): Internetworking with TCP/IP, Vol. 1, *Prentice Hall*, Englewood Cliffs, New Jersey, 1991.

[A-3] POSTEL (J.): Internet protocol, RFC 791, *Internet Engineering Task Force*, September 1981.

[A-4] MILLS (D.): Network time protocol (v3), RFC 1305, *Internet Engineering Task Force*, April 1992.

[A-5] EASTLAKE (D.), CROCKER (S.) and SCHILLER (J.): Randomness recommendations for security, RFC 1750, *Internet Engineering Task Force*, December 1994.

[A-6] BOLOT (J.-C.), TURLETTI (T.) and WAKEMAN (I.): Scalable feedback control for multicast video distribution in the internet, in *SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 58-67, ACM, London, August 1994.

[A-7] BUSSE (I.), DEFFNER (B.) and SCHULZRINNE (H.): Dynamic QOS control of multimedia applications based on RTP, *Computer Communications*, January 1996.

[A-8] FLOYD (S.) and JACOBSON (V.): The synchronization of periodic routing messages, in *SIGCOMM Symposium on Communications Architectures and Protocols* (D. P. Sidhu, ed.,), pp. 33-44, ACM, (San Francisco, California) September 1993.

[A-9] CADZOW (J.A.): Foundations of digital signal processing and data analysis, Macmillan New York, 1987.

[A-10] ISO/IEC 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.*

[A-11] MOCKAPETRIS (P.): Domain names – Concepts and facilities, STD 13, RFC 1034, *Internet Engineering Task Force*, November 1987.

[A-12] MOCKAPETRIS (P.): Domain names – Implementation and specification, STD 13, RFC 1035, *Internet Engineering Task Force*, November 1987.

[A-13] BRADEN (R.): Requirements for internet hosts – Application and support, STD 3, RFC 1123, *Internet Engineering Task Force*, October 1989.

[A-14] REKHTER (Y.), MOSKOWITZ (R.), KARRENBERG (D.) and de GROOT (G.): Address allocation for private internets, RFC 1597, *Internet Engineering Task Force*, March 1994.

[A-15] LEAR (E.), FAIR (E.), CROCKER (D.) and KESSLER (T.): Network 10 considered harmful (some practices should not be codified), RFC 1627, *Internet Engineering Task Force*, July 1994.

[A-16] CROCKER (D.): Standard for the format of ARPA internet text messages, STD 11, RFC 822, *Internet Engineering Task Force*, August 1982.

[A-17] FELLER (W.): An Introduction to Probability Theory and its Applications, Vol. 1, John Wiley and Sons, third ed., New York, 1968.

[A-18] BALENSON (D.): Privacy enhancement for internet electronic mail: Part III: algorithms, modes, and identifiers, RFC 1423, *Internet Engineering Task Force*, February 1993.

[A-19] VOYDOCK (V.L.) and KENT (S.T.): Security mechanisms in high-level network protocols, *ACM Computing Surveys* , Vol. 15, pp. 135-171, June 1983.

[A-20] RIVEST (R.): The MD5 message-digest algorithm, RFC 1321, *Internet Engineering Task Force*, April 1992.

ANNEX B

RTP Profile

See the introduction to Annex A; all the warnings mentioned there apply to this annex as well. An informative reference to the full IETF document can be found in Appendix II; however, this annex contains all information needed for the implementation of Recommendation H.323.

B.1 Introduction

This profile defines aspects of RTP left unspecified in Annex A. This profile is intended for the use within audio and video conferences with minimal session control. In particular, no support for the negotiation of parameters or membership control is provided. The profile is expected to be useful in sessions where no negotiation or membership control are used (e.g. using the static payload types and the membership indications provided by RTCP), but this profile may also be useful in conjunction with a higher-level control protocol.

Use of this profile occurs by use of the appropriate applications; there is no explicit indication by port number, protocol identifier or the like.

Other profiles may make different choices for the items specified here.

B.2 RTP and RTCP packet forms and protocol behaviour

The subclause "RTP profiles and payload format specification" enumerates a number of items that can be specified or modified in a profile. This subclause addresses these items. Generally, this profile follows the default and/or recommended aspects of the RTP specification.

RTP data header: The standard format of the fixed RTP data header is used (one marker bit).

Payload types: Static payload types are defined in B.6 Payload type definitions.

RTP data header additions: No additional fixed fields are appended to the RTP data header.

RTP data header extensions: No RTP header extensions are defined, but applications operating under this profile may use such extensions. Thus, applications should not assume that the RTP header X bit is always zero and should be prepared to ignore the header extension. If a header extension is defined in the future, that definition must specify the contents of the first 16 bits in such a way that multiple different extensions can be identified.

RTCP packet types: No additional RTCP packet types are defined by this profile specification.

RTCP report interval: The suggested constants are to be used for the RTCP report interval calculation.

SR/RR extension: No extension section is defined for the RTCP SR or RR packet.

SDES use: Applications may use any of the SDES items described. While CNAME information is sent every reporting interval, other items should be sent only every fifth reporting interval.

Security: The RTP default security services are not the default under this profile.

String-to-key mapping: See Appendix II for this informative information.

Underlying protocol: Any underlying protocol is allowed an described in Appendix IV that meets certain requirements.

Transport mapping: The standard mapping of RTP and RTCP to transport-level addresses is used.

Encapsulation: No encapsulation of RTP packets is specified.

B.3 Payload types

See Appendix II for information on registering new payload types.

Note that not all encodings to be used by RTP need to be assigned a static payload type. Non-RTP means beyond the scope of this annex (such as directory services or invitation protocols) may be used to establish a dynamic mapping between a payload type drawn from the range 96-127 and an encoding. For implementor convenience, this profile contains descriptions of encodings which do not currently have a static payload type assigned to them.

The available payload type space is relatively small. Thus, new static payload types are assigned only if the following conditions are met:

- The encoding is of interest to the Internet community at large.
- It offers benefits compared to existing encodings and/or is required for interoperation with existing, widely deployed conferencing or multimedia systems.
- The description is sufficient to build a decoder.

B.4 Audio

B.4.1 Encoding-independent recommendations

For applications which send no packets during silence, the first packet of a talkspurt (first packet after a silence period) is distinguished by setting the marker bit in the RTP data header. Applications without silence suppression set the bit to zero.

The RTP clock rate used for generating the RTP timestamp is independent of the number of channels and the encoding; it equals the number of sampling periods per second. For N-channel encodings, each sampling period (say, 1/8000 of a second) generates N samples. (This terminology is standard, but somewhat confusing, as the total number of samples generated per second is then the sampling rate times the channel count.)

If multiple audio channels are used, channels are numbered left-to- right, starting at one. In RTP audio packets, information from lower-numbered channels precedes that from higher-numbered channels.

For more than two channels, the convention should use the following notation:

l left

r right

c centre

s surround

F front

R rear

Channels	Description	Channel					
		1	2	3	4	5	6
2	stereo	l	r				
3		l	r	c			
4	quadrophonic	Fl	Fr	Rl	Rr		
4		l	c	r	S		
5		Fl	Fr	Fc	Sl	Sr	
6		l	lc	c	r	rc	S

Samples for all channels belonging to a single sampling instant must be within the same packet. The interleaving of samples from different channels depends on the encoding. General guidelines are given in B.4.2, Guidelines for sample-based audio encodings.

The sampling frequency should be drawn from the set: 8000, 11 025, 16 000, 22 050, 24 000, 32 000, 44 100 and 48 000 Hz. (The Apple Macintosh computers have native sample rates of 22 254.54 and 11 127.27, which can be converted to 22 050 and 11 025 with acceptable quality by dropping 4 or 2 samples in a 20 ms frame.) However, most audio encodings are defined for a more restricted set of sampling frequencies. Receivers should be prepared to accept multichannel audio, but may choose to only play a single channel.

The following recommendations are default operating parameters. Applications should be prepared to handle other values. The ranges given are meant to give guidance to application writers, allowing a set of applications conforming to these guidelines to interoperate without additional negotiation. These guidelines are not intended to restrict operating parameters for applications that can negotiate a set of interoperable parameters, e.g. through a conference control protocol.

For packetized audio, the default packetization interval should have a duration of 20 ms, unless otherwise noted when describing the encoding. The packetization interval determines the minimum end-to-end delay; longer packets introduce less header overhead but higher delay and make packet loss more noticeable. For non-interactive applications such as lectures or links with severe bandwidth constraints, a higher packetization delay may be appropriate. A receiver should accept packets representing between 0 and 200 ms of audio data. This restriction allows reasonable buffer sizing for the receiver.

B.4.2 Guidelines for sample-based audio encodings

In sample-based encodings, each audio sample is represented by a fixed number of bits. Within the compressed audio data, codes for individual samples may span octet boundaries. An RTP audio packet may contain any number of audio samples, subject to the constraint that the number of bits per sample times the number of samples per packet yields an integral octet count. Fractional encodings produce less than one octet per sample.

The duration of an audio packet is determined by the number of samples in the packet.

For sample-based encodings producing one or more octets per sample, samples from different channels sampled at the same sampling instant are packed in consecutive octets. For example, for a two-channel encoding, the octet sequence is (left channel, first sample), (right channel, first sample), (left channel, second sample), (right channel, second sample) .... For multi-octet encodings, octets are transmitted in network byte order (i.e. most significant octet first).

The packing of sample-based encodings producing less than one octet per sample is encoding-specific.

B.4.3 Guidelines for frame-based audio encodings

Frame-based encodings encode a fixed-length block of audio into another block of compressed data, typically also of fixed length. For frame-based encodings, the sender may choose to combine several such frames into a single message. The receiver can tell the number of frames contained in a message since the frame duration is defined as part of the encoding.

For frame-based codecs, the channel order is defined for the whole block. That is, for two-channel audio, right and left samples are coded independently, with the encoded frame for the left channel preceding that for the right channel.

All frame-oriented audio codecs should be able to encode and decode several consecutive frames within a single packet. Since the frame size for the frame-oriented codecs is given, there is no need to use a separate designation for the same encoding, but with different number of frames per packet.

B.4.4 Audio encodings

The characteristics of standard audio encodings are shown in Table B.1 and their payload types are listed in Table B.2.

See Appendix II for information on any coding not listed in Table B.1. Support for such codings is not part of Recommendation H.323.

Table B.1/H.225.0 – Properties of audio encodings

Encoding	Sample/frame	Bits/sample	ms/frame
G722	Sample	8	
G728	Frame	N/A	2.5
PCMA	Sample	8	
PCMU	Sample	8	
G723	Frame	N/A	30
G729	Frame	N/A	10

B.4.4.1 G722

G722 is specified in Recommendation G.722, "7 kHz audio-coding within 64 kbit/s".

B.4.4.2 G728

G728 is specified in Recommendation G.728, "Coding of speech at 16 kbit/s using low-delay code excited linear prediction".

B.4.4.3 PCMA

PCMA is specified in Recommendation G.711. Audio data is encoded as eight bits per sample, after logarithmic scaling.

B.4.4.4 PCMU

PCMU is specified in Recommendation G.711. Audio data is encoded as eight bits per sample, after logarithmic scaling.

B.5 Video

The following video encodings are currently defined, with their abbreviated names used for identification. See Appendix II for any coding not described here. Such coding are not part of Recommendation H.323.

H261

The encoding is specified in Recommendation H.261. The packetization and RTP-specific properties are described in Annex C.

H263

The encoding is specified in Recommendation H.263. The packetization and RTP-specific properties are described in Annex E.

The following procedure is to be followed by H.323 entities wanting to transmit H.263 (1996 or 1998) video streams:

- In an **OpenLogicalChannel** message of H.245, a sender that wants to use the legacy payload format for H.263 (1996) widely used in the industry shall signal H.263 (1996) features only and shall omit the h2250LogicalChannelParameters.mediaPacketization.
- In an **OpenLogicalChannel** message of H.245, a sender that wants to use the payload format for H.263 (1996) defined in RFC 2190 shall specify **h2250LogicalChannelParameters.rtpPayloadType** as follows: { rfc-number = 2190, payloadType = 34 }.

- In general, in an **OpenLogicalChannel** message of H.245, a sender shall specify the payload format according to the semantics defined in Recommendation H.245. This is particularly to be followed for signalling the H.263+ (1998) payload format (as defined in Annex A) and potential successors.

B.6 Payload type definitions

Table B.2 defines this profile's static payload type values for the PT field of the RTP data header. In addition, payload type values in the range 96-127 may be defined dynamically through a conference control protocol, which is beyond the scope of this Recommendation. For example, a session directory could specify that for a given session, payload type 96 indicates PCMU encoding, 8000 Hz sampling rate, 2 channels. The payload type range marked "reserved" has been set aside so that RTCP and RTP packets can be reliably distinguished (see A.11, Summary of protocol constants).

An RTP source emits a single RTP payload type at any given time; the interleaving of several RTP payload types in a single RTP session is not allowed, but multiple RTP sessions may be used in parallel to send multiple media. The payload types currently defined in this profile carry either audio or video, but not both. However, it is allowed to define payload types that combine several media, e.g. audio and video, with appropriate separation in the payload format. Session participants agree through mechanisms beyond the scope of this specification on the set of payload types allowed in a given session. This set may, for example, be defined by the capabilities of the applications used, negotiated by a conference control protocol or established by agreement between the human participants.

All current video encodings use a timestamp frequency of 90 000 Hz, the same as the MPEG presentation time stamp frequency. This frequency yields exact integer timestamp increments for the typical 24 (HDTV), 25 (PAL), and 29.97 (NTSC) and 30 Hz (HDTV) frame rates and 50, 59.94 and 60 Hz field rates. While 90 kHz is the recommended rate for future video encodings used within this profile, other rates are possible. However, it is not sufficient to use the video frame rate (typically between 15 and 30 Hz) because that does not provide adequate resolution for typical synchronization requirements when calculating the RTP timestamp corresponding to the NTP timestamp in an RTCP SR packet (see Annex A). The timestamp resolution must also be sufficient for the jitter estimate contained in the receiver reports.

The standard video encodings and their payload types are listed in Table B.2.

Table B.2/H.225.0 – Payload Types (PT) for standard audio and video encodings

PT	Encoding name	Audio/video (A/V)	Clock rate (Hz)	Channels (audio)
0	PCMU	A	8 000	1
8	PCMA	A	8 000	1
9	G722	A	8 000	1
4	G723	A	8 000	1
15	G728	A	8 000	1
18	G729	A	8 000	1
31	H261	V	90 000	
34	H263	V	90 000	
96-127	Dynamic	?		
NOTE – Payload types 1-7, 10-14, 16-30, and 30-95 are reserved. See Appendix II for more information.				

B.7 Port Assignment

As specified in the RTP protocol definition, RTP data is to be carried on an even UDP port number and the corresponding RTCP packets are to be carried on the next higher (odd) port number.

Applications operating under this profile may use any such UDP port pair. For example, the port pair may be allocated randomly by a session management program. A single fixed port number pair cannot be required because multiple applications using this profile are likely to run on the same host, and there are some operating systems that do not allow multiple processes to use the same UDP port with different multicast addresses.

However, port numbers 5004 and 5005 have been registered for use with this profile for those applications that choose to use them as the default pair. Applications that operate under multiple profiles may use this port pair as an indication to select this profile if they are not subject to the constraint of the previous paragraph. Applications need not have a default and may require that the port pair be explicitly specified. The particular port numbers were chosen to lie in the range above 5000 to accommodate port number allocation practice within the Unix operating system, where port numbers below 1024 can only be used by privileged processes and port numbers between 1024 and 5000 are automatically assigned by the operating system.

ANNEX C

RTP payload format for H.261 video streams

See the introduction to Annex A; all the warnings mentioned there apply to this annex as well. An informative reference to the full IETF document can be found in Appendix III; however, this annex contains all information needed for the implementation of Recommendation H.323.

C.1 Introduction

Recommendation H.261 [12] specifies the encodings used by ITU-T compliant video-conference codecs. Although these encodings were originally specified for fixed data rate ISDN circuits, experiments have shown that they can also be used over packet-switched networks such as the Internet.

The purpose of this annex is to specify the RTP payload format for encapsulating H.261 video streams in RTP (see Annex A).

C.2 Structure of the packet stream

C.2.1 Overview of Recommendation H.261

The H.261 coding is organized as a hierarchy of groupings. The video stream is composed of a sequence of images, or frames, which are themselves organized as a set of Groups of Blocks (GOB). Note that H.261 "pictures" are referred as "frames" in this Recommendation. Each GOB holds a set of three lines of 11 Macro Blocks (MB). Each MB carries information on a group of 16×16 pixels: luminance information is specified for 4 blocks of 8×8 pixels, while chrominance information is given by two "red" and "blue" colour difference components at a resolution of only 8×8 pixels. These components and the codes representing their sampled values are as defined in ITU-R Recommendation BT.601 [C-3].

This grouping is used to specify information at each level of the hierarchy:

– At the frame level, one specifies information such as the delay from the previous frame, the image format, and various indicators.

- At the GOB level, one specifies the GOB number and the default quantifier that will be used for the MBs.
- At the MB level, one specifies which blocks are present and which did not change, and optionally a quantifier and motion vectors.

Blocks which have changed are encoded by computing the Discrete Cosine Transform (DCT) of their coefficients, which are then quantized and Huffman encoded (Variable Length Codes).

The H.261 Huffman encoding includes a special "GOB start" pattern, composed of 15 zeroes followed by a single 1, that cannot be imitated by any other code words. This pattern is included at the beginning of each GOB header (and also at the beginning of each frame header) to mark the separation between two GOBs, and is in fact used as an indicator that the current GOB is terminated. The encoding also includes a stuffing pattern, composed of seven zeroes followed by four ones; that stuffing pattern can only be entered between the encoding of MBs, or just before the GOB separator.

C.2.2 Considerations for packetization

H.261 codecs designed for operation over ISDN circuits produce a bit stream composed of several levels of encoding specified by Recommendation H.261 and companion Recommendations. The bits resulting from the Huffman encoding are arranged in 512-bit frames, containing 2 bits of synchronization, 492 bits of data and 18 bits of error correcting code. The 512-bit frames are then interlaced with an audio stream and transmitted over $p \times 64$ kbit/s circuits according to Recommendation H.221 [1].

When transmitting over the Internet, we will directly consider the output of the Huffman encoding. All the bits produced by the Huffman encoding stage will be included in the packet. We will not carry the 512-bit frames, as protection against bit errors can be obtained by other means. Similarly, we will not attempt to multiplex audio and video signals in the same packets, as UDP and RTP provide a much more efficient way to achieve multiplexing.

Directly transmitting the result of the Huffman encoding over an unreliable stream of UDP datagrams would, however, have poor error resistance characteristics. The result of the hierarchical structure of H.261 bit stream is that one needs to receive the information present in the frame header to decode the GOBs, as well as the information present in the GOB header to decode the MBs. Without precautions, this would mean that one has to receive all the packets that carry an image in order to properly decode its components.

If each image could be carried in a single packet, this requirement would not create a problem. However, a video image or even one GOB by itself can sometimes be too large to fit in a single packet. Therefore, the MB is taken as the unit of fragmentation. Packets must start and end on a MB boundary, i.e. a MB cannot be split across multiple packets. Multiple MBs may be carried in a single packet when they will fit within the maximal packet size allowed. This practice is recommended to reduce the packet send rate and packet overhead.

To allow each packet to be processed independently for efficient resynchronization in the presence of packet losses, some state information from the frame header and GOB header is carried with each packet to allow the MBs in that packet to be decoded. This state information includes the GOB number in effect at the start of the packet, the macroblock address predictor (i.e. the last MBA encoded in the previous packet), the quantizer value in effect prior to the start of this packet (GQUANT, MQUANT or zero in case of a beginning of GOB) and the reference Motion Vector Data (MVD) for computing the true MVDs contained within this packet. The bit stream cannot be fragmented between a GOB header and MB 1 of that GOB.

Moreover, since the compressed MB may not fill an integer number of octets, the data header contains two three-bit integers, SBIT and EBIT, to indicate the number of unused bits in the first and last octets of the H.261 data, respectively.

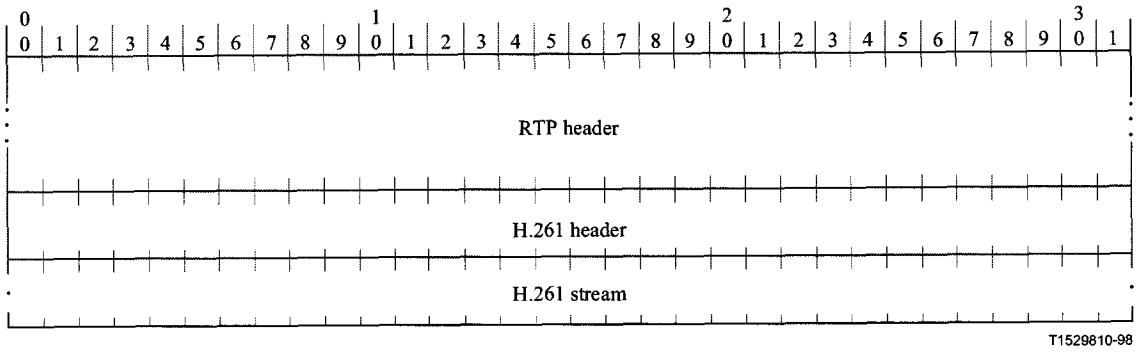
C.3 Specification of the packetization scheme

C.3.1 Usage of RTP

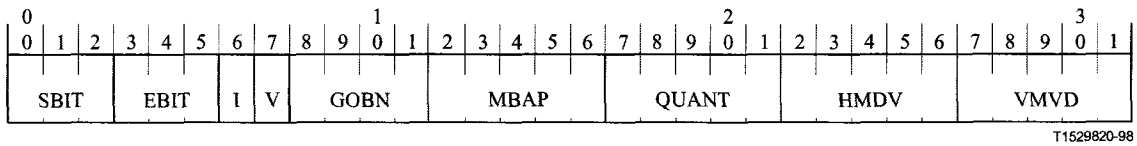
The H.261 information is carried as payload data within the RTP protocol. The following fields of the RTP header are specified:

- The payload type should specify H.261 payload format (see Annex B).
- The RTP timestamp encodes the sampling instant of the first video image contained in the RTP data packet. The RTP timestamp shall be the same on successive packets if a video image of the same video picture occupies more than one packet. For H.261 video streams, the RTP timestamp is based on a 90 kHz clock. This clock rate is a multiple of the natural H.261 frame rate (i.e. 30 000/1001 or approx. 29.97 Hz). That way, for each frame time, the clock is just incremented by the multiple and this removes inaccuracy in calculating the timestamp. Furthermore, the initial value of the timestamp is random (unpredictable) to make known-plaintext attacks on encryption more difficult, see RTP (Annex A). Note that if multiple frames are encoded in a packet (e.g. when there are very little changes between two images), it is necessary to calculate display times for the frames after the first using the timing information in the H.261 frame header. This is required because the RTP timestamp only gives the display time of the first frame in the packet.
- The marker bit of the RTP header is set to one in the last packet of a video frame, and otherwise, must be zero. Thus, it is not necessary to wait for a following packet (which contains the start code that terminates the current frame) to detect that a new frame should be displayed.

The H.261 data will follow the RTP header, as in:



The H.261 header is defined as following:



The fields in the H.261 header have the following meanings:

Start Bit Position (SBIT): 3 bits – Number of bits that should be ignored in the first data octet.

End Bit Position (EBIT): 3 bits – Number of bits that should be ignored in the last data octet.

INTRA-frame encoded data (I): 1 bit – Set to 1 if this stream contains only INTRA-frame coded blocks. Set to 0 if this stream may or may not contain INTRA-frame coded blocks. The sense of this bit may not change during the course of the session.

Motion Vector flag (V): 1 bit – Set to 0 if motion vectors are not used in this stream. Set to 1 if motion vectors may or may not be used in this stream. The sense of this bit may not change during the course of the session.

GOB Number (GOBN): 4 bits – Encodes the GOB number in effect at the start of the packet. Set to 0 if the packet begins with a GOB header.

Macroblock Address Predictor (MBAP): 5 bits – Encodes the macroblock address predictor (i.e. the last MBA encoded in the previous packet). This predictor ranges from 0-32 (to predict the valid MBAs 1-33), but because the bit stream cannot be fragmented between a GOB header and MB 1, the predictor at the start of the packet can never be 0. Therefore, the range is 1-32, which is biased by –1 to fit in 5 bits. For example, if MBAP is 0, the value of the MBA predictor is 1. Set to 0 if the packet begins with a GOB header.

Quantizer (QUANT): 5 bits – Quantizer value (MQUANT or GQUANT) in effect prior to the start of this packet. Set to 0 if the packet begins with a GOB header.

Horizontal Motion Vector Data (HMVD): 5 bits – Reference horizontal motion vector data (MVD). Set to 0 if V flag is 0 or if the packet begins with a GOB header. HMVD values are 5-bit 2's complement numbers directly representing the values [–16, +15], where –16 is not used.

Vertical motion vector data (VMVD): 5 bits – Reference vertical motion vector data (MVD). Set to 0 if V flag is 0 or if the packet begins with a GOB header. VMVD values are 5-bit 2's complement numbers directly representing the values [–16, +15], where –16 is not used.

Note that the I and V flags are hint flags, i.e. they can be inferred from the bit stream. They are included to allow decoders to make optimizations that would not be possible if these hints were not provided before bit stream was decoded. Therefore, these bits cannot change for the duration of the stream. A conformant implementation can always set V = 1 and I = 0.

Horizontal and vertical motion vector data must be set to zero when the MTYPE of the last MB encoded in the previous packet was not motion compensated.

C.3.2 Recommendations for operation with hardware codecs

Packetizers for hardware codecs can trivially figure out GOB boundaries using the GOB-start pattern included in the H.261 data. (Note that software encoders already know the boundaries.) The cheapest packetization implementation is to packetize at the GOB level all the GOBs that fit in a packet. But when a GOB is too large, the packetizer has to parse it to do MB fragmentation. (Note that only the

Huffman encoding must be parsed and that it is not necessary to fully decompress the stream, so this requires relatively little processing; example implementations can be found in Appendix III.) It is recommended that MB level fragmentation be used when feasible in order to obtain more efficient packetization. Using this fragmentation scheme reduces the output packet rate and therefore reduces the overhead.

At the receiver, the data stream can be depacketized and directed to a hardware codec's input. If the hardware decoder operates at a fixed bit rate, synchronization may be maintained by inserting the stuffing pattern between MBs (i.e. between packets) when the packet arrival rate is slower than the bit rate.

C.3.3 Packet loss issues

On the Internet, most packet losses are due to network congestion rather than transmission errors. Using UDP, no mechanism is available at the sender to know if a packet has been successfully received. It is up to the application, i.e. coder and decoder, to handle the packet loss. Each RTP packet includes a sequence number field which can be used to detect packet loss.

Recommendation H.261 uses the temporal redundancy of video to perform compression. This differential coding (or INTER-frame coding) is sensitive to packet loss. After a packet loss, parts of the image may remain corrupt until all corresponding MBs have been encoded in INTRA-frame mode (i.e. encoded independently of past frames). There are several ways to mitigate packet loss:

1) One way is to use only INTRA-frame encoding and MB level conditional replenishment. That is, only MBs that change (beyond some threshold) are transmitted.

2) Another way is to adjust the INTRA-frame encoding refreshment rate according to the packet loss observed by the receivers. Recommendation H.261 specifies that a MB is INTRA-frame encoded at least every 132 times it is transmitted. However, the INTRA-frame refreshment rate can be raised in order to speed the recovery when the measured loss rate is significant.

3) The fastest way to repair a corrupted image is to request an INTRA-frame coded image refreshment after a packet loss is detected. One means to accomplish this is for the decoder to send to the coder a list of packets lost. The coder can decide to encode every MB of every GOB of the following video frame in INTRA-frame mode (i.e. Full INTRA-frame encoded), or if the coder can deduce from the packet sequence numbers which MBs were affected by the loss, it can save bandwidth by sending only those MBs in INTRA-frame mode. This mode is particularly efficient in point-to-point connection or when the number of decoders is low. The next subclause specifies how the refresh function may be implemented.

C.3.4 Use of optional H.261-specific control packets

This specification defines two H.261-specific RTCP control packets, "Full INTRA-frame Request" and "Negative Acknowledgement", described in the next subclause Their purpose is to speed up refreshment of the video in those situations where their use is feasible. Support of these H.261-specific control packets by the H.261 sender is optional; in particular, early experiments have shown that the usage of this feature could have very negative effects when the number of sites is very large. Thus, these control packets should be used with caution.

The H.261-specific control packets differ from normal RTCP packets in that they are not transmitted to the normal RTCP destination transport address for the RTP session (which is often a multicast address). Instead, these control packets are sent directly via unicast from the decoder to the coder. The destination port for these control packets is the same port that the coder uses as a source port for transmitting RTP (data) packets. Therefore, these packets may be considered "reverse" control packets.

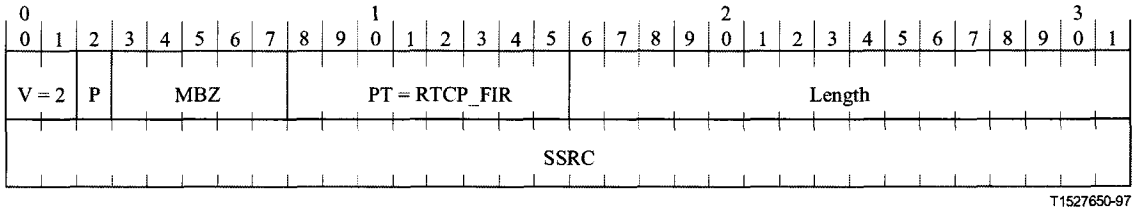
As a consequence, these control packets may only be used when no RTP mixers or translators intervene in the path from the coder to the decoder. If such intermediate systems do intervene, the address of the coder would no longer be present as the network-level source address in packets received by the decoder, and in fact, it might not be possible for the decoder to send packets directly to the coder.

Some reliable multicast protocols use similar NACK control packets transmitted over the normal multicast distribution channel, but they typically use random delays to prevent a NACK implosion problem . The goal of such protocols is to provide reliable multicast packet delivery at the expense of delay, which is appropriate for applications such as a shared whiteboard.

On the other hand, interactive video transmission is more sensitive to delay and does not require full reliability. For video applications it is more effective to send the NACK control packets as soon as possible, i.e. as soon as a loss is detected, without adding any random delays. In this case, multicasting the NACK control packets would generate useless traffic between receivers since only the coder will use them. But this method is only effective when the number of receivers is small, e.g. if the H.261-specific control packets are used only in point-to-point connections or in point-to-multipoint connections when there are less than 10 participants in the conference.

C.3.5 Control packets definition

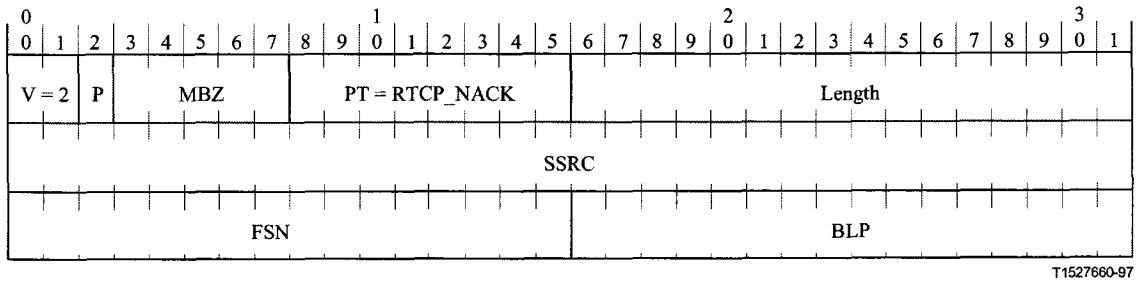
C.3.5.1 Full INTRA-frame Request (FIR) packet



This packet indicates that a receiver requires a full encoded image in order to either start decoding with an entire image or to refresh its image and speed the recovery after a burst of lost packets. The receiver requests the source to force the next image in full "INTRA-frame" coding mode, i.e. without using differential coding. The various fields are defined in the RTP specification [Annex A]. SSRC is the synchronization source identifier for the sender of this packet. The value of the packet type (PT) identifier is the constant RTCP_FIR (192).

C.3.5.2 Negative Acknowledgements (NACK) packet

The format of the NACK packet is as follows:



The various fields T, P, PT, length and SSRC are defined in the RTP specification (see Annex A). The value of the Packet Type (PT) identifier is the constant RTCP_NACK (193). SSRC is the synchronization source identifier for the sender of this packet.

The two remaining fields have the following meanings:

First Sequence Number (FSN): 16 bits – Identifies the first sequence number lost.

Bitmask of following Lost Packets (BLP): 16 bits – A bit is set to 1 if the corresponding packet has been lost, and set to 0 otherwise. BLP is set to 0 only if no packet other than that being NACKed (using the FSN field) has been lost. BLP is set to 0x00001 if the packet corresponding to the FSN and the following packet have been lost, etc.

C.4 Bibliography

[C-1] ITU-T Recommendation H.221 (1997), *Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices.*

[C-2] ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at p × 64 kbit/s.*

[C-3] ITU-R Recommendation BT.601 (1997), *Digital Methods of Transmitting Television Information.*

ANNEX D

RTP payload format for H.261A video streams

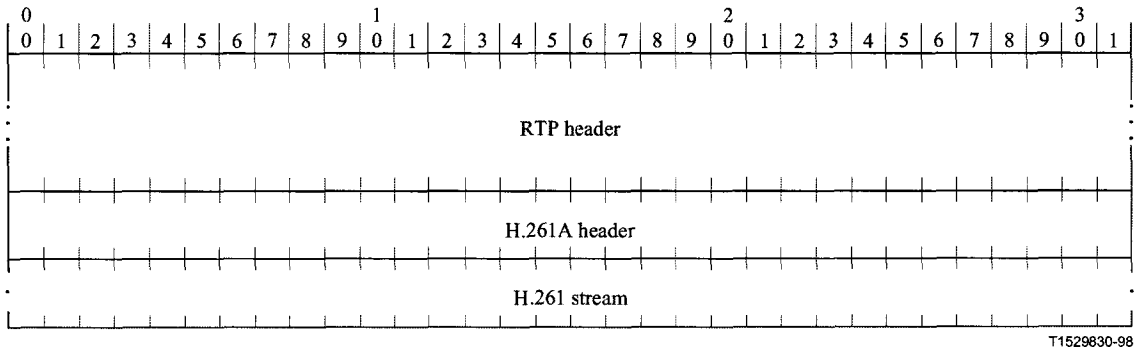
D.1 Introduction

To facilitate interfacing H.323 video streams to the SCN via gateways, Recommendation H.323 defines a modified form of the RTP H.261 video payload. This eases buffer management and interoperability with remote SCN codecs. Support of the H.261A payload type is signalled using H.245 capability sets and in the **openLogicalChannel** message using RTP dynamic payload types.

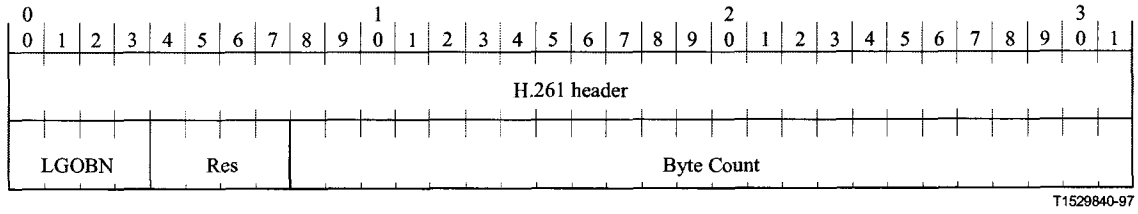
D.2 H.261A RTP packetization

This version is an extension of the version described in Annex C except that an additional 32-bit word is appended to the H.261 header. The procedures that are described in Annex C also apply to this annex.

The H.261A data will follow the RTP header, as in:



The H.261A header is defined as:



The fields in the H.261A header have the following meanings:

H.261 header: 32 bits – As described in Annex C.

Last GOB Number (LGOBN): 4 bits – The GOB number of the last GOB in the RTP packet (max GOB number is 12 for Recommendation H.261).

Reserved (RES): Reserved.

Byte Count: 24 bits – Indicates the cumulative number of octets that have been sent in the H.261 stream part of the RTP packets. If the last byte of a packet is only partially filled (as indicated by EBIT), then it is not counted in the cumulative byte count. This modulo 2^{24} byte count starts at a random value and is never reset.

Both of the additional fields may be used when packets are lost or delivered out of order. The Byte Count can be used to determine how much stuffing will be needed in the SCN stream and facilitates buffer management. The last GOB number simplifies determining which GOBs have been lost due to packet loss.

ANNEX E

Video packetization

An RTP payload format for H.263 video is specified in IETF RFC 2190 for H.263 video bitstreams that do not contain the new features adopted in version 2 (the 1998 version) of Recommendation H.263 (the features using PLUSPTYPE or annexes subsequent to Annex H/H.263). An additional payload format which supports the enhanced features of H.263 version 2 bitstreasms will be specified at a later date. A legacy packetization format widely used in industry (not as specified in

IETF RFC 2190) may only be used if the peer indicated support for this format in the capability exchange.

Subclause B.5 describes the procedure to use to signal H.263 video streams.

ANNEX F

Audio packetization

This annex describes RTP packetization details for audio codecs standardized by the ITU.

F.1 G.723.1

This Recommendation specifies a coded representation that can be used for compressing the speech signal component of multimedia services at a very low bit rate. A G.723.1 frame can be one of three sizes: 24 bytes (6.3 kbit/s frame), 20 bytes (5.3 kbit/s frame), or 4 bytes. These 4-byte frames are called SID frames (Silence Insertion Descriptor) and are used to specify comfort noise parameters. There is no restriction on how 4, 20, and 24 byte frames are intermixed. The least significant two bits of the first octet in the frame determine the frame size and codec type (refer to Table 5/G.723.1 and Table 6/G.723.1 for more information on bit order). It is possible to switch between the two rates at any 30 ms frame boundary. Both (5.3 kbit/s and 6.4 kbit/s) rates are a mandatory part of the encoder and decoder. This coder was optimized to represent speech with near-toll quality at the above rates using a limited amount of complexity.

All the bits of the encoded bit stream are transmitted always from the least significant bit towards the most significant bit.

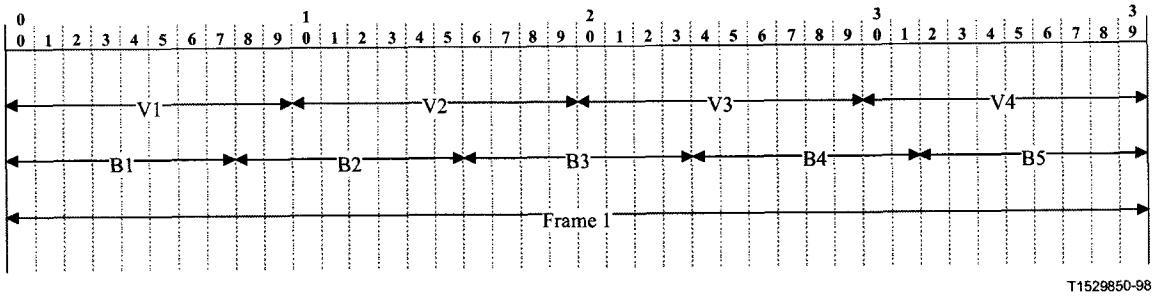
G.723.1 packetization conforms to Annex B except for the packetization interval (30 ms vs. 20 ms default):

1) The first packet of a talkspurt (first packet after a silence period) is distinguished by setting the marker bit in the RTP data header.
2) The sampling frequency (RTP clock frequency) is 8000 Hz.
3) The packetization interval shall have a duration of 30 ms (one frame) as opposed to the default packetization of 20 ms.
4) Codecs should be able to encode and decode several consecutive frames within a single packet.
5) A receiver should accept packets representing between 0 and 180 ms of audio data as opposed to the default of 0 and 200 ms.

F.2 G.728

1) *Frame packetization*:

   A G.728 frame (4 vectors: V1-V4, 10 bits each, V1 is the older – first to be played) is organized into 5 bytes (B1-B5). Referring to the figure below, the principle for bit order is "maintenance of bit significance". Bits from older vectors are more significant than bit from newer vectors. The Most Significant Bit (MSB) of the frame goes to MSB of B1 and the Least Significant Bit (LSB) of the frame goes to LSB of B5. For clarification: more significant bits from each vector are put in more significant bits of B1-B5 (the more significant bits of lower number B).



For example:

B1 contains 8 most significant bits of V1, MSB of V1 is MSB of B1.

B2 contains 2 Least Significant bits of V2, the more significant of the two in its MSB, and 6 most significant bits of V2, the most significant of them is more significant at B2 also.

B1 shall be put first to the packet (most significant byte in RTP) and B5 last.

2) *Multi Frame packetization*:

ending a single frame in an RTP packet might cause considerable network overhead. Therefore, sending a multi-frame packet is allowed in the following manner:

An RTP G.728 packet shall contain a whole number of frames.

Older frames (to be played first) shall be put first into the RTP packet.

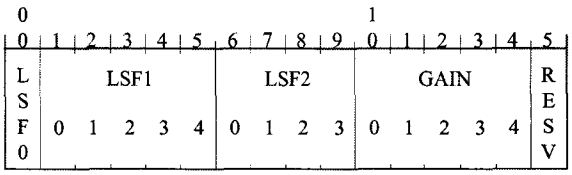
The time stamp would reflect the capturing time of the first sample, in the first vector (V1) of the first frame (the oldest information in the packet).

3) The marker bit shall retain the same meaning assigned to it in Recommendation H.225.

F.3 G.729

This Recommendation specifies a coded representation that can be used for compressing the speech signal component of multi-media services at a bit rate of 8 kbit/s. This coder was optimized to represent speech with toll or wireline quality at 8 kbit/s. This coder has an inherent robustness against random bit errors as well as against randomly and bursty erased frames. It represents speech with a high quality when operating in a noisy environment. A complexity-reduced version of the G.729 algorithm is specified in Annex A/G.729. The speech coding algorithms in the main body of Recommendation G.729 and in Annex A/G.729 are fully interoperable with each other, so there is no need to further distinguish between them.

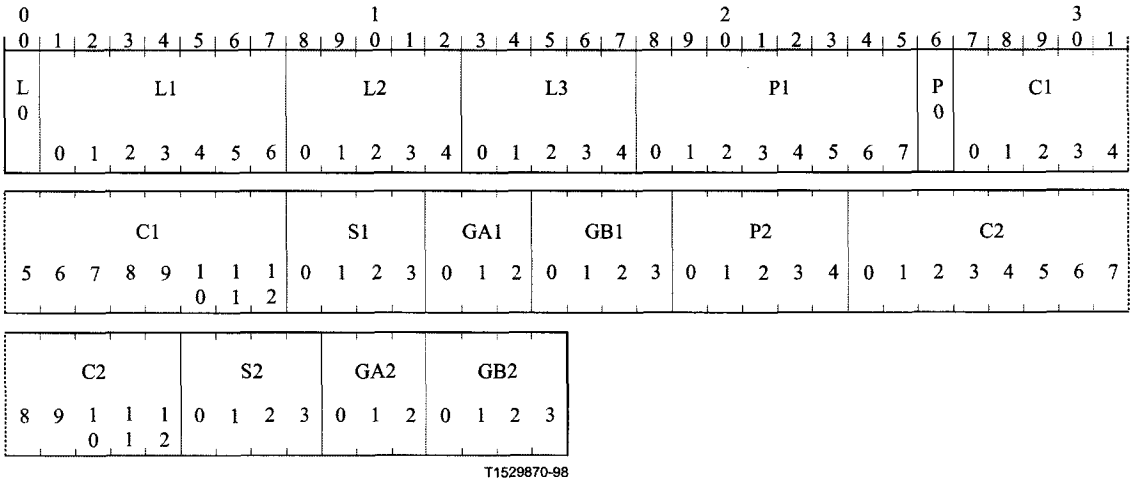
A Voice Activity Detector (VAD) and Comfort Noise Generator (CNG) algorithm in Annex B/G.729 is recommended for digital simultaneous voice and data applications and can be used in conjunction with Recommendation G.729 or Annex A/G.729. A G.729 or Annex A/G.729 frame contains 10 octets, while the Annex B/G.729 comfort noise frame occupies 2 octets:



RESV = Reserved (zero)

T1529860-98

The transmitted parameters of a G.729 and G.729 Annex A 10-ms frame, consisting of 80 bits, are defined in Table 8/G.729. The mapping of the these parameters is given below. Bits are numbered as Internet order, that is, the most significant bit is bit 0.



T1529870-98

An RTP packet may consist of zero or more G.729 or G.729 Annex A frames, followed by zero or one G.729 Annex B payloads. The presence of a comfort noise frame can be deduced from the length of the RTP payload.

1) The first packet of a talkspurt (first packet after a silence period) is distinguished by setting the marker bit in the RTP header.

2) The sampling frequency (RTP clock frequency) is 8000 Hz.

3) The default packetization interval should have a duration of 20 ms. While 20 ms is the strongly recommended value, in some situations it may be desireable to send 10 ms packets. For example, consider a transition from voiced to unvoiced in the first 10 ms of the packet. If a 20 ms packetization interval were mandatory, then the transmitter would need to wait until speech is active again.

4) Codecs should be able to encode and decode several consecutive frames within a single packet.

5) A receiver should accept packets representing between 0 and 200 ms of audio data.

F.4 Silence suppression

Recommendation H.225 states that coders may send silence frames before the stop transmission during a silence period. Since not all audio coders have in-band signalling for silence, a general mechanism at the RTP level should be defined. An example might be sending an empty RTP packet. This is for further study.

ANNEX G

Gatekeeper-to-gatekeeper communication

For further study.

ANNEX H

H.225.0 Message Syntax (ASN.1)

This Recommendation defines protocols for RAS (essentially a gatekeeper protocol) and call signalling (essentially protocol data units which reside in a User-to-user Information Element). These protocols are defined together in the following ASN.1 tree. Semantic definitions for the messages and various elements appear in previous clauses.

```
H323-MESSAGES DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS
        SIGNED{},
        ENCRYPTED{},
        HASHED{},
        ChallengeString,
        TimeStamp,
        RandomVal,
        Password,
        EncodedPwdCertToken,
        ClearToken,
        CryptoToken,
        AuthenticationMechanism
    FROM H235-SECURITY-MESSAGES

H323-UserInformation ::= SEQUENCE               -- root for all Q.931 related ASN.1
{
    h323-uu-pdu H323-UU-PDU,
    user-data   SEQUENCE
    {
        protocol-discriminator                  INTEGER   (0..255),
        user-information                        OCTET STRING (SIZE(1..131)),
        ...
    } OPTIONAL,
    ...
}

H323-UU-PDU ::= SEQUENCE
{
    h323-message-body  CHOICE
    {
        setup                   Setup-UUIE,
        callProceeding          CallProceeding-UUIE,
        connect                 Connect-UUIE,
        alerting                Alerting-UUIE,
        userInformation         Information-UUIE,
        releaseComplete         ReleaseComplete-UUIE,
        facility                Facility-UUIE,
        ...,
```

```
        progress                Progress-UUIE,
        empty                   NULL            -- used when a FACILITY message is sent,
                                                -- but the Facility-UUIE is not to be invoked
                                                -- (possible when transporting supplementary
                                                -- services messages)
    },
    nonStandardData         NonStandardParameter OPTIONAL,
    ...,
    h4501SupplementaryService   SEQUENCE OF OCTET STRING OPTIONAL
                                -- each sequence of octet string is defined as one
                                -- H4501SupplementaryService APDU as defined in
                                -- Table 3/H.450.1

    h245Tunneling           BOOLEAN,
                                -- if TRUE, tunneling of H.245 messages is enabled
    h245Control             SEQUENCE OF OCTET STRING OPTIONAL,
                                -- each octet string may contain exactly
                                -- one H.245 PDU
    nonStandardControl      SEQUENCE OF NonStandardParameter OPTIONAL
}

Alerting-UUIE ::= SEQUENCE
{
    protocolIdentifier      ProtocolIdentifier,
    destinationInfo         EndpointType,
    h245Address             TransportAddress OPTIONAL,
    ...,
    callIdentifier          CallIdentifier,
    h245SecurityMode        H245Security OPTIONAL,
    tokens                  SEQUENCE OF ClearToken OPTIONAL,
    cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
    fastStart               SEQUENCE OF OCTET STRING OPTIONAL,
}

CallProceeding-UUIE ::= SEQUENCE
{
    protocolIdentifier      ProtocolIdentifier,
    destinationInfo         EndpointType,
    h245Address             TransportAddress OPTIONAL,
    ...,
    callIdentifier          CallIdentifier,
    h245SecurityMode        H245Security OPTIONAL,
    tokens                  SEQUENCE OF ClearToken OPTIONAL,
    cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
    fastStart               SEQUENCE OF OCTET STRING OPTIONAL,
}

Connect-UUIE ::= SEQUENCE
{
    protocolIdentifier      ProtocolIdentifier,
    h245Address             TransportAddress OPTIONAL,
    destinationInfo         EndpointType,
    conferenceID            ConferenceIdentifier,
    ...,
    callIdentifier          CallIdentifier,
    h245SecurityMode        H245Security OPTIONAL,
    tokens                  SEQUENCE OF ClearToken OPTIONAL,
    cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
    fastStart               SEQUENCE OF OCTET STRING OPTIONAL,
}
```

```
Information-UUIE ::=SEQUENCE
{
        protocolIdentifier          ProtocolIdentifier,
        ...
        callIdentifier              CallIdentifier
}

ReleaseComplete-UUIE ::= SEQUENCE
{
        protocolIdentifier          ProtocolIdentifier,
        reason                      ReleaseCompleteReason OPTIONAL,
        ...,
        callIdentifier              CallIdentifier
}

ReleaseCompleteReason ::= CHOICE
{
        noBandwidth                 NULL,       -- bandwidth taken away or ARQ denied
        gatekeeperResources         NULL,       -- exhausted
        unreachableDestination      NULL,       -- no transport path to the destination
        destinationRejection        NULL,       -- rejected at destination
        invalidRevision             NULL,
        noPermission                NULL,       -- called party's gatekeeper rejects
        unreachableGatekeeper       NULL,       -- terminal cannot reach gatekeeper for ARQ
        gatewayResources            NULL,
        badFormatAddress            NULL,
        adaptiveBusy                NULL,       -- call is dropping due to LAN crowding
        inConf                      NULL,       -- no address in AlternativeAddress
        undefinedReason             NULL,
        ...,
        facilityCallDeflection      NULL,       -- call was deflected using a Facility message
        securityDenied              NULL,       -- incompatible security settings
        calledPartyNotRegistered    NULL,       -- used by gatekeeper when endpoint has
                                                -- preGrantedARQ to bypass ARQ/ACF
        callerNotRegistered             NULL    -- used by gatekeeper when endpoint has
                                                -- preGrantedARQ to bypass ARQ/ACF
}

Setup-UUIE ::= SEQUENCE
{
        protocolIdentifier          ProtocolIdentifier,
        h245Address                 TransportAddress OPTIONAL,
        sourceAddress               SEQUENCE OF AliasAddress OPTIONAL,
        sourceInfo                  EndpointType,
        destinationAddress          SEQUENCE OF AliasAddress OPTIONAL,
        destCallSignalAddress       TransportAddress OPTIONAL,
        destExtraCallInfo           SEQUENCE OF AliasAddress OPTIONAL,      -- Note 1
        destExtraCRV                SEQUENCE OF CallReferenceValue OPTIONAL, -- Note 1
        activeMC                    BOOLEAN,
        conferenceID                ConferenceIdentifier,
        conferenceGoal              CHOICE
        {
                create              NULL,
                join                NULL,
                invite              NULL,
                ...,
                capability-negotiation      NULL,
                callIndependentSupplementaryService     NULL
        },
        callServices            QseriesOptions  OPTIONAL,
        callType                CallType,
```

```
        ...,
        sourceCallSignalAddress  TransportAddress OPTIONAL,
        remoteExtensionAddress   AliasAddress OPTIONAL,
        callIdentifier           CallIdentifier,
        h245SecurityCapability   SEQUENCE OF H245Security OPTIONAL,
        tokens                   SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens             SEQUENCE OF CryptoH323Token OPTIONAL,
        fastStart                SEQUENCE OF OCTET STRING OPTIONAL,
        mediaWaitForConnect      BOOLEAN
        canOverlapSend           BOOLEAN
}

Facility-UUIE ::= SEQUENCE
{
        protocolIdentifier       ProtocolIdentifier,
        alternativeAddress       TransportAddress OPTIONAL,
        alternativeAliasAddress  SEQUENCE OF AliasAddress OPTIONAL,
        conferenceID             ConferenceIdentifier OPTIONAL,
        reason                   FacilityReason,
        ...,
        callIdentifier           CallIdentifier,
        destExtraCallInfo        SEQUENCE OF AliasAddress OPTIONAL,
        remoteExtensionAddress   AliasAddress OPTIONAL,
        tokens                   SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens             SEQUENCE OF CryptoH323Token OPTIONAL,
        conferences              SEQUENCE OF ConferenceList OPTIONAL
        h245Address              TransportAddress OPTIONAL
}

ConferenceList ::= SEQUENCE
{
        conferenceID             ConferenceIdentifier OPTIONAL,
        conferenceAlias          AliasAddress OPTIONAL,
        nonStandardData          NonStandardParameter OPTIONAL,
        ...
}

FacilityReason ::= CHOICE
{
        routeCallToGatekeeper    NULL,        -- call must use gatekeeper model
                                              -- gatekeeper is alternativeAddress
        callForwarded            NULL,
        routeCallToMC            NULL,
        undefinedReason          NULL,
        ...,
        conferenceListChoice     NULL
        starth245                NULL         -- recipient should connect to h245Address
}

Progress-UUIE ::= SEQUENCE
{
        protocolIdentifier       ProtocolIdentifier,
        destinationInfo          EndpointType,
        h245Address              TransportAddress OPTIONAL,
        callIdentifier           CallIdentifier,
        h245SecurityMode         H245Security OPTIONAL,
        ...,
        tokens                   SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens             SEQUENCE OF CryptoH323Token OPTIONAL,
        fastStart                SEQUENCE OF OCTET STRING OPTIONAL,
}
```

```
TransportAddress ::= CHOICE
{
        ipAddress    SEQUENCE
        {
                ip                      OCTET STRING (SIZE(4)),
                port                    INTEGER(0..65535)
        },
        ipSourceRoute                   SEQUENCE
        {
                ip                      OCTET STRING (SIZE(4)),
                port                    INTEGER(0..65535),
                route                   SEQUENCE OF OCTET STRING(SIZE(4)),
                routing                 CHOICE
                {
                        strict  NULL,
                        loose   NULL,
                        ...
                },
                ...
        },
        ipxAddress   SEQUENCE
        {
                node                    OCTET STRING (SIZE(6)),
                netnum                  OCTET STRING (SIZE(4)),
                port                    OCTET STRING (SIZE(2))
        },
        ip6Address   SEQUENCE
        {
                ip                      OCTET STRING (SIZE(16)),
                port                    INTEGER(0..65535),
                ...
        },
        netBios                         OCTET STRING (SIZE(16)),
        nsap                            OCTET STRING (SIZE(1..20)),
        nonStandardAddress              NonStandardParameter,
        ...
}

EndpointType ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        vendor                  VendorIdentifier OPTIONAL,
        gatekeeper              GatekeeperInfo OPTIONAL,
        gateway                 GatewayInfo OPTIONAL,
        mcu                     McuInfo OPTIONAL,       -- mc must be set as well
        terminal                TerminalInfo OPTIONAL,
        mc                      BOOLEAN,                -- shall not be set by itself
        undefinedNode           BOOLEAN,
        ...
}

GatewayInfo ::= SEQUENCE
{
        protocol                SEQUENCE OF SupportedProtocols OPTIONAL,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...
}
```

```
SupportedProtocols ::= CHOICE
{
        nonStandardData         NonStandardParameter,
        h310                    H310Caps,
        h320                    H320Caps,
        h321                    H321Caps,
        h322                    H322Caps,
        h323                    H323Caps,
        h324                    H324Caps,
        voice                   VoiceCaps,
        t120-only               T120OnlyCaps,
        ...,
        nonStandardProtocol     NonStandardProtocol
}

H310Caps ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        dataRatesSupported      SEQUENCE OF DataRate OPTIONAL,
        supportedPrefixes       SEQUENCE OF SupportedPrefix
}

H320Caps ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        dataRatesSupported      SEQUENCE OF DataRate OPTIONAL,
        supportedPrefixes       SEQUENCE OF SupportedPrefix
}

H321Caps ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        dataRatesSupported      SEQUENCE OF DataRate OPTIONAL,
        supportedPrefixes       SEQUENCE OF SupportedPrefix
}

H322Caps ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        dataRatesSupported      SEQUENCE OF DataRate OPTIONAL,
        supportedPrefixes       SEQUENCE OF SupportedPrefix
}

H323Caps ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        dataRatesSupported      SEQUENCE OF DataRate OPTIONAL,
        supportedPrefixes       SEQUENCE OF SupportedPrefix
}
```

```
H324Caps ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        dataRatesSupported      SEQUENCE OF DataRate OPTIONAL,
        supportedPrefixes       SEQUENCE OF SupportedPrefix
}

VoiceCaps ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        dataRatesSupported      SEQUENCE OF DataRate OPTIONAL,
        supportedPrefixes       SEQUENCE OF SupportedPrefix
}

T120OnlyCaps ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        dataRatesSupported      SEQUENCE OF DataRate OPTIONAL,
        supportedPrefixes       SEQUENCE OF SupportedPrefix
}

NonStandardProtocol ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        dataRatesSupported      SEQUENCE OF DataRate OPTIONAL,
        supportedPrefixes       SEQUENCE OF SupportedPrefix,
        ...
}

McuInfo ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...
}

TerminalInfo ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...
}

GatekeeperInfo ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        ...
}

VendorIdentifier ::= SEQUENCE
{
        vendor                  H221NonStandard,
        productId               OCTET STRING (SIZE(1..256)) OPTIONAL,    -- per vendor
        versionId               OCTET STRING (SIZE(1..256)) OPTIONAL,    -- per product
        ...
}
```

```
H221NonStandard ::= SEQUENCE
{       t35CountryCode          INTEGER(0..255),        -- country, as per T.35
        t35Extension            INTEGER(0..255),        -- assigned nationally
        manufacturerCode        INTEGER(0..65535),      -- assigned nationally
        ...
}

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

NonStandardIdentifier           ::= CHOICE
{
        object                  OBJECT IDENTIFIER,
        h221NonStandard         H221NonStandard,
        ...
}

AliasAddress ::= CHOICE
{
        e164                    IA5String (SIZE (1..128)) (FROM ("0123456789No. *,")),
        h323-ID                 BMPString (SIZE (1..256)),      -- Basic ISO/IEC 10646-1 (Unicode)
        ...,
        url-ID                  IA5String (SIZE(1..512)),       -- URL style address
        transportID             TransportAddress,
        email-ID                IA5String (SIZE(1..512)),       -- rfc822-compliant email address
        partyNumber             PartyNumber
}

PartyNumber ::= CHOICE
{
        publicNumber PublicPartyNumber,
                                                        -- the numbering plan is according to
                                                        -- Recommendations E.163 and E.164.
        dataPartyNumber NumberDigits,
                                                        -- not used, value reserved.
        telexPartyNumber NumberDigits,
                                                        -- not used, value reserved.
        privateNumber PrivatePartyNumber,
        nationalStandardPartyNumber NumberDigits,
                                                        -- not used, value reserved.
        ...
}

PublicPartyNumber ::= SEQUENCE
{
        publicTypeOfNumber      PublicTypeOfNumber,
        publicNumberDigits      NumberDigits
}

PrivatePartyNumber ::= SEQUENCE
{
        privateTypeOfNumber     PrivateTypeOfNumber,
        privateNumberDigits     NumberDigits
}

NumberDigits ::= IA5String (SIZE (1..128)) (FROM ("0123456789No. *,"))
```

```
PublicTypeOfNumber ::= CHOICE
{
        unknown                 NULL,
                                                -- if used number digits carry prefix indicating type
                                                -- of number according to national recommendations.
        internationalNumber     NULL,
        nationalNumber          NULL,
        networkSpecificNumber   NULL,
                                                -- not used, value reserved
        subscriberNumber        NULL,
        abbreviatedNumber       NULL,
                                                -- valid only for called party number at the outgoing
                                                -- access, network substitutes appropriate number.
          ...
}

PrivateTypeOfNumber ::= CHOICE
{
        unknown                 NULL,
        level2RegionalNumber    NULL,
        level1RegionalNumber    NULL,
        pISNSpecificNumber      NULL,
        localNumber             NULL,
        abbreviatedNumber       NULL,
        ...
}

Endpoint ::= SEQUENCE
{
        nonStandardData         NonStandardParameter OPTIONAL,
        aliasAddress            SEQUENCE OF AliasAddress OPTIONAL,
        callSignalAddress       SEQUENCE OF TransportAddress OPTIONAL,
        rasAddress              SEQUENCE OF TransportAddress OPTIONAL,
        endpointType            EndpointType OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        priority                INTEGER(0..127) OPTIONAL,
        remoteExtensionAddress  SEQUENCE OF AliasAddress OPTIONAL,
        destExtraCallInfo       SEQUENCE OF AliasAddress OPTIONAL,
        ...
}

AlternateGK ::= SEQUENCE
{
        rasAddress              TransportAddress,
        gatekeeperIdentifier    GatekeeperIdentifier  OPTIONAL,
        needToRegister          BOOLEAN,
        priority                INTEGER (0..127),
        ...
}

AltGKInfor ::=SEQUENCE
{
        alternategatekeeper     SEQUENCE OF AlternateGK,
        altGKisPermanent        BOOLEAN,
        ...
}
```

```
SecurityServiceMode ::= CHOICE
{
        nonStandard             NonStandardParameter,
        none                    NULL,
        default                 NULL,
        ...                     -- can be extended with other specific modes
}

SecurityCapabilities ::= SEQUENCE
{
      nonStandard               NonStandardParameter OPTIONAL,
      encryption                SecurityServiceMode,
      authenticaton             SecurityServiceMode,
      integrity                 SecurityServiceMode,
        ...
}

H245Security ::= CHOICE
{
      nonStandard               NonStandardParameter,
      noSecurity                NULL,
      tls                       SecurityCapabilities,
      ipsec                     SecurityCapabilities,
      ...
}


QseriesOptions ::= SEQUENCE
{
      q932Full          BOOLEAN, -- if true, indicates full support for Q.932
      q951Full          BOOLEAN, -- if true, indicates full support for Q.951
      q952Full          BOOLEAN, -- if true, indicates full support for Q.952
      q953Full          BOOLEAN, -- if true, indicates full support for Q.953
      q955Full          BOOLEAN, -- if true, indicates full support for Q.955
      q956Full          BOOLEAN, -- if true, indicates full support for Q.956
      q957Full          BOOLEAN, -- if true, indicates full support for Q.957
      q954Info          Q954Details,
      ...
}

Q954Details ::= SEQUENCE
{
      conferenceCalling         BOOLEAN,
      threePartyService         BOOLEAN,
      ...
}

GloballyUniqueID        ::=  OCTET STRING (SIZE(16))
ConferenceIdentifier    ::=  GloballyUniqueID
RequestSeqNum           ::=  INTEGER (1..65535)
GatekeeperIdentifier    ::=  BMPString (SIZE(1..128))
BandWidth               ::=  INTEGER (0.. 4294967295)     -- in 100s of bits
CallReferenceValue      ::=  INTEGER (0..65535)
EndpointIdentifier      ::=  BMPString (SIZE(1..128))
ProtocolIdentifier      ::=  OBJECT IDENTIFIER
                                                          -- shall be set to
                                                          -- {itu-t (0) recommendation (0) h (8) 2250
                                                          -- version (0) 2}
TimeToLive              ::=  INTEGER (1..4294967295)      --in seconds
```

```
CallIdentifier ::= SEQUENCE
{
        guid                    GloballyUniqueID,
        ...
}


EncryptIntAlg ::= CHOICE
{       -- core encryption algorithms for RAS message integrity
        nonStandard             NonStandardParameter,
        isoAlgorithm            OBJECT IDENTIFIER,  -- defined in ISO/IEC 9979
        ...
}
NonIsoIntegrityMechanism ::= CHOICE
{       -- HMAC mechanism used, no truncation, tagging may be necessary!
        hMAC-MD5                NULL,
        hMAC-iso10118-2-s       EncryptIntAlg,      -- according to ISO/IEC 10118-2 using
                                                    -- EncryptIntAlg as core block encryption algorithm
                                                    -- (short MAC)
        hMAC-iso10118-2-l       EncryptIntAlg,      -- according to ISO/IEC 10118-2 using
                                                    -- EncryptIntAlg as core block encryption algorithm
                                                    -- (long MAC)
        hMAC-iso10118-3         OBJECT IDENTIFIER,  -- according to ISO/IEC 10118-3 using
                                                    -- OID as hash function (OID is SHA-1,
                                                    -- RIPE-MD160,
                                                    -- RIPE-MD128)
        ...
}

IntegrityMechanism ::= CHOICE
{       -- for RAS message integrity
        nonStandard             NonStandardParameter,
        digSig                  NULL,               -- indicates to apply a digital signature
        iso9797                 OBJECT IDENTIFIER,  -- according to ISO/IEC 9797 using OID as
                                                    -- core encryption algorithm (X-CBC MAC)
        nonIsoIM                NonIsoIntegrityMechanism,
        ...
}

ICV ::= SEQUENCE
{
        algorithmOID    OBJECT IDENTIFIER,  -- the algorithm used to compute the signature
        icv             BIT STRING          -- the computed cryptographic integrity check value
                                            -- or signature
}

FastStartToken ::= ClearToken (WITH COMPONENTS {..., timeStampPRESENT, dhkeyPRESENT,
generalIDPRESENT                            -- set to 'alias' -- })
EncodedFastStartToken ::= TYPE-IDENTIFIER.&Type (FastStartToken)
CryptoH323Token::= CHOICE
{
        cryptoEPPwdHash SEQUENCE
        {
                alias           AliasAddress,       -- alias of entity generating hash
                timeStamp       TimeStamp,          -- timestamp used in hash
                token           HASHED    { EncodedPwdCertToken -- generalID set to 'alias' -- }
        },
```

```
        cryptoGKPwdHash  SEQUENCE
          {
              gatekeeperId        GatekeeperIdentifier,          -- GatekeeperID of GK generating hash
              timeStamp           TimeStamp,                     -- timestamp used in hash
              token               HASHED    { EncodedPwdCertToken  -- generalID set to Gatekeeperid -- }
        },
        cryptoEPPwdEncr           ENCRYPTED
                          { EncodedPwdCertToken                  -- generalID set to Gatekeeperid --},
        cryptoGKPwdEncr           ENCRYPTED
                          { EncodedPwdCertToken                  -- generalID set to Gatekeeperid --},
        cryptoEPCert              SIGNED { EncodedPwdCertToken -- generalID set to Gatekeeperid -- },
        cryptoGKCert              SIGNED { EncodedPwdCertToken -- generalID set to alias -- },
        cryptoFastStart           SIGNED { EncodedFastStartToken },
        nestedcryptoToken         CryptoToken,
        ...
}


DataRate ::= SEQUENCE
{
        nonStandardData           NonStandardParameter OPTIONAL,
        channelRate               BandWidth,
        channelMultiplier         INTEGER (1..256) OPTIONAL,
        ...
}

SupportedPrefix ::= SEQUENCE
{
        nonStandardData           NonStandardParameter OPTIONAL,
        prefix                    AliasAddress,
        ...
}


RasMessage ::= CHOICE
{
        gatekeeperRequest             GatekeeperRequest,
        gatekeeperConfirm             GatekeeperConfirm,
        gatekeeperReject              GatekeeperReject,
        registrationRequest           RegistrationRequest,
        registrationConfirm           RegistrationConfirm,
        registrationReject            RegistrationReject,
        unregistrationRequest         UnregistrationRequest,
        unregistrationConfirm         UnregistrationConfirm,
        unregistrationReject          UnregistrationReject,
        admissionRequest              AdmissionRequest,
        admissionConfirm              AdmissionConfirm,
        admissionReject               AdmissionReject,
        bandwidthRequest              BandwidthRequest,
        bandwidthConfirm              BandwidthConfirm,
        bandwidthReject               BandwidthReject,
        disengageRequest              DisengageRequest,
        disengageConfirm              DisengageConfirm,
        disengageReject               DisengageReject,
        locationRequest               LocationRequest,
        locationConfirm               LocationConfirm,
        locationReject                LocationReject,
        infoRequest                   InfoRequest,
        infoRequestResponse           InfoRequestResponse,
        nonStandardMessage            NonStandardMessage,
        unknownMessageResponse        UnknownMessageResponse,
```

```
        ...,
        requestInProgress           RequestInProgress,
        resourcesAvailableIndicate  ResourcesAvailableIndicate,
        resourcesAvailableConfirm   ResourcesAvailableConfirm,
        infoRequestAck              InfoRequestAck,
        infoRequestNak              InfoRequestNak
}

GatekeeperRequest ::= SEQUENCE --(GRQ)
{
        requestSeqNum               RequestSeqNum,
        protocolIdentifier          ProtocolIdentifier,
        nonStandardData             NonStandardParameter OPTIONAL,
        rasAddress                  TransportAddress,
        endpointType                EndpointType,
        gatekeeperIdentifier        GatekeeperIdentifier OPTIONAL,
        callServices                QseriesOptions  OPTIONAL,
        endpointAlias               SEQUENCE OF AliasAddress OPTIONAL,
        ...,
        alternateEndpoints          SEQUENCE OF Endpoint OPTIONAL,
        tokens                      SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens                SEQUENCE OF CryptoH323Token OPTIONAL,
        authenticationCapability    SEQUENCE OF AuthenticationMechanism OPTIONAL,
        algorithmOIDs               SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
        integrity                   SEQUENCE OF IntegrityMechanism OPTIONAL,
        integrityCheckValue         ICV OPTIONAL
}

GatekeeperConfirm ::= SEQUENCE --(GCF)
{
        requestSeqNum               RequestSeqNum,
        protocolIdentifier          ProtocolIdentifier,
        nonStandardData             NonStandardParameter OPTIONAL,
        gatekeeperIdentifier        GatekeeperIdentifier  OPTIONAL,
        rasAddress                  TransportAddress,
        ...,
        alternateGatekeeper         SEQUENCE OF AlternateGK OPTIONAL,
        authenticationMode          AuthenticationMechanism OPTIONAL,
        tokens                      SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens                SEQUENCE OF CryptoH323Token OPTIONAL,
        algorithmOID                OBJECT IDENTIFIER OPTIONAL,
        integrity                   SEQUENCE OF IntegrityMechanism OPTIONAL,
        integrityCheckValue         ICV OPTIONAL
}

GatekeeperReject ::= SEQUENCE --(GRJ)
{
        requestSeqNum               RequestSeqNum,
        protocolIdentifier          ProtocolIdentifier,
        nonStandardData             NonStandardParameter OPTIONAL,
        gatekeeperIdentifier        GatekeeperIdentifier  OPTIONAL,
        rejectReason                GatekeeperRejectReason,
        ...,
        altGKInfo                   AltGKInfo OPTIONAL,
        tokens                      SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens                SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue         ICV OPTIONAL
}
```

```
GatekeeperRejectReason ::= CHOICE
{
        resourceUnavailable             NULL,
        terminalExcluded                NULL,       -- permission failure, not a resource failure
        invalidRevision                 NULL,
        undefinedReason                 NULL,
        ...,
        securityDenial                  NULL
}

RegistrationRequest ::= SEQUENCE --(RRQ)
{
        requestSeqNum               RequestSeqNum,
        protocolIdentifier          ProtocolIdentifier,
        nonStandardData             NonStandardParameter OPTIONAL,
        discoveryComplete           BOOLEAN,
        callSignalAddress           SEQUENCE OF TransportAddress,
        rasAddress                  SEQUENCE OF TransportAddress,
        terminalType                EndpointType,
        terminalAlias               SEQUENCE OF AliasAddress OPTIONAL,
        gatekeeperIdentifier        GatekeeperIdentifier  OPTIONAL,
        endpointVendor              VendorIdentifier,
        ...,
        alternateEndpoints          SEQUENCE OF Endpoint OPTIONAL,
        timeToLive                  TimeToLive OPTIONAL,
        tokens                      SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens                SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue         ICV OPTIONAL,
        keepAlive                   BOOLEAN,
        endpointIdentifier          EndpointIdentifier OPTIONAL,
        willSupplyUUIEs             BOOLEAN
}

RegistrationConfirm ::= SEQUENCE --(RCF)
{
        requestSeqNum               RequestSeqNum,
        protocolIdentifier          ProtocolIdentifier,
        nonStandardData             NonStandardParameter OPTIONAL,
        callSignalAddress           SEQUENCE OF TransportAddress,
        terminalAlias               SEQUENCE OF AliasAddress OPTIONAL,
        gatekeeperIdentifier        GatekeeperIdentifier  OPTIONAL,
        endpointIdentifier          EndpointIdentifier,
        ...,
        alternateGatekeeper         SEQUENCE OF AlternateGK OPTIONAL,
        timeToLive                  TimeToLive OPTIONAL,
        tokens                      SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens                SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue         ICV OPTIONAL,
        willRespondToIRR            BOOLEAN,
        preGrantedARQ               SEQUENCE
        {
                makeCall                            BOOLEAN,
                useGKCallSignalAddressToMakeCall    BOOLEAN,
                answerCall                          BOOLEAN,
                useGKCallSignalAddressToAnswer      BOOLEAN,
                ...
        } OPTIONAL
}
```

```
RegistrationReject ::= SEQUENCE --(RRJ)
{
        requestSeqNum           RequestSeqNum,
        protocolIdentifier      ProtocolIdentifier,
        nonStandardData         NonStandardParameter OPTIONAL,
        rejectReason            RegistrationRejectReason,
        gatekeeperIdentifier    GatekeeperIdentifier  OPTIONAL,
        ...,
        altGKInfo               AltGKInfo OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

RegistrationRejectReason ::= CHOICE
{
        discoveryRequired               NULL,   -- registration permission has aged
        invalidRevision                 NULL,
        invalidCallSignalAddress        NULL,
        invalidRASAddress               NULL,   -- supplied address is invalid
        duplicateAlias                  SEQUENCE OF AliasAddress,
                                                -- alias registered to another endpoint
        invalidTerminalType             NULL,
        undefinedReason                 NULL,
        transportNotSupported           NULL,   -- one or more of the transports
        ...,
        transportQOSNotSupported        NULL,   -- endpoint QOS not supported
        resourceUnavailable             NULL,   -- gatekeeper resources exhausted
        invalidAlias                    NULL,   -- alias not consistent with gatekeeper rules
        securityDenial                  NULL
}

UnregistrationRequest ::= SEQUENCE --(URQ)
{
        requestSeqNum           RequestSeqNum,
        callSignalAddress       SEQUENCE OF TransportAddress,
        endpointAlias           SEQUENCE OF AliasAddress OPTIONAL,
        nonStandardData         NonStandardParameter OPTIONAL,
        endpointIdentifier      EndpointIdentifier OPTIONAL,
        ...,
        alternateEndpoints      SEQUENCE OF Endpoint OPTIONAL,
        gatekeeperIdentifier    GatekeeperIdentifier OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        reason                  UnregRequestReason OPTIONAL
}

UnregRequestReason ::= CHOICE
{
        reregistrationRequired  NULL,
        ttlExpired              NULL,
        securityDenial          NULL,
        undefinedReason         NULL,
        ...
}
```

```
UnregistrationConfirm ::= SEQUENCE --(UCF)
{
        requestSeqNum           RequestSeqNum,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

UnregistrationReject ::= SEQUENCE --(URJ)
{

        requestSeqNum           RequestSeqNum,
        rejectReason            UnregRejectReason,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        altGKInfo               AltGKInfo OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

UnregRejectReason ::= CHOICE
{
        notCurrentlyRegistered  NULL,
        callInProgress          NULL,
        undefinedReason         NULL,
        ...,
        permissionDenied        NULL,   -- requesting user not allowed to unregister
                                        -- specified user
        securityDenial          NULL
}

AdmissionRequest ::= SEQUENCE --(ARQ)
{
        requestSeqNum           RequestSeqNum,
        callType                CallType,
        callModel               CallModel OPTIONAL,
        endpointIdentifier      EndpointIdentifier,
        destinationInfo         SEQUENCE OF AliasAddress OPTIONAL,      -- Note 1
        destCallSignalAddress   TransportAddress OPTIONAL,              -- Note 1
        destExtraCallInfo       SEQUENCE OF AliasAddress OPTIONAL,
        srcInfo                 SEQUENCE OF AliasAddress,
        srcCallSignalAddress    TransportAddress OPTIONAL,
        bandWidth               BandWidth,
        callReferenceValue      CallReferenceValue,
        nonStandardData         NonStandardParameter OPTIONAL,
        callServices            QseriesOptions  OPTIONAL,
        conferenceID            ConferenceIdentifier,
        activeMC                BOOLEAN,
        answerCall              BOOLEAN,  -- answering a call
        ...,
        canMapAlias             BOOLEAN,  -- can handle alias address
        callIdentifier          CallIdentifier,
        srcAlternatives         SEQUENCE OF Endpoint OPTIONAL,
        destAlternatives        SEQUENCE OF Endpoint OPTIONAL,
        gatekeeperIdentifier    GatekeeperIdentifier OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
```

```
        transportQOS            TransportQOS OPTIONAL,
        willSupplyUUIEs         BOOLEAN
}

CallType ::= CHOICE
{
        pointToPoint            NULL,           -- Point-to-point
        oneToN                  NULL,           -- no interaction (FFS)
        nToOne                  NULL,           -- no interaction (FFS)
        nToN                    NULL,           -- interactive (multipoint)
        ...
}

CallModel ::= CHOICE
{
        direct                  NULL,
        gatekeeperRouted        NULL,
        ...
}

TransportQOS ::= CHOICE
{
        endpointControlled      NULL,
        gatekeeperControlled    NULL,
        noControl               NULL,
        ...
}

AdmissionConfirm ::= SEQUENCE --(ACF)
{
        requestSeqNum           RequestSeqNum,
        bandWidth               BandWidth,
        callModel               CallModel,
        destCallSignalAddress   TransportAddress,
        irrFrequency            INTEGER (1..65535) OPTIONAL,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        destinationInfo         SEQUENCE OF AliasAddress OPTIONAL,
        destExtraCallInfo       SEQUENCE OF AliasAddress OPTIONAL,
        destinationType         EndpointType OPTIONAL,
        remoteExtensionAddress  SEQUENCE OF AliasAddress OPTIONAL,
        alternateEndpoints      SEQUENCE OF Endpoint OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        transportQOS            TransportQOS OPTIONAL,
        willRespondToIRR        BOOLEAN,
        uuiesRequested          UUIEsRequested
}

UUIEsRequested ::= SEQUENCE
{
        setup                   BOOLEAN,
        callProceeding          BOOLEAN,
        connect                 BOOLEAN,
        alerting                BOOLEAN,
        userInformation         BOOLEAN,
        releaseComplete         BOOLEAN,
        facility                BOOLEAN,
        progress                BOOLEAN,
```

```
        empty                   BOOLEAN,
        ...
}

AdmissionReject ::= SEQUENCE --(ARJ)
{
        requestSeqNum           RequestSeqNum,
        rejectReason            AdmissionRejectReason,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        altGKInfo               AltGKInfo OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        CallSignalAddress       SEQUENCE OF TransportAddress OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

AdmissionRejectReason ::= CHOICE
{
        calledPartyNotRegistered    NULL,   -- cannot translate address
        invalidPermission           NULL,   -- permission has expired
        requestDenied               NULL,   -- no bandwidth available
        undefinedReason             NULL,
        callerNotRegistered         NULL,
        routeCallToGatekeeper       NULL,
        invalidEndpointIdentifier   NULL,
        resourceUnavailable         NULL,
        ...,
        securityDenial              NULL,
        qosControlNotSupported      NULL,
        incompleteAddress           NULL
}

BandwidthRequest ::= SEQUENCE --(BRQ)
{
        requestSeqNum           RequestSeqNum,
        endpointIdentifier      EndpointIdentifier,
        conferenceID            ConferenceIdentifier,
        callReferenceValue      CallReferenceValue,
        callType                CallType OPTIONAL,
        bandWidth               BandWidth,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        callIdentifier          CallIdentifier,
        gatekeeperIdentifier    GatekeeperIdentifier OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        answeredCall            BOOLEAN
}

BandwidthConfirm ::= SEQUENCE --(BCF)
{
        requestSeqNum           RequestSeqNum,
        bandWidth               BandWidth,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}
```

```
BandwidthReject ::= SEQUENCE --(BRJ)
{
        requestSeqNum           RequestSeqNum,
        rejectReason            BandRejectReason,
        allowedBandWidth        BandWidth,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        altGKInfo               AltGKInfo OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

BandRejectReason ::= CHOICE
{
        notBound                NULL,   -- discovery permission has aged
        invalidConferenceID     NULL,   -- possible revision
        invalidPermission       NULL,   -- true permission violation
        insufficientResources   NULL,
        invalidRevision         NULL,
        undefinedReason         NULL,
        ...,
        securityDenial          NULL
}

LocationRequest ::= SEQUENCE --(LRQ)
{
        requestSeqNum           RequestSeqNum,
        endpointIdentifier      EndpointIdentifier OPTIONAL,
        destinationInfo         SEQUENCE OF AliasAddress,
        nonStandardData         NonStandardParameter OPTIONAL,
        replyAddress            TransportAddress,
        ...,
        sourceInfo              SEQUENCE OF AliasAddress OPTIONAL,
        canMapAlias             BOOLEAN, -- can handle alias address
        gatekeeperIdentifier    GatekeeperIdentifier OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

LocationConfirm ::= SEQUENCE --(LCF)
{
        requestSeqNum           RequestSeqNum,
        callSignalAddress       TransportAddress,
        rasAddress              TransportAddress,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        destinationInfo         SEQUENCE OF AliasAddress OPTIONAL,
        destExtraCallInfo       SEQUENCE OF AliasAddress OPTIONAL,
        destinationType         EndpointType OPTIONAL,
        remoteExtensionAddress  SEQUENCE OF AliasAddress OPTIONAL,
        alternateEndpoints      SEQUENCE OF Endpoint OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}
```

```
LocationReject ::= SEQUENCE --(LRJ)
{
        requestSeqNum           RequestSeqNum,
        rejectReason            LocationRejectReason,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        altGKInfo               AltGKInfo OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

LocationRejectReason ::= CHOICE
{
        notRegistered           NULL,
        invalidPermission       NULL,       -- exclusion by administrator or feature
        requestDenied           NULL,       -- cannot find location
        undefinedReason         NULL,
        ...,
        securityDenial          NULL
}

DisengageRequest ::= SEQUENCE --(DRQ)
{
        requestSeqNum           RequestSeqNum,
        endpointIdentifier      EndpointIdentifier,
        conferenceID            ConferenceIdentifier,
        callReferenceValue      CallReferenceValue,
        disengageReason         DisengageReason,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        callIdentifier          CallIdentifier,
        gatekeeperIdentifier    GatekeeperIdentifier OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        answeredCall            BOOLEAN
}

DisengageReason ::= CHOICE
{
        forcedDrop              NULL,       -- gatekeeper is forcing the drop
        normalDrop              NULL,       -- associated with normal drop
        undefinedReason         NULL,
        ...
}

DisengageConfirm ::= SEQUENCE --(DCF)
{
        requestSeqNum           RequestSeqNum,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}
```

```
DisengageReject ::= SEQUENCE --(DRJ)
{
        requestSeqNum           RequestSeqNum,
        rejectReason            DisengageRejectReason,
        nonStandardData         NonStandardParameter OPTIONAL,
        ...,
        altGKInfo               AltGKInfo OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

DisengageRejectReason ::= CHOICE
{
        notRegistered           NULL,       -- not registered with gatekeeper
        requestToDropOther      NULL,       -- cannot request drop for others
        ...,
        securityDenial          NULL
}

InfoRequest ::= SEQUENCE --(IRQ)
{
        requestSeqNum           RequestSeqNum,
        callReferenceValue      CallReferenceValue,
        nonStandardData         NonStandardParameter OPTIONAL,
        replyAddress            TransportAddress OPTIONAL,
        ...,
        callIdentifier          CallIdentifier,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        uuiesRequested          UUIEsRequested OPTIONAL
}
InfoRequestResponse ::= SEQUENCE --(IRR)
{
        nonStandardData             NonStandardParameter OPTIONAL,
        requestSeqNum               RequestSeqNum,
        endpointType                EndpointType,
        endpointIdentifier          EndpointIdentifier,
        rasAddress                  TransportAddress,
        callSignalAddress           SEQUENCE OF TransportAddress,
        endpointAlias               SEQUENCE OF AliasAddress OPTIONAL,
        perCallInfo                 SEQUENCE OF SEQUENCE
        {
                nonStandardData     NonStandardParameter OPTIONAL,
                callReferenceValue  CallReferenceValue,
                conferenceID        ConferenceIdentifier,
                originator          BOOLEAN OPTIONAL,
                audio               SEQUENCE OF RTPSession OPTIONAL,
                video               SEQUENCE OF RTPSession OPTIONAL,
                data                SEQUENCE OF TransportChannelInfo OPTIONAL,
                h245                TransportChannelInfo,
                callSignaling       TransportChannelInfo,
                callType            CallType,
                bandWidth           BandWidth,
                callModel           CallModel,
                ...,
                callIdentifier      CallIdentifier,
                tokens              SEQUENCE OF ClearToken OPTIONAL,
                cryptoTokens        SEQUENCE OF CryptoH323Token OPTIONAL,
```

```
            substituteConfIDs       SEQUENCE OF ConferenceIdentifier,
            pdu                     SEQUENCE OF SEQUENCE
            {
                h323pdu             H323-UU-PDU,
                sent                BOOLEAN         -- TRUE is sent, FALSE is received
            } OPTIONAL
        } OPTIONAL,
        ...,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        needResponse            BOOLEAN
}

TransportChannelInfo ::= SEQUENCE
{
        sendAddress             TransportAddress OPTIONAL,
        recvAddress             TransportAddress OPTIONAL,
        ...
}

RTPSession ::= SEQUENCE
{
        rtpAddress              TransportChannelInfo,
        rtcpAddress             TransportChannelInfo,
        cname                   PrintableString,
        ssrc                    INTEGER (1..4294967295),
        sessionId               INTEGER (1..255),
        associatedSessionIds    SEQUENCE OF INTEGER (1..255),
        ...
}

InfoRequestAck ::= SEQUENCE --(IACK)
{
        requestSeqNum           RequestSeqNum,
        nonStandardData         NonStandardParameter OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        ...
}

InfoRequestNak ::= SEQUENCE --(INAK)
{
        requestSeqNum           RequestSeqNum,
        nonStandardData         NonStandardParameter OPTIONAL,
        nakReason               InfoRequestNakReason,
        altGKInfo               AltGKInfo OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        ...
}

InfoRequestNakReason ::= CHOICE
{
        notRegistered           NULL,       -- not registered with gatekeeper
        securityDenial          NULL,
        undefinedReason         NULL,
        ...
}
```

```
NonStandardMessage ::= SEQUENCE
{
        requestSeqNum           RequestSeqNum,
        nonStandardData         NonStandardParameter,
        ...,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

UnknownMessageResponse ::= SEQUENCE   -- (XRS)
{
        requestSeqNum           RequestSeqNum,
        ...,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL
}

RequestInProgress ::= SEQUENCE          -- (RIP)
{
        requestSeqNum           RequestSeqNum,
        nonStandardData         NonStandardParameter   OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        delay                   INTEGER(1..65535),
        ...
}

ResourcesAvailableIndicate ::= SEQUENCE --(RAI)
{
        requestSeqNum           RequestSeqNum,
        protocolIdentifier      ProtocolIdentifier,
        nonStandardData         NonStandardParameter OPTIONAL,
        endpointIdentifier      EndpointIdentifier,
        protocols               SEQUENCE OF SupportedProtocols,
        almostOutOfResources    BOOLEAN,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        ...
}

ResourcesAvailableConfirm ::= SEQUENCE --(RAC)
{
        requestSeqNum           RequestSeqNum,
        protocolIdentifier      ProtocolIdentifier,
        nonStandardData         NonStandardParameter OPTIONAL,
        tokens                  SEQUENCE OF ClearToken OPTIONAL,
        cryptoTokens            SEQUENCE OF CryptoH323Token OPTIONAL,
        integrityCheckValue     ICV OPTIONAL,
        ...
}

END             -- of ASN.1
```

APPENDIX I

RTP/RTCP algorithms

The referenced informative material may be found in the following proposed Internet Standard:

SCHULZRINNE (H.), CASNER (S.), FREDERICK (R.) and JACOBSON (V.), RTP: A Transport Protocol for Real-Time Applications, RFC 1889, *Internet Engineering Task Force*, 1996.

APPENDIX II

RTP profile

The referenced informative material may be found in the following proposed Internet Standard:

SCHULZRINNE (H.), RTP Profile for Audio and Video Conferences with Minimal Control, RFC 1890, *Internet Engineering Task Force*, 1996.

APPENDIX III

H.261 packetization

The referenced informative material may be found in the following proposed Internet Standard:

TURLETTI (T.), HUITEMA (C.), RTP Payload Format for H.261 Video Stream, RFC 2032, *Internet Engineering Task Force*, 1996.

APPENDIX IV

H.225.0 operation on different packet-based network protocol stacks

This appendix provides additional details concerning the operation of H.225.0 on various actual packet-based network protocol stacks. This appendix: non-normative. Packet-based networks used in this Recommendation shall provide both reliable and unreliable modes of operation, including a means to distinguish packet boundaries.

IV.1 TCP/IP/UDP

Note that UDP can fragment and re-assemble large video packets, but that failure to perform MB packetization may lead to the loss of an entire GOB.

IP multicast should be used for GRQ distribution as opposed to media access layer broadcast.

<table>
<tr><th>Unreliable delivery applications</th><th>Call signalling and H.245 channel</th></tr>
<tr><td rowspan="2">UDP</td><td>TPKT
— —
TCP</td></tr>
<tr><td colspan="2">IP</td></tr>
<tr><td colspan="2">Link Layer</td></tr>
<tr><td colspan="2">Physical Layer</td></tr>
</table>

IV.1.1 Discovering the gatekeeper

IV.1.1.1 Discovery using multicast address or well-known port

Following the gatekeeper discovery and registration procedures described in clause 7/H.323, endpoints should use the following multicast address or well known port when attempting to discover the gatekeeper as appropriate for their network configuration:

- Gatekeeper UDP Discovery Multicast Address 224.0.1.41
- Gatekeeper UDP Discovery Port 1718
- Gatekeeper UDP Registration and Status Port 1719

Note that implementations should pay attention to the scope of the multicast so as to not flood the Internet with discovery messages.

IV.1.1.2 Discovery using DNS (informative)

IV.1.1.2.1 A URL for gatekeepers

As a first step, note that a gatekeeper is identified by a transport address and a gatekeeperIdentifier, which is a string. A gatekeeper is a particular resource on the Internet, so it is reasonable to specify it in a Uniform Resource Locator (URL). The protocol spoken by the gatekeeper is RAS, so the URL for a gatekeeper could be given by:

ras://gkID@domainname

gkID is the gatekeeperIdentifier, and domainname is a DNS domain name which identifies the gatekeeper's domain. Note that this is not necessarily a Fully Qualified Domain Name (FQDN) with an A-record – it is not required that this domain name has a physical transport interface with an IP number recorded in the DNS. If it is a FQDN, however, it is reasonable to insist that its IP number is that of the gatekeeper to which the URL refers. In this case, it is allowed to add an optional port number to the URL:

ras://gkID@domainname:port_no.

If no port number is given, then the well known value of 1719 is taken as a default.

The more interesting case is when this is not an FQDN, and then the domain name does not refer to a transport address listed in the DNS. The domain name then can refer to a pure "gatekeeper zone of authority". The next subclause explains how to find the gatekeeper in this case.

IV.1.1.2.2 Finding the URL

The URL does not solve the problem of locating the gatekeeper, it just gives a standard format for the information to find. The problem is how to produce a transport address and gatekeeperIdentifier for RAS signalling given the domain name of a gatekeeper.

If the gatekeeper has an IETF RFC 822-compliant identifier, it is easy to extract a domain name from the RFC 822-compliant identifier of a gatekeeper. In fact, it may be convenient to give RFC 822-compliant identifiers to endpoints, and then to stipulate that the domain name part of the identifier refers to the gatekeeper domain.

IV.1.1.2.2.1 The SRV resource record query

The first solution uses the fact that the gatekeeper is basically a system service, and the transport address of a named system service can be extracted from DNS by using a query for a new type of DNS Resource Record, called SRV (for "service location record"). Given a domain name, an SRV record query will be made for the transport address of the ras service for that domain. The domain name itself, or one returned in the SRV response, is used as the gatekeeperIdentifier.

This simple solution will soon be standard. The problem is that almost no current DNS client or server implementations support the SRV resource record yet. Unless the DNS client knows about the SRV resource record type, it is not possible for it to pass on queries for this resource record. Until this support becomes widespread, there is a reasonable chance that the SRV query will fail.

IV.1.1.2.2.2 The TXT record query

All current DNS implementations support the TXT resource record. Basically this is some free text that can be returned for each domain name. It is possible to store many TXT resources for a single domain. The standard stipulates that all TXT records will be returned when a query is made for them.

It is possible to use TXT queries if the SRV queries fail. Assume the same convention for extracting a domain name that was suggested above. Either RFC 822 compliant strings (email "-like" names) or IETF RFC 1768 compliant strings (URLs) can be used for gatekeeperIdentifiers. In either case the domain name is used to make a DNS TXT query for the domain name. The returned resource records are lines of free text, and the terminal will then look for lines in the response of the form:

ras [< gk id>@]<domain name >[:<portno>] [<priority>]

The **<gk id>** field is an optional gatekeeper ID which is separate from the domain name. If this field is missing, then the domain itself is assumed to be the gatekeeper ID.

The **<domain name>** field can be either the name of the A-record which contains the gatekeeper's IP address, or a raw IP address in dotted form. The domain name need not be fully qualified; if it is not, the subdomain in which the TXT record was found should be appended to it to form the fully qualified A-record name.

The optional **[:<portno>]** can be used to specify a port number other than the standard RAS port.

The optional **[<priority>]** field specifies the order in which the listed gatekeepers should be accessed for discovery or LRQ queries if there is more than one ras TXT record. Lower numbers have higher priority.

Note that this format, if the <gk id> field is missing, assumes that the gatekeeper IDs are in fact legal domain names. However, if it is necessary for a single host to support multiple logical gatekeepers, each with a separate ID, the format will support this. This is because separate A-records can contain the same IP address.

White spaces are used as delimiters between **ras** and **gk id** if present or **domain name**, and between **portno** and **priority**. White spaces consist of any number of spaces or tabs.

Examples of valid gatekeeper TXT records:

ras gk1

ras gk1.company.com

ras gk1:1500 3

ras 172.11.22.33:1500 2

The client parses the returned lines, and from them obtains the transport address of the gatekeeper within that domain to which it can send RAS messages.

Since DNS requires a server to return all TXT records associated with a domain name, the client can filter out and process only those records which are useful to it. It also allows DNS to return an ordered list of gatekeepers which can serve as alternatives and back-ups as defined in this Recommendation under the AlternateGK ASN.1 structure.

Note that the server returned in such a query might be an actual transport address in dotted decimal notation, or it could be an FQDN which itself requires an A-record query in DNS to determine the

transport address. The advantage of using an FQDN is the usual hiding of actual IP numbers. The advantage of using IP numbers is that a second DNS query is avoided, thus speeding up this pre-call setup time.

IV.1.1.2.3 Gatekeeper processing of email-IDs during ARQ and LRQ

When the **destinationInfo** field of an ARQ or LRQ message contains an **email-ID** alias address, the gatekeeper should first check its registration database for the alias. If it cannot be resolved, the gatekeeper should parse the alias to recover its domain portion. If no domain is given, the gatekeeper may generate a default domain. The domain is then used to locate one or more gatekeepers, using the procedures in IV.1.1.2.2. The gatekeeper may then query all gatekeepers thus found with an LRQ/LCF/LRJ message exchange.

Note that more than one gatekeeper may have corresponding TXT records in a single DNS domain. Consequently, a single DNS domain can "contain" more than one H.323 zone. Therefore, even if a gatekeeper cannot resolve an email-ID whose domain portion is one of its default domains, it may still query other zones in the same DNS domain.

If the gatekeeper is presented with an unregistered alias which is an **h323-id** and the ID can be interpreted as a legal user portion of an RFC 822 name, the gatekeeper may interpret the alias as if it were an email-ID in its default domain and attempt to locate the alias in some other gatekeeper. Similarly, an email-ID from an incoming LRQ may be stripped of its domain name by the gatekeeper so that it may be located as an h323-ID.

IV.1.2 Endpoint-to-endpoint communications

Endpoints which wish to receive calls from endpoints outside the zone of their gatekeeper should use the following port for the Call Signalling channel:

– Endpoint TCPCall Signalling Port 1720

While it is permitted to use dynamic values for these ports to allow multiple endpoints in a single device, it must be understood that this will prevent interoperation with endpoints outside the zone of the gatekeeper except via a gateway in the zone.

IV.2 SPX/IPX

Note that since there is no network re-assembly of large packets, the use of MB fragmentation is essential.

Unreliable delivery applications	**H.245 channel call signalling channel**
PXP	SPX
IPX	
Link Layer	
Physical Layer	

IV.2.1 Discovering the gatekeeper

In IPX terminology, a "socket" is the equivalent of a "port" in IP and a "TSAP Identifier" in this Recommendation and Recommendation H.323.

On IPX based networks, the gatekeepers should advertise the "gatekeeper service type" defined below to allow endpoints to locate them on a network. Likewise, endpoints should query for the "gatekeeper service type" to find the location of the nearest gatekeeper.

– Gatekeeper Service Type FFS

NOTE – The service type is referred to as the SAP socket in some IPX documentation.

IV.2.2 Endpoint-to-endpoint communication

Endpoints which wish to receive calls from endpoints outside the zone of their gatekeeper should use the following sockets for Call signalling.

– Endpoint IPX Call Signalling Port FFS

While it is permitted to use dynamic values for these sockets to allow multiple endpoints in a single device, it must be understood that this will prevent interoperation with endpoints outside the zone of the gatekeeper except via a gateway in the zone.

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure
Series Z	Programming languages