

**Superseded by a more recent version**



INTERNATIONAL TELECOMMUNICATION UNION

**ITU-T**

TELECOMMUNICATION  
STANDARDIZATION SECTOR  
OF ITU

**H.225.0**

(11/96)

**SERIES H: TRANSMISSION OF NON-TELEPHONE  
SIGNALS**

Infrastructure of audiovisual services – Transmission  
multiplexing and synchronization

---

**Media stream packetization and synchronization  
on non-guaranteed quality of service LANs**

ITU-T Recommendation H.225.0  
Superseded by a more recent version

(Previously CCITT Recommendation)

---

# Superseded by a more recent version

## ITU-T H-SERIES RECOMMENDATIONS TRANSMISSION OF NON-TELEPHONE SIGNALS

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	H.200–H.399
General	H.200–H.219
<b>Transmission multiplexing and synchronization</b>	<b>H.220–H.229</b>
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

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

# **Superseded by a more recent version**

**ITU-T RECOMMENDATION H.225.0**

## **MEDIA STREAM PACKETIZATION AND SYNCHRONIZATION ON NON-GUARANTEED QUALITY OF SERVICE LAN**

### **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 Local Area Networks (LANs), 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 non-guaranteed quality of service LAN can be managed to provide conversational services in H.323 equipment.

### **Source**

ITU-T Recommendation H.225.0 was prepared by ITU-T Study Group 15 (1993-1996) and was approved under the WTSC Resolution No. 1 procedure on the 8th of November 1996.

# Superseded by a more recent version

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

© ITU 1997

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the ITU.

# Superseded by a more recent version

## CONTENTS

	<b>Page</b>
1 Scope .....	1
2 References .....	3
3 Definitions .....	5
4 Conventions .....	5
5 Abbreviations .....	5
5.1 General Abbreviations .....	5
5.2 RAS Message Abbreviations .....	7
6 Packetization and synchronization mechanism .....	7
6.1 General approach .....	7
6.2 Use of RTP/RTCP .....	11
6.2.1 Audio .....	12
6.2.2 Video messages .....	13
6.2.3 Data messages .....	13
7 Initialization and Call Setup message definitions .....	14
7.1 Use of Q.931 .....	14
7.2 Common Q.931 Information Elements .....	16
7.2.1 Header Information Elements .....	16
7.2.2 Message-specific information elements .....	17
7.3 Q.931 Message details .....	23
7.3.1 Alerting .....	23
7.3.2 Call Proceeding .....	24
7.3.3 Connect .....	24
7.3.4 Connect Acknowledge .....	25
7.3.5 Disconnect .....	26
7.3.6 User Information .....	26
7.3.7 Notify .....	26
7.3.8 Progress .....	26
7.3.9 Release .....	27
7.3.10 Release Complete .....	27
7.3.11 Setup .....	28
7.3.12 Setup Acknowledge .....	30
7.3.13 Status .....	30
7.3.14 Status Inquiry .....	30
7.4 Q.932 Message Details .....	30

# Superseded by a more recent version

	<b>Page</b>
7.4.1 Facility.....	30
7.4.2 Hold.....	31
7.4.3 Hold Acknowledge.....	31
7.4.4 Hold Reject.....	32
7.4.5 Retrieve .....	32
7.4.6 Retrieve Acknowledge .....	32
7.4.7 Retrieve Reject .....	33
7.5 Q.931 timer values.....	33
7.6 H.225.0 RAS Message Common Parts.....	33
7.7 Required Support of RAS Messages .....	39
7.8 Terminal and gateway discovery messages.....	40
7.9 Terminal and gateway registration messages .....	41
7.10 Terminal/gatekeeper unregistration messages.....	43
7.11 Terminal to gatekeeper admission messages.....	44
7.12 Terminal to Gatekeeper Requests for Changes in Bandwidth.....	46
7.13 Location Request messages .....	48
7.14 Disengage messages .....	49
7.15 Status Request messages .....	50
7.16 Non-Standard message .....	52
7.17 Message not understood .....	52
8 Mechanisms for maintaining QOS .....	52
8.1 General approach and assumptions .....	52
8.2 Use of RTCP in measuring QOS.....	53
8.2.1 Sender Reports .....	53
8.2.2 Receiver Reports .....	53
8.3 Audio/video jitter procedures .....	54
8.4 Audio/video skew procedures .....	54
8.5 Procedures for maintaining QOS.....	54
8.6 Echo control.....	55
Annex A – RTP/RTCP .....	55
A.1 Introduction .....	56
A.2 RTP use scenarios.....	57
A.2.1 Simple multicast audio conference .....	57
A.2.2 Audio and video conference.....	58
A.2.3 Mixers and translators	58

# Superseded by a more recent version

	<b>Page</b>
A.3	Definitions ..... 58
A.4	Byte order, alignment, and time format..... 60
A.5	RTP Data Transfer Protocol ..... 61
A.5.1	RTP Fixed header fields ..... 61
A.5.2	Multiplexing RTP sessions ..... 62
A.5.3	Profile-specific modifications to the RTP header ..... 63
A.6	RTP Control Protocol – RTCP ..... 64
A.6.1	RTCP packet format..... 65
A.6.2	RTCP transmission interval ..... 66
A.6.3	Sender and Receiver Reports ..... 69
A.6.4	SDES: Source description RTCP packet..... 76
A.6.5	BYE: Goodbye RTCP packet..... 78
A.6.6	APP: Application-defined RTCP packet..... 78
A.7	RTP translators and mixers ..... 78
A.7.1	General description ..... 79
A.7.2	RTCP processing in translators ..... 80
A.7.3	RTCP processing in mixers..... 81
A.7.4	Cascaded mixers..... 82
A.8	SSRC identifier allocation and use ..... 82
A.8.1	Probability of collision ..... 82
A.8.2	Collision resolution and loop detection..... 83
A.9	Security..... 85
A.10	RTP over Network and Transport Protocols ..... 85
A.11	Summary of Protocol Constants..... 85
A.11.1	RTCP packet types ..... 86
A.11.2	SDES types..... 86
A.12	RTP Profiles and Payload Format Specifications ..... 87
A.13	Appendix I algorithms ..... 88
A.14	Bibliography ..... 88
Annex B – RTP Profile..... 89	
B.1	Introduction ..... 89
B.2	RTP and RTCP Packet Forms and Protocol Behaviour ..... 90
B.3	Payload Types ..... 90
B.4	Audio ..... 91
B.4.1	Encoding-independent recommendations ..... 91
B.4.2	Guidelines for sample-based audio encodings ..... 92
B.4.3	Guidelines for frame-based audio encodings ..... 92

# Superseded by a more recent version

	<b>Page</b>
B.4.4 Audio encodings.....	92
B.5 Video .....	93
B.6 Payload Type definitions.....	93
B.7 Port Assignment .....	94
Annex C – RTP payload format for H.261 video streams.....	95
C.1 Introduction .....	95
C.2 Structure of the packet stream .....	95
C.2.1 Overview of Recommendation H.261.....	95
C.2.2 Considerations for packetization.....	95
C.3 Specification of the packetization scheme .....	96
C.3.1 Usage of RTP .....	96
C.3.2 Recommendations for operation with hardware codecs .....	98
C.3.3 Packet loss issues .....	98
C.3.4 Use of optional H.261-specific control packets .....	99
C.3.5 Control packets definition .....	100
C.4 Bibliography .....	100
Annex D – RTP payload format for H.261A video streams.....	101
D.1 Introduction .....	101
D.2 H.261A RTP packetization.....	101
Appendix I – RTP/RTCP.....	102
Appendix II – RTP profile .....	102
Appendix III – H.261 packetization .....	102
Appendix IV .....	102
IV.1 TCP/IP/UDP.....	102
IV.1.1 Discovering the Gatekeeper .....	103
IV.1.2 Endpoint to Endpoint Communications .....	103
IV.2 SPX/IPX .....	103
IV.2.1 Discovering the Gatekeeper .....	103
IV.2.2 Endpoint to Endpoint Communication.....	104



# Superseded by a more recent version

## Recommendation H.225.0

### MEDIA STREAM PACKETIZATION AND SYNCHRONIZATION ON NON-GUARANTEED QUALITY OF SERVICE LAN

(Geneva, 1996)

The ITU-T,

*considering*

the widespread adoption of and the increasing use of the H.320 Recommendation 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 the H.322 or H.323 Recommendations 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 terminals on a non-guaranteed quality of service LAN, or between H.323 terminals and 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. This gateway, terminal descriptions, and procedures are described in H.323 while H.225.0 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.

H.225.0 is intended to operate over a variety of different LANs, including IEEE 802.3, Token Ring, etc. Thus, H.225.0 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 terminals and H.323 gateways on the same LAN, using the same transport protocol.*** This LAN may be a single segment or ring, or it logically could be an enterprise data network comprising multiple LANs 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 LANs may result in poor performance. The possible means by which quality of service might be assured on this LAN network, or on the Internet in general is beyond the scope of this Recommendation. However, H.225.0 provides a means for the user of H.323 equipment to determine that quality problems are the result of LAN 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.

## Superseded by a more recent version

H.323/H.225.0 are intended to extend H.320/H.221 conferences/connections onto the non-guaranteed QOS LAN environment. As such the primary conference model<sup>1</sup> 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.

H.225.0 makes use of RTP/RTCP (Real-time Transport Protocol/Real-Time Transport Control Protocol) for media stream packetization and synchronization for all underlying LANs (see Annexes A, B, and C). Please note that the usage of RTP/RTCP as specified in H.225.0 is not tied in any way to the usage of TCP/IP/UDP. H.225.0 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. H.225.0 contains details on the usage of RTP/RTCP.

In H.221, audio, video, data, and control are multiplexed into one or more synchronized physical SCN calls. On the LAN 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  $P \times 64$  kbit/s call, e.g. 2 by 64 kbit/s, 3 by 64 kbit/s, etc. Thus, on the LAN side, for example, there are only single "connection" calls with a maximum rate limited to 128 kbit/s, not  $2 \times 64$  kbit/s fixed rate calls. Another example has single "connection" LAN calls with a maximum rate limited to 384 kbit/s interworking with  $6 \times 64$  kbit/s on the WAN side<sup>2</sup>. The primary rationale of this approach is to put complexity in the gateway rather than the terminal and to avoid extending onto the LAN 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 LAN 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 H.323/H.225.0. Note that the H.323 terminal is indirectly aware of the H.320 transfer rates via the video maximum bit rate fields in H.245 and shall not transmit at rates that exceed these rates.

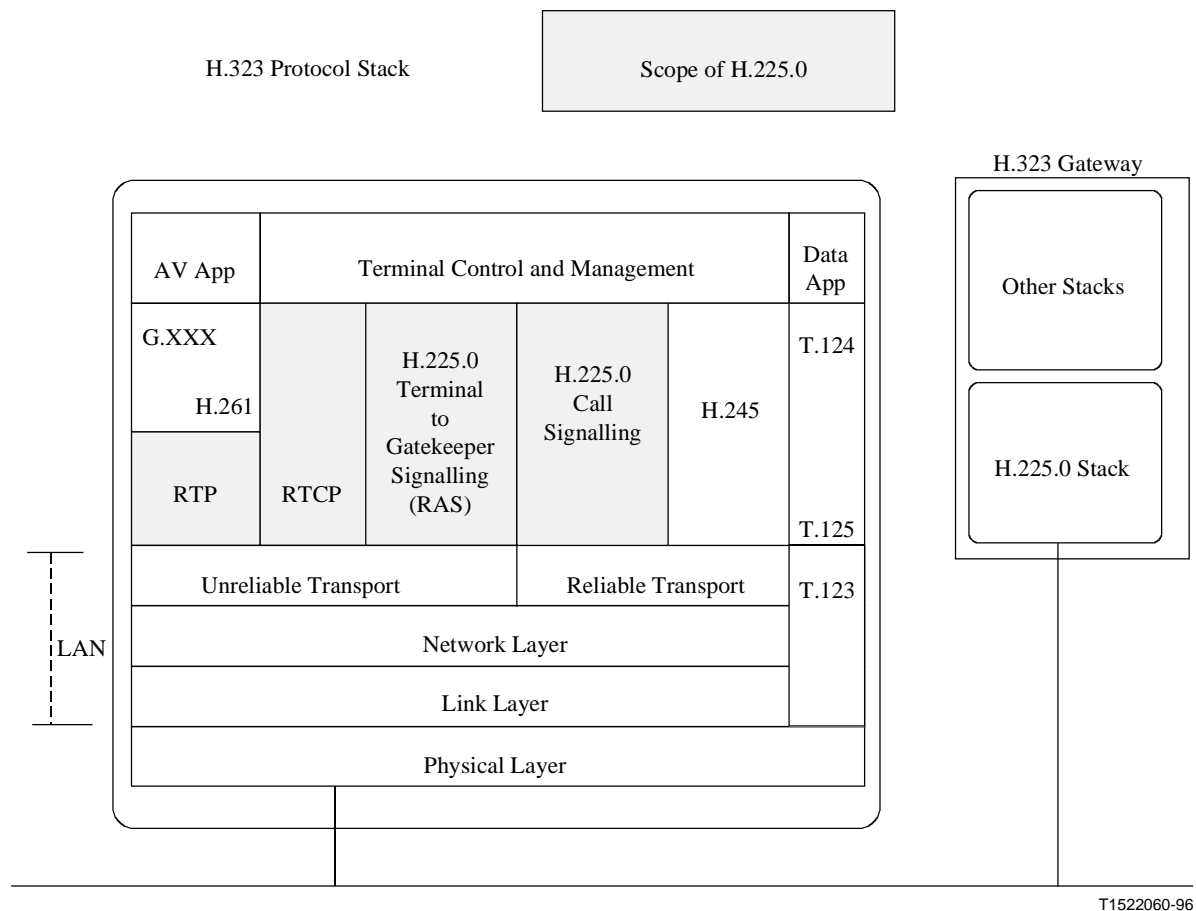
H.225.0 is designed so that, with an H.323 gateway, interoperability with H.320 (1990), H.320 (1993), and H.320 (1996) terminals is possible. However, some features of H.225.0 may be directed toward allowing enhanced operations with future versions of H.320. It is also possible that the quality of service on the H.320 side may vary based on the features and capabilities of the H.323 gateway (see Figure 1).

---

<sup>1</sup> An optional broadcast only conference model is under consideration; of necessity the broadcast model does not provide tight admissions or conference control.

<sup>2</sup> 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.

# Superseded by a more recent version



**Figure 1/H.225.0 – Scope of H.225.0**

The general approach of H.225.0 is to provide a means of synchronizing packets that make use of the underlying LAN/transport facilities. H.225.0 does not require all media and control to be mixed into a single stream, which is then packetized. The framing mechanisms of 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 LAN environment.
- H.245 and Q.931 can be sent over reliable links provided by the LAN.
- 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*.

## Superseded by a more recent version

- [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 kbit/s using conjugate structure algebraic-code-excited linear-prediction (CS-ACELP).*
- [6] ITU-T Recommendation H.221 (1995), *Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices.*
- [7] ITU-T Recommendation H.230 (1995), *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 (1996), *System for establishing communication between audiovisual terminals using digital channels up to 2 Mbit/s.*
- [10] ITU-T Recommendation H.243 (1996), *Procedures for establishing communication between three or more audiovisual terminals using digital channels up to 1920 Kbit/s.*
- [11] ITU-T Recommendation H.320 (1996), *Narrow-band visual telephone systems and terminal equipment.*
- [12] ITU-T Recommendation T.122 (1993), *Multipoint communication service for audiographics and audiovisual conferencing service definition.*
- [13] ITU-T Recommendation T.123 (1994), *Protocol stacks for audiographic and audiovisual teleconference applications.*
- [14] ITU-T Recommendation T.125 (1994), *Multipoint communication service protocol specification.*
- [15] ITU-T Recommendation H.321 (1996), *Adaptation of H.320 visual telephone terminals to B-ISDN environments.*
- [16] ITU-T Recommendation H.322 (1996), *Visual telephone systems and terminal equipment for local area networks which provide a guaranteed quality of service.*
- [17] ITU-T Recommendation H.324 (1996), *Terminal for low bit rate multimedia communications.*
- [18] ITU-T Recommendation H.310 (1996), *Broadband and audiovisual communication systems and terminals.*
- [19] ITU-T Recommendation Q.931 (1993), *ISDN user-network interface layer 3 specification for basic call control.*
- [20] ITU-T Recommendation Q.932 (1993), *Generic procedures for the control of ISDN supplementary services.*
- [21] ITU-T Recommendation X.680 (1994), *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- [22] ITU-T Recommendation X.691 (1995), *Information technology – ASN.1 encoding rules – Specification of Packed Encoding Rules (PER).*
- [23] ITU-T Recommendation H.245 (1996), *Control protocol for multimedia communication.*
- [24] CCITT Recommendation E.164 (1991), *Numbering plan for the ISDN era.*

# Superseded by a more recent version

- [25] ISO/IEC 10646-1:1993 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.*
- [26] ITU-T Recommendation Q.950 (1993), *Supplementary services protocols, structure and general principles.*
- [27] ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at  $p \times 64$  kbit/s.*
- [28] ITU-T Recommendation Q.850 (1993), *Usage of cause and location in the digital subscriber Signalling System No. 1 and the Signalling System No. 7 ISDN user part.*

## 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 H.225.0 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/second is abbreviated kbit/s and is measured in units of 1000. Thus, 64 kbit/s (kbps) is exactly 64 000 bit/s.

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

### 5.1 General Abbreviations

This Recommendation uses the following 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

## Superseded by a more recent version

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"

# Superseded by a more recent version

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  
IRR Information Request Response  
IRQ Information Request  
LCF Location Confirm  
LRJ Location Reject  
LRQ Location Request  
RCF Registration Confirm  
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 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.

## Superseded by a more recent version

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<sup>3</sup>, 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<sup>4</sup>.

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 LAN side multi-media conference (centralized vs distributed/multicast) is negotiated on a per connection basis<sup>5</sup>. 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 in a single packet as long as whole messages are sent; there shall be no fragmentation of H.245 messages across packets.

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 LAN environment<sup>6</sup>. 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/LAN stacks of T.123 as appropriate. T.120 shall be conveyed over the LAN 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

---

<sup>3</sup> A terminal that is not registered with a gatekeeper is not required to send an admissions request.

<sup>4</sup> 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.

<sup>5</sup> 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).

<sup>6</sup> Fast Update of full frames, MBs, or GOBs may be requested via H.245 signalling.



## Superseded by a more recent version

additional TSAP identifiers. All H.245 logical channels used are uni-directional except for those associated with T.120, which are bi-directional.

**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 LAN address and differ only by TSAP identifier, some manufacturers may choose to use different LAN addresses for audio and video. The only requirement is that the convention of Annexes A/B should be followed in the numbering of TSAP identifiers in the RTP session<sup>7</sup>.

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 below.		

<sup>7</sup> 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.

## Superseded by a more recent version

**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 (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 (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 LAN 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; H.245 provides this association which may be used for audio/video synchronization.

## Superseded by a more recent version

In general, two types of conference modes of operation on the LAN 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 H.323; the messages of H.225.0 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 LAN addresses for audio and video, but for each LAN address the convention of Annexes A/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 H.225.0. Implementations shall follow the RTP profile (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/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/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 LAN to LAN 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 H.225.0 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 LAN, 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 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 LAN 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 H.245 to that signalled in the H.245 **FlowControlCommands**, H.245 logical channel commands, and the T.120 flow control mechanism.

## Superseded by a more recent version

When the H.323 LAN terminal is connected to an H.323 gateway, the gateway shall use the means of 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 LAN need not match that in H.221.
- Audio bandwidth on the LAN may match that in 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 LAN 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 H.225.0 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. 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 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. 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.

When sending 48/56 kbit/s PCM, the H.323 gateway shall pad the extra 1 or 2 bits in each octet, and use the RTP values for PCMA or PCMU (8 or 0). For  $\mu$ -law the padding consists of "1" in both the 7<sup>th</sup> and 8<sup>th</sup> bit. For A-law the 7<sup>th</sup> bit shall be 0 and the 8<sup>th</sup> bit 1. In the reverse direction the H.323 gateway shall truncate 64 kbit/s G.711 on the LAN side to fit the G.711 rate being used in H.320. Thus, on the LAN side only 64 kbit/s G.711 shall be used.

When sending 48/56 kbit/s G.722 toward the LAN, the H.323 gateway shall pad the extra 1 or 2 bits in each octet, and use dynamic RTP payload types as signalled by 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 LAN side to fit the G.711 rate being used in H.320. Thus, on the LAN side only 64 kbit/s G.722 shall be used.

## Superseded by a more recent version

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 H.261 shall be used. Dynamic payload types may be used for H.263 or other ITU-T algorithms for which payload types do not exist.

**Marker (M):** The marker bit should be set except in cases where it would increase end-to-end delay.

In order to recover from the loss of video packets, H.245's **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.3.5.2 error recovery method (3) may be impractical if the NACK does not arrive within one frame time.

H.261 is packetized on the LAN 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 MTU (Maximum Transfer Unit) on a given LAN 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 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 LAN specific advice on video packetization.

### 6.2.3 Data messages

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

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

See 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.

# Superseded by a more recent version

The protocol to be used by H.224 on the LAN is for further study.

## 7 Initialization and Call Setup message definitions

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

### 7.1 Use of Q.931

Implementations shall follow the Q.931 as specified in H.225.0. Terminals may also support optional Q.931, Q.932, and Q.95x messages. The use of such messages is for further study. The Q.931 messages shall contain all of the mandatory information elements and may contain any of the optional information elements as defined in Q.931 as described in H.225.0. 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 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 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 protocols. The endpoints shall be able to ignore unknown information elements contained in a Q.931 message without disturbing operation.

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 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 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.

In this H.225.0, all references are to the 1993 version of Q.931/Q.932. 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 LAN 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 LAN 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. However, the gatekeeper may act as a Q.931 NETWORK to optionally support various of the Q.95x

## Superseded by a more recent version

supplementary services. The details of such procedures are for further study; the ability to treat the gatekeeper as a NETWORK is associated with support of the Q.95x supplementary services.

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/H.225.0 call setup using Q.931 on the LAN:

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

Call establishment messages	Transmit (M, F, O, CM) (Note 1)	Receive and act on [M, F, O (Note 2), CM]
Alerting	M	M
Call Proceeding	O	CM (Note 3)
Connect	M	M
Connect Acknowledge	F	F
Progress	O	O
Setup	M	M
Setup Acknowledge	O	O
<b>Call clearing messages</b>		
Disconnect	F	F
Release	F	F
Release Complete	M (Note 4)	M
<b>Call information phase messages</b>		
Resume	O	O
Resume Acknowledge	O	O
Resume Reject	O	O
Suspend	O	O
Suspend Acknowledge	O	O
Suspend Reject	O	O
User Information	O	O
<b>Miscellaneous messages</b>		
Congestion Control	F	F
Information	O	O
Notify	O	O
Status	M (Note 5)	M
Status Inquiry	O	M

# Superseded by a more recent version

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

Call establishment messages	Transmit (M, F, O, CM) (Note 1)	Receive and act on [M, F, O (Note 2), CM]
<b>Q.932 messages</b>		
Facility	M	M
Hold	O	O
Hold Acknowledge	O	O
Hold Reject	O	O
Retrieve	O	O
Retrieve Acknowledge	O	O
Retrieve Reject	O	O
<p>NOTE 1 – M Mandatory F Forbidden O Optional CM Conditional Mandatory</p> <p>Something is CM if it is required once an option is supported.</p> <p>NOTE 2 – 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.</p> <p>NOTE 3 – Terminals intended to use gateways shall receive and act on CALL PROCEEDING.</p> <p>NOTE 4 – Release Complete is required for any situation in which the H.225.0 reliable call signalling channel is open. If this channel is not open, H.245 session end may be used to terminate the conference.</p> <p>NOTE 5 – 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 know to the receiver.</p>		

## 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 is 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 end-point.

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.



## Superseded by a more recent version

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 number 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.

### 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 section 4.5.1/Q.931 and Table 4-3/Q.931. These rules shall be followed. The escape mechanism (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 LAN-to-LAN call it may be ignored by the receiver. 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 #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 endpoint, 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 #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 #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 #4.1*

- Shall be present if information transfer rate is set to 'multirate'.

## Superseded by a more recent version

- The extension bit (bit 8) shall be set to '1'.
- The bits 1-7 shall indicate the bandwidth needed for the call as defined in the following (Note, that in contrast to 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 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 LAN 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 LAN by the ACF/BCF messages.

### *Layer 1 protocol – octet #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  $\mu$ -law '00010') to indicate a voice-only call and H.221/H.242 ('00101') to indicate an H.323 videophone call.

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

### *Layer 2 protocol identifier – octet #6*

- Shall not be present.

### *Layer 3 protocol identifier – octet #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 #3 Coding Standard (bits 8-7)*

- Set to "00" for CCITT (ITU-T) standardized coding.

##### *Call State Value (octet #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.

## Superseded by a more recent version

### 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 #3 Extension (bit 8)*

- Set to '1'.

*Type of number (octet #3, bits 5-7)*

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

*Numbering plan identification (octet #3, bits 1-4)*

- Encoded following the values and rules of Table 4-9/Q.931. If set to "1001" in a LAN originated call, this indicates that:
  - 1) the E.164 address is not present in SETUP; and
  - 2) the call will be routed via an H323\_ID or transport 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 #3 Extension (bit 8)*

- Set to '1'.

*Type of number (octet #3, bits 5-7)*

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

*Numbering plan identification (octet #3, bits 1-4)*

- Encoded following the values and rules of Table 4-9/Q.931. If set to "1001" in a LAN originated call, this indicates that:
  - 1) the E.164 address is not present in SETUP; and
  - 2) the call will be routed via an H323\_ID or transport address in the user-to-user information.

*Octet #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.

# Superseded by a more recent version

## 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.

## 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 Facility

Encoded following Figure 8-2/Q.932 and Table 8-5/Q.932.

The Facility IE PDU shall be formed according to ROSE (uses X.208 [Specification of ASN.1] and X.209 [Specification of basic encoding rules for ASN.1]) as defined in Recommendations Q.932 and Q.952.

For the call forwarding case, the ROSE invoke component shall be completed as follows:

```
invokeIdentifier = sequence number
operationValue = callRerouting
argument =
{
    reroutingReason = cd           // cd = call deflection
    calledAddress = forwarded-to E164 // get alias from Facility-UUIE
    reroutingCounter = x
}
```

The **Facility-UUIE** will be encapsulated within the ROSE PDU as defined in Recommendation Q.932. If the forwarded-to endpoint cannot be specified with an E.164 address, the forwarding terminal shall supply either the **alternativeAddress** or **alternativeAliasAddress**.

In cases unique to H.323 (i.e. the FacilityReason codes found under the **Facility-UUIE** description found in H.225.0), the *reroutingReason* will be **cd** and the *reason* of the **Facility-UUIE** will contain the actual reason for the deflection. This means that the receiver of the Facility message must always check the **Facility-UUIE** reason.

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 completed in nearly the same manner as for forwarding. The **conferenceID** shall indicate the conference to join and the reason in the **Facility-UUIE** shall be **routeCallToMC**.

To instruct the calling endpoint to signal the called endpoint through the called endpoint's gatekeeper, the Facility IE would be completed in nearly the same manner as for forwarding. The reason in the **Facility-UUIE** shall be **routeCallToGatekeeper**.

Possible extensions for H.225.0 are for further study.

## Superseded by a more recent version

### 7.2.2.14 High layer compatibility

For further study.

### 7.2.2.15 Keypad facility

Encoded following Figure 4-24/Q.931.

### 7.2.2.16 Low layer compatibility

For further study.

### 7.2.2.17 More data

Shall not be used.

### 7.2.2.18 Network-specific facilities

Shall not be used.

### 7.2.2.19 Notification indicator

Encoded following Figure 4-28/Q.931 and Table 4-19/Q.931.

### 7.2.2.20 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 #3, bits 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.

### 7.2.2.21 Repeat indicator

Shall not be used.

### 7.2.2.22 Restart indicator

Shall not be used.

### 7.2.2.23 Segmented message

Shall not be used. Note that there is no critical upper limit on the message size in H.323/H.225.0.

### 7.2.2.24 Sending complete

Encoded following Figure 4-33/Q.931.

# Superseded by a more recent version

No restrictions apply.

## 7.2.2.25 Signal

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

No restrictions apply.

## 7.2.2.26 Transit network selection

Shall not be used.

## 7.2.2.27 User-user

Encoded following Figure 4-36/Q.931 and Table 4-26/Q.931.

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/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 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 Recommendation X.691. Note that the ASN.1 structure begins with **H323-UserInformation**.

**H323-MESSAGES DEFINITIONS AUTOMATIC TAGS ::= BEGIN**

**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,
    }
}
```

## Superseded by a more recent version

```

userInformation      UI-UUIE,
releaseComplete     ReleaseComplete-UUIE,
facility             Facility-UUIE,
...
},
nonStandardData     NonStandardParameter OPTIONAL,
...
}

```

- For the user-information field the rules specified in 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 do refer to messages that are generated by H.323 terminals only. Regardless of the specified sizes, messages forwarded from the SCN side may have different (larger) sizes.

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

#### 7.3.1 Alerting

Follow Table 3-2/Q.931 (1993 version) as modified below in Table 5.

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."

**Table 5/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
Channel identification	FFS	NA
Progress indicator	O	2-4
Display	O	2-82
Signal	O	2-3
High layer compatibility	FFS	NA
User-to-User	M (Note)	2-131
NOTE – The User-to-User IE contains the ASN.1 shown below. The size shown is understood as the size of the <b>user-data</b> structure in <b>H323-UserInformation</b> and does not include the <b>h323-UU-PDU</b> . The total size of <b>H323-UserInformation</b> is limited to 65 536 octets.		

The following information shall be provided in the ASN.1 User-to-User IE:

```

Alerting-UUIE ::= SEQUENCE
{
  protocolIdentifier      ProtocolIdentifier,
  destinationInfo        EndpointType,
  h245Address             TransportAddress OPTIONAL,
  ...
}

```

## Superseded by a more recent version

<b>protocolIdentifier</b> – set by the called endpoint to the version of H.225.0 supported.
<b>destinationInfo</b> – contains an <b>EndpointType</b> to allow the caller to determine whether the call involves a gateway or not.
<b>h245Address</b> – 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 or Connect.

### 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 6.

**Table 6/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
Channel identification	FFS	NA
Progress indicator	O	2-4
Display	O	2-82
High layer compatibility	FFS	NA
User-to-User	M	2-131

The following information shall be provided in the ASN.1 User-to-User IE:

```

CallProceeding-UUIE ::= SEQUENCE
{
    protocolIdentifier    ProtocolIdentifier,
    destinationInfo      EndpointType,
    h245Address          TransportAddress OPTIONAL,
    ...
}
    
```

<b>protocolIdentifier</b> – set by the called endpoint to the version of H.225.0 supported.
<b>destinationInfo</b> – contains an EndpointType to allow the caller to determine whether the call involves a gateway or not.
<b>h245Address</b> – 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.

### 7.3.3 Connect

Follow Table 3-4/Q.931, as modified below.

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. See Table 7.



# Superseded by a more recent version

Table 7/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 1)	5-6
Channel identification	FFS	NA
Progress indicator	O	2-4
Display	O	2-82
Date/Time <sup>6</sup>	O	8
High layer compatibility	FFS	NA
Low layer compatibility	FFS	NA
User-to-User	M (Note 2)	2-131
NOTE 1 – BC is mandatory if the message is between a terminal and a gateway.		
NOTE 2 – The User-to-User IE contains the ASN.1 below.		

The following information shall be provided in the ASN.1 User-to-User IE:

```

Connect-UUIE ::= SEQUENCE
{
    protocolIdentifier      ProtocolIdentifier,
    h245Address             TransportAddress OPTIONAL,
    destinationInfo        EndpointType,
    conferenceID            ConferenceIdentifier,
    ...
}
    
```

<b>protocolIdentifier</b> – set by the called endpoint to the version of H.225 supported.
<b>h245Address</b> – 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 or CALL PROCEEDING.
<b>destinationInfo</b> – contains an EndpointType to allow the caller to determine whether the call involves a gateway or not.
<b>conferenceID</b> – will contain a unique number to allow the conference to be uniquely identified from all others as received in the SETUP.

## 7.3.4 Connect Acknowledge

Follow Table 3-5/Q.931 as modified below (see Table 8).

This message shall not be sent.

# Superseded by a more recent version

Table 8/H.225.0

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.

## 7.3.6 User Information

This message may be sent to provide supplementary 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 9).

Table 9/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

```
UI-UUIE ::= SEQUENCE
{
    protocolIdentifier ProtocolIdentifier,
    ...
}
```

## 7.3.7 Notify

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

The contents and semantics of a NOTIFY message received from the network are defined in Table 3-8/Q.931.

## 7.3.8 Progress

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

## Superseded by a more recent version

The contents and semantics of a PROGRESS message received are defined in Table 3-9/Q.931.

### 7.3.9 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.

### 7.3.10 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 LAN environment, the single step method of sending only Release Complete is used.

Follow Table 3-11/Q.931. The following modifications apply (see Table 10).

**Table 10/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
Display	O	2-82
Signal	O	2-3
User-to-User	M	2-131
NOTE – Either the Cause IE or the <b>ReleaseCompleteReason</b> shall be present.		

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

**ReleaseComplete-UUIE ::= SEQUENCE**

```
{
    protocolIdentifier      ProtocolIdentifier,
    reason                  ReleaseCompleteReason OPTIONAL,
    ...
}
```

**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
}
```

## Superseded by a more recent version

```

inConf          NULL,          -- no address in AlternativeAddress
undefinedReason NULL,
...
}
    
```

<b>protocolIdentifier</b> – set by the calling endpoint to the version of H.225.0 supported.
<b>reason</b> – more information on why the call was released.

### 7.3.11 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 below (see Table 11).

**Table 11/H.225.0**

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
Channel identification	FFS	NA
Progress indicator	F	NA
Network specific facilities	F	NA
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 LAN side only calls.		
NOTE 2 – If an ARQ was previously sent, the CRV used here shall be the same.		

## Superseded by a more recent version

The following information shall be provided in the ASN.1 User-to-User IE:

```

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,      -- see Note
    destExtraCRV            SEQUENCE OF CallReferenceValue OPTIONAL, -- see Note
    activeMC                BOOLEAN,
    conferenceID            ConferenceIdentifier,
    conferenceGoal          CHOICE
    {
        create              NULL,
        join                 NULL,
        invite               NULL,
        ...
    },
    callServices            QseriesOptions OPTIONAL,
    callType                CallType,
    ...
}

```

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.

<b>protocolIdentifier</b> – set by the calling endpoint to the version of H.225.0 supported.
<b>h245Address</b> – this is a specific transport address on which the calling endpoint or gatekeeper handling the call would like to establish 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.
<b>sourceAddress</b> – contains the H323_IDs for the source; the E.164 number of the source is in the Q.931 part of SETUP. The primary address shall be first.
<b>sourceInfo</b> – contains an <b>EndpointType</b> to allow the called party to determine whether the call involves a gateway or not.
<b>destinationAddress</b> – this is the address the endpoint wishes to be connected to. 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 IE.
<b>destCallSignalAddress</b> – needed to inform the gatekeeper of the destination terminal's call signalling transport address; redundant in the direct terminal-to-terminal case.
<b>destExtraCallInfo</b> – 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.
<b>destExtraCRV</b> – CRVs for the additional SCN calls specified by <b>destExtraCallInfo</b> . Their use is for further study.
<b>activeMC</b> – indicates that the calling endpoint is under the influence of an active MC.
<b>conferenceID</b> – unique conference identifier.

# Superseded by a more recent version

**conferenceGoal** – indicates a desire to join an existing conference, start a new conference, or to invite a party to join an existing conference.

**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.

## 7.3.12 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.

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

## 7.3.13 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.

## 7.3.14 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 Recommendation Q.932 (1993). Refer to Recommendation Q.932 for further details.

The message type information element shall be encoded following the rules of 8.1/Q.932.

### 7.4.1 Facility

The FACILITY message may be used to request or acknowledge a supplementary service. It shall be used to provide information on where a call should be directed as part of call transfer or a terminal indicating that the incoming call must go through a gatekeeper.

Follow Table 7-2/Q.932. The following modifications apply (see Table 12).

**Table 12/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
Facility	M	8-*
Display	O	2-82
User-to-User	M	2-131

# Superseded by a more recent version

Facility-UUIE ::= SEQUENCE

```
{
    protocolIdentifier      ProtocolIdentifier,
    alternativeAddress      TransportAddress OPTIONAL,
    alternativeAliasAddress SEQUENCE OF AliasAddress OPTIONAL,
    conferenceID           ConferenceIdentifier OPTIONAL,
    reason                 FacilityReason,
    ...
}
```

FacilityReason ::= CHOICE

```
{
    routeCallToGatekeeper  NULL,      -- call must use gatekeeper model
                                -- gatekeeper is alternativeAddress
    callForwarded          NULL,
    routeCallToMC          NULL,
    undefinedReason        NULL,
    ...
}
```

<b>protocolIdentifier</b> – set by the calling endpoint to the version of H.225.0 supported.
<b>alternativeAddress</b> – this is a specific transport address to which the calling party should direct the call; if present <b>alternativeAliasAddress</b> is not needed.
<b>alternativeAliasAddress</b> – contains aliases that can be used to re-direct the call; if an alias is provided <b>alternativeAddress</b> is not needed.
<b>conferenceID</b> – unique conference identifier.
<b>reason</b> – more information about the facility message.

## 7.4.2 Hold

HOLD message is used to put an existing call in hold. Hold Ack is used by the User to indicate that the Hold function has been successfully performed.

Follow Table 7-3/Q.932. The following modifications apply (see Table 13).

**Table 13/H.225.0 – Hold**

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

## 7.4.3 Hold Acknowledge

HOLD message is used to put an existing call in hold. Hold Ack is used by the User to indicate that the Hold function has been successfully performed.

Follow Table 7-4/Q.932. The following modifications apply (see Table 14).

# Superseded by a more recent version

**Table 14/H.225.0 – Hold 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

## 7.4.4 Hold Reject

This message is sent to indicate the denial of a request to hold a call.

Follow Table 7-5/Q.932. The following modifications apply (see Table 15).

**Table 15/H.225.0 – Hold Reject**

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
Cause	M	4-32
User-user	FFS	*

## 7.4.5 Retrieve

This message shall be sent to request the retrieval of a held call.

Follow Table 7-7/Q.932. The following modifications apply (see Table 16).

**Table 16/H.225.0 – Retrieve**

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
Channel identification	FFS	NA
Display	O	2-82
User-user	FFS	*

## 7.4.6 Retrieve Acknowledge

This message is sent to indicate that the Retrieve has been successful.

Follow Table 7-8/Q.932. The following modifications apply (see Table 17).



# Superseded by a more recent version

**Table 17/H.225.0 – Retrieve 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
Channel identification	FFS	NA
Display	O	2-82
User-user	FFS	*

## 7.4.7 Retrieve Reject

This message shall be sent to indicate the inability to perform the requested Retrieve. Follow Table 7-9/Q.932. The following modifications apply (see Table 18).

**Table 18/H.225.0 – Retrieve Reject**

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	M	4-32
Display	O	2-82
User-user	FFS	*

## 7.5 Q.931 timer values

Two Q.931 timers shall be supported:

- The "setup timer" (T303/see Table 9-1/Q.931 and Table 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 4 seconds.

- The "establishment timer"(T301/see Table 9-1/Q.931 and Table 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 LAN side values of these timers is the same as that used in the SCN.

Other timers may be supported as part of optional Q.931, Q.932, and Q.95x features.

## 7.6 H.225.0 RAS Message Common Parts

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

## Superseded by a more recent version

**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 = "LAN Address plus TSAP identifier" language of H.323. Instead, the terms common in each transport domain are used.

```
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),
        ...
    }
}
```

## Superseded by a more recent version

```
    },
    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,
    ...
}

H310Caps ::= SEQUENCE
{
    nonStandardData  NonStandardParameter OPTIONAL,
    ...
}

H320Caps ::= SEQUENCE
{
    nonStandardData  NonStandardParameter OPTIONAL,
    ...
}

H321Caps ::= SEQUENCE
{
    nonStandardData  NonStandardParameter OPTIONAL,
    ...
}
```

## Superseded by a more recent version

```
H322Caps ::= SEQUENCE
{
    nonStandardData NonStandardParameter OPTIONAL,
    ...
}

H323Caps ::= SEQUENCE
{
    nonStandardData NonStandardParameter OPTIONAL,
    ...
}

H324Caps ::= SEQUENCE
{
    nonStandardData NonStandardParameter OPTIONAL,
    ...
}

VoiceCaps ::= SEQUENCE
{
    nonStandardData NonStandardParameter OPTIONAL,
    ...
}

T120OnlyCaps ::= SEQUENCE
{
    nonStandardData NonStandardParameter OPTIONAL,
    ...
}

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
    ...
}
```

## Superseded by a more recent version

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

```
AliasAddress ::= CHOICE
{
    e164          IA5String (SIZE (1..128)) (FROM ("0123456789#*,")),
    h323-ID      BMPString (SIZE (1..256)),      -- Basic ISO/IEC 10646-1 (Unicode)
    ...
}
```

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.

```
QseriesOptions ::= SEQUENCE
{
    q932Full      BOOLEAN,      -- if true, indicates full support for Recommendation Q.932
    q951Full      BOOLEAN,      -- if true, indicates full support for Recommendation Q.951
    q952Full      BOOLEAN,      -- if true, indicates full support for Recommendation Q.952
    q953Full      BOOLEAN,      -- if true, indicates full support for Recommendation Q.953
    q955Full      BOOLEAN,      -- if true, indicates full support for Recommendation Q.955
    q956Full      BOOLEAN,      -- if true, indicates full support for Recommendation Q.956
    q957Full      BOOLEAN,      -- if true, indicates full support for Recommendation Q.957
    q954Info      Q954Details,
    ...
}
```

```
Q954Details ::= SEQUENCE
{
    conferenceCalling      BOOLEAN,
    threePartyService      BOOLEAN,
    ...
}
```

Commonly used values include:

```
ConferenceIdentifier ::= OCTET STRING (SIZE (16))
RequestSeqNum        ::= INTEGER (1..65535)
GatekeeperIdentifier ::= BMPString (SIZE(128))
BandWidth            ::= INTEGER (1.. 4294967295)      -- in 100s of bits
CallReferenceValue   ::= INTEGER (1..65535)
EndpointIdentifier    ::= BMPString (SIZE(128))
ProtocolIdentifier    ::= OBJECT IDENTIFIER
                    -- shall be set to
                    -- {itu-t(0) recommendation(0) h(8) h225-0 version(0) 1}
```

```
NonStandardParameter ::= SEQUENCE
{
    nonStandardIdentifier NonStandardIdentifier,
    data                  OCTET STRING
}
```

## Superseded by a more recent version

```
H221NonStandard ::= SEQUENCE
{
    t35CountryCode    INTEGER(0..255),    -- country, as per Recommendation T.35
    t35Extension      INTEGER(0..255),    -- assigned nationally
    manufacturerCode  INTEGER(0..65535), -- assigned nationally
    ...
}
```

```
NonStandardIdentifier ::= CHOICE
{
    object            OBJECT IDENTIFIER,
    h221NonStandard  H221NonStandard,
    ...
}
```

The root structure for RAS messages is:

```
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,
    ...
}
```

# Superseded by a more recent version

## 7.7 Required Support of RAS Messages

Table 19 shows the RAS messages that are supported by different endpoint types.

**Table 19/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 3)	M	M	O
BRJ	M	M	M	O
IRQ		M	M	
IRR	M			M
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

NOTE 1 – M Mandatory  
O Optional  
F Forbidden  
CM Conditionally Mandatory  
blank indicates "Not Applicable"

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

NOTE 3 – Note that 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.

# Superseded by a more recent version

## 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.

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

```
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,
    ...
}
```

<b>requestSeqNum</b> – 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.
--

<b>rasAddress</b> – this is the transport address that this endpoint uses for registration and status messages.
---

<b>endpointType</b> – this specifies the type(s) of the terminal that is registering (the MC bit shall not be set by itself).
---

<b>gatekeeperIdentifier</b> – string to identify the gatekeeper that the terminal would like to receive permission to register from. A missing or null string <b>gatekeeperIdentifier</b> indicates that the terminal is interested in any available gatekeeper.
--

<b>callServices</b> – provides information on support of optional Q-Series protocols to gatekeeper and called terminal.
---

```
GatekeeperConfirm ::= SEQUENCE --(GCF)
{
    requestSeqNum      RequestSeqNum,
    protocolIdentifier ProtocolIdentifier,
    nonStandardData    NonStandardParameter OPTIONAL,
    gatekeeperIdentifier GatekeeperIdentifier OPTIONAL,
    rasAddress         TransportAddress,
    ...
}
```

<b>requestSeqNum</b> – this shall be the same value that was passed in the GRQ.
---

<b>gatekeeperIdentifier</b> – string to identify gatekeeper that is sending the GCF.
--

<b>rasAddress</b> – this is the transport address that the gatekeeper uses for registration and status messages.
--

```
GatekeeperReject ::= SEQUENCE --(GRJ)
{
    requestSeqNum      RequestSeqNum,
    protocolIdentifier ProtocolIdentifier,
```



## Superseded by a more recent version

```

nonStandardData      NonStandardParameter OPTIONAL,
gatekeeperIdentifier GatekeeperIdentifier OPTIONAL,
rejectReason         GatekeeperRejectReason,
...
}

GatekeeperRejectReason ::= CHOICE
{
    resourceUnavailable      NULL,
    terminalExcluded         NULL,      -- permission failure, not a resource failure
    invalidRevision          NULL,
    undefinedReason          NULL,
    ...
}

```

<b>requestSeqNum</b> – this shall be the same value that was passed in the GRQ.
---

<b>gatekeeperIdentifier</b> – string to identify gatekeeper that is sending the GRJ.
--

<b>rejectReason</b> – codes for why the GRQ was rejected by this gatekeeper.
--

### 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.

```

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,
    ...
}

```

<b>requestSeqNum</b> – 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.
--

<b>protocolIdentifier</b> – identifies the H.225.0 vintage of the sending terminal.
---

<b>discoveryComplete</b> – 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 <b>discoveryRequired</b> or <b>notRegistered</b> respectively. This indicates that the endpoint should perform the discovery procedure (either dynamic or static) before issuing the RRQ with <b>discoveryComplete</b> set to TRUE.
---

<b>callSignalAddress</b> – this is the call control transport address for this endpoint. If multiple transports are supported, they must be registered all at once.
---

<b>rasAddress</b> – this is the registration and status transport address for this endpoint.
--

## Superseded by a more recent version

**terminalType** – this specifies the type(s) of the terminal 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.

**terminalAlias** – this optional value is a list of external addresses, by which external (to the LAN) terminals may identify this terminal such as E.164 numbers or H323\_IDs. 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. Note that multiple E.164 addresses or H323\_IDs may refer to the same transport addresses. All of the endpoint's alias's 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.

```
RegistrationConfirm ::= SEQUENCE --(RCF)
{
    requestSeqNum      RequestSeqNum,
    protocolIdentifier ProtocolIdentifier,
    nonStandardData    NonStandardParameter OPTIONAL,
    callSignalAddress  SEQUENCE OF TransportAddress,
    terminalAlias       SEQUENCE OF AliasAddress OPTIONAL,
    gatekeeperIdentifier GatekeeperIdentifier OPTIONAL,
    endpointIdentifier EndpointIdentifier,
    ...
}
```

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

**protocolIdentifier** – identifies the vintage of the accepting gatekeeper.

**callSignalAddress** – this is an array of transport addresses for H.225.0 call control 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 external addresses, by which external (to the LAN) terminals may identify this terminal such as E.164 numbers or H323\_IDs.

**gatekeeper identifier** – 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.

```
RegistrationReject ::= SEQUENCE --(RRJ)
{
    requestSeqNum      RequestSeqNum,
    protocolIdentifier ProtocolIdentifier,
    nonStandardData    NonStandardParameter OPTIONAL,
    rejectReason       RegistrationRejectReason,
    gatekeeperIdentifier GatekeeperIdentifier OPTIONAL,
    ...
}
```

## Superseded by a more recent version

<b>requestSeqNum</b> – this shall be the same value that was passed in the RRQ.
<b>protocolIdentifier</b> – identifies the vintage of the rejecting gatekeeper.
<b>rejectReason</b> – the reason for the rejection of the registration.
<b>gatekeeper identifier</b> – string to identify the gatekeeper that has rejected the terminal's registration.

```

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
    ...
}

```

### 7.10 Terminal/gatekeeper unregistration messages

The URQ requests that the association between a terminal and a gatekeeper be broken. Note that unregister is bi-directional; 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.

```

UnregistrationRequest ::= SEQUENCE --(URQ)
{
    requestSeqNum      RequestSeqNum,
    callSignalAddress  SEQUENCE OF TransportAddress,
    endpointAlias       SEQUENCE OF AliasAddress OPTIONAL,
    nonStandardData    NonStandardParameter OPTIONAL,
    endpointIdentifier  EndpointIdentifier OPTIONAL,
    ...
}

```

<b>requestSeqNum</b> – 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.
<b>callSignalAddress</b> – this is one or more of the transport call signalling addresses for this endpoint which are to be unregistered.
<b>endpointAlias</b> – this optional value is a list of external addresses, by which external (to the LAN) terminals may identify this terminal such as E.164 numbers or H323_IDs. 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.
<b>endpointIdentifier</b> – confirmation of identity; not sent by the gatekeeper.

```

UnregistrationConfirm ::= SEQUENCE --(UCF)
{
    requestSeqNum      RequestSeqNum,
    nonStandardData    NonStandardParameter OPTIONAL,
    ...
}

```

# Superseded by a more recent version

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

```
UnregistrationReject ::= SEQUENCE --(URJ)
{
    requestSeqNum      RequestSeqNum,
    rejectReason       UnregRejectReason,
    nonStandardData    NonStandardParameter OPTIONAL,
    ...
}
```

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

**rejectReason** – the reason for the rejection of the unregistration.

```
UnregRejectReason ::= CHOICE
{
    notCurrentlyRegistered NULL,
    callInProgress         NULL,
    undefinedReason        NULL,
    ...
}
```

## 7.11 Terminal to gatekeeper admission messages

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

```
AdmissionRequest ::= SEQUENCE --(ARQ)
{
    requestSeqNum      RequestSeqNum,
    callType           CallType,
    callModel          CallModel OPTIONAL,
    endpointIdentifier EndpointIdentifier,
    destinationInfo    SEQUENCE OF AliasAddress OPTIONAL, -- see Note
    destCallSignalAddress TransportAddress OPTIONAL, -- see Note
    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
    ...
}
```

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.

## Superseded by a more recent version

```

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,
    ...
}

```

<b>requestSeqNum</b> – 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.
<b>callType</b> – using this value, gatekeeper can attempt to determine 'real' bandwidth usage. The default value is <b>pointToPoint</b> 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.
<b>callModel</b> – if <b>direct</b> , the endpoint is requesting the direct terminal to terminal call model. If <b>gatekeeperRouted</b> , the endpoint is requesting the gatekeeper mediated model. The gatekeeper is not required to comply with this request.
<b>endpointIdentifier</b> – this is an endpoint identifier that was assigned to the terminal by RCF, probably the E.164 address or H323_ID. It is used as a security measure to help ensure that this is a registered terminal within its zone.
<b>destinationInfo</b> – sequence of external addresses for the destination terminal, such as E.164 addresses or H323_IDs.
<b>destCallSignalAddress</b> – transport address used at the destination for call signalling.
<b>destExtraCallInfo</b> – contains external addresses for multiple calls.
<b>srcInfo</b> – sequence of external addresses for the source terminal, such as E.164 addresses or H323_IDs.
<b>srcCallSignalAddress</b> – transport address used at the source for call signalling.
<b>bandWidth</b> – the number of 100 bit/s requested for the bi-directional 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.
<b>callReferenceValue</b> – 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.
<b>callServices</b> – provides information on support of optional Q-series protocols to gatekeeper and called terminal.
<b>conferenceID</b> – unique conference identifier.
<b>activeMC</b> – if TRUE, the calling party has an active MC; otherwise FALSE.
<b>answerCall</b> – used to indicate to a gatekeeper that a call is incoming.

```

AdmissionConfirm ::= SEQUENCE --(ACF)
{
    requestSeqNum      RequestSeqNum,
    bandWidth          BandWidth,
    callModel          CallModel,
}

```

## Superseded by a more recent version

```

destCallSignalAddress  TransportAddress,
irrFrequency           INTEGER (1..65535) OPTIONAL,
nonStandardData        NonStandardParameter OPTIONAL,
...
}

```

<b>requestSeqNum</b> – this shall be the same value that was passed in the ARQ.
<b>bandWidth</b> – the allowed maximum bandwidth for the call; may be less than that requested.
<b>callModel</b> – tells terminal whether call signalling sent on destCallSignalAddress goes to a gatekeeper or to a terminal. A value of <b>gatekeeperRouted</b> indicates that call signalling is being passed via the gatekeeper, while <b>direct</b> indicates that the endpoint-to-endpoint call mode is in use.
<b>destCallSignalAddress</b> – the transport address to send Q.931 call signalling, but may be an endpoint or gatekeeper address depending on the call model in use.
<b>irrFrequency</b> – 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.

```

AdmissionReject ::= SEQUENCE --(ARJ)
{
    requestSeqNum      RequestSeqNum,
    rejectReason       AdmissionRejectReason,
    nonStandardData    NonStandardParameter 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,
    ...
}

```

<b>requestSeqNum</b> – this shall be the same value that was passed in the ARQ.
<b>rejectReason</b> – reason the bandwidth request was denied.

### 7.12 Terminal to Gatekeeper Requests for Changes in Bandwidth

The **BRQ** message requests that an endpoint be granted a changed LAN 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**.

```

BandwidthRequest ::= SEQUENCE --(BRQ)
{
    requestSeqNum      RequestSeqNum,
    endpointIdentifier  EndpointIdentifier,
}

```

## Superseded by a more recent version

```

conferenceID           ConferenceIdentifier,
callReferenceValue     CallReferenceValue,
callType              CallType OPTIONAL,
bandWidth             BandWidth,
nonStandardData       NonStandardParameter OPTIONAL,
...
}

```

**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, probably the E.164 address or H323\_ID. It is used as a security measure to help ensure that this is a registered terminal within its zone.

**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 bit/s increments requested for the call. This is an absolute value that includes only audio and video bitstreams not counting headers and overhead.

```

BandwidthConfirm      ::= SEQUENCE --(BCF)
{
    requestSeqNum       RequestSeqNum,
    bandWidth          BandWidth,
    nonStandardData    NonStandardParameter OPTIONAL,
    ...
}

```

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

**BandWidth** – the maximum allowed at this time in increments of 100 bit/s.

```

BandwidthReject      ::= SEQUENCE --(BRJ)
{
    requestSeqNum       RequestSeqNum,
    rejectReason       BandRejectReason,
    allowedBandWidth   BandWidth,
    nonStandardData    NonStandardParameter OPTIONAL,
    ...
}

```

```

BandRejectReason     ::= CHOICE
{
    notBound           NULL,           -- discovery permission has aged
    invalidConferenceID NULL,           -- possible revision
    invalidPermission  NULL,           -- true permission violation
    insufficientResources NULL,
    invalidRevision    NULL,
    undefinedReason    NULL,
    ...
}

```

# Superseded by a more recent version

<b>requestSeqNum</b> – this shall be the same value that was passed in the BRQ.
<b>rejectReason</b> – the reason the change was rejected by the gatekeeper.
<b>allowedBandWidth</b> – the maximum allowed at this time in increments of 100 bit/s including the current allocation.

## 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**.

```
LocationRequest ::= SEQUENCE --(LRQ)
{
    requestSeqNum      RequestSeqNum,
    endpointIdentifier  EndpointIdentifier OPTIONAL,
    destinationInfo    SEQUENCE OF AliasAddress,
    nonStandardData    NonStandardParameter OPTIONAL,
    replyAddress       TransportAddress,
    ...
}
```

<b>requestSeqNum</b> – 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.
<b>endpointIdentifier</b> – this is an endpoint identifier that was assigned to the terminal by RCF, probably the E.164 address or H323_ID. It is used as a security measure to help ensure that this is a registered terminal within its zone.
<b>destinationInfo</b> – sequence of external addresses for the destination terminal, such as E.164 addresses or H323_IDs.
<b>replyAddress</b> – transport address to send the LCF/LRQ.

```
LocationConfirm ::= SEQUENCE --(LCF)
{
    requestSeqNum      RequestSeqNum,
    callSignalAddress  TransportAddress,
    rasAddress         TransportAddress,
    nonStandardData    NonStandardParameter OPTIONAL,
    ...
}
```

<b>requestSeqNum</b> – this shall be the same value that was passed in the LRQ.
<b>callSignalAddress</b> – the transport address 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.
<b>rasAddress</b> – registration, admissions, and status address for the located endpoint.

```
LocationReject ::= SEQUENCE --(LRJ)
{
    requestSeqNum      RequestSeqNum,
    rejectReason       LocationRejectReason,
    nonStandardData    NonStandardParameter OPTIONAL,
    ...
}
```



# Superseded by a more recent version

```
LocationRejectReason ::= CHOICE
{
    notRegistered          NULL,
    invalidPermission      NULL,      -- exclusion by administrator or feature
    requestDenied          NULL,      -- cannot find location
    undefinedReason        NULL,
    ...
}
```

<b>requestSeqNum</b> – this shall be the same value that was passed in the LRQ.
---

<b>rejectReason</b> – reason the location request was denied.
---

## 7.14 Disengage messages

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.

```
DisengageRequest ::= SEQUENCE --(DRQ)
{
    requestSeqNum          RequestSeqNum,
    endpointIdentifier      EndpointIdentifier,
    conferenceID            ConferenceIdentifier,
    callReferenceValue      CallReferenceValue,
    disengageReason         DisengageReason,
    nonStandardData         NonStandardParameter OPTIONAL,
    ...
}
```

```
DisengageReason ::= CHOICE
{
    forcedDrop              NULL,      -- gatekeeper is forcing the drop
    normalDrop              NULL,      -- associated with normal drop
    undefinedReason         NULL,
    ...
}
```

<b>requestSeqNum</b> – 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.
--

<b>endpointIdentifier</b> – this is an endpoint identifier that was assigned to the terminal by RCF, probably the E.164 address or H323_ID. It is used as a security measure to help ensure that this is a registered terminal within its zone.
---

<b>conferenceID</b> – ID of the call that is to have the bandwidth released.
--

<b>callReferenceValue</b> – 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.
--

<b>disengageReason</b> – the reason the change was requested by the gatekeeper or the terminal.
---

## Superseded by a more recent version

```
DisengageConfirm ::= SEQUENCE --(DCF)
{
    requestSeqNum RequestSeqNum,
    nonStandardData NonStandardParameter OPTIONAL,
    ...
}
```

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

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

```
DisengageReject ::= SEQUENCE --(DRJ)
{
    requestSeqNum RequestSeqNum,
    rejectReason DisengageRejectReason,
    nonStandardData NonStandardParameter OPTIONAL,
    ...
}
DisengageRejectReason ::= CHOICE
{
    notRegistered NULL, -- not registered with gatekeeper
    requestToDropOther NULL, -- cannot request drop for others
    ...
}
```

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

### 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 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.

```
InfoRequest ::= SEQUENCE --(IRQ)
{
    requestSeqNum RequestSeqNum,
    callReferenceValue CallReferenceValue,
    nonStandardData NonStandardParameter OPTIONAL,
    replyAddress TransportAddress OPTIONAL,
    ...
}
```

**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.

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

# Superseded by a more recent version

```
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,
        callSignalling      TransportChannelInfo,
        callType            CallType,
        bandWidth           BandWidth,
        callModel           CallModel,
        ...
    } OPTIONAL,
    ...
}

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),
    ...
}
```

# Superseded by a more recent version

<b>requestSeqNum</b> – this shall contain the sequence number from the IRR or one (1) for an unsolicited report to the gatekeeper.
<b>endpointType</b> – provides information about the endpoint.
<b>endpointIdentifier</b> – value assigned by the gatekeeper in the ACF.
<b>rasAddress</b> – address for registration, admissions, etc.
<b>callSignalAddress</b> – address of H.225.0 call signalling.
<b>endpointAlias</b> – alias(s) for endpoint.
<b>callReferenceValue</b> – Q.931 CRV of that call that the response is about.
<b>conferenceID</b> – unique conference identifier.
<b>originator</b> – if TRUE the endpoint being queried was the call originator, if FALSE the endpoint was the call destination.
<b>callType</b> – provides information on call topology.
<b>bandwidth</b> – current usage in increments of 100 bit/s; includes only audio and video excluding headers and overhead.

## 7.16 Non-Standard message

The **NonStandardMessage** structure is as follows:

```
NonStandardMessage ::= SEQUENCE
{
    requestSeqNum      RequestSeqNum,
    nonStandardData    NonStandardParameter,
    ...
}
```

## 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.

```
UnknownMessageResponse ::= SEQUENCE -- XRS
{
    requestSeqNum      RequestSeqNum,
    ...
}

END -- of ASN.1
```

## 8 Mechanisms for maintaining QOS

### 8.1 General approach and assumptions

Transport QOS (Quality of Service) on a LAN includes such characteristics as:

- Bit error rate.
- Packet loss rate.
- Delay.

# Superseded by a more recent version

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 LAN 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 multi-media load. The assumption is made that all LAN multimedia terminals are H.323 terminals, and all will attempt to reduce LAN usage as congestion rises rather than "steal" bandwidth from each other.

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

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 H.225.0. 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 H.225.0 to measure QOS:

- 1) Fraction Lost.
- 2) The cumulative packets lost.
- 3) The extended highest sequence number received.
- 4) Interarrival jitter.

## Superseded by a more recent version

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 LAN 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 LAN 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 LAN 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 LAN. 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 LAN→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.

## Superseded by a more recent version

- 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 LAN 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/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 the 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 earphones. 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 [10] to ISO/IEC 10646-1, which also appears in the H.225.0 references clause. In some cases a reference will appear to Appendix I; such references are for informative purposes only. All details required to implement H.323/H.225.0 are contained in this Annex and other related annexes and documents 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/H.225.0.

Readers should also note that the terminology used in this Annex differs somewhat from that used in H.323/H.225.0 according to the following table:

H.323/H.225.0 term	Annex A (RTP/RTCP) term
Media stream	Data
Transport address	Transport address
LAN address	Network address
TSAP identifier	Port
Annex A	Specification or document
Shall	Must
Should	Should

## Superseded by a more recent version

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 sections shall be considered informative.

Finally, implementors are reminded to implement RTP only as described in H.225.0, including Annexes A, B, and C, which contain details and clarifications relevant to H.323/H.225.0. In all cases, the text of H.225.0 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 document defines RTP, consisting of two closely-linked parts:

- The Real-time Transport Protocol (RTP), 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 set-up, 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 document.

RTP represents a new style of protocol following the principles of application level framing and integrated layer processing proposed by Clark and Tennenhouse [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 document 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.



## Superseded by a more recent version

Therefore, in addition to this document, 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

## Superseded by a more recent version

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 document.

## Superseded by a more recent version

**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" [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).

## Superseded by a more recent version

**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 document.

### 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 [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.

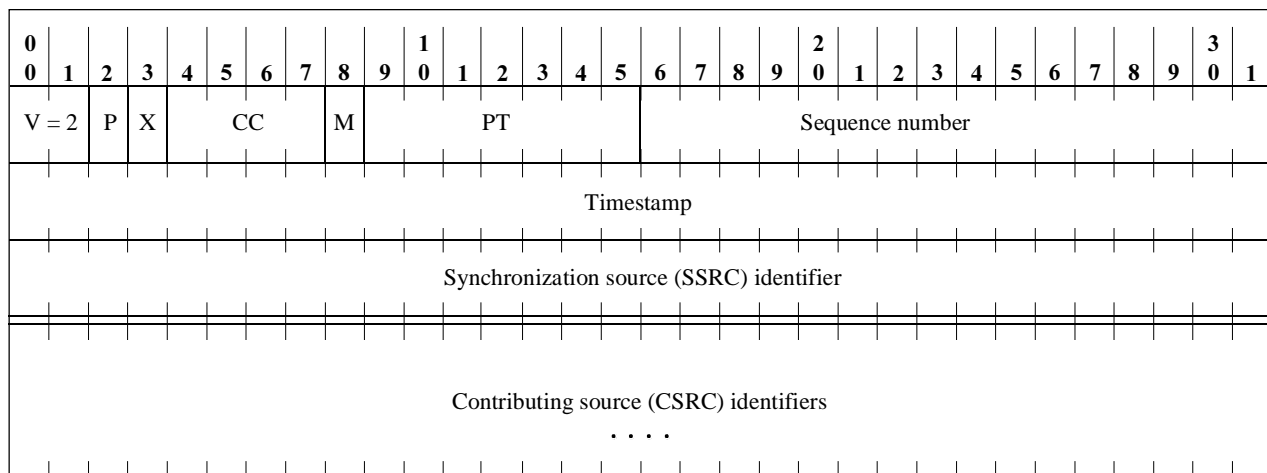
Wallclock time (absolute time) is represented using the timestamp format of the Network Time Protocol (NTP), which is in seconds relative to 0h UTC on 1 January 1900 [4]. The full resolution NTP timestamp is a 64-bit unsigned fixed-point number with the integer part in the first 32 bits and the fractional 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.

# Superseded by a more recent version

## A.5 RTP Data Transfer Protocol

### A.5.1 RTP Fixed header fields

The RTP header has the following format:



T1527560-97

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 [5].

## Superseded by a more recent version

**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 Appendix I. 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 [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

## Superseded by a more recent version

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.

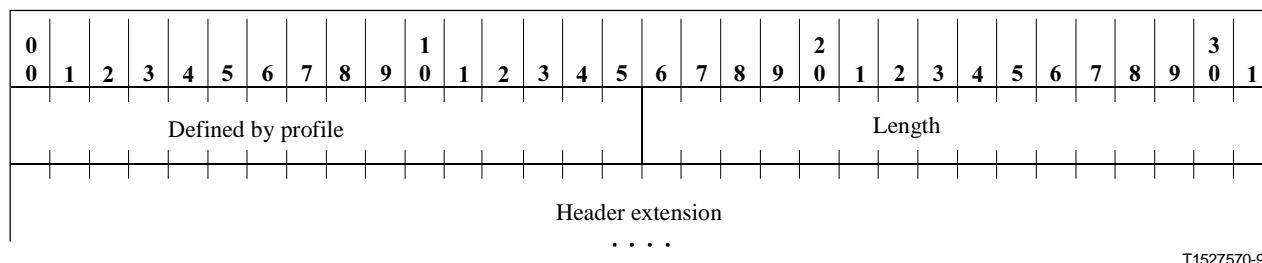
If it turns out that additional functionality is needed in common across all profiles, then a new version of RTP should be defined to make a permanent change to the fixed header.

#### 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.

## Superseded by a more recent version

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 [6] and [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 (see A.6.4.1, CNAME: Canonical end-point identifier 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



## Superseded by a more recent version

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 document, may be needed.

Functions 1 to 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 mis-addressed RTP data packets or other unrelated packets.

# Superseded by a more recent version

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 Appendix I. 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).

## A.6.2 RTCP transmission interval

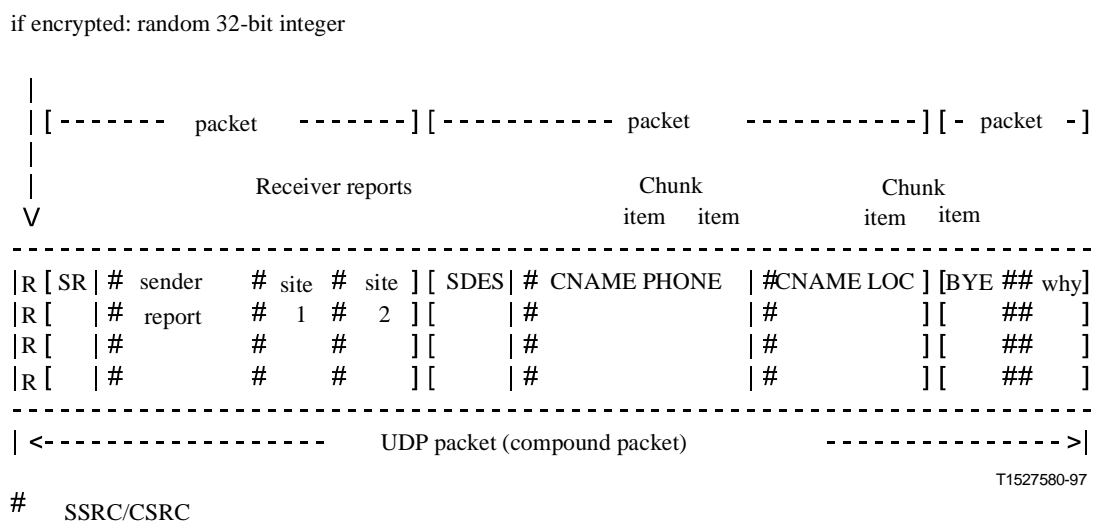


Figure A.1/H.225.0 – Example of an RTCP compound packet

## Superseded by a more recent version

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 Appendix I 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 [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.

## Superseded by a more recent version

- 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 Appendix I). 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 (SDS) 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 SDS 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 SDS 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 SDS 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

## Superseded by a more recent version

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

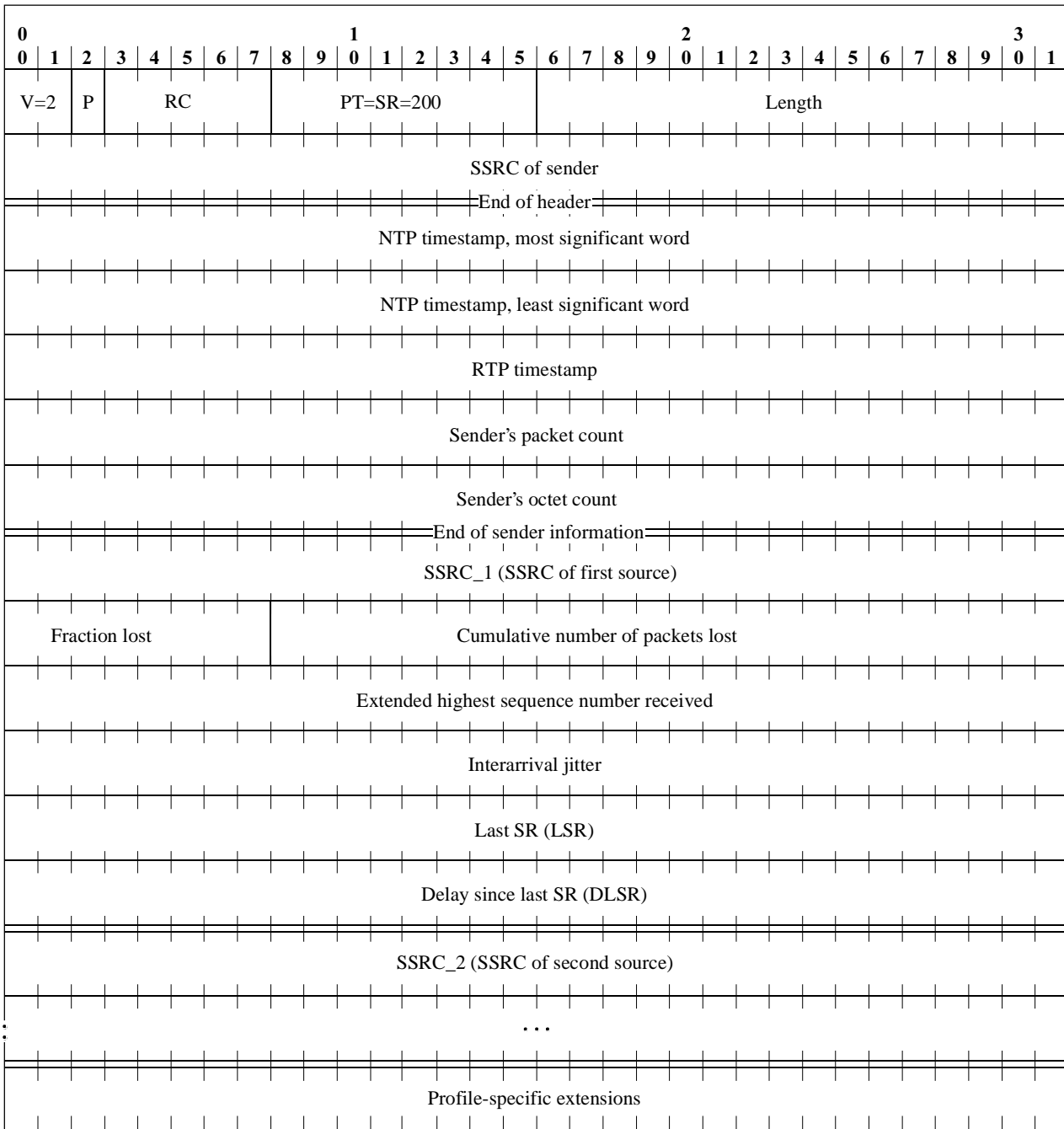
RTP receivers provide reception quality feedback using RTCP report packets which may take one of two forms depending upon whether or not the receiver is also a sender. The only difference between the Sender Report (SR) and Receiver Report (RR) forms, besides the packet type code, is that the sender report 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 is given in A.7, RTP Translators and Mixers.

# Superseded by a more recent version

## A.6.3.1 SR: Sender Report RTCP packet



T1527590-97

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.

## Superseded by a more recent version

**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 Appendix I. If the loss is negative

## Superseded by a more recent version

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 Appendix I.

**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 Appendix I. 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  $S_i$  is the RTP timestamp from packet  $i$ , and  $R_i$  is the time of arrival in RTP timestamp units for packet  $i$ , then for two packets  $i$  and  $j$ ,  $D$  may be expressed as:

$$D(i,j)=(R_j-R_i)-(S_j-S_i)=(R_j-S_j)-(R_i-S_i)$$

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:

$$J=J+(|D(i-1,i)|-J)/16$$

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 [9]. A sample implementation is shown in Appendix I.

**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



## Superseded by a more recent version

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.

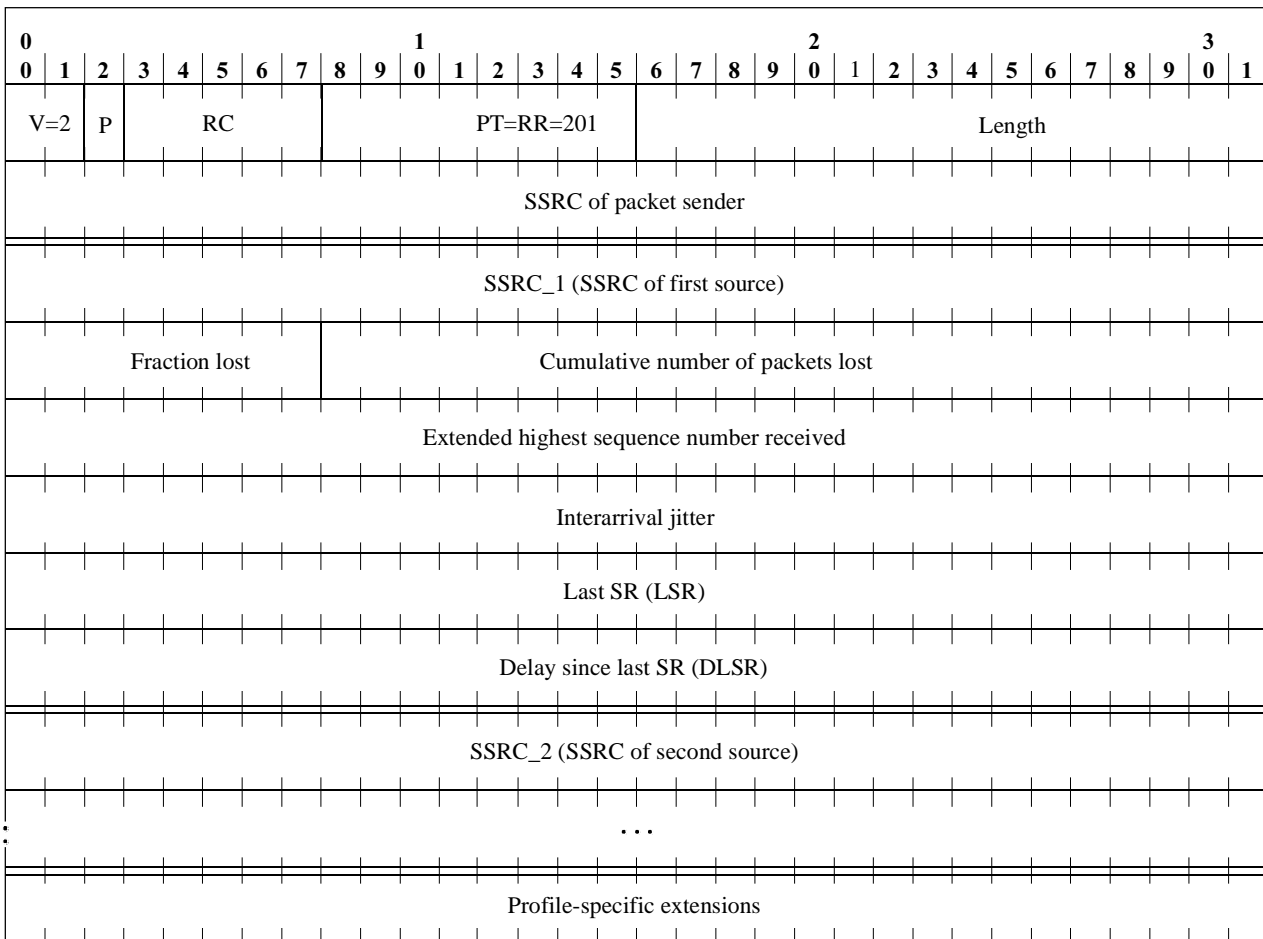
### A.6.3.2 Receiver report RTCP packet

```
[10 Nov 1995 11:33:25.125]          [10 Nov 1995 11:33:36.5]
n                SR(n)                A=b710:8000 (46864.500 s)
----->
                v                    ^
ntp_sec =0xb44db705 v                ^ dlsr=0x0005:4000 ( 5.250s)
ntp_frac=0x20000000 v                ^ lsr =0xb705:2000 (46853.125s)
(3024992016.125 s) v                ^
r                v                    ^ RR(n)
----->
                |<-DLSR->|
                (5.250 s)

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**

## Superseded by a more recent version



T1527600-97

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, that 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.

## Superseded by a more recent version

### A.6.3.4 Analyzing 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.

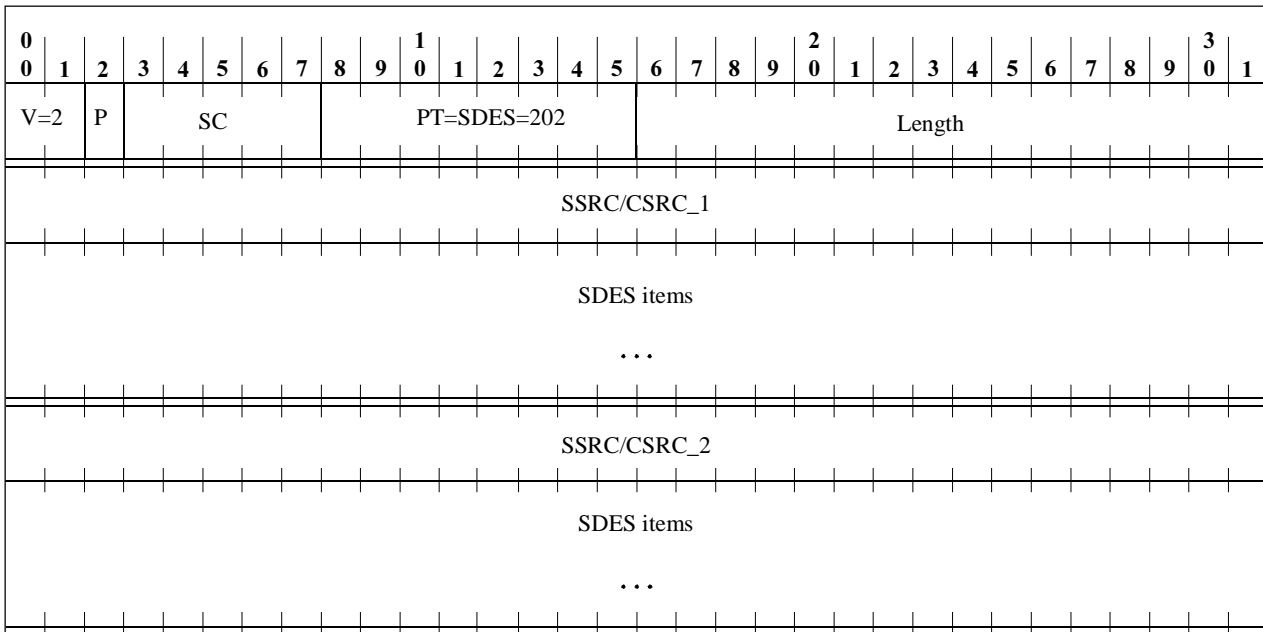
From the sender information, a third-party monitor can calculate the average payload data rate and the average packet rate over an interval without receiving the data. Taking the ratio of the two gives the average payload size. If it can be assumed that packet loss is independent of packet size, then 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.

# Superseded by a more recent version

## A.6.4 SDES: Source description RTCP packet



T1527610-97

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 sections.

**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 [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 #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.

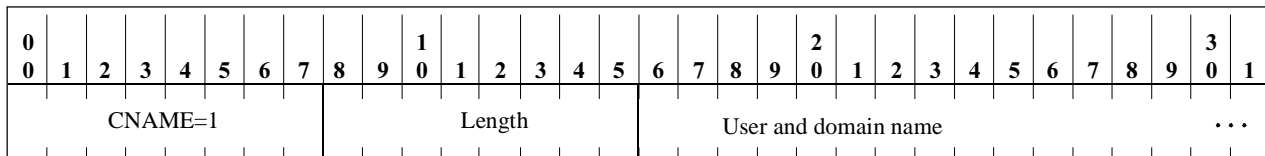
End systems send one SDES packet containing their own source identifier (the same as the SSRC in the fixed RTP header). A mixer sends one SDES packet containing a chunk for each contributing

## Superseded by a more recent version

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



T1527620-97

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 [11], RFC 1035 [12] and 2.1 of RFC 1123 [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

## Superseded by a more recent version

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 [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 [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.

# Superseded by a more recent version

## 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.

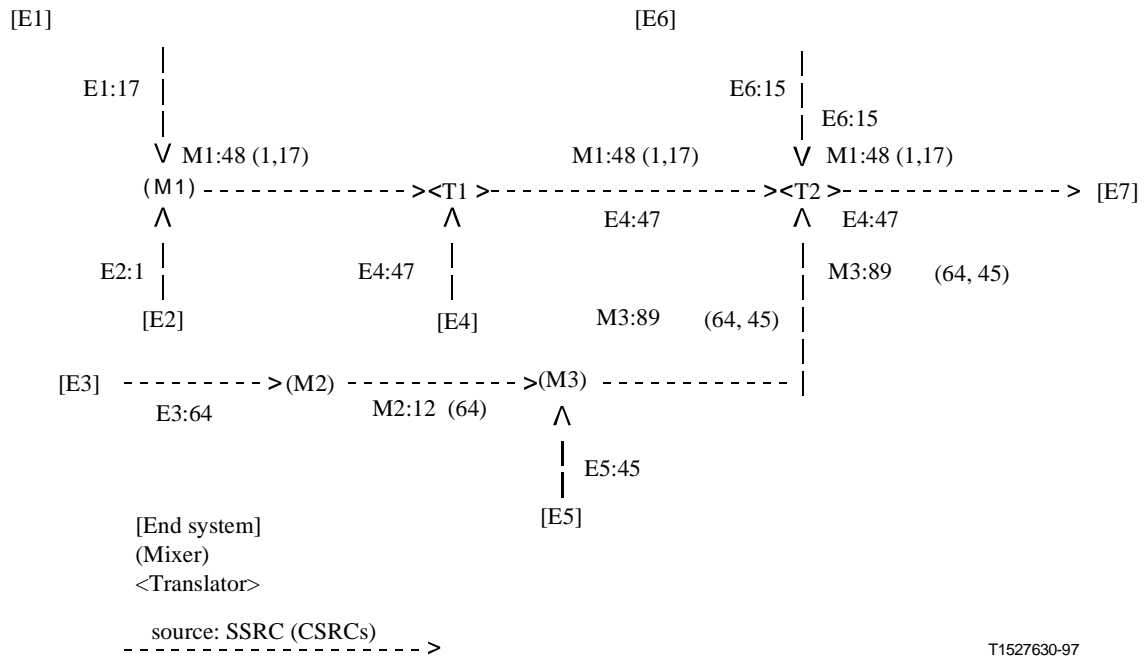
**Mixer:** Receives streams of RTP data packets from one or more sources, possibly changes the data format, combines the streams in some manner and then forwards the combined stream. Since the timing among multiple input sources will not generally be synchronized, the mixer will make timing adjustments 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 do not have

# Superseded by a more recent version

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 can not do inter-media synchronization of the original streams. A multi-media mixer could do it.



**Figure A.3/H.225.0 – Sample RTP network with end systems, mixers and translators**

A collection of mixers and translators is shown in Figure A.3 above to illustrate their effect on SSRC and CSRC identifiers. In the figure, end systems are shown as rectangles (named E), translators as triangles (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.

## 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



## Superseded by a more recent version

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 that 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.

## Superseded by a more recent version

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 cannot 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 Figure A.3. 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 Appendix I.

#### 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  $1 - e^{-\frac{N^2}{2^{L+1}}}$ . 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.

# Superseded by a more recent version

## 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 do not 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 lookup. (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).

## Superseded by a more recent version

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

## Superseded by a more recent version

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.

## Superseded by a more recent version

The RTP payload type (PT) constants are defined in profiles rather than in this document. 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 Appendix I. 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

Abbreviation	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 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

Abbreviation	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.

# Superseded by a more recent version

## 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

## Superseded by a more recent version

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 Appendix II 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 Appendix I algorithms

This appendix 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.

- [1] CLARK (D.D.), TENNENHOUSE (D.L.): Architectural considerations for a new generation of protocols, in *SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 200-208, IEEE, Philadelphia, Pennsylvania, Sept. 1990, *Computer Communications Review*, Vol. 20 (4), Sept. 1990. (Informative reference)
- [2] COMER (D.E.), *Internetworking with TCP/IP*, Vol. 1, Prentice Hall, Englewood Cliffs, New Jersey, 1991.
- [3] POSTEL (J.): Internet protocol, RFC 791, *Internet Engineering Task Force*, Sept. 1981.
- [4] MILLS (D.): *Network time protocol (v3)*, RFC 1305, *Internet Engineering Task Force*, April 1992.
- [5] EASTLAKE (D.), CROCKER (S.), SCHILLER (J.): Randomness recommendations for security, RFC 1750, *Internet Engineering Task Force*, Dec. 1994.
- [6] BOLOT (J.-C.), TURLETTI (T.), 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, England) Aug. 1994.
- [7] BUSSE (I.), DEFFNER (B.), SCHULZRINNE (H.): Dynamic QOS control of multimedia applications based on RTP, *Computer Communications*, Jan. 1996.
- [8] FLOYD (S.), 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), Sept. 1993.
- [9] CADZOW (J.A.): *Foundations of digital signal processing and data analysis* New York, Macmillan, New York, 1987.



## Superseded by a more recent version

- [10] ISO/IEC 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and basic Multilingual Plane*.
- [11] MOCKAPETRIS (P.): Domain names – concepts and facilities, STD 13, RFC 1034, *Internet Engineering Task Force*, Nov. 1987.
- [12] MOCKAPETRIS (P.): Domain names – implementation and specification, STD 13, RFC 1035, *Internet Engineering Task Force*, Nov. 1987.
- [13] BRADEN (R.): Requirements for internet hosts – application and support, STD 3, RFC 1123, *Internet Engineering Task Force*, Oct. 1989.
- [14] REKHTER (Y.), MOSKOWITZ (R.), KARREBERG (D.), de GROOT (G.): Address allocation for private internets, RFC 1597, *Internet Engineering Task Force*, March 1994.
- [15] LEAR (E.), FAIR (E.), CROCKER (D.), KESSLER (T.): Network 10 considered harmful (some practices shouldn't be codified), RFC 1627, *Internet Engineering Task Force*, July 1994.
- [16] CROCKER (D.): Standard for the format of ARPA internet text messages, STD 11, RFC 822, *Internet Engineering Task Force*, Aug. 1982.
- [17] FELLER (W.): An Introduction to Probability Theory and its Applications, *John Wiley and Sons*, Volume 1 , third ed., New York 1968.
- [18] BALENSON (D.): Privacy enhancement for internet electronic mail: Part III: algorithms, modes, and identifiers, RFC 1423, *Internet Engineering Task Force*, Feb. 1993.
- [19] VOYDOCK (V.L.), KENT (S.T.): Security mechanisms in high-level network protocols, *ACM Computing Surveys* , Vol. 15, pp. 135-171, June 1983.
- [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.

# Superseded by a more recent version

## B.2 RTP and RTCP Packet Forms and Protocol Behaviour

The section "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 and 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 memo (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.

# Superseded by a more recent version

## 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 center
- S surround
- F front
- R rear

Channels	Description	Channel					
		1	2	3	4	5	6
2	Stereo  Quadrasonic	l	r				
3		l	r	c			
4		F1	Fr	R1	Rr		
4		l	c	r	S		
5		F1	Fr	Fc	S1	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, and B.4.3, Guidelines for Frame-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 multi-channel audio, but may choose to only play a single channel.

## Superseded by a more recent version

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 the H.323 Recommendation.

# Superseded by a more recent version

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	

## B.4.4.1 G722

G722 is specified in Recommendation G.722.

## B.4.4.2 G728

G728 is specified in Recommendation G.728.

## 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 codings are not part of the H.323 Recommendation.

**H261:** the encoding is specified in Recommendation H.261. The packetization and RTP-specific properties are described in Annex C.

## 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 document. 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.

## Superseded by a more recent version

All current video encodings use a timestamp frequency of 90 000 Hz, the same as the MPEG presentation timestamp 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
15	G728	A	8 000	
31	H261	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.

# Superseded by a more recent version

## 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 [2] in C.4 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 memo 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 (GOBs). Note that H.261 "pictures" are referred as "frames" in this document. Each GOB holds a set of 3 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" color difference components at a resolution of only 8×8 pixels. These components and the codes representing their sampled values are as defined in the ITU-R Recommendation 601 [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

## Superseded by a more recent version

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 may be the same on successive packets if a video image 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),

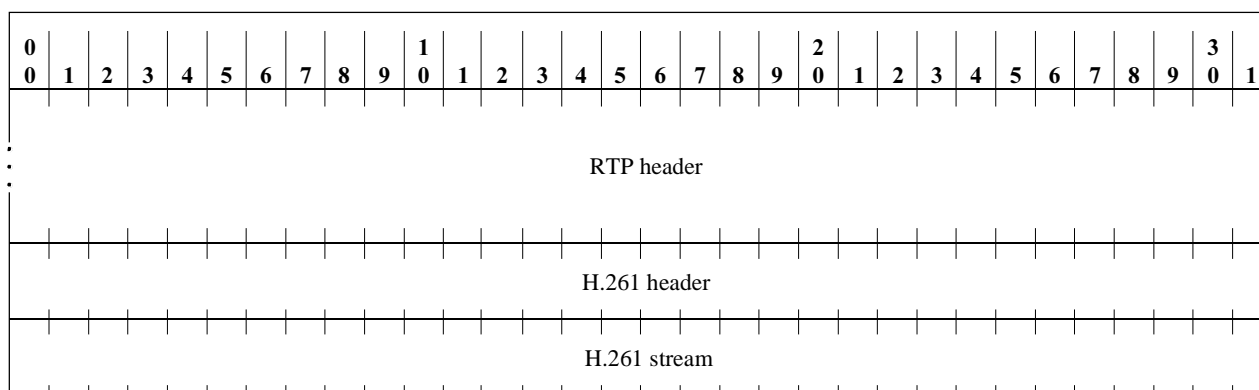


## Superseded by a more recent version

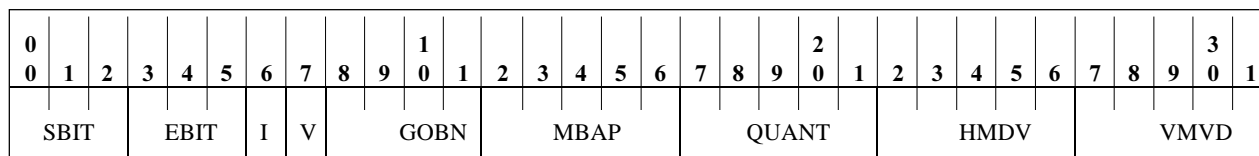
it is necessary to calculate display times for the frames after the first using the timing information in the H.261 frame header. This is required because the RTP timestamp only gives the display time of the first frame in the packet.

- 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:



T1527640-97

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

## Superseded by a more recent version

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 (MQANT 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. The H.261 Recommendation specifies that a MB is INTRA-frame encoded at least every 132 times it is transmitted. However, the INTRA-

## Superseded by a more recent version

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.

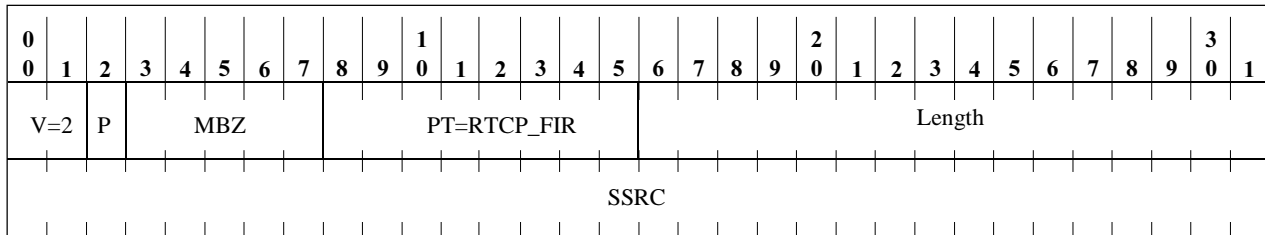
Some reliable multicast protocols use similar NACK control packets transmitted over the normal multicast distribution channel, but they typically use random delays to prevent a NACK implosion problem. The goal of such protocols is to provide reliable multicast packet delivery at the expense of delay, 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, for example, 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.

# Superseded by a more recent version

## C.3.5 Control packets definition

### C.3.5.1 Full INTRA-frame Request (FIR) packet

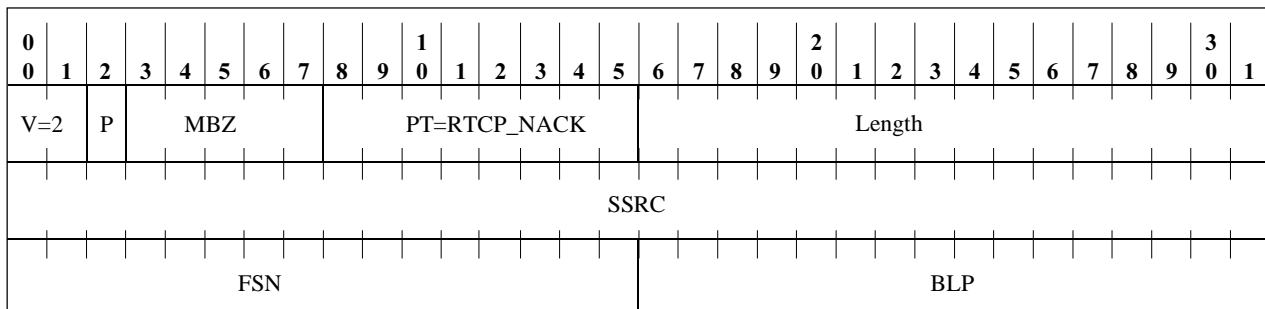


T1527650-97

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 follow:



T1527660-97

The various fields T, P, PT, length and SSRC are defined in the RTP specification (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 0x0001 if the packet corresponding to the FSN and the following packet have been lost, etc.

## C.4 Bibliography

- [1] ITU-T Recommendation H.221 (1995), *Frame structure for a 64 to 1920 kbit/s channel in audiovisual services*.
- [2] ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at  $p \times 64$  kbit/s*.

# Superseded by a more recent version

[3] ITU-R Recommendation 601 (1986), *Digital methods of transmitting television information*.

## ANNEX D

### RTP payload format for H.261A video streams

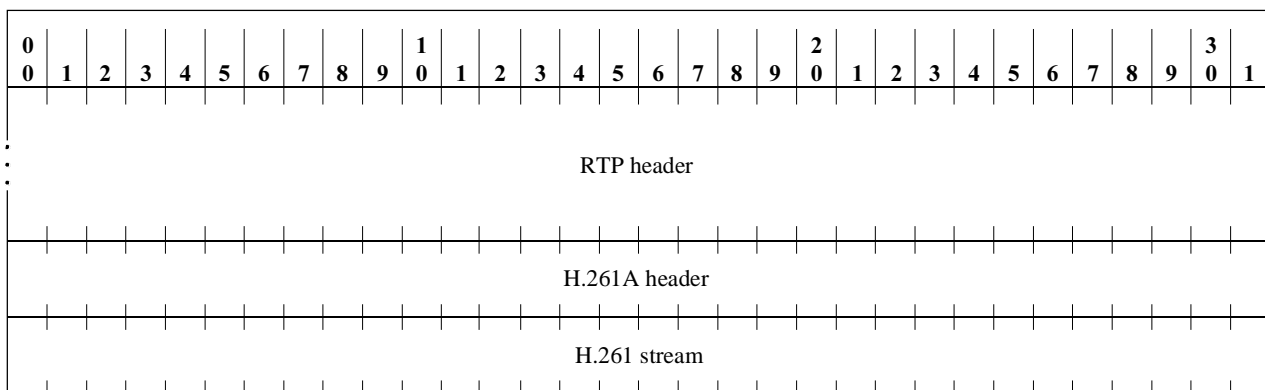
#### D.1 Introduction

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.

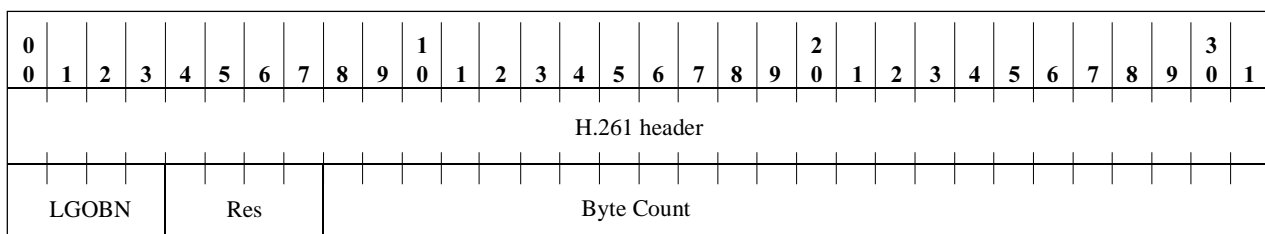
#### D.2 H.261A RTP packetization

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.261 header is defined as:



T1527670-97

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 H.261).

# Superseded by a more recent version

**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.

## APPENDIX I

### RTP/RTCP

The referenced informative material may be found in the following proposed Internet Standard:

- SCHULZRINNE (H.), CASNER (S.), FREDERICK (R.), 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 XXX, *Internet Engineering Task Force*, 1996.

## APPENDIX IV

This Appendix provides additional details concerning the operation of H.225.0 on various actual LAN protocol stacks. This Appendix is non-normative. LANs used in H.225.0 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.

## Superseded by a more recent version

<b>Unreliable delivery applications</b>	<b>Call signalling and H.245 channel</b>
UDP	TPKT —— TCP
IP	
Link layer	
Physical layer	

### IV.1.1 Discovering the Gatekeeper

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

### 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.

<b>Unreliable delivery applications</b>	<b>H.245 channel call signalling channel</b>
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 H.225.0/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	eeee
-------------------------	------

## **Superseded by a more recent version**

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                      bbbb

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.



# Superseded by a more recent version

## ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Telephone network and ISDN
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media
<b>Series H</b>	<b>Transmission of non-telephone signals</b>
Series I	Integrated services digital network
Series J	Transmission of sound-programme and television signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	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
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminal equipments and protocols for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communication
Series Z	Programming languages