

Superseded by a more recent version



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.245

(03/96)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS
Infrastructure of audiovisual services – Communication
procedures

Control protocol for multimedia communication

ITU-T Recommendation H.245
Superseded by a more recent version

(Previously CCITT Recommendation)

Superseded by a more recent version

ITU-T H-SERIES RECOMMENDATIONS AUDIOVISUAL AND MULTIMEDIA SYSTEMS

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

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

Superseded by a more recent version

FOREWORD

The ITU-T (Telecommunication Standardization Sector) is a permanent organ of the International Telecommunication Union (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 (Helsinki, March 1-12, 1993).

ITU-T Recommendation H.245 was prepared by ITU-T Study Group 15 (1993-1996) and was approved under the WTSC Resolution No. 1 procedure on the 19th of March 1996.

NOTE

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

© 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

	<i>Page</i>
1 Scope	1
2 References	1
3 Definitions	3
4 Abbreviations	4
5 General	5
5.1 Master-slave determination	5
5.2 Capability exchange	6
5.3 Logical channel signalling procedures	6
5.4 Receive terminal close logical channel request	7
5.5 H.223 multiplex table entry modification	7
5.6 Audiovisual and data mode request	7
5.7 Round-trip delay determination	7
5.8 Maintenance loops	7
5.9 Commands and indications	7
6 Messages – Syntax	7
7 Messages – Semantic definitions	33
7.1 Master Slave Determination messages	33
7.1.1 Master Slave Determination	33
7.1.2 Master Slave Determination Acknowledge	33
7.1.3 Master Slave Determination Reject	33
7.1.4 Master Slave Determination Release	33
7.2 Terminal capability messages	33
7.2.1 Overview	33
7.2.2 Terminal Capability Set	34
7.2.3 Terminal Capability Set Acknowledge	42
7.2.4 Terminal Capability Set Reject	42
7.2.5 Terminal Capability Set Release	43
7.3 Logical channel signalling messages	43
7.3.1 Open Logical Channel	43
7.3.2 Open Logical Channel Acknowledge	45
7.3.3 Open Logical Channel Reject	45
7.3.4 Open Logical Channel Confirm	45
7.3.5 Close Logical Channel	46
7.3.6 Close Logical Channel Acknowledge	46
7.3.7 Request Channel Close	46
7.3.8 Request Channel Close Acknowledge	47
7.3.9 Request Channel Close Reject	47
7.3.10 Request Channel Close Release	47
7.4 Multiplex Table signalling messages	47
7.4.1 Multiplex Entry Send	47
7.4.2 Multiplex Entry Send Acknowledge	47
7.4.3 Multiplex Entry Send Reject	48
7.4.4 Multiplex Entry Send Release	48
7.5 Request Multiplex Table signalling messages	48
7.5.1 Request Multiplex Entry	48
7.5.2 Request Multiplex Entry Acknowledge	48
7.5.3 Request Multiplex Entry Reject	48
7.5.4 Request Multiplex Entry Release	49

Superseded by a more recent version

	<i>Page</i>
7.6	Request Mode messages 49
7.6.1	Request Mode..... 49
7.6.2	Request Mode Acknowledge..... 51
7.6.3	Request Mode Reject..... 51
7.6.4	Request Mode Release 52
7.7	Round Trip Delay messages..... 52
7.7.1	Round Trip Delay request 52
7.7.2	Round Trip Delay response..... 52
7.8	Maintenance Loop messages..... 52
7.8.1	Maintenance Loop request 52
7.8.2	Maintenance Loop Acknowledge..... 52
7.8.3	Maintenance Loop Reject..... 52
7.8.4	Maintenance Loop Command Off..... 53
7.9	Commands 53
7.9.1	Send Terminal Capability Set..... 53
7.9.2	Encryption 53
7.9.3	Flow Control 53
7.9.4	End session..... 54
7.9.5	Miscellaneous Command 54
7.10	Indications 55
7.10.1	Function not Supported 55
7.10.2	Miscellaneous Indication..... 55
7.10.3	Jitter Indication..... 56
7.10.4	H.223 Skew Indication..... 57
7.10.5	New ATM Virtual Channel Indication..... 57
7.10.6	User Input..... 57
8	Procedures 57
8.1	Introduction..... 57
8.1.1	Method of specification..... 58
8.1.2	Communication between protocol entity and protocol user 58
8.1.3	Peer-to-peer communication 58
8.1.4	SDL diagrams..... 58
8.1.5	SDL key..... 59
8.2	Master slave determination procedures..... 60
8.2.1	Introduction 60
8.2.2	Communication between the MSDSE and the MSDSE user 61
8.2.3	Peer-to-peer MSDSE communication 62
8.2.4	MSDSE procedures 63
8.3	Capability exchange procedures 65
8.3.1	Introduction 65
8.3.2	Communication between CESE and CESE user 70
8.3.3	Peer-to-peer CESE communication..... 72
8.3.4	CESE procedures..... 73
8.4	Unidirectional Logical Channel signalling procedures..... 78
8.4.1	Introduction 78
8.4.2	Communication between the LCSE and the LCSE user 80
8.4.3	Peer-to-peer LCSE communication..... 82
8.4.4	LCSE procedures..... 83
8.5	Bidirectional Logical Channel signalling procedures 91
8.5.1	Introduction 91
8.5.2	Communication between the B-LCSE and the B-LCSE user 93
8.5.3	Peer-to-peer B-LCSE communication..... 96
8.5.4	B-LCSE procedures..... 97
8.6	Close Logical Channel procedures 106
8.6.1	Introduction 106
8.6.2	Communication between CLCSE and CLCSE user..... 107
8.6.3	Peer-to-peer CLCSE communication 109
8.6.4	CLCSE procedures 109

Superseded by a more recent version

Page

8.7	H.223 Multiplex Table procedures	114
8.7.1	Introduction	114
8.7.2	Communication between MTSE and MTSE user	116
8.7.3	Peer-to-peer MTSE communication.....	118
8.7.4	MTSE procedures.....	119
8.8	Request Multiplex Entry procedures.....	120
8.8.1	Introduction	120
8.8.2	Communication between RMESE and RMESE user	126
8.8.3	Peer-to-peer RMESE communication	128
8.8.4	RMESE procedures	128
8.9	Mode Request procedures	130
8.9.1	Introduction	130
8.9.2	Communication between MRSE and MRSE user	133
8.9.3	Peer-to-peer MRSE communication.....	136
8.9.4	MRSE procedures	136
8.10	Round trip delay procedures	142
8.10.1	Introduction	142
8.10.2	Communication between the RTDSE and the RTDSE user.....	143
8.10.3	Peer-to-peer RTDSE communication.....	144
8.10.4	RTDSE procedures.....	145
8.11	Maintenance Loop procedures	147
8.11.1	Introduction	147
8.11.2	Communication between the MLSE and the MLSE user.....	148
8.11.3	Peer-to-peer MLSE communication.....	150
8.11.4	MLSE procedures.....	151
	Annex A – Object identifier assignments.....	158
	Appendix I – Overview of ASN.1 syntax.....	159
I.1	Introduction to ASN.1	159
I.2	Basic ASN.1 data types.....	159
I.3	Aggregate data types.....	161
I.4	Object Identifier type	161
	Appendix II – Examples of H.245 procedures.....	162
II.1	Introduction.....	162
II.2	Master Slave Determination Signalling Entity.....	162
II.3	Capability Exchange Signalling Entity	167
II.4	Logical Channel Signalling Entity	169
II.5	Close Logical Channel Signalling Entity	172
II.6	Multiplex Table Signalling Entity.....	173
II.7	Mode Request Signalling Entity	175
II.8	Round Trip Delay Signalling Entity.....	177
	Appendix III – Summary of procedure timers and counters.....	180
III.1	Timers	180
III.2	Counters	180

Superseded by a more recent version

SUMMARY

This Recommendation specifies syntax and semantics of terminal information messages as well as procedures to use them for in-band negotiation at the start of or during communication. The messages cover receiving and transmitting capabilities as well as mode preference from the receiving end, logical channel signalling, and Control and Indication. Acknowledged signalling procedures are specified to ensure reliable audiovisual and data communication.

Superseded by a more recent version

Recommendation H.245

CONTROL PROTOCOL FOR MULTIMEDIA COMMUNICATION

(Geneva, 1996)

1 Scope

This Recommendation specifies syntax and semantics of terminal information messages as well as procedures to use them for in-band negotiation at the start of or during communication. The messages cover receiving and transmitting capabilities as well as mode preference from the receiving end, logical channel signalling, and Control and Indication. Acknowledged signalling procedures are specified to ensure reliable audiovisual and data communication.

This Recommendation covers a wide range of applications, including storage/retrieval, messaging and distribution services as well as conversational. It applies to, but is not limited to, multimedia systems that use the multiplexes defined in Recommendations H.222.0 and H.223. These different systems share the same syntax and semantics, and are therefore bit-wise compatible. Some of the procedures are applicable to all systems, while the others are more specific to particular systems.

The different systems that make use of this Recommendation may specify the use of different transport protocols. However, it is intended to be used with a reliable transport layer, that is, one that provides guaranteed delivery of correct data.

NOTE – There should be no confusion with the T.120 management system, which is carried within the data stream, and covers different functionalities from those described here – the H.245 stream and the T.120-data stream are complementary.

2 References

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

- [1] CCITT Recommendation G.711 (1988), *Pulse Code Modulation (PCM) of voice frequencies*.
- [2] CCITT Recommendation G.722 (1988), *7 kHz audio-coding within 64 kbit/s*.
- [3] ITU-T Recommendation G.723.1 (1996), *Speech coders: Dual rate speech coder for multimedia communication transmitting at 5.3 and 6.3 kbit/s*.
- [4] CCITT Recommendation G.728 (1992), *Coding of speech at 16 kbit/s using low-delay code excited linear prediction*.
- [5] ITU-T Recommendation H.221 (1995), *Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices*.
- [6] ITU-T Recommendation H.222.0 (1996) | ISO/IEC 13818-1:1997, *Information technology – Generic coding of moving pictures and associated audio information: Systems*.
- [7] ITU-T Recommendation H.222.1 (1996), *Multimedia multiplex and synchronization for audiovisual communication in ATM environments*.
- [8] ITU-T Recommendation H.223 (1996), *Multiplexing protocol for low bitrate multimedia communication*.

Superseded by a more recent version

- [9] ITU-T Recommendation H.224 (1994), *A real time control protocol for simplex applications using the H.221 LSD/HSD/MLP channels.*
- [10] ITU-T Recommendation H.230 (1995), *Frame-synchronous control and indication signals for audiovisual systems.*
- [11] ITU-T Recommendation H.233 (1995), *Confidentiality system for audiovisual services.*
- [12] ITU-T Recommendation H.234 (1994), *Encryption key management and authentication system for audiovisual services.*
- [13] ITU-T Recommendation H.261 (1993), *Video Codec for audiovisual services at $p \times 64$ kbit/s.*
- [14] ITU-T Recommendation H.262 (1996) | ISO/IEC 13818-2:1997, *Registration of copyright identifier.*
- [15] ITU-T Recommendation H.263 (1996), *Video coding for low bit rate communication.*
- [16] ITU-T Recommendation H.281 (1994), *A far end camera control protocol for videoconferences using H.224.*
- [17] ITU-T Recommendation H.320 (1996), *Narrow-band visual telephone systems and terminal equipment.*
- [18] ITU-T Recommendation H.324 (1996), *Terminal for low bit rate multimedia communication.*
- [19] ITU-T Recommendations I.363.1, I.363.3, I.363.5 (1996), *B-ISDN ATM adaptation layer specification: Type 1 AAL, type 3/4 AAL, type 5 AAL.*
- [20] ITU-T Recommendation Q.2931 (1995), *Broadband Integrated Services Digital Network (B-ISDN) – Digital Subscriber Signalling System No. 2 (DSS 2) – User-network interface (UNI) layer 3 specification for basic call/connection control.*
- [21] ITU-T Recommendation T.30 (1996), *Procedures for document facsimile transmission in the general switched telephone network.*
- [22] CCITT Recommendation T.35 (1991), *Procedure for the allocation of CCITT defined codes for non-standard facilities.*
- [23] CCITT Recommendation T.51 (1992), *Latin based coded character sets for telematic services.*
- [24] ITU-T T.84 (1996) | ISO/IEC 10918-3 (1996), *Information technology – Digital compression and coding of continuous-tone still images – Extensions.*
- [25] ITU-T Recommendation T.120 (1996), *Data protocols for multimedia conferencing.*
- [26] ITU-T Recommendation T.434 (1996), *Binary file transfer format for the telematic services.*
- [27] ITU-T Recommendation V.14 (1993), *Transmission of start-stop characters over synchronous bearer channels.*
- [28] ITU-T Recommendation V.34 (1996), *A modem operating at data signalling rates of up to 28 800 bit/s for use on the general switched telephone network and on leased point-to-point 2-wire telephone-type circuits.*
- [29] ITU-T Recommendation V.42 (1996), *Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion.*
- [30] ITU-T Recommendation X.680 (1994), *Information technology – Abstract Syntax Notation One (ASN.1) – Specification of basic notation.*
- [31] ITU-T Recommendation X.691 (1995), *Information technology – ASN.1 encoding rules – Specification of Packed Encoding Rules (PER).*
- [32] ISO/IEC 3309:1993, *Information technology – Telecommunications and information exchange between systems – High-level Data link Control (HDLC) procedures – Frame structure.*
- [33] ISO/IEC 11172-2:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 2: Video.*
- [34] ISO/IEC 11172-3:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 3: Audio.*

Superseded by a more recent version

- [35] ISO/IEC 13818-3:1995, *Information technology – Generic coding of moving pictures and associated audio information – Part 3: Audio*.
- [36] ISO/IEC 13818-6:1996, *Information technology – Generic coding of moving pictures and associated audio information – Part 6: Digital storage media command and control*.
- [37] ISO/IEC TR 9577 (1993), *Information technology – Telecommunications information exchange between systems – Protocol identification in the network layer*.

3 Definitions

For the purposes of this Recommendation, the following definitions apply:

- 3.1 bi-directional logical channel:** A bi-directional channel consists of a pair of associated transmission path between two terminals, one in each direction of transmission.
- 3.2 capability:** A terminal has a particular capability if it is able to encode and transmit or receive and decode that particular signal.
- 3.3 channel:** A channel is a unidirectional link between two end points.
- 3.4 command:** A command is a message that requires action but no explicit response.
- 3.5 elementary stream:** Elementary Stream is a generic term for a coded video, coded audio or other coded bitstream.
- 3.6 entry:** The word entry is used to refer to elements in sets or tables, such as capability sets and multiplex tables.
- 3.7 in-band:** In-band messages are those that are transported within the channel or logical channel to which they refer.
- 3.8 indication:** An indication is a message that contains information but does not require action or response.
- 3.9 logical channel:** A logical channel is a unidirectional path or bidirectional path for the transmission of information.
- 3.10 logical channel number:** A logical channel number is a number that identifies a single logical channel.
- 3.11 logical channel signalling:** Logical channel signalling is a set of procedures that are used to open and close logical channels.
- 3.12 master terminal:** A master terminal is the terminal that is determined as being the master terminal by the master slave determination procedure defined in this Recommendation, or by some other procedure.
- 3.13 medium type:** A medium type is a single form of information that is presented to a user or the data representing that information: video, audio and text are example Medium Types.
- 3.14 mode:** A mode is a set of elementary streams that a terminal is transmitting, intends to transmit, or would like to receive.
- 3.15 multimedia communication:** Multimedia communication refers to the transmission and/or reception of signals of two or more Medium Types simultaneously.
- 3.16 non-standard:** Not conforming to a national or international standard referenced in this Recommendation.
- 3.17 multipoint:** Multipoint refers to the simultaneous interconnection of three or more terminals to allow communication among several sites through the use of multipoint control units (bridges) that centrally direct the flow of information.
- 3.18 request:** A request is a message that results in action by the remote terminal and requires an immediate response from it.

Superseded by a more recent version

- 3.19 response:** A response is a message that is the response to a request.
- 3.20 session:** A session is a period of communication between two terminals which may be conversational or non-conversational (for example retrieval from a database).
- 3.21 slave terminal:** A slave terminal is the terminal that is determined as being the slave terminal by the master-slave determination procedure defined in this Recommendation, or by some other procedure.
- 3.22 support:** The ability to operate in a given mode, however a requirement to support a mode does not mean that the mode must actually be used at all times: unless prohibited, other modes may be used by mutual negotiation.
- 3.23 terminal:** A terminal is any endpoint and may be a user's terminal or some other communication system such as an MCU or an information server.
- 3.24 unidirectional logical channel:** A unidirectional logical channel is a path for the transmission of a single elementary stream from one terminal to another.

4 Abbreviations

For the purposes of this Recommendation, the following abbreviations apply:

AAL	ATM Adaptation Layer
AL1, 2, 3	H.223 Adaptation Layers 1, 2 and 3
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
B-LCSE	Bi-directional Logical Channel Signalling Entity
CESE	Capability Exchange Signalling Entity
CLCSE	Close Logical Channel Signalling Entity
CIF	Common Intermediate Format (of a video picture: refer to Recommendations H.261 and H.263)
CPCS	Common Part Convergence Sublayer (of ATM Adaptation Layer 5)
DTMF	Dual Tone Multi-Frequency
DSM-CC	Digital Storage Media – Command and Control
GOB	Group of Blocks (of a video picture: refer to Recommendations H.261 and H.263)
GSTN	General Switched Telephone Network
HDLC	High-level Data Link Control
HRD	Hypothetical Reference Decoder (refer to Recommendations H.261 and H.263)
IV	Initialisation Vector (used for encryption: refer to Recommendations H.233 and H.234)
LAPM	Link Access Protocol for Modems
LCSE	Logical Channel Signalling Entity
MCU	Multipoint Control Unit
MLSE	Maintenance Loop Signalling Entity
MPI	Minimum Picture Interval

Superseded by a more recent version

MSDSE	Master Slave Determination Signalling Entity
MTSE	Multiplex Table Signalling Entity
MRSE	Mode Request Signalling Entity
PCR	Program Clock Reference (refer to ITU-T Rec. H.222.0 ISO/CEI 13818-1)
PDU	Protocol Data Unit
PID	Packet Identifier (refer to ITU-T Rec. H.222.0 ISO/CEI 13818-1)
QCIF	Quarter CIF
RMESE	Request Multiplex Entry Signalling Entity
RTDSE	Round Trip Delay Signalling Entity
SDL	Specification and Description Language
SDU	Service Data Unit
SE	Session Exchange message (used for encryption: refer to Recommendations H.233 and H.234)
SQCIF	Sub QCIF
STD	System Target Decoder (refer to ITU-T Rec. H.222.0 ISO/CEI 13818-1)
VC	ATM Virtual channel

5 General

This Recommendation provides a number of different services, some of which are expected to be applicable to all terminals that use it and some that are more specific to particular ones. Procedures are defined to allow the exchange of audiovisual and data capabilities; to request the transmission of a particular audiovisual and data mode; to manage the logical channels used to transport the audiovisual and data information; to establish which terminal is the master terminal and which is the slave terminal for the purposes of managing logical channels; to carry various control and indication signals; to control the bit rate of individual logical channels and the whole multiplex; and to measure the round trip delay, from one terminal to the other and back. These procedures are explained in more detail below.

Following this general introduction, there are clauses detailing the message syntax and semantics and the procedures. The syntax has been defined using ASN.1 notation [30] and the semantics define the meaning of syntax elements as well as providing syntactic constraints that are not specified in the ASN.1 syntax. The procedures clause defines the protocols that use the messages defined in the other sections.

Although not all of the messages and procedures defined in this Recommendation will be applicable to all terminals, no indication of such restrictions is given here. These restrictions are the responsibility of the recommendations that use this Recommendation.

This Recommendation has been defined to be independent of the underlying transport mechanism, but is intended to be used with a reliable transport layer, that is, one that provides guaranteed delivery of correct data.

5.1 Master-slave determination

Conflicts may arise when two terminals involved in a call initiate similar events simultaneously and only one such event is possible or desired, for example, when resources are available for only one occurrence of the event. To resolve such conflicts, one terminal shall act as a master and the other terminal shall act as a slave terminal. Rules specify how the master and slave terminal shall respond at times of conflict.

Superseded by a more recent version

The master-slave determination procedure allow terminals in a call to determine which terminal is the master and which terminal is the slave. The terminal status, once determined, remains constant for the call duration.

5.2 Capability exchange

The capability exchange procedures are intended to ensure that the only multimedia signals to be transmitted are those that can be received and treated appropriately by the receive terminal. This requires that the capabilities of each terminal to receive and decode be known to the other terminal. It is not necessary that a terminal understand or store all in-coming capabilities; those that are not understood, or can not be used shall be ignored, and no fault shall be considered to have occurred.

The total capability of a terminal to receive and decode various signals is made known to the other terminal by transmission of its capability set.

Receive capabilities describe the terminal's ability to receive and process in-coming information streams. Transmitters shall limit the content of their transmitted information to that which the receiver has indicated it is capable of receiving. The absence of a receive capability indicates that the terminal cannot receive (is a transmitter only).

Transmit capabilities describe the terminal's ability to transmit information streams. Transmit capabilities serve to offer receivers a choice of possible modes of operation, so that the receiver may request the mode which it prefers to receive. The absence of a transmit capability indicates that the terminal is not offering a choice of preferred modes to the receiver (but it may still transmit anything within the capability of the receiver).

These capability sets provide for more than one stream of a given medium type to be sent simultaneously. For example, a terminal may declare its ability to receive (or send) two independent H.262 video streams and two independent G.722 audio streams at the same time. Capability messages have been defined to allow a terminal to indicate that it does not have fixed capabilities, but that they depend on which other modes are being used simultaneously. For example, it is possible to indicate that higher resolution video can be decoded when a simpler audio algorithm is used; or that either two low resolution video sequences can be decoded or a single high resolution one. It is also possible to indicate trade-offs between the capability to transmit and the capability to receive.

Non-standard capabilities and control messages may be issued using the NonStandardParameter structure. Note that while the meaning of non-standard messages is defined by individual organizations, equipment built by any manufacturer may signal any non-standard message, if the meaning is known.

Terminals may reissue capability sets at any time.

5.3 Logical channel signalling procedures

An acknowledged protocol is defined for the opening and closing of logical channels which carry the audiovisual and data information. The aim of these procedures is to ensure that a terminal is capable of receiving and decoding the data that will be transmitted on a logical channel at the time the logical channel is opened rather than at the time the first data is transmitted on it and to ensure that the receive terminal is ready to receive and decode the data that will be transmitted on the logical channel before that transmission starts. The OpenLogicalChannel message includes a description of the data to be transported, for example. H.262 MP and ML at 6 Mbit/s. Logical channels should only be opened when there is sufficient capability to receive data on all open logical channels simultaneously.

A part of this protocol is concerned with the opening of bidirectional channels. To avoid conflicts which may arise when two terminals initiate similar events simultaneously, one terminal is defined as the master terminal, and the other as the slave terminal. A protocol is defined to establish which terminal is the master and which is the slave. However, systems that use this Recommendation may specify the procedure specified in this Recommendation or another means of determining which terminal is the master and which is the slave.

Superseded by a more recent version

5.4 Receive terminal close logical channel request

A logical channel is opened and closed from the transmitter side. A mechanism is defined which allows a receive terminal to request the closure of an incoming logical channel. The transmit terminal may accept or reject the logical channel closure request. A terminal may, for example, use these procedures to request the closure of an incoming logical channel which, for whatever reason, cannot be decoded. These procedures may also be used to request the closure of a bidirectional logical channel by the terminal that did not open the channel.

5.5 H.223 multiplex table entry modification

The H.223 multiplex table associates each octet within an H.223 MUX message with a particular logical channel number. The H.223 multiplex table may have up to 15 entries. A mechanism is provided that allows the transmit terminal to specify and inform the receiver of new H.223 multiplex table entries. A receive terminal may also request the retransmission of a multiplex table entry.

5.6 Audiovisual and data mode request

When the capability exchange protocol has been completed, both terminals will be aware of each other's capability to transmit and receive as specified in the capability descriptors that have been exchanged. It is not mandatory for a terminal to declare all its capabilities; it need only declare those that it wishes to be used.

A terminal may indicate its capabilities to transmit. A terminal that receives transmission capabilities from the remote terminal may request a particular mode to be transmitted to it. A terminal indicates that it does not want its transmission mode to be controlled by the remote terminal by sending no transmission capabilities.

5.7 Round-trip delay determination

It may be useful in some applications to have knowledge of the round-trip delay between a transmit terminal and a receive terminal. A mechanism is provided to measure this round-trip delay. This mechanism may also be useful as a means to detect whether the remote terminal is still functioning.

5.8 Maintenance loops

Procedures are specified to establish maintenance loops. It is possible to specify the loop of a single logical channel either as a digital loop or decoded loop, and the loop of the whole multiplex.

5.9 Commands and indications

Commands and indications are provided for various purposes: video/audio, active/inactive signals to inform the user; fast update request for source switching in multipoint applications are some examples. Neither commands nor indications elicit response messages from the remote terminal. Commands force an action at the remote terminal whilst indications merely provide information and do not force any action.

A command is defined to allow the bit rate of logical channels and the whole multiplex to be controlled from the remote terminal. This has a number of purposes: interworking with terminals using multiplexes in which only a finite number of bit rates are available; multipoint applications where the rates from different sources should be matched; and flow control in congested networks.

6 Messages – Syntax

This clause specifies the syntax of messages using the notation defined in ASN.1 [30]. Messages shall be encoded for transmission by applying the packed encoding rules specified in [31] using the basic aligned variant. The first bit in each octet which is transmitted is the most significant bit of the octet as is specified in Recommendation X.691.

Superseded by a more recent version

MULTIMEDIA-SYSTEM-CONTROL DEFINITIONS AUTOMATIC TAGS ::= BEGIN

-- Export all symbols

-- =====
-- Top level Messages
-- =====

```
MultimediaSystemControlMessage ::=CHOICE
{
    request      RequestMessage,
    response     ResponseMessage,
    command      CommandMessage,
    indication   IndicationMessage,
    ...
}
```

-- A RequestMessage results in action and requires an immediate response

```
RequestMessage ::=CHOICE
{
    nonStandard      NonStandardMessage,
    masterSlaveDetermination MasterSlaveDetermination,
    terminalCapabilitySet TerminalCapabilitySet,
    openLogicalChannel OpenLogicalChannel,
    closeLogicalChannel CloseLogicalChannel,
    requestChannelClose RequestChannelClose,
    multiplexEntrySend MultiplexEntrySend,
    requestMultiplexEntry RequestMultiplexEntry,
    requestMode      RequestMode,
    roundTripDelayRequest RoundTripDelayRequest,
    maintenanceLoopRequest MaintenanceLoopRequest,
    ...
}
```

-- A ResponseMessage is the response to a request Message

```
ResponseMessage ::=CHOICE
{
    nonStandard      NonStandardMessage,
    masterSlaveDeterminationAck MasterSlaveDeterminationAck,
    masterSlaveDeterminationReject MasterSlaveDeterminationReject,
    terminalCapabilitySetAck TerminalCapabilitySetAck,
    terminalCapabilitySetReject TerminalCapabilitySetReject,
    openLogicalChannelAck OpenLogicalChannelAck,
    openLogicalChannelReject OpenLogicalChannelReject,
    closeLogicalChannelAck CloseLogicalChannelAck,
    ...
}
```

Superseded by a more recent version

requestChannelCloseAck	RequestChannelCloseAck,
requestChannelCloseReject	RequestChannelCloseReject,
multiplexEntrySendAck	MultiplexEntrySendAck,
multiplexEntrySendReject	MultiplexEntrySendReject,
requestMultiplexEntryAck	RequestMultiplexEntryAck,
requestMultiplexEntryReject	RequestMultiplexEntryReject,
requestModeAck	RequestModeAck,
requestModeReject	RequestModeReject,
roundTripDelayResponse	RoundTripDelayResponse,
maintenanceLoopAck	MaintenanceLoopAck,
maintenanceLoopReject	MaintenanceLoopReject,
...	

}

-- *A CommandMessage requires action, but no explicit response*

CommandMessage	::=CHOICE
{	
nonStandard	NonStandardMessage,
maintenanceLoopOffCommand	MaintenanceLoopOffCommand,
sendTerminalCapabilitySet	SendTerminalCapabilitySet,
encryptionCommand	EncryptionCommand,
flowControlCommand	FlowControlCommand,
endSessionCommand	EndSessionCommand,
miscellaneousCommand	MiscellaneousCommand,
...	
}	

-- *An IndicationMessage is information that does not require action or response*

IndicationMessage	::=CHOICE
{	
nonStandard	NonStandardMessage,
functionNotSupported	FunctionNotSupported,
masterSlaveDeterminationRelease	MasterSlaveDeterminationRelease,
terminalCapabilitySetRelease	TerminalCapabilitySetRelease,
openLogicalChannelConfirm	OpenLogicalChannelConfirm,
requestChannelCloseRelease	RequestChannelCloseRelease,
multiplexEntrySendRelease	MultiplexEntrySendRelease,
requestMultiplexEntryRelease	RequestMultiplexEntryRelease,
requestModeRelease	RequestModeRelease,
miscellaneousIndication	MiscellaneousIndication,
jitterindication	Jitterindication,
h223Skewindication	H223Skewindication,
}	

Superseded by a more recent version

```
newATMVCindication      NewATMVCindication,
userinput               UserinputIndication,
...
}

-- SequenceNumber is defined here as it is used in a number of Messages

SequenceNumber          ::=INTEGER (0..255)

-- =====
-- Non standard Message definitions
-- =====

NonStandardMessage      ::=SEQUENCE
{
  nonStandardData       NonStandardParameter,
  ...
}

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

NonStandardIdentifier   ::=CHOICE
{
  object                OBJECT IDENTIFIER,
  h221NonStandard       SEQUENCE
  {
    t35CountryCode      INTEGER (0..255),           -- country, per T.35
    t35Extension        INTEGER (0..255),           -- assigned nationally
    manufacturerCode    INTEGER (0..65535)         -- assigned nationally
  }
}

-- =====
-- Master-slave determination definitions
-- =====

MasterSlaveDetermination ::=SEQUENCE
{
  terminalType          INTEGER (0..255),
  statusDeterminationNumber
  INTEGER (0..16777215),
  ...
}

MasterSlaveDeterminationAck ::=SEQUENCE
{
  decision              CHOICE
  {
    master              NULL,
    slave               NULL
  },
  ...
}

MasterSlaveDeterminationReject ::=SEQUENCE
{
  cause                CHOICE
  {
    identicalNumbers    NULL,
    ...
  },
  ...
}
```

Superseded by a more recent version

```
MasterSlaveDeterminationRelease ::=SEQUENCE
{
  ...
}

-----
-- Capability exchange definitions
-----

TerminalCapabilitySet ::=SEQUENCE
{
  sequenceNumber          SequenceNumber,

  protocolIdentifier      OBJECT IDENTIFIER,
                          -- shall be set to the value
                          -- {itu-t(0) recommendation(0)h(8) 245 version(0) 1}

  multiplexCapability     MultiplexCapability OPTIONAL,

  capabilityTable         SET SIZE (1..256) OF CapabilityTableEntry OPTIONAL,

  capabilityDescriptors   SET SIZE (1..256) OF CapabilityDescriptor OPTIONAL,
  ...
}

V75 Capability ::=SEQUENCE
{
  audioHeader             BOOLEAN,
  ...
}

CapabilityTableEntry ::=SEQUENCE
{
  capabilityTableEntryNumber CapabilityTableEntryNumber,
  capability                Capability OPTIONAL
}

CapabilityDescriptor ::=SEQUENCE
{
  capabilityDescriptorNumber CapabilityDescriptorNumber,
  simultaneousCapabilities  SET SIZE (1..256) OF AlternativeCapabilitySet OPTIONAL
}

AlternativeCapabilitySet ::=SEQUENCE SIZE (1..256) OF CapabilityTableEntryNumber

CapabilityTableEntryNumber ::=INTEGER (1..65535)

CapabilityDescriptorNumber ::=INTEGER (0..255)

TerminalCapabilitySetAck ::=SEQUENCE
{
  sequenceNumber          SequenceNumber,
  ...
}

TerminalCapabilitySetReject ::=SEQUENCE
{
  sequenceNumber          SequenceNumber,
  cause                   CHOICE
  {
    unspecified           NULL,
    undefinedTableEntryUsed NULL,
    descriptorCapacityExceeded NULL,
    tableEntryCapacityExceeded CHOICE
  }
}
```

Superseded by a more recent version

```
{
  highestEntryNumberProcessed      CapabilityTableEntryNumber,
  noneProcessed                    NULL
},
...
},
...
}
TerminalCapabilitySetRelease      ::=SEQUENCE
{
  ...
}

-- =====
-- Capability exchange definitions: top level capability description
-- =====

Capability                        ::=CHOICE
{
  nonStandard                     NonStandardParameter,

  receiveVideoCapability           VideoCapability,
  transmitVideoCapability          VideoCapability,
  receiveAndTransmitVideoCapability VideoCapability,

  receiveAudioCapability           AudioCapability,
  transmitAudioCapability          AudioCapability,
  receiveAndTransmitAudioCapability AudioCapability,

  receiveDataApplicationCapability DataApplicationCapability,
  transmitDataApplicationCapability DataApplicationCapability,
  receiveAndTransmitDataApplicationCapability DataApplicationCapability,

  h233EncryptionTransmitCapability BOOLEAN,
  h233EncryptionReceiveCapability SEQUENCE
  {
    h233IVResponseTime            INTEGER (0..255),           -- units milliseconds
    ...
  },
  ...
}

-- =====
-- Capability exchange definitions: Multiplex capabilities
-- =====

MultiplexCapability              ::=CHOICE
{
  nonStandard                     NonStandardParameter,
  h222Capability                  H222Capability,
  h223Capability                  H223Capability,
  v76Capability                   V76Capability,
  ...
}

H222Capability                   ::= SEQUENCE
{
  numberOfVCs                     INTEGER (1..256),
  vcCapability                     SET OF VCCapability,
  ...
}
```


Superseded by a more recent version

```

h223MultiplexTableCapability          CHOICE
{
    basic                             NULL,
    enhanced                          SEQUENCE
    {
        maximumNestingDepth          INTEGER (1..15),
        maximumElementListSize      INTEGER (2..255),
        maximumSubElementListSize   INTEGER (2..255),
        ...
    }
},
...
}
V76Capability                        ::=SEQUENCE
{
    suspendResumeCapabilitywAddress   BOOLEAN,
    suspendResumeCapabilitywoAddress  BOOLEAN,
    rejCapability                     BOOLEAN,
    sREJCapability                    BOOLEAN,
    mREJCapability                    BOOLEAN,
    crc8bitCapability                 BOOLEAN,
    crc16bitCapability                BOOLEAN,
    crc32bitCapability                BOOLEAN,
    uihCapability                     BOOLEAN,
    numOfDLCS                         INTEGER (2..8191),
    twoOctetAddressFieldCapability    BOOLEAN,
    loopBackTestCapability            BOOLEAN,
    n401Capability                    INTEGER (1..4095),
    maxWindowSizeCapabilityCapability INTEGER (1..127),
    v75Capability                     V75Capability,
    ...
}

-- =====
-- Capability exchange definitions: Video capabilities
-- =====

VideoCapability                      ::=CHOICE
{
    nonStandard                       NonStandardParameter ,
    h261VideoCapability                H261VideoCapability,
    h262VideoCapability                H262VideoCapability,
    h263VideoCapability                H263VideoCapability,
    is11172VideoCapability             IS11172VideoCapability,
    ...
}

H261VideoCapability                  ::=SEQUENCE
{
    qcifMPI                           INTEGER (1..4) OPTIONAL,      -- units 1/29.97 Hz
    cifMPI                             INTEGER (1..4) OPTIONAL,    -- units 1/29.97 Hz
    temporalSpatialTradeOffCapability  BOOLEAN,
    maxBitRate                         INTEGER (1..19200),          -- units of 100 bit/s
    stillimageTransmission             BOOLEAN,                    -- Annex D of H.261
    ...
}

H262VideoCapability                  ::=SEQUENCE
{
    profileAndLevel-SPatML             BOOLEAN,
    profileAndLevel-MPatLL             BOOLEAN,
    profileAndLevel-MPatML             BOOLEAN,
    profileAndLevel-MPatH-14           BOOLEAN,
    profileAndLevel-MPatHL             BOOLEAN,
}

```

Superseded by a more recent version

```

profileAndLevel-SNRatLL          BOOLEAN,
profileAndLevel-SNRatML          BOOLEAN,
profileAndLevel-SpatialatH-14    BOOLEAN,
profileAndLevel-HPatML           BOOLEAN,
profileAndLevel-HPatH-14         BOOLEAN,
profileAndLevel-HPatHL           BOOLEAN,
videoBitRate                     INTEGER (0.. 1073741823) OPTIONAL, -- units 400 bits/s
vbvBufferSize                    INTEGER (0.. 262143) OPTIONAL,   -- units 16384 bits
samplesPerLine                   INTEGER (0..16383) OPTIONAL,    -- units samples/line
linesPerFrame                    INTEGER (0..16383) OPTIONAL,    -- units lines/frame
framesPerSecond                  INTEGER (0..15) OPTIONAL,       -- frame_rate_code
luminanceSampleRate              INTEGER (0..4294967295) OPTIONAL, -- units samples/s
...
}

```

```

H263VideoCapability              ::=SEQUENCE
{
    sqcifMPI                      INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    qcifMPI                       INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    cifMPI                        INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    cif4MPI                       INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    cif16MPI                      INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    maxBitRate                    INTEGER (1..192400),            -- units 100 bit/s
    unrestrictedVector            BOOLEAN,
    arithmeticCoding              BOOLEAN,
    advancedPrediction            BOOLEAN,
    pbFrames                      BOOLEAN,
    temporalSpatialTradeOffCapability BOOLEAN,
    hrd-B                         INTEGER (0..524287) OPTIONAL,   -- units 128 bits
    bppMaxKb                      INTEGER (0..65535) OPTIONAL,    -- units 1024 bits
    ...
}

```

```

IS11172VideoCapability          ::=SEQUENCE
{
    constrainedBitstream          BOOLEAN,
    videoBitRate                 INTEGER (0..1073741823) OPTIONAL, -- units 400 bits/s
    vbvBufferSize                INTEGER (0..262143) OPTIONAL,   -- units 16 384 bits
    samplesPerLine               INTEGER (0..16383) OPTIONAL,    -- units samples/line
    linesPerFrame                INTEGER (0..16383) OPTIONAL,    -- units lines/frame
    pictureRate                  INTEGER (0..15) OPTIONAL,       -- frame_rate_code
    luminanceSampleRate          INTEGER (0..4294967295) OPTIONAL, -- units samples/sec
    ...
}

```

```

-- =====
-- Capability exchange definitions: Audio capabilities
-- =====

```

```

-- For an H.222 multiplex, the integers indicate the size of the STD buffer in units of 256 octets
-- For an H.223 multiplex, the integers indicate the maximum number of audio frames per AL-SDU

```

```

AudioCapability                ::=CHOICE
{
    nonStandard                  NonStandardParameter,
    g711Alaw64k                 INTEGER (1..256),
    g711Alaw56k                 INTEGER (1..256),
    g711Ul原因64k                INTEGER (1..256),
    g711Ul原因56k                INTEGER (1..256),

    g722-64k                    INTEGER (1..256),
    g722-56k                    INTEGER (1..256),
    g722-48k                    INTEGER (1..256),
}

```

Superseded by a more recent version

g723	SEQUENCE	
{		
maxAl-sduAudioFrames	INTEGER (1..256),	
silenceSuppression	BOOLEAN	
},		
g728	INTEGER (1..256),	
g729	INTEGER (1..256),	
g729Annex A	INTEGER (1..256),	
is11172AudioCapability	IS11172AudioCapability,	
is13818AudioCapability	IS13818AudioCapability,	
...		
}		
IS11172AudioCapability	::=SEQUENCE	
{		
audioLayer1	BOOLEAN,	
audioLayer2	BOOLEAN,	
audioLayer3	BOOLEAN,	
audioSampling32k	BOOLEAN,	
audioSampling44k1	BOOLEAN,	
audioSampling48k	BOOLEAN,	
singleChannel	BOOLEAN,	
twoChannels	BOOLEAN,	
bitRate	INTEGER (1..448),	-- units kbits/s
...		
}		
IS13818AudioCapability	::=SEQUENCE	
{		
audioLayer1	BOOLEAN,	
audioLayer2	BOOLEAN,	
audioLayer3	BOOLEAN,	
audioSampling16k	BOOLEAN,	
audioSampling22k05	BOOLEAN,	
audioSampling24k	BOOLEAN,	
audioSampling32k	BOOLEAN,	
audioSampling44k1	BOOLEAN,	
audioSampling48k	BOOLEAN,	
singleChannel	BOOLEAN,	
twoChannels	BOOLEAN,	
threeChannels2-1	BOOLEAN,	
threeChannels3-0	BOOLEAN,	
fourChannels2-0-2-0	BOOLEAN,	
fourChannels2-2	BOOLEAN,	
fourChannels3-1	BOOLEAN,	
fiveChannels3-0-2-0	BOOLEAN,	
fiveChannels3-2	BOOLEAN,	
lowFrequencyEnhancement	BOOLEAN,	
multilingual	BOOLEAN,	
bitRate	INTEGER (1..1130),	-- units kbits/s
...		
}		

Superseded by a more recent version

=====
-- Capability exchange definitions: Data capabilities
=====

```
DataApplicationCapability ::=SEQUENCE
{
  application CHOICE
  {
    nonStandard NonStandardParameter,
    t120 DataProtocolCapability,
    dsm-cc DataProtocolCapability,
    userData DataProtocolCapability,
    t84 SEQUENCE
    {
      t84Protocol DataProtocolCapability,
      t84Profile T84Profile
    },
    t434 DataProtocolCapability,
    h224 DataProtocolCapability,
    nlpid SEQUENCE
    {
      nlpidProtocol DataModeProtocol,
      nlpidData OCTET STRING
    },
    dsvdControl NULL,
    h222DataPartitioning DataProtocolCapability,
    ...
  },
  maxBitRate INTEGER (0..4294967295), -- units 100 bits/s
  ...
}

DataProtocolCapability ::=CHOICE
{
  nonStandard NonStandardParameter,
  v14buffered NULL,
  v42lapm NULL, -- may negotiate to V.42 bis
  hdlcFrameTunneling NULL,
  h310SeparateVCStack NULL,
  h310SingleVCStack NULL,
  transparent NULL,
  ...
}

T84Profile ::=CHOICE
{
  t84Unrestricted NULL,
  t84Restricted SEQUENCE
  {
    qcif BOOLEAN,
    cif BOOLEAN,
    ccir601Seq BOOLEAN,
    ccir601Prog BOOLEAN,
    hdtvSeq BOOLEAN,
    hdtvProg BOOLEAN,

    g3FacsMH200x100 BOOLEAN,
    g3FacsMH200x200 BOOLEAN,
    g4FacsMMR200x100 BOOLEAN,
    g4FacsMMR200x200 BOOLEAN,
    jbig200x200Seq BOOLEAN,
    jbig200x200Progr BOOLEAN,
    jbig300x300Seq BOOLEAN,
    jbig300x300Prog BOOLEAN,
  }
}
```

Superseded by a more recent version

```
digPhotoLow          BOOLEAN,
digPhotoMedSeq       BOOLEAN,
digPhotoMedProg      BOOLEAN,
digPhotoHighSeq      BOOLEAN,
digPhotoHighProg     BOOLEAN,
...
}
}

-- =====
-- Logical channel signalling definitions
-- =====

-- 'Forward' is used to refer to transmission in the direction from the terminal making the
-- original request for a logical channel to the other terminal, and 'reverse' is used to refer
-- to the opposite direction of transmission, in the case of a bidirectional channel request.

OpenLogicalChannel ::=SEQUENCE
{
    forwardLogicalChannelNumber LogicalChannelNumber,

    forwardLogicalChannelParameters SEQUENCE
    {
        portNumber INTEGER (0..65535) OPTIONAL,
        dataType DataType,
        multiplexParameters CHOICE
        {
            h222LogicalChannelParameters H222LogicalChannelParameters,
            h223LogicalChannelParameters H223LogicalChannelParameters,
            v76LogicalChannelParameters V76LogicalChannelParameters,
            ...
        },
        ...
    },
    -- Used to specify the reverse channel for bidirectional open request

    reverseLogicalChannelParameters SEQUENCE
    {
        dataType DataType,
        multiplexParameters CHOICE
        {
            -- H.222 parameters are never present in reverse direction
            h223LogicalChannelParameters H223LogicalChannelParameters,
            v76LogicalChannelParameters V76LogicalChannelParameters,
            ...
        }OPTIONAL, -- Not present for H.222
        ...
    }OPTIONAL, -- Not present for unidirectional channel request
    ...
}

LogicalChannelNumber ::=INTEGER (1..65535)

V75Parameters ::=SEQUENCE
{
    audioHeaderPresent BOOLEAN
}
}
```


Superseded by a more recent version

v75Parameters	V75Parameters,
...	
}	
V76HDLParameters	::=SEQUENCE
{	
crcLength	CRCLength,
n401	INTEGER (1..127),
loopbackTestProcedure	BOOLEAN,
...	
}	
CRCLength	::=CHOICE
{	
crc8bit	NULL,
crc16bit	NULL,
crc32bit	NULL,
...	
}	
OpenLogicalChannelAck	::=SEQUENCE
{	
forwardLogicalChannelNumber	LogicalChannelNumber,
reverseLogicalChannelParameters	SEQUENCE
{	
reverseLogicalChannelNumber	LogicalChannelNumber
portNumber	INTEGER (0..65535) OPTIONAL,
multiplexParameters	CHOICE
{	
h222LogicalChannelParameters	H222LogicalChannelParameters,
-- H.223 parameters are never present in reverse direction	
...	
} OPTIONAL,	-- Not present for H.223
...	
} OPTIONAL,	-- Not present for unidirectional channel request
...	
}	
}	
OpenLogicalChannelReject	::=SEQUENCE
{	
forwardLogicalChannelNumber	LogicalChannelNumber,
cause	CHOICE
{	
unspecified	NULL,
unsuitableReverseParameters	NULL,
dataTypeNotSupported	NULL,
dataTypeNotAvailable	NULL,
unknownDataType	NULL,
dataTypeALCombinationNotSupported	NULL,
...	
},	
...	
}	
OpenLogicalChannelConfirm	::=SEQUENCE
{	
forwardLogicalChannelNumber	LogicalChannelNumber,
...	
}	
CloseLogicalChannel	::=SEQUENCE
{	
logicalChannelNumber	LogicalChannelNumber,
source	CHOICE
{	
user	NULL,
lcse	NULL
},	
...	
}	

Superseded by a more recent version

```
CloseLogicalChannelAck ::=SEQUENCE
{
    logicalChannelNumber LogicalChannelNumber,
    ...
}

RequestChannelClose ::=SEQUENCE
{
    logicalChannelNumber LogicalChannelNumber,
    ...
}

RequestChannelCloseAck ::=SEQUENCE
{
    logicalChannelNumber LogicalChannelNumber,
    ...
}

RequestChannelCloseReject ::=SEQUENCE
{
    logicalChannelNumber LogicalChannelNumber,
    cause CHOICE
    {
        unspecified NULL,
        ...
    },
    ...
}

RequestChannelCloseRelease ::=SEQUENCE
{
    logicalChannelNumber LogicalChannelNumber,
    ...
}

-- =====
-- H.223 multiplex table definitions
-- =====

MultiplexEntrySend ::=SEQUENCE
{
    sequenceNumber SequenceNumber,
    multiplexEntryDescriptors SET SIZE (1..15) OF MultiplexEntryDescriptor,
    ...
}

MultiplexEntryDescriptor ::=SEQUENCE
{
    multiplexTableEntryNumber MultiplexTableEntryNumber,
    elementList SEQUENCE SIZE (1..256) OF MultiplexElement OPTIONAL
}

MultiplexElement ::=SEQUENCE
{
    type CHOICE
    {
        logicalChannelNumber INTEGER(0..65535),
        subElementList SEQUENCE SIZE (2..255) OF MultiplexElement
    },
    repeatCount CHOICE
    {
        finite INTEGER (1..65535), -- repeats of type
        untilClosingFlag NULL -- used for last element
    }
}
```

Superseded by a more recent version

MultiplexTableEntryNumber	::=INTEGER (1..15)
MultiplexEntrySendAck { sequenceNumber multiplexTableEntryNumber ... }	::=SEQUENCE SequenceNumber, SET SIZE (1..15) OF MultiplexTableEntryNumber,
MultiplexEntrySendReject { sequenceNumber rejectionDescriptions ... }	::=SEQUENCE SequenceNumber, SET SIZE (1..15) OF MultiplexEntryRejectionDescriptions,
MultiplexEntryRejectionDescriptions { multiplexTableEntryNumber cause { unspecifiedCause descriptorTooComplex ... }, ... }	::=SEQUENCE MultiplexTableEntryNumber, CHOICE NULL, NULL,
MultiplexEntrySendRelease { multiplexTableEntryNumber ... }	::=SEQUENCE SET SIZE (1..15) OF MultiplexTableEntryNumber,
RequestMultiplexEntry { entryNumbers ... }	::=SEQUENCE SET SIZE (1..15) OF MultiplexTableEntryNumber,
RequestMultiplexEntryAck { entryNumbers ... }	::=SEQUENCE SET SIZE (1..15) OF MultiplexTableEntryNumber,
RequestMultiplexEntryReject { entryNumbers rejectionDescriptions ... }	::=SEQUENCE SET SIZE (1..15) OF MultiplexTableEntryNumber, SET SIZE (1..15) OF RequestMultiplexEntryRejectionDescriptions,
RequestMultiplexEntryRejectionDescriptions { multiplexTableEntryNumber cause { unspecifiedCause ... }, ... }	::=SEQUENCE MultiplexTableEntryNumber, CHOICE NULL,
RequestMultiplexEntryRelease { entryNumbers ... }	::=SEQUENCE SET SIZE (1..15) OF MultiplexTableEntryNumber,

Superseded by a more recent version

-- Request mode definitions

-- RequestMode is a list, in order of preference, of modes that a terminal would like
-- to have transmitted to it.

```
RequestMode ::=SEQUENCE
{
    sequenceNumber      SequenceNumber,
    requestedModes      SEQUENCE SIZE (1..256) OF ModeDescription,
    ...
}
```

```
RequestModeAck ::=SEQUENCE
{
    sequenceNumber      SequenceNumber,
    response            CHOICE
    {
        willTransmitMostPreferredMode    NULL,
        willTransmitLessPreferredMode    NULL,
        ...
    },
    ...
}
```

```
RequestModeReject ::=SEQUENCE
{
    sequenceNumber      SequenceNumber,
    cause              CHOICE
    {
        modeUnavailable    NULL,
        multipointConstraint    NULL,
        requestDenied      NULL,
        ...
    },
    ...
}
```

```
RequestModeRelease ::=SEQUENCE
{
    ...
}
```

-- Request mode definitions: Mode description

```
ModeDescription ::=SET SIZE (1..256) OF ModeElement
```

```
ModeElement ::= SEQUENCE
{
    type            CHOICE
    {
        nonStandard    NonStandardParameter,
        videoMode      VideoMode,
        audioMode      AudioMode,
        dataMode       DataMode,
        encryptionMode EncryptionMode,
        ...
    },
    h223ModeParameters    H223ModeParameters OPTIONAL,
    ...
}
```

Superseded by a more recent version

```
H223ModeParameters ::=SEQUENCE
{
  adaptationLayerType CHOICE
  {
    nonStandard NonStandardParameter,
    al1Framed NULL,
    al1NotFramed NULL,
    al2WithoutSequenceNumbers NULL,
    al2WithSequenceNumbers NULL,
    al3 SEQUENCE
    {
      controlFieldOctets INTEGER(0..2),
      sendBufferSize INTEGER(0..16777215) -- units octets
    },
    ...
  },
  segmentableFlag BOOLEAN,
  ...
}

=====
-- Request mode definitions: Video modes
=====

VideoMode ::=CHOICE
{
  nonStandard NonStandardParameter,
  h261VideoMode H261VideoMode,
  h262VideoMode H262VideoMode,
  h263VideoMode H263VideoMode,
  is11172VideoMode IS11172VideoMode,
  ...
}

H261VideoMode ::=SEQUENCE
{
  resolution CHOICE
  {
    qcif NULL,
    cif NULL
  },
  bitRate INTEGER(1..19200) -- units 100 bit/s
  stillImageTransmission BOOLEAN,
  ...
}

H262VideoMode ::=SEQUENCE
{
  profileAndLevel CHOICE
  {
    profileAndLevel-SPatML NULL,
    profileAndLevel-MPatLL NULL,
    profileAndLevel-MPatML NULL,
    profileAndLevel-MPatH-14 NULL,
    profileAndLevel-MPatHL NULL,
    profileAndLevel-SNRatLL NULL,
    profileAndLevel-SNRatML NULL,
    profileAndLevel-SpatialatH-14 NULL,
    profileAndLevel-HPatML NULL,
    profileAndLevel-HPatH-14 NULL,
    profileAndLevel-HPatHL NULL,
    ...
  },
  ...
}
```

Superseded by a more recent version

```

videoBitRate          INTEGER(0..1073741823) OPTIONAL,  -- units 400 bits/sec
vbvBufferSize         INTEGER(0..262143) OPTIONAL,    -- units 16 384 bits
samplesPerLine        INTEGER(0..16383) OPTIONAL,    -- units samples/line
linesPerFrame         INTEGER(0..16383) OPTIONAL,    -- units lines/frame
framesPerSecond       INTEGER(0..15) OPTIONAL,      -- frame_rate_code
luminanceSampleRate   INTEGER(0..4294967295) OPTIONAL, -- units samples/s
...
}

H263VideoMode        ::=SEQUENCE
{
    resolution        CHOICE
    {
        sqcif         NULL,
        qcif          NULL,
        cif           NULL,
        cif4          NULL,
        cif16         NULL,
        ...
    },
    bitRate           INTEGER(1..19200),              -- units 100 bit/s
    unrestrictedVector BOOLEAN,
    arithmeticCoding  BOOLEAN,
    advancedPrediction BOOLEAN,
    pbFrames          BOOLEAN,
    ...
}

IS11172VideoMode    ::=SEQUENCE
{
    constrainedBitStream BOOLEAN,
    videoBitRate      INTEGER (0..1073741823) OPTIONAL, -- units 400 bits/s
    vbvBufferSize     INTEGER (0..262143) OPTIONAL,    -- units 16 384 bits
    samplesPerLine    INTEGER (0..16383) OPTIONAL,    -- units samples/line
    linesPerFrame     INTEGER (0..16383) OPTIONAL,    -- units lines/frame
    pictureRate       INTEGER (0..15) OPTIONAL,
    luminanceSampleRate INTEGER (0..4294967295) OPTIONAL, -- units samples/s
    ...
}

-----
-- Request mode definitions: Audio modes
-----

AudioMode            ::=CHOICE
{
    nonStandard       NonStandardParameter,
    g711Alaw64k       NULL,
    g711Alaw56k       NULL,
    g711Ulaw64k       NULL,
    g711Ulaw56k       NULL,

    g722-64k          NULL,
    g722-56k          NULL,
    g722-48k          NULL,

    g728              NULL,
    g729              NULL,
    v76               NULL,
    g723              CHOICE
    {
        noSilenceSuppressionLowRate NULL,
        noSilenceSuppressionHighRate NULL,
        silenceSuppressionLowRate   NULL,
        silenceSuppressionHighRate  NULL
    },
}

```

Superseded by a more recent version

```
is11172AudioMode
is13818AudioMode

...
}

IS11172AudioMode ::=SEQUENCE
{
    audioLayer CHOICE
    {
        audioLayer1 NULL,
        audioLayer2 NULL,
        audioLayer3 NULL
    },
    audioSampling CHOICE
    {
        audioSampling32k NULL,
        audioSampling44k1 NULL,
        audioSampling48k NULL
    },
    multichannelType CHOICE
    {
        singleChannel NULL,
        twoChannelStereo NULL,
        twoChannelDual NULL
    },
    bitRate INTEGER (1..448), -- units kbit/s
    ...
}

IS13818AudioMode ::=SEQUENCE
{
    audioLayer CHOICE
    {
        audioLayer1 NULL,
        audioLayer2 NULL,
        audioLayer3 NULL
    },
    audioSampling CHOICE
    {
        audioSampling16k NULL,
        audioSampling22k05 NULL,
        audioSampling24k NULL,
        audioSampling32k NULL,
        audioSampling44k1 NULL,
        audioSampling48k NULL
    },
    multichannelType CHOICE
    {
        singleChannel NULL,
        twoChannelStereo NULL,
        twoChannelDual NULL,
        threeChannels2-1 NULL,
        threeChannels3-0 NULL,
        fourChannels2-0-2-0 NULL,
        fourChannels2-2 NULL,
        fourChannels3-1 NULL,
        fiveChannels3-0-2-0 NULL,
        fiveChannels3-2 NULL
    },
    lowFrequencyEnhancement BOOLEAN,
```

Superseded by a more recent version

```
multilingual                BOOLEAN,

bitRate                     INTEGER (1..1130),           -- units kbit/s
...
}

=====
-- Request mode definitions: Data modes
=====

DataMode                    ::=SEQUENCE
{
  application                CHOICE
  {
    nonStandard              NonStandardParameter,
    t120                     DataProtocolCapability,
    dsm-cc                   DataProtocolCapability,
    userData                 DataProtocolCapability,
    t84                      DataProtocolCapability,
    t434                    DataProtocolCapability,
    h224                    DataProtocolCapability,
    nIpid                   SEQUENCE
    {
      nIpidProtocol          DataProtocolCapability,
      nIpidData              OCTET STRING
    },
    dsvdControl              NULL,
    h222DataPartitioning    DataProtocolCapability,
    ...
  }
  bitRate                   INTEGER (0..4294967295),     -- units 100 kbit/s
  ...
}

=====
-- Request mode definitions: Encryption modes
=====

EncryptionMode              ::=CHOICE
{
  nonStandard               NonStandardParameter,
  h233Encryption            NULL,
  ...
}

=====
-- Round Trip Delay definitions
=====

RoundTripDelayRequest       ::=SEQUENCE
{
  sequenceNumber            SequenceNumber,
  ...
}

RoundTripDelayResponse      ::=SEQUENCE
{
  sequenceNumber            SequenceNumber,
  ...
}
```

Superseded by a more recent version

-- Maintenance Loop definitions

MaintenanceLoopRequest ::=SEQUENCE
{
 sequenceNumber SequenceNumber,
 type CHOICE
 {
 systemLoop NULL,
 mediaLoop LogicalChannelNumber,
 logicalChannelLoop LogicalChannelNumber,
 ...
 },
 ...
}

MaintenanceLoopAck ::=SEQUENCE
{
 type CHOICE
 {
 systemLoop NULL,
 mediaLoop LogicalChannelNumber,
 logicalChannelLoop LogicalChannelNumber,
 ...
 },
 ...
}

MaintenanceLoopReject ::=SEQUENCE
{
 type CHOICE
 {
 systemLoop NULL,
 mediaLoop LogicalChannelNumber,
 logicalChannelLoop LogicalChannelNumber,
 ...
 },
 cause CHOICE
 {
 canNotPerformLoop NULL,
 ...
 },
 ...
}

MaintenanceLoopOffCommand ::=SEQUENCE
{
 ...
}

-- Command Message definitions

-- Command Message: Send Terminal Capability Set

SendTerminalCapabilitySet ::=CHOICE
{
 specificRequest SEQUENCE
 {
 multiplexCapability BOOLEAN,
 capabilityTableEntryNumbers SET SIZE(1..65535) OF CapabilityTableEntry Number OPTIONAL,
 capabilityDescriptorNumbers SET SIZE(1..256) OF CapabilityDescriptorNumber OPTIONAL,
 ...
 },
}

Superseded by a more recent version

```
genericRequest          NULL,
...
}

-----
-- Command Message: Encryption
-----

EncryptionCommand      ::=CHOICE
{
    encryptionSE        OCTET STRING,          -- per H.233, but no error protection
    encryptionIVRequest NULL,                -- requests new IV
    encryptionAlgorithmID SEQUENCE
    {
        h233AlgorithmIdentifier SequenceNumber,
        associatedAlgorithm      NonStandardParameter
    },
    ...
}

-----
-- Command Message: Flow Control
-----

FlowControlCommand     ::=SEQUENCE
{
    scope                CHOICE
    {
        logicalChannelNumber LogicalChannelNumber,
        virtualChannelID      INTEGER (0..65535),
        wholeMultiplex        NULL
    },
    restriction          CHOICE
    {
        maximumBitRate        INTEGER (0..16777215), -- units 100 kbits/s
        noRestriction          NULL
    },
    ...
}

-----
-- Command Message: Change or End Session
-----

EndSessionCommand      ::=CHOICE
{
    nonStandard          NonStandardParameter,

    disconnect           NULL,
    gstnOptions          CHOICE
    {
        telephonyMode        NULL,
        v8bis                NULL,
        v34DSVD              NULL,
        v34DuplexFAX         NULL,
        v34H324              NULL,
        ...
    },
    ...
}
```

Superseded by a more recent version

=====
-- *Command Message: Miscellaneous H.230-like commands*
=====

```
MiscellaneousCommand ::=SEQUENCE
{
    logicalChannelNumber LogicalChannelNumber,
    type CHOICE
    {
        equaliseDelay NULL, -- same as H.230 ACE
        zeroDelay NULL, -- same as H.230 ACZ
        multipointModeCommand NULL,
        cancelMultipointModeCommand NULL,
        videoFreezePicture NULL,
        videoFastUpdatePicture NULL,
        videoFastUpdateGOB SEQUENCE
        {
            firstGOB INTEGER (0..17),
            numberOfGOBs INTEGER (1..18)
        },
        videoTemporalSpatialTradeOff INTEGER (0..31), -- commands a trade-off value
        videoSendSyncEveryGOB NULL,
        videoSendSyncEveryGOBCancel NULL,
        ...
    },
    ...
}
```

=====
-- *Indication Message definitions*
=====

=====
-- *Indication Message: Function not supported*
=====

-- *This is used to return a complete request, response or command that is not recognized*

```
FunctionNotSupported ::=CHOICE
{
    request RequestMessage,
    response ResponseMessage,
    command CommandMessage
}
```

=====
-- *Indication Message: Miscellaneous H.230-like indication*
=====

```
Miscellaneousindication ::=SEQUENCE
{
    logicalChannelNumber LogicalChannelNumber,
    type CHOICE
    {
        logicalChannelActive NULL, -- same as H.230 AIA and VIA
        logicalChannelInactive NULL, -- same as H.230 AIM and VIS

        multipointConference NULL,
        cancelMultipointConference NULL,

        multipointZeroComm NULL, -- same as H.230 MIZ
        cancelMultipointZeroComm NULL, -- same as H.230 cancel MIZ
    }
}
```

Superseded by a more recent version

```
multipointSecondaryStatus      NULL,                -- same as H.230 MIS
cancelMultipointSecondaryStatus NULL,                -- same as H.230 cancel MIS

videoindicateReadyToActivate   NULL,                -- same as H.230 VIR

videoTemporalSpatialTradeOff   INTEGER (0..31),     -- indicates current trade-off

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

```
=====  
-- Indication Message: Jitter Indication  
=====
```

```
JitterIndication ::=SEQUENCE
{
  scope CHOICE
  {
    logicalChannelNumber LogicalChannelNumber,
    virtualChannelID     INTEGER (0..65535),
    wholeMultiplex       NULL
  },
  estimatedReceivedJitterMantissa INTEGER (0..3),
  estimatedReceivedJitterExponent INTEGER (0..7),
  skippedFrameCount        INTEGER (0..15) OPTIONAL,
  additionalDecoderBuffer   INTEGER (0.. 262143) OPTIONAL, -- 262143 = 2^18 - 1
  ...
}
```

```
=====  
-- Indication Message: H.223 logical channel skew  
=====
```

```
H223SkewIndication ::=SEQUENCE
{
  logicalChannelNumber1 LogicalChannelNumber,
  logicalChannelNumber2 LogicalChannelNumber,
  skew                  INTEGER (0..4095),      -- units milliseconds
  ...
}
```

```
=====  
-- Indication Message: New ATM virtual channel indication  
=====
```

```
NewATMVCindication ::=SEQUENCE
{
  resourceID      INTEGER(0..65535),
  bitRate         INTEGER(1..65535),      -- units 64 kbits/s
  bitRateLockedToPCRClock    BOOLEAN,
  bitRateLockedToNetworkClock  BOOLEAN,
  aal CHOICE
  {
    aal1 SEQUENCE
    {
      clockRecovery CHOICE
      {
        nullClockRecovery NULL,
        srtsClockRecovery NULL,
        adaptiveClockRecovery NULL,
        ...
      },

```

Superseded by a more recent version

```
errorCorrection          CHOICE
{
  nullErrorCorrection    NULL,
  longInterleaver       NULL,
  shortInterleaver      NULL,
  errorCorrectionOnly    NULL,
  ...
},
structuredDataTransfer  BOOLEAN,
partiallyFilledCells    BOOLEAN,
...
},
aal5                    SEQUENCE
{
  forwardMaximumSDUSize INTEGER (0..65535),    -- units octets
  backwardMaximumSDUSize INTEGER (0..65535),    -- units octets
  ...
},
...
},
multiplex              CHOICE
{
  NoMultiplex          NULL,
  transportStream      NULL,
  programStream        NULL,
  ...
},
...
}
```

```
-- =====
-- Indication Message: user input
-- =====
```

```
UserInputIndication    ::=CHOICE
{
  nonStandard           NonStandardParameter,
  alphanumeric          GeneralString,
  ...
}
```

END

Superseded by a more recent version

7 Messages – Semantic definitions

This clause provides semantic definitions and constraints on the syntax elements defined in the previous clause.

MultimediaSystemControlMessage: Is a choice of message types. Messages defined in this Recommendation are classified as request, response, command and indication messages.

RequestMessage: A request message results in an action by the remote terminal and requires an immediate response from it. The nonStandardRequest may be used to send non-standard requests.

ResponseMessage: A response message is the response to a request message. The nonStandardResponse may be used to send non-standard responses.

CommandMessage: A command message requires action but no explicit response. The nonStandardCommand may be used to send non-standard commands.

IndicationMessage: An indication contains information that does not require action or response. The nonStandardIndication may be used to send non-standard indications.

NonStandardParameter: This may be used to indicate a non-standard parameter. It consists of an identity and the actual parameters, which are coded as an octet string.

NonStandardIdentifier: Is used to identify the type of non-standard parameter. It is either an object identifier, or an H.221 type of identifier that is an octet string consisting of exactly four octets which are country code (octet 1 as in T.35 [22]; octet 2*), manufacturer code (next two octets*), *=assigned nationally. The manufacturer codes are the same as those assigned for use in Recommendation H.320 [17].

7.1 Master Slave Determination messages

This set of messages is used by a protocol to determine which terminal is the master terminal and which is the slave terminal.

7.1.1 Master Slave Determination

This is sent from a MSDSE to a peer MSDSE.

terminalType is a number that identifies different types of terminal, such as, terminals, MCUs and gateways. The allocation of values to terminal types is outside the scope of this Recommendation.

statusDeterminationNumber is a random number in the range $0..2^{24}-1$.

7.1.2 Master Slave Determination Acknowledge

This is used to confirm whether the terminal is the master terminal or the slave terminal, as indicated by decision. When decision is of type master, the terminal receiving this message is the master terminal and when decision is of type slave, it is the slave terminal.

7.1.3 Master Slave Determination Reject

This is used to reject the MasterSlaveDetermination message. When the cause is of type identicalNumbers, the rejection was due to both random numbers being equivalent and the terminal types being the same.

7.1.4 Master Slave Determination Release

This is sent in the case of a time-out.

7.2 Terminal capability messages

This set of messages is for the secure exchange of capabilities between the two terminals.

7.2.1 Overview

The transmitting terminal assigns each individual mode the terminal is capable of operating in a number in a capabilityTable. For example, G.723 audio, G.728 audio, and CIF H.263 video would each be assigned separate numbers.

Superseded by a more recent version

These capability numbers are grouped into `AlternativeCapabilitySet` structures. Each `AlternativeCapabilitySet` indicates that the terminal is capable of operating in exactly one mode listed in the set. For example, an `AlternativeCapabilitySet` listing {G.711, G.723.1, G.728} means that the terminal can operate in any one of those audio modes, but not more than one.

These `AlternativeCapabilitySet` structures are grouped into `simultaneousCapabilities` structures. Each `simultaneousCapabilities` structure indicates a set of modes the terminal is capable of using simultaneously. For example, a `simultaneousCapabilities` structure containing the two `AlternativeCapabilitySet` structures {H.261, H.263} and {G.711, G.723.1, G.728} means that the terminal can operate either of the video codecs simultaneously with any one of the audio codecs. The `simultaneousCapabilities` set {{H.261}, {H.261, H.263}, {G.711, G.723.1, G.728}} means the terminal can operate two video channels and one audio channel simultaneously: one video channel per H.261, another video channel per either H.261 or H.263, and one audio channel per either G.711, G.723.1, or G.728.

NOTE – The actual capabilities stored in the `capabilityTable` are often more complex than presented here. For example, each H.263 capability indicates details including ability to support various picture formats at given minimum picture intervals, and ability to use optional coding modes.

The terminal's total capabilities are described by a set of `CapabilityDescriptor` structures, each of which is a single `simultaneousCapabilities` structure and a `capabilityDescriptorNumber`. By sending more than one `CapabilityDescriptor`, the terminal may signal dependencies between operating modes by describing different sets of modes which it can simultaneously use. For example, a terminal issuing two `CapabilityDescriptor` structures, one {{H.261, H.263}, {G.711, G.723.1, G.728}} as in the previous example, and the other {{H.262}, {G.711}}, means the terminal can also operate the H.262 video codec, but only with the low-complexity G.711 audio codec.

Terminals may dynamically add capabilities during a communication session by issuing additional `CapabilityDescriptor` structures, or remove capabilities by sending revised `CapabilityDescriptor` structures. All terminals shall transmit at least one `CapabilityDescriptor` structure.

7.2.2 Terminal Capability Set

This message contains information about the terminal's capability to transmit and receive. It also indicates the version of this Recommendation that is in use. It is sent from an out-going CESE to a peer in-coming CESE.

`sequenceNumber` is used to label instances of `TerminalCapabilitySet` so that the corresponding response can be identified.

`protocolIdentifier` is used to indicate the version of this Recommendation that is in use. Annex A lists the object identifiers defined for use by this Recommendation.

`multiplexCapability` indicates capabilities relating to multiplexing and network adaptation. A terminal shall include `multiplexCapability` in the first `V75Capability` indicates capabilities of the V.75 control entity. The `audioHeader` indicates the capability of the V.75 audio header `TerminalCapabilitySet`, sent.

7.2.2.1 Capability Table

A `Capability Table` is a numbered list of capabilities. A terminal shall be capable of everything that it lists in its `capability table`, but shall not necessarily be capable of simultaneously performing more than one of them.

A `TerminalCapabilitySet` may contain zero or more `CapabilityTableEntry`s. At the start, no table entries are defined. When a `CapabilityTableEntry` is received, it replaces the previously received `CapabilityTableEntry` with the same `CapabilityTableEntryNumber`. A `CapabilityTableEntry` without a `Capability` may be used to remove the previously received `CapabilityTableEntry` with the same `CapabilityTableEntryNumber`.

7.2.2.2 Capability Descriptors

`CapabilityDescriptors` are used to indicate a terminal's capability to transmit and receive. Each `CapabilityDescriptor` provides an independent statement about the terminal's capabilities.

Superseded by a more recent version

capabilityDescriptorNumber is used to number CapabilityDescriptors. If a terminal has a preference for the mode it would like to transmit or receive, and wishes to express this when transmitting its capabilities, it may do so by giving CapabilityDescriptors that relate to its preferred mode or modes small values of capabilityDescriptorNumber.

simultaneousCapabilities is a set of AlternativeCapabilitySet. It is used to list the simultaneous capabilities of the terminal.

An AlternativeCapabilitySet is a sequence of CapabilityTableEntryNumbers. Only those CapabilityTableEntries that have been defined shall be present in an AlternativeCapabilitySet, although it is possible to define CapabilityTableEntries and refer to them in the same TerminalCapabilitySet. If a terminal has a preference for the mode it would like to transmit or receive, and wishes to express this when transmitting its capabilities, it may do so by listing elements in AlternativeCapabilitySets in order of decreasing preference.

A terminal shall be capable of simultaneously performing any one capability from each AlternativeCapabilitySet listed in simultaneousCapabilities.

At least one capability descriptor shall have the following structure: there shall be at least one AlternativeCapabilitySet containing only capabilities of a single medium type for each medium type that the terminal can support. This is to ensure that the remote terminal can select a mode of transmission that includes at least one instance of each medium type that the receiver can support.

NOTE 1 – A repetition of a capability in an AlternativeCapabilitySet is redundant and conveys no further information, while the repetition of a capability in different AlternativeCapabilitySets in the same CapabilityDescriptor indicates the possibility of an additional, simultaneous, instance of the particular capability.

NOTE 2 – Terminals that can not vary the allocation of resources can indicate their capability completely by use of a single CapabilityDescriptor.

7.2.2.3 Capability

The choices receiveVideoCapability, receiveAudioCapability and receiveDataApplicationCapability indicate the capability to receive according to the respective VideoCapability, AudioCapability and DataApplicationCapability.

The choices transmitVideoCapability, transmitAudioCapability and transmitDataApplicationCapability indicate the capability to transmit according to the respective VideoCapability, AudioCapability and DataApplicationCapability.

The choices receiveAndTransmitVideoCapability, receiveAndTransmitAudioCapability and receiveAndTransmitDataApplicationCapability indicate the capability to receive and transmit according to the respective VideoCapability, AudioCapability and DataApplicationCapability. These code points may be useful for indicating that the receive and transmit capabilities are not independent.

The boolean h233EncryptionTransmitCapability, when true, indicates that the terminal supports encryption according to Recommendations H.233 and H.234 [11] and [12] respectively.

h233IVResponseTime is measured in units of milliseconds, and indicates the minimum time the receiver requires the transmitter to wait after the completion of transmission of an IV message before starting to use the new IV. The means of transmitting the IV is not defined in this Recommendation.

7.2.2.4 Multiplex capabilities

MultiplexCapability indicates capabilities relating to a multiplexing and network adaptation. A terminal shall send MultiplexCapability in the first TerminalCapability sent. Unless stated otherwise, these are capabilities to receive.

H222Capability: indicates multiplexing and network adaptation capabilities that are specific to the multiplex defined in Recommendation H.222.1 [7].

numberOfVCs indicates how many simultaneous ATM Virtual Channels (VCs) can be supported by the terminal. This includes any VCs that transport H.245, T.120, DSM-CC or any other data, and all VCs that carry audiovisual information. It does not include the VC used for Q.2931 signalling [20].

vcCapability is a set, of size equal to the value of numberOfVCs, that indicates the capabilities present for each available VC.

Superseded by a more recent version

The sequence aal1, when present, indicates the capability for ATM adaptation layer 1 and which of its options, as specified in I.363 [19], are supported. The codepoints are defined in Table 1.

TABLE 1/H.245

ATM Adaptation Layer 1 codepoints

ASN.1 codepoint	Semantic meaning of codepoint
nullClockRecovery	Null source clock frequency recovery method: synchronous circuit transport
srtsClockRecovery	Synchronous residual timestamp source clock frequency recovery method
adaptiveClockRecovery	Adaptive clock source clock frequency recovery method
nullErrorCorrection	No error correction is supported
longInterleaver	The forward error correction method for loss sensitive signal transport is supported
shortInterleaver	The forward error correction method for delay sensitive signal transport is supported
errorCorrectionOnly	The forward error correction method without cell interleaving is supported
structuredDataTransfer	Structured data transfer is supported
partiallyFilledCells	Partially filled cells is supported

The sequence aal5, when present, indicates the capability for ATM adaptation layer 5, and which of its options as specified in I.363 [19], are supported.

forwardMaximumSDUSize and backwardMaximumSDUSize indicate the maximum CPCS-SDU size in the forward and reverse directions, measured in octets. Either aal1 or aal5 or both shall be present.

The booleans transportStream and programStream, when equal to true, indicate the capability to support the Transport Stream and Program Stream multiplexes respectively [6].

availableBitRates indicates the bit rate capabilities for the VC. It is a sequence of different bit rates that can be supported, measured in units of 64 kbit/s. Bit rates are listed in decreasing order, that is, the highest bit rate supported is listed first. Supported bit rates can be listed as individual values using the field singleBitRate, or as a rangeOfBitRates between lowerBitRate and HigherBitRate, indicating that all values between this lower limit and higher limit, including these limits, are supported. The bit rates indicated are measured at the AAL-SAP.

H223Capability: indicates capabilities specific to the H.223 multiplex [8].

The boolean transportWithI-frames, when true, indicates that the terminal is capable of sending and receiving control channel messages using LTP I-frames as defined in Recommendation V.42 [29].

The booleans videoWithAL1, videoWithAL2, videoWithAL3, audioWithAL1, audioWithAL2, audioWithAL3, dataWithAL1, dataWithAL2 and dataWithAL3, when true, indicate the capability to receive the stated medium type (video, audio, or data) using the stated adaptation layer (AL1, AL2, or AL3).

The integers maximumAL2SDUSize and maximumAL3SDUSize indicate the maximum number of octets in each SDU that the terminal can receive when using adaptation layer types 2 and 3 respectively.

maximumDelayJitter indicates the maximum peak-to-peak multiplexing jitter that the transmitter shall cause. It is measured in milliseconds. Multiplexing jitter is defined as the difference in time of delivery of the first octet of an audio frame when delivered in the multiplexed stream and when it would be delivered at constant bit rate without a multiplex.

h223MultiplexTableCapability: indicates the terminal's ability to receive and process multiplex table entries.

Superseded by a more recent version

basic indicates that the multiplex can only receive basic MultiplexEntryDescriptors as defined in Recommendation H.223 [8].

enhanced indicates that the multiplex can receive enhanced MultiplexEntryDescriptors with the additional parameters defined below.

maximumNestingDepth indicates the maximum nesting depth of recursively invoked subElementList fields. MultiplexEntryDescriptors which do not use the subElementList field shall be considered to have a nesting depth of zero.

maximumElementListSize indicates the maximum number of fields in the ASN.1 SEQUENCE.

maximumSubElementListSize indicates the maximum number of subelements in the SubElementList.

V76Capability: indicates capabilities specific to the V.76 multiplex.

The suspendResumeCapabilitywAddress indicates the capability of supporting V.76 suspend/resume with an address field. The suspendResumeCapabilitywoAddress indicates the capability of supporting V.76 suspend/resume without an address field.

rejCapability indicates the capability of the V.76 multiplex error control function to perform reject.

sREJCapability indicates the capability of the multiplex error control function to perform selective reject.

mREJCapability indicates the capability of multiplex error control function to perform multiple selective reject.

8bitCRCCapability is the capability of the multiplex to use 8-bit CRC.

crc16bitCapability is the capability of multiplex to use 16-bit CRC.

32bitCRCCapability is the capability of the multiplex to use 32-bit CRC.

uihCapability indicates support of V.76 UIH frames.

numOfDLCS indicates the number of DLCs which the V.76 multiplex can support.

twoOctetAddressFieldCapability indicates the ability of the V.76 multiplex to support an address field of two octets.

loopBackTestCapability indicates the support of loop back per Recommendation V.76.

n401Capability indicates the maximum value of N401 described in Recommendation V.76.

maxWindowSizeCapability indicates the maximum window size the V.76 multiplex can support.

7.2.2.5 Video capabilities

This indicates video capabilities. The indication of more than a single capability within a single VideoCapability does not indicate simultaneous processing capability. Simultaneous processing capability can be indicated by instances of VideoCapability in different AlternativeCapabilitySets in a single CapabilityDescriptor.

H261VideoCapability: indicates H.261 [13] capabilities.

If present, qcifMPI indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of QCIF pictures, and if not present, no capability for QCIF pictures is indicated.

If present, cifMPI indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of CIF pictures, and if not present, no capability for CIF pictures is indicated.

The boolean temporalSpatialTradeOffCapability, when true, indicates that the encoder is able to vary its trade-off between temporal and spatial resolution as commanded by the remote terminal. It has no meaning when part of a receive capability.

maxBitRate indicates the maximum bit rate in units of 100 bit/s at which a transmitter can transmit video or a receiver can receive video.

Superseded by a more recent version

stillImage Transmission indicates the capability for still images as specified in Annex D/H.261.

H262VideoCapability: indicates H.262 [14] capabilities.

The list of booleans indicate the capability of processing the particular profiles and levels: a value of true indicates that such operation is possible, while a value of false indicates that such operation is not possible. An encoder shall produce bitstreams compliant to the specifications of a profile and level for which it has indicated capability, but also within the limitations imposed by the optional fields (see below). A decoder shall be able to accept all bit streams conforming to a profile and level for which it has indicated capability, provided it is within the limitations indicated by the optional fields. The optional fields are integers with units defined in Table 2.

TABLE 2/H.245

Units for H.262 codepoints

ASN.1 codepoint	Units for referenced parameter
videoBitRate	400 bits per second
vbvBufferSize	16 384 bits
samplesPerLine	Samples per line
linesPerFrame	Lines per frame
framesPerSecond	The index, frame_rate_code, into Table 6-4/H.262
luminanceSampleRate	Samples per second

H263VideoCapability: indicates H.263 [15] capabilities.

If present, sqcifMPI indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of SQCIF pictures, and if not present, no capability for SQCIF pictures is indicated.

If present, qcifMPI indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of QCIF pictures, and if not present, no capability for QCIF pictures is indicated.

If present, cifMPI indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of CIF pictures, and if not present, no capability for CIF pictures is indicated.

If present, cif4MPI indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of 4CIF pictures, and if not present, no capability for 4CIF pictures is indicated.

If present, cif16MPI indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of 16CIF pictures, and if not present, no capability for 16CIF pictures is indicated.

maxBitRate indicates the maximum bit rate in units of 100 bit/s at which a transmitter can transmit video or a receiver can receive video.

The booleans unrestrictedVector, arithmeticCoding, advancedPrediction, and pbFrames, when true, indicate the capability to transmit and/or receive these optional modes defined in the annexes of H.263.

The boolean temporalSpatialTradeOffCapability, when true, indicates that the encoder is able to vary its trade-off between temporal and spatial resolution as commanded by the remote terminal. It has no meaning when part of a receive capability.

The integer hrd-B, when present, indicates the HRD parameter B, and is measured in units of 128 bits. When not present, the default value defined in Recommendation H.263 applies. It is a receiver capability and has no meaning in transmission capability sets.

Superseded by a more recent version

The integer `bppMax`, when present, indicates the maximum number of bits for one coded picture that the receiver can receive and decode correctly, and is measured in units of 1024 bits. When not present, the default value defined in Recommendation H.263 applies. It is a receiver capability and has no meaning in transmission capability sets.

The values of `MPI` are applicable when all of the optional modes, for which capability is indicated, are being used, as well as when any combination of them is used. A terminal may signal the capability for a smaller `MPI` when some options are not used by transmitting another `VideoCapability` including this smaller `MPI` and indicating the reduced set of options.

IS11172 VideoCapability: indicates IS11172-2 [33] capabilities.

`constrainedBitstream` indicates the capability for bit streams in which constrained parameters flag is set to “1”: a value of true indicates that such operation is possible, while a value of false indicates that such operation is not possible. An encoder shall produce bits streams within the limitations imposed by the optional fields (see below). A decoder shall be able to accept all bit streams within the limitations indicated by the optional fields. The optional fields are integers with units defined in Table 3.

TABLE 3/H.245

Units for IS11172-2 codepoints

ASN.1 codepoint	Units for referenced parameter
<code>videoBitRate</code>	400 bits per second
<code>vbvBufferSize</code>	16 384 bits
<code>samplesPerLine</code>	Samples per line
<code>linesPerFrame</code>	Lines per frame
<code>pictureRate</code>	Refer to 2.4.3.2/IS11172-2
<code>luminanceSample Rate</code>	Samples per second

7.2.2.6 Audio capabilities

This indicates audio capabilities. The indication of more than a single capability within a single `AudioCapability` does not indicate simultaneous processing capability. Simultaneous processing capability can be indicated by instances of `AudioCapability` in different `AlternativeCapabilitySets` in a single `CapabilityDescriptor`.

The capability to transmit and/or receive G-Series audio is indicated by a choice of integers. When an H.222.1 multiplex is used, these numbers refer to the available STD buffer size in units of 256 octets. When an H.223 multiplex is used, these numbers refer to the maximum number of audio frames per AL-SDU. The exact meaning of the codepoints is given in Table 4.

IS11172AudioCapability: indicates the ability to process audio coded according to ISO/IEC 11172-3 [34].

Booleans that have the value of true indicate that the particular mode of operation is possible, while a value of false indicates that it is not. The booleans `audioLayer1`, `audioLayer2` and `audioLayer3` indicate which audio coding layers can be processed. The booleans `audioSampling32k`, `audioSampling44k1` and `audioSampling48k` indicate which of the audio sample rates, 32KHz, 44.1KHz and 48KHz respectively, can be processed. The booleans `singleChannel` and `twoChannels` indicate capability for single channel and stereo/dual channel operation respectively. The integer `bitRate` indicates the maximum audio bit rate capability, and is measured in units of kbits per second.

IS13818AudioCapability: indicates the ability to process audio coded according to ISO/IEC 13818-3 [35].

Booleans that have the value of true indicate that the particular mode of operation is possible, while a value of false indicates that it is not. The booleans `audioLayer1`, `audioLayer2` and `audioLayer3` indicate which audio coding layers can be processed. The booleans `audioSampling16k`, `audioSampling22k05`, `audioSampling24k`, `audioSampling32k`, `audioSampling44k1` and `audioSampling48k` indicate which of the audio sample rates, 16KHz, 22.05KHz, 24KHz, 32KHz, 44.1KHz and 48KHz respectively, can be processed.

Superseded by a more recent version

TABLE 4/H.245

G-Series audio codepoints

ASN.1 codepoint	Semantic meaning of codepoint
g711Alaw64k	G.711 audio at 64 kbit/s, A-law
g711Alaw56k	G.711 audio at 56 kbit/s, A-law, truncated to 7 bits
g711Ulaw64k	G.711 audio at 64 kbit/s, μ -law
g711Ulaw56k	G.711 audio at 56 kbit/s, μ -law, truncated to 7 bits
g722-64k	G.722 7 kHz audio at 64 kbit/s
g722-56k	G.722 7 kHz audio at 56 kbit/s
g722-48k	G.722 7 kHz audio at 48 kbit/s
g723	G.723.1 at either 5.3 or 6.4 kbit/s
g728	G.728 audio at 16 kbit/s
g729	G.729 audio at 8 kbit/s
g-729AnnexA	G.729 Annex A at 8 kbit/s

The booleans concerned with multi-channel operation indicate capability to operate in the particular modes, as specified in Table 5.

TABLE 5/H.245

ISO/IEC 13818-3 multi-channel codepoints

ASN.1 codepoint	Semantic meaning of codepoint
singleChannel	One channel, using the 1/0 configuration. Single channel mode (as in ISO/IEC 11172-3)
twoChannels	Two channels, using the 2/0 configuration. Stereo or dual channel mode (as in ISO/IEC 11172-3)
threeChannels2-1	Three channels, using the 2/1 configuration. Left, Right and single surround channel
threeChannels3-0	Three channels, using the 3/0 configuration. Left, Centre and Right, without surround channel
fourChannels2-0-2-0	Four channels, using the 2/0 + 2/0 configuration. Left and Right of the first programme and Left and Right of the second programme
fourChannels2-2	Four channels, using the 2/2 configuration. Left, Right, Left surround and Right surround
fourChannels3-1	Four channels, using the 3/1 configuration. Left, Centre, Right, and a single surround channel
fiveChannels3-0-2-0	Five channels, using the 3/0 + 2/0 configuration. Left, Centre and Right of the first programme and Left and Right of the second programme
fiveChannels3-2	Five channels, using the 3/2 configuration. Left, Centre, Right, Left surround and Right surround

The boolean lowFrequencyEnhancement indicates the capability for a low frequency enhancement channel.

The boolean multilingual, when true, indicates the capability to support up to seven multilingual channels, and when false that no multilingual channel is supported.

The integer bitRate indicates the maximum audio bit rate capability, and is measured in units of kbits per second.

Superseded by a more recent version

7.2.2.7 Data Application capabilities

This indicates data capabilities. The indication of more than a single capability within a single `DataApplicationCapability` does not indicate simultaneous processing capability. Simultaneous processing capability can be indicated by instances of `DataApplicationCapability` in different `AlternativeCapabilitySets` in a single `CapabilityDescriptor`.

Recommendations that use this Recommendation may place restrictions on which of these modes may be signalled.

Some of the data capabilities require bidirectional logical channels, for example, to run a retransmission protocol. This requirement is implicitly included in the appropriate capability codepoints.

DataApplicationCapability: is a list of data applications and bit rates. Each data application indicated shall be supported by one or more `DataProtocolCapability`.

`maxBitRate` indicates the maximum bit rate in units of 100 bit/s at which a transmitter can transmit video or a receiver can receive the given data application.

`t120` indicates the capability to support the T.120 [25] protocol.

`dsm-cc` indicates the capability to support the DSM-CC [36] protocol.

`userData` indicates the capability to support unspecified user data from external data ports.

`t84` indicates the capability to support the transfer of T.84 [24] type images (JPEG, JBIG, Facsimile Gr.3/4).

`t434` indicates the capability to support the transfer of T.434 [26] telematic binary files.

`h224` indicates the capability to support the real-time simplex device control protocol H.224 [9].

`nlpid` indicates the capability to support the network layer protocol as specified by `nlpidData` as defined in ISO/IEC TR 9577 [37]. These protocols include Internet Protocol (IP) and IETF Point-to-Point Protocol (PPP), among others.

NOTE – The use of the NLPID is extensively described in IETC-RFC1490, 'Multiprotocol Interconnect over Frame Relay'.

`V76Control` indicates the capability of the V76 terminal to support an out-of-band control channel.

`h222DataPartitioning` indicates the capability to support the modified and restricted usage of data partitioning of Recommendation H.262, as specified in Recommendation H.222.1, in which the enhancement data is transmitted as a data channel supported by the listed `DataProtocolCapability`.

DataProtocolCapability: contains a list of data protocols.

`v14buffered` indicates the capability to support a specified data application using buffered V.14 [27].

`v42lapm` indicates the capability to support a specified data application using the LAPM protocol defined in V.42 [29].

`hdlcFrameTunneling` indicates the capability to support a specified data application using HDLC Frame Tunneling. Refer to 4.5.2 of ISO/IEC 3309 [32].

`h310SeparateVCStack` indicates the capability to support a specified data application using the protocol stack defined in Recommendation H.310 for the transport of H.245 messages over a separate ATM-VC to that used for audiovisual communication.

`H310SingleVCStack` indicates the capability to support a specified data application using the protocol stack defined in Recommendation H.310 for the transport of H.245 messages in the same ATM-VC as that used for audiovisual communication.

`transparent` indicates the capability to support a specified data application using transparent data transfer.

T84Profile: indicates the types of still image profile that the terminal is able to support.

`T84Unrestricted` provides no indication of the type of T.84 still image that the terminal is able to support: information in the T.84 layer should be used to determine whether a particular image can be received.

Superseded by a more recent version

T84Restricted indicates the type of T.84 still image that the terminal is able to support.

qcif indicates the support of a sequential colour YCrCb type image with QCIF resolution.

cif indicates the support of a sequential colour YCrCb type image with CIF resolution.

ccir601Seq indicates the support of a sequential colour YCrCb type image with CCIR601 resolution.

ccir601Prog indicates the support of a progressive colour YCrCb type image with CCIR601 resolution.

hdtvSeq indicates the support of a sequential colour YCrCb type image with HDTV resolution.

hdtvProg indicates the support of a progressive colour YCrCb type image with HDTV resolution.

g3FacsmH200x100 indicates the support of a sequential Facsimile Gr. 3 MH (Modified Huffman) coded bi-level image at the normal (200 × 100 ppi) resolution.

g3FacsmH200x200 indicates the support of a sequential Facsimile Gr. 3 MH (Modified Huffman) coded bi-level image at the high (200 × 200 ppi) resolution.

g4FacsmMR200x100 indicates the support of a sequential Facsimile Gr. 4 MMR (Modified Modified Reed) coded bi-level image at the normal (200 × 100 ppi) resolution.

g4FacsmMR200x200 indicates the support of a sequential Facsimile Gr. 4 MMR (Modified Modified Reed) coded bi-level image at the high (200 × 200 ppi) resolution.

jbig200x200Seq indicates the support of a sequential bi-level JBIG coded bi-level image at the 200 × 200 ppi resolution.

jbig200x200Prog indicates the support of a progressive bi-level JBIG coded bi-level image at the 200 × 200 ppi resolution.

jbig300x300Seq indicates the support of a sequential bi-level JBIG coded bi-level image at the 300 × 300 ppi resolution.

jbig300x300Prog indicates the support of a progressive bi-level JBIG coded bi-level image at the 300 × 300 ppi resolution.

digPhotoLow indicates the support of a sequential JPEG coded colour image of up to 720 × 576 image size.

digPhotoMedSeq indicates the support of a sequential JPEG coded colour image of up to 1440 × 1152 image size.

digPhotoMedProg indicates the support of a progressive JPEG coded colour image of up to 1440 × 1152 image size.

digPhotoHighSeq indicates the support of a sequential JPEG coded colour image of up to 2880 × 2304 image size.

digPhotoHighProg indicates the support of a progressive JPEG coded colour image of up to 2880 × 2304 image size.

7.2.3 Terminal Capability Set Acknowledge

This is used to confirm receipt of a TerminalCapabilitySet from the peer CESE.

The sequenceNumber shall be the same as the sequenceNumber in the TerminalCapabilitySet for which this is the confirmation.

7.2.4 Terminal Capability Set Reject

This is used to reject a TerminalCapabilitySet from the peer CESE.

The sequenceNumber shall be the same as the sequenceNumber in the TerminalCapabilitySet for which this is the negative acknowledgement.

Superseded by a more recent version

The reasons for sending this message are given in Table 6.

TABLE 6/H.245

Reasons for rejecting a TerminalCapabilitySet

ASN.1 codepoint	Cause
unspecified	No cause for rejection specified
undefinedTableEntryUsed	A capability descriptor made reference to a capabilityTable entry that is not defined
descriptorCapacityExceeded	The terminal was incapable of storing all of the information in the TerminalCapabilitySet
tableEntryCapacityExceeded	The terminal was incapable of storing more entries than that indicates in highestEntryNumberProcessed or else could not store any

7.2.5 Terminal Capability Set Release

This is sent in the case of a time out.

7.3 Logical channel signalling messages

This set of messages is for logical channel signalling. The same set of messages is used for unidirectional and bidirectional logical channel signalling; however, some parameters are only present in the case of bidirectional logical channel signalling.

“Forward” is used to refer to transmission in the direction from the terminal making the original request for a logical channel to the other terminal, and “reverse” is used to refer to the opposite direction of transmission, in the case of a bidirectional channel request.

7.3.1 Open Logical Channel

This is used to attempt to open a unidirectional logical channel connection between an outgoing LCSE and a peer incoming LCSE and to open a bidirectional logical channel connection between an outgoing B-LCSE and a peer incoming B-LCSE.

forwardLogicalChannelNumber: indicates the logical channel number of the forward logical channel that is to be opened.

forwardLogicalChannelParameters: include parameters associated with the logical channel in the case of attempting to open a unidirectional channel and parameters associated with the forward logical channel in the case of attempting to open a bidirectional channel.

reverseLogicalChannelParameters: include parameters associated with the reverse logical channel in the case of attempting to open a bidirectional channel. Its presence indicates that the request is for a bidirectional logical channel with the stated parameters, and its absence indicates that the request is for a unidirectional logical channel.

NOTE – H.222 parameters are not included in reverseLogicalChannel Parameters as their values are not known to the terminal initiating the request.

PortNumber is a user-to-user parameter that may be used by a user for such purposes as associating an input or output port, or higher layer channel number, with the logical channel.

dataType indicates the data that is to be carried on the logical channel.

If it is nullData, the logical channel will not be used for the transport of elementary stream data, but only for adaptation layer information – if video is to be transmitted in one direction only, but a retransmission protocol is to be used, such as AL3 defined in Recommendation H.223, a return channel is needed to transport the retransmission requests – it may also be used to describe a logical channel that only contains PCR values in the case of H.222.1 Transport Streams [7].

Superseded by a more recent version

Terminals capable only of unidirectional (transmit or receive) operation on media types which make use of bidirectional channels shall send capabilities only for the supported direction of operation. The reverse direction shall use the nullData type, for which no capability is necessary. Transmit-only terminals should send transmit capabilities, but terminals should not assume that the absence of transmit capabilities implies that transmit-only operation is not possible.

If it is of type VideoCapability, AudioCapability, the logical channel may be used for any of the variations indicated by each individual capability: and it shall be possible to switch between these variations using only signalling that is in-band to the logical channel – for example, in the case of H.261 video, if both QCIF and CIF are indicated, it shall be possible to switch between these on a picture-by-picture basis. In the case of DataApplicationCapability, only one instance of a capability can be indicated since there is no in-band signalling allowing a switch between variations.

If it is encryptionData, the logical channel will be used for the transport of encryption information as specified.

H222LogicalChannelParameters: is used to indicate parameters specific to using Recommendation H.222.1 [7]. It shall be present in forwardLogicalChannelParameters and shall not be present in reverseLogicalChannelParameters.

resourceID indicates in which ATM Virtual Channel the logical channel is to be transported. The means by which this parameter is associated with an ATM Virtual Channel is not specified in this Recommendation.

subChannelID indicates which H.222.1 sub-channel is used for the logical channel. It shall be equal to the PID in a Transport Stream and the stream_id in a Program Stream.

pcr-pid indicates the PID used for the transport of Program Clock References when the Transport Stream is used. It shall be present when the ATM virtual channel carries a Transport Stream and shall not be present when the ATM virtual channel carries a Program Stream.

programDescriptors is an optional octet string, which, if present, contains one or more descriptors, as specified in Recommendations H.222.0 and H.222.1, that describe the program that the information to be carried in the logical channel is a part of.

streamDescriptors is an optional octet string, which, if present, contains one or more descriptors, as specified in Recommendations H.222.0 and H.222.1, that describe the information that is to be carried in the logical channel.

H223LogicalChannelParameters: is used to indicate parameters specific to using Recommendation H.223 [8]. It shall be present in forwardLogicalChannelParameters and reverseLogicalChannelParameters.

adaptationLayerType indicates which adaptation layer and options will be used on the logical channel. The codepoints are as follows:

- nonStandard;
- al1Framed (AL1 framed mode);
- al1NotFramed (AL1 unframed mode);
- al2WithoutSequenceNumbers (AL2 with no sequence numbers present);
- al2WithSequenceNumbers (AL2 with sequence numbers present); and
- al3 (AL3, indicating the number of control field octets that will be present and the size of the send buffer, B_s , that will be used, the size being measured in octets).

segmentableFlag, when equal to true indicates that the channel is designated to be segmentable, and when equal to false indicates that the channel is designated to be non-segmentable.

V76LogicalChannel Parameters: is used to indicate parameters specific to using V.76.

audioheader is used to indicate the use of an audio header on the logical channel. This is a valid parameter for channels of the data type audio.

suspendResume is used to indicate that the channel may use the suspend/resume procedures to suspend other logical channels.

eRM indicates that the logical channel shall perform error recovery procedures as defined in V.76.

Superseded by a more recent version

uNERM indicates that the logical channel shall operate in non-error recovery mode as defined in V.76.

For description of n401, windowSize and loopbackTestProcedure, see 12.2.1/V.42 and its subclauses.

crcLength is an optional parameter that indicates the CRC length used in error recovery mode. If this parameter is not present, the default CRC length shall be used. 8bitCRC indicates to use an 8-bit CRC crc16bit indicates use of the 16-bit CRC and 32bitCRC indicates to use a 32-bit CRC as defined in V.76.

recovery is an optional parameter that indicates the error recovery procedures defined in V.76. If this parameter is not present, the default error recovery procedure shall be used. sEJ indicates to use the selective frame reject procedure and mEJ indicates to use the multiple selective reject procedure as defined in V.76.

wAddress indicates that the suspend/resume channel shall use the address field as defined in Recommendation V.76, woAddress indicates that the suspend/resume channel shall not use the address field.

uiH indicates the use of V.76 UIH frames.

rej indicates the use of the reject procedure in Recommendation V.76.

V75Parameters is used to indicate parameter specific to using V.75, audioHeaderPresent indicates the presence of the V.75 audio header.

7.3.2 Open Logical Channel Acknowledge

This is used to confirm acceptance of the logical channel connection from the peer LCSE to B-LCSE. In the case of a request for a unidirectional logical channel, it indicates acceptance of that unidirectional logical channel. In the case of a request for a bidirectional logical channel, it indicates acceptance of that bidirectional logical channel, and indicates the appropriate parameters of the reverse channel.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel that is being opened.

reverseLogicalChannelParameters is present if, and only if, responding to a bidirectional channel request.

reverseLogicalChannelNumber indicates the logical channel number of the reverse channel.

portNumber is a user-to-user parameter that may be used by a user for such purposes as associating an input or output port, or higher layer channel number, with the reverse logical channel.

multiplexParameters indicate parameters specific to the multiplex, H.222 or H.223, that is used to transport the reverse logical channel.

NOTE – H.223 parameters are not included in reverseLogicalChannelParameters as their values were specified in the OpenLogicalChannel request message.

7.3.3 Open Logical Channel Reject

This is used to reject the logical channel connection request from the peer LCSE or B-LCSE.

NOTE – In the case of a bidirectional channel request, rejection applies to both forward and reverse channels. It is not possible to accept one and reject the other.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel specified in the request that is being rejected.

The cause field indicates the reason for rejection of the logical channel establishment. The cause values are given in Table 7.

7.3.4 Open Logical Channel Confirm

This is used in bidirectional signalling to indicate to the incoming B-LCSE that the reverse channel is open and can be used for transmission.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel which was opened.

Superseded by a more recent version

TABLE 7/H.245

Reasons for rejecting a OpenLogicalChannel

ASN.1 codepoint	Cause
unspecified	No cause for rejection specified
unsuitableReverseParameters	This shall only be used to reject a bidirectional logical channel request when the only reason for rejection is that the requested reverseLogicalChannelParameters are inappropriate. Such a rejection shall immediately be followed by initiating procedures to open a similar but acceptable bidirectional logical channel.
dataTypeNotSupported	The terminal was not capable of supporting the dataType indicated in OpenLogicalChannel
dataTypeNotAvailable	The terminal was not capable of supporting the dataType indicated in OpenLogicalChannel simultaneously with the dataTypes of logical channels that are already open
unknownDataType	The terminal did not understand the dataType indicated in OpenLogicalChannel
dataTypeALCombinationNotSupported	The terminal was not capable of supporting the dataType indicated in OpenLogicalChannel simultaneously with the Adaptation Layer type indicated in H.233 LogicalChannelParameters

7.3.5 Close Logical Channel

This is used by the outgoing LCSE or B-LCSE to close a logical channel connection between two peer LCSEs or B-LCSEs.

NOTE – In the case of a bidirectional logical channel, this closes both forward and reverse channels. It is not possible to close one and not the other.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that is to be closed.

The source of the logical channel release is given in Table 8.

TABLE 8/H.245

Sources of logical channel release

ASN.1 codepoint	Cause
user	The LCSE or B-LCSE-user is the source of the release
lcse	The LCSE or B-LCSE is the source of the release. This may occur as a result of a protocol error

7.3.6 Close Logical Channel Acknowledge

This is used to confirm the closing of a logical channel connection.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that is being closed.

7.3.7 Request Channel Close

This is used by the outgoing CLCSE to request the closing of a logical channel connection between two peer LCSEs.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that is requested to close.

Superseded by a more recent version

7.3.8 Request Channel Close Acknowledge

This is used by the incoming CLCSE to indicate that the logical channel connection will be closed.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that it has been requested to close.

7.3.9 Request Channel Close Reject

This is used by the incoming CLCSE to indicate that the logical channel connection will not be closed.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that it has been requested to close.

The cause field indicates the reason for rejection of the request to close the logical channel. The only valid cause value is unspecified.

7.3.10 Request Channel Close Release

This is sent by the outgoing CLCSE in the case of a time out.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that it has requested to close.

7.4 Multiplex Table signalling messages

This set of messages is for the secure transmission of H.223 multiplex table entries from the transmitter to the receiver.

7.4.1 Multiplex Entry Send

This is used to send H.223 multiplex table entries from the transmitter to the receiver. It is sent from an outgoing MTSE and a peer incoming MTSE.

sequenceNumber is used to label instances of MultiplexEntrySend so that the corresponding response can be identified.

MultiplexEntryDescriptors is a set of 1 to 15 MultiplexEntryDescriptors.

MultiplexEntryDescriptor: describes a single multiplex table entry. It includes the MultiplexTableEntryNumber and a list of MultiplexElements. A missing element list indicates that the entry is deactivated.

MultiplexElement: is a recursive structure that describes a single element and a repeat count. If of type logicalChannelNumber, the element indicates a single segment from the given logical channel, and the repeat count indicates the length of the segment in octets. If of type subElementList, the element indicates a sequence of nested MultiplexElements, and the repeat count indicates the number of times to repeat the sequence. In either case, if the repeatCount field is untilClosingFlag, this means to repeat the element indefinitely until the closing flag of the MUX-PDU.

In each MultiplexEntryDescriptor, the repeatCount of the final MultiplexElement in the elementList shall be set to “untilClosingFlag”, and the repeatCount of all other MultiplexElements in the elementList shall be set to “finite”. This ensures that all multiplex table entries define a multiplex sequence pattern of indefinite length, repeating until the closing flag of the MUX-PDU. A multiplexEntryDescriptor with a missing elementlist field shall indicate a deactivated entry.

Each MultiplexEntrySend indication may contain up to 15 MultiplexEntryDescriptors, each describing a single multiplex table entry. Multiplex entries may be sent in any order.

7.4.2 Multiplex Entry Send Acknowledge

This is used to confirm receipt of one or more multiplexEntryDescriptors from a MultiplexEntrySend from the peer MTSE.

The sequenceNumber shall be the same as the sequenceNumber in the MultiplexEntrySend for which this is the confirmation.

multiplexTableEntryNumber indicates which multiplex table entries are being confirmed.

Superseded by a more recent version

7.4.3 Multiplex Entry Send Reject

This is used to reject one or more multiplexEntryDescriptors from a MultiplexEntrySend from the peer MTSE.

The sequenceNumber shall be the same as the sequenceNumber in the MultiplexEntrySend for which this is the rejection.

MultiplexEntryRejectionDescriptions specifies which table entries are being rejected, and why. The causes of rejection are given in Table 9.

TABLE 9/H.245

Reasons for rejecting a MultiplexEntrySend

ASN.1 codepoint	Cause
unspecified	No cause for rejection specified
descriptorTooComplex	The MultiplexEntryDescriptor exceeded the capability of the receive terminal

7.4.4 Multiplex Entry Send Release

This is sent by the outgoing MTSE in the case of a time-out.

multiplexTableEntryNumber indicates which multiplex table entries have timed out.

7.5 Request Multiplex Table signalling messages

This set of messages is for the secure request of retransmission of one or more MultiplexEntryDescriptors from the transmitter to the receiver.

7.5.1 Request Multiplex Entry

This is used to request the retransmission of one or more MultiplexEntryDescriptors.

entryNumbers is a list of the MultiplexTableEntryNumbers of the MultiplexEntryDescriptors for which retransmission is requested.

7.5.2 Request Multiplex Entry Acknowledge

This is used by the incoming RMESE to indicate that the multiplex entry will be transmitted.

entryNumbers is a list of the MultiplexTableEntryNumbers of the MultiplexEntryDescriptors will be transmitted.

7.5.3 Request Multiplex Entry Reject

This is used by the incoming RMESE to indicate that the multiplex entry will be transmitted. entryNumbers is a list of the MultiplexTableEntryNumbers of the MultiplexEntryDescriptors will not be transmitted. The values of MultiplexTableEntryNumbers should match the values of MultiplexTableEntryNumber in rejectionDescriptions, otherwise errors may occur during operation.

RequestMultiplexEntryRejectionDescriptions specifies which table entries are being rejected, and why. The causes of rejection are given in Table 10.

TABLE 10/H.245

Reasons for rejecting a MultiplexEntrySend

ASN.1 codepoint	Cause
unspecified	No cause for rejection specified

Superseded by a more recent version

7.5.4 Request Multiplex Entry Release

This is sent by the outgoing RMESE in the case of a time-out.

entryNumbers is a list of the MultiplexTableEntryNumbers of the MultiplexEntryDescriptors for which time-out has occurred.

7.6 Request Mode messages

This set of messages is used by a receive terminal to request particular modes of transmission from the transmit terminal.

7.6.1 Request Mode

This is used to request particular modes of transmission from the transmit terminal. It is a list, in order of preference (most preferable first), of modes that the terminal would like to receive. Each mode is described using a ModeDescription.

sequenceNumber is used to label instances of RequestMode so that the corresponding response can be identified.

ModeDescription: is a set of one or more ModeElements.

ModeElement: is used to describe a mode element, that is, one of the constituent parts of a complete mode description. It indicates the type of elementary stream that is requested and optionally how it is requested to be multiplexed.

type is used to indicate the type of elementary stream that is requested. It is a choice of VideoMode, AudioMode, DataMode, and EncryptionMode.

h223ModeParameters is used to indicate parameters specific to using Recommendation H.223 [8].

adaptationLayerType indicates which adaptation layer and options are requested for the requested type. The codepoints are as follows:

- nonStandard;
- al1Framed (AL1 framed mode);
- al1NotFramed (AL1 unframed mode);
- al2WithoutSequenceNumbers (AL2 with no sequence numbers present);
- al2WithSequenceNumbers (AL2 with sequence numbers present); and
- al3 (AL3, indicating the number of control field octets that will be present and the size of the send buffer, B_s , that will be used, the size being measured in octets).

segmentableFlag, when equal to true indicates that segmentable multiplexing is requested, and when equal to false indicates that non-segmentable multiplexing is requested.

7.6.1.1 Video Mode

This is a choice of VideoModes.

H261VideoMode: Indicates the requested picture resolution (either QCIF or CIF), bit rate, in units of 100 bit/s, and still picture transmission.

H262VideoMode: Indicates the requested profile and level, and the optional fields, if present, indicate the requested values of the parameters given. The optional fields are integers with units defined in Table 2.

H263VideoMode: Indicates the requested picture resolution (SQCIF, QCIF, CIF, 4CIF and 16CIF) and bit rate, in units of 100 bit/s.

The booleans unrestrictedVector, arithmeticCoding, advancedPrediction, and pbFrames, when true, indicate that it is requested to use these optional modes that are defined in the annexes of Recommendation H.263.

IS11172VideoMode: indicates request for constrainedBitstream and the optional fields, if present, indicate the requested values of the parameters given. The optional fields are integers with units defined in Table 3.

Superseded by a more recent version

7.6.1.2 Audio Mode

This is a choice of AudioModes.

The exact meaning of the G-Series audio codepoints is given in Table 3. There are four options for G.723.1 audio, to allow either of the bit rates (the low bit rate of 5.3 kbit/s or the high bit rate of 6.3 kbit/s) to be requested with or without the use of silence suppression.

IS11172AudioMode: is used to request audio coded according to ISO/IEC 11172-3 [34].

audioLayer indicates which coding layer is requested: either audioLayer1, audioLayer2 or audioLayer3.

audioSampling indicates which sample rate is requested: audioSampling32k, audioSampling44k1 and audioSampling48k indicate the audio sample rates 32 kHz, 44.1 kHz and 48 kHz respectively.

multichannelType indicates which multichannel mode is requested: singleChannel, twoChannelStereo and twoChannelDual request single channel, stereo and dual channel operation respectively.

bitRate indicates the requested audio bit rate, and is measured in units of kbits per second.

IS13818AudioMode: is used to request audio coded according to ISO/IEC 13818-3 [35].

audioLayer indicates which coding layer is requested: either audioLayer1, audioLayer2 or audioLayer3.

audioSampling indicates which sample rate is requested: audioSampling16k, audioSampling22k05, audioSampling24k, audioSampling32k, audioSampling44k1 and audioSampling48k indicate the audio sample rates 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz and 48 kHz respectively.

multichannelType indicates which multichannel mode is requested as specified in Table 11.

The boolean lowFrequencyEnhancement, when true, requests a low frequency enhancement channel.

The boolean multilingual, when true, requests up to seven multilingual channels.

bitRate indicates the requested audio bit rate, and is measured in units of kbits per second.

TABLE 11/H.245

ISO/IEC 13818-3 multichannel codepoints

ASN.1 Codepoint	Semantic meaning of codepoint
singleChannel	One channel, using the 1/0 configuration. Single channel mode (as in ISO/IEC 11172-3).
twoChannelStereo	Two channels using the 2/0 configuration, stereo channel mode (as in ISO/IEC 11172-3).
twoChannelDual	Two channels using the 2/0 configuration, dual channel mode (as in ISO/IEC 11172-3).
threeChannels2-1	Three channels, using the 2/1 configuration. Left, Right and single surround channel.
threeChannels3-0	Three channels, using the 3/0 configuration. Left, Centre and Right, without surround channel.
fourChannels2-0-2-0	Four channels, using the 2/0 + 2/0 configuration. Left and Right of the first programme and Left and Right of the second programme.
fourChannels2-2	Four channels, using the 2/2 configuration. Left, Right, Left surround and Right surround.
fourChannels3-1	Four channels, using the 3/1 configuration. Left, Centre, Right, and a single surround channel.
fiveChannels3-0-2-0	Five channels, using the 3/0 + 2/0 configuration. Left, Centre and Right of the first programme and Left and Right of the second programme.
fiveChannels3-2	Five channels, using the 3/2 configuration. Left, Centre, Right, Left surround and Right surround.

Superseded by a more recent version

7.6.1.3 Data Mode

This is a choice of data applications and bit rates.

bitRate indicates the requested bit rate in units of 100 bit/s.

t120 requests the use of the T.120 [25] protocol.

dsm-cc requests the use of the DSM-CC [36] protocol.

userData requests the use of unspecified user data from external data ports.

t84 requests the use of T.84 [24] for the transfer of such images (JPEG, JBIG, Facsimile Gr.3/4).

t434 requests the use of Recommendation T.434 [26] for the transfer of telematic binary files.

h224 requests the use of the real-time simplex device control protocol H.224 [9].

nlpid requests the use of the specified network link layer data application.

V76Control requests the use of the V76 terminal to support an out-of-band control channel.

H222DataPartitioning requests the use of the modified and restricted usage of data partitioning of Recommendation H.262, as specified in Recommendation H.222.1, in which the enhancement data is transmitted as a data channel supported by the listed DataProtocolCapability.

7.6.1.4 Encryption Mode

This is a choice of encryption modes.

h233Encryption requests the use of encryption according to H.233 and H.234 [11] [12].

7.6.2 Request Mode Acknowledge

This is sent to confirm that the transmit terminal intends to transmit in one of the modes requested by the receive terminal.

The sequenceNumber shall be the same as the sequenceNumber in the RequestMode for which this is the confirmation.

The response field indicates the action from the remote terminal.

The possible responses are given in Table 12.

TABLE 12/H.245

Confirmation responses to Request Mode

ASN.1 codepoint	Response
willTransmitMostPreferredMode	The transmit terminal will change to the receiver's most preferred mode
willTransmitLessPreferredMode	The transmit terminal will change to one of the receiver's preferred mode, but not the most preferred mode

7.6.3 Request Mode Reject

This is sent to reject the request by the receive terminal.

The sequenceNumber shall be the same as the sequenceNumber in the RequestMode for which this is the response.

The cause field indicates the reason for rejection of the requested mode.

Superseded by a more recent version

The possible responses are given in Table 13.

TABLE 13/H.245

Rejection responses to Request Mode

ASN.1 codepoint	Response
modeUnavailable	The transmit terminal will not change its mode of transmission as the requested modes are not available
multipointConstraint	The transmit terminal will not change its mode of transmission due to a multipoint constraint
requestDenied	The transmit terminal will not change its mode of transmission

7.6.4 Request Mode Release

This is used by the outgoing MRSE in the case of a time-out.

7.7 Round Trip Delay messages

This set of messages is used by a terminal to determine the round trip delay between two communicating terminals. It also enables a H.245 user to determine whether the peer H.245 protocol entity is alive.

7.7.1 Round Trip Delay request

This is sent from the outgoing RTDSE to the incoming RTDSE.

sequenceNumber is used to label instances of RoundTripDelayRequest so that the corresponding response can be identified.

7.7.2 Round Trip Delay response

This is sent from the incoming RTDSE to the outgoing RTDSE.

The sequenceNumber shall be the same as the sequenceNumber in the RoundTripDelayRequest for which this is the response.

7.8 Maintenance Loop messages

This set of messages is used by a terminal to perform maintenance loop functions.

7.8.1 Maintenance Loop request

This is sent to request a particular type of loop back. The types mediaLoop and logicalChannelLoop request the loopback of only one logical channel as indicated by LogicalChannelNumber, while the type systemLoop refers to all logical channels. The exact definition of these types is system specific and outside the scope of this Recommendation.

7.8.2 Maintenance Loop Acknowledge

This is used to confirm that the terminal will perform the loop as requested.

7.8.3 Maintenance Loop Reject

This is used to confirm that the terminal will not perform the loop as requested.

A terminal may use the cause canNotPerformLoop to indicate that it does not have the capability to perform the requested loop.

Superseded by a more recent version

7.8.4 Maintenance Loop Command Off

On receipt of this command, the terminal shall disconnect all loops and restore audio, video and data paths to their normal condition.

7.9 Commands

A command message requires action but no explicit response.

7.9.1 Send Terminal Capability Set

specificRequest commands the far-end terminal to indicate its transmit and receive capabilities by sending one or more TerminalCapabilitySets that contain the information requested, as specified below. This command may be sent at any time to elicit the capabilities of the remote terminal, for example, following an interruption or other cause for uncertainty; however, such messages should not be sent repetitively without strong cause.

A terminal shall only request the transmission of capabilityTableEntryNumbers and capabilityDescriptorNumbers that it has previously received. A terminal shall ignore any requests to transmit capabilityTableEntryNumbers and capabilityDescriptorNumbers that it has not previously transmitted and no fault shall be considered to have occurred.

The boolean multiplexCapability, when true, requests the transmission of the MultiplexCapability.

capabilityTableEntryNumbers is a set of the CapabilityTableEntryNumbers that indicate the CapabilityTableEntries that the terminal requests to be transmitted.

capabilityDescriptorNumbers is a set of the CapabilityDescriptorNumbers that indicate the CapabilityDescriptors that the terminal requests to be transmitted.

genericRequest commands the far-end terminal to send its entire terminal capability set.

7.9.2 Encryption

This command is used to exchange encryption capabilities and to command the transmission of an initialization vector (IV), refer to Recommendations H.233 and H.234 [11] and [12], respectively.

encryptionSE is an H.233 Session Exchange (SE) message, except that the error protection bits described in Recommendation H.233 shall not be applied.

encryptionIVRequest commands the far-end encryptor to transmit a new IV in a logical channel opened for encryptionData.

encryptionAlgorithmIdentifier indicates to the receiver that the sending terminal will associate the given H.233 Algorithm Identifier value with the non-standard encryption algorithm.

7.9.3 Flow Control

This command is used to specify the upper limit of bit rate of either a single logical channel or the whole multiplex. A terminal may send this command to restrict the bit rate that the far-end terminal sends. A terminal that receives this command shall comply with it.

When scope is of type logicalChannelNumber the limit applies to the given logical channel, when scope is of type virtualChannelID the limit applies to the given ATM virtual channel, and when scope is of type wholeMultiplex, the limit applies to the whole multiplex.

maximumBitRate is measured in units of 100 bits/s averaged over non-overlapping consecutive periods of one second. When this is present, the specified limit supersedes any previous limit, whether higher or lower. When it is not present any previous restriction on the bit rate for the channel is no longer applicable.

The point at which the bit rate limit is applied, and the specification of which bits are included in the calculation of bit rate is not specified in this Recommendation, but should be specified by recommendations that use this Recommendation.

Superseded by a more recent version

Each transmission of this command affects a specific logical channel or the entire multiplex. More than one such command may be in effect at the same time, up to the number of open logical channels plus one, for the overall multiplex limitation.

NOTE – When the bit rate that can be transmitted on a logical channel is constrained to particular values, for example G.723.1 audio, and the request is to transmit at a rate lower than the lowest rate at which it would normally operate, it shall respond by stopping transmission on the logical channel.

7.9.4 End session

This command indicates the end of the H.245 session. After transmitting EndSessionCommand, the terminal shall not send any more of the messages defined in this Recommendation.

disconnect indicates that the connection will be dropped.

gstnOptions: is a choice of alternatives that will occur after ending the H.245 session when a V-Series modem is used over the GSTN.

The possible options are given in Table 14.

TABLE 14/H.245

Options after EndSessionCommand when using a V-Series modem over the GSTN

ASN.1 codepoint	Option
telephonyMode	The terminal shall initiate the clear-down procedures defined in the V-Series modem Recommendation, except that it shall not physically disconnect the GSTN connection
v8bis	The terminal shall initiate the clear-down procedures defined in the V-Series modem Recommendation and enter a V.8 <i>bis</i> session
v34V76	The terminal shall preserve the V.34 modem connection, but use it to support V76
v34DuplexFAX	The terminal shall preserve the V.34 modem connection, but use it to support T.30 FAX [21]
v34H324	The terminal shall preserve the V.34 modem connection, but use it to support H.324 [18]

7.9.5 Miscellaneous Command

This is used for a variety of commands, some of which are present in Recommendations H.221 and H.230 [5] and [10], respectively.

logicalChannelNumber indicates the logical channel number to which the command applies. It shall indicate a logical channel opened for video data when the type is one of videoFreezePicture, videoFastUpdatePicture, videoFastUpdateGOB, videoTemporalSpatialTradeOff, videoSendSyncEveryGOB, and videoSendSyncEveryGOBCancel.

equaliseDelay and zeroDelay shall have the same meaning as the commands ACE and ACZ defined in H.230 [10].

multipointModeCommand commands that a terminal in receipt shall comply with all requestMode requests issued by the MCU. An example of a mode change is an audio coding change from G.711 to G.728.

cancelMultipointModeCommand cancels a previously sent multipointModeCommand command.

videoFreezePicture commands the video decoder to complete updating the current video frame and subsequently display the frozen picture until receipt of the appropriate freeze-picture release control signal.

Superseded by a more recent version

videoFastUpdatePicture commands the video encoder to enter the fast-update mode at its earliest opportunity.

videoFastUpdateGOB commands the far-end video encoder to perform a fast update of one or more GOBs. firstGOB indicates the number of the first GOB to be updated, and numberOfGOBs indicates the number of GOBs to be updated. It shall only be used with video compression algorithms that define GOBs, for example, Recommendations H.261 and H.263.

videoTemporalSpatialTradeOff commands the far-end video encoder to change its trade-off between temporal and spatial resolution. A value of 0 commands a high spatial resolution and a value of 31 commands a high frame rate. The values from 0 to 31 indicate monotonically a desire for higher frame rate. Actual values do not correspond to precise values of spatial resolution or frame rate.

videoSendSyncEveryGOB commands the far-end video encoder to use sync for every GOB as defined in Recommendation H.263 [15], until the command videoSendSyncEveryGOBCancel is received, from which time the far-end video encoder may decide the frequency of GOB syncs. These commands shall only be used with video encoded according to Recommendation H.263.

7.10 Indications

An indication contains information that does not require action or response.

7.10.1 Function not Supported

This is used to return requests, responses and commands that are not understood to the transmitter of them.

The whole of the RequestPDU, ResponsePDU or CommandPDU is returned.

If a terminal receives a request, response or command that it does not understand, either because it is non-standard or has been defined in a subsequent revision of this Recommendation, it should respond by sending FunctionNotSupported.

7.10.2 Miscellaneous Indication

This is used for a variety of indications, some of which are present in Recommendations H.221 and H.230 [5] and [10], respectively.

logicalChannelNumber indicates the logical channel number to which the indication applies. It shall indicate a logical channel opened for video data when the type is videoIndicateReadyToActivate, and videoTemporalSpatialTradeOff.

logicalChannelInactive is used to indicate that the content of the logical channel does not represent a normal signal. It is analogous to AIM and VIS defined in H.230 [10].

logicalChannelActive is complementary to logicalChannelInactive. It is analogous to AIA and VIA defined in Recommendation H.230. MultipointZeroComm, cancelMultipointZeroComm, multipointSecondaryStatus and cancelMultipointSecondaryStatus shall have the same meaning as MIZ, cancelMIZ, MIS and cancelMIS respectively, as defined in H.230 [10].

multipointConference indicates that the terminal is joined to an H.243 multipoint conference and the terminal is expected to obey bit rate symmeterization. However, bit rate symmeterization will be enforced via FlowControl messages. Note that multipointConference has exactly the same meaning as MCC in Recommendation H.230 [10]. Note that multipointConference, like MCC, does not require mode symmetry.

videoIndicateReadyToActivate shall have the same meaning as VIR defined in Recommendation H.230, that is, it is transmitted by a terminal whose user has decided not to send video unless he will also receive video from the other end.

videoTemporalSpatialTradeOff indicates to the far-end video decoder its current trade-off between temporal and spatial resolution. A value of 0 indicates a high spatial resolution and a value of 31 indicates a high frame rate. The values from 0 to 31 indicate monotonically a higher frame rate. Actual values do not correspond to precise values of spatial

Superseded by a more recent version

resolution or frame rate. A terminal that has indicated temporalSpatialTradeOffCapability shall transmit this indication whenever it changes its trade-off and when a video logical channel is initially opened.

7.10.3 Jitter Indication

This is used to indicate the amount of jitter, as estimated by the receive terminal, of a logical channel. It may be useful for choice of bit-rate and buffer control in video channels, or to determine an appropriate rate of transmission of timing information, etc. The video encoder will then have the option of using this information to restrict the video bit-rate or the video decoder buffer fluctuations to help prevent decoder buffer underflow or overflow, given the occurring jitter. If the encoder takes this option, it will enable correct operation for existing designs of video decoder buffers, regardless of the amplitude of received jitter, as well as allow correct operation with minimum delay.

When scope is of type logicalChannelNumber, the information applies to the given logical channel, when scope is of type virtualChannelID, the information applies to the given ATM virtual channel, and when scope is of type wholeMultiplex, the information applies to the whole multiplex.

estimatedReceivedJitterMantissa and estimatedReceivedJitterExponent provide an estimate of the jitter that has been received by the terminal that has sent the message.

estimatedReceivedJitterMantissa indicates the mantissa of the jitter estimate as given in Table 15.

TABLE 15/H.245

Mantissa of estimatedReceivedJitterMantissa in JitterIndication

estimatedReceivedJitterMantissa	Mantissa
0	1
1	2.5
2	5
3	7.5

estimatedReceivedJitterExponent indicates the exponent of the jitter estimate as given in Table 16.

TABLE 16/H.245

Exponent of estimatedReceivedJitterExponent in JitterIndication

estimatedReceivedJitterExponent	Exponent
0	Out of range
1	1 μ s
2	10 μ s
3	100 μ s
4	1 ms
5	10 ms
6	100 ms
7	1 s

The jitter estimate is obtained by multiplying the mantissa by the exponent, unless estimatedReceivedJitterExponent is equal to zero, in which case the estimate is just known to be more than 7.5 s.

Superseded by a more recent version

skippedFrameCount indicates how many frames have been skipped by the decoder since the last JitterControlPDU was received. Since the maximum value that can be encoded is 15, if this option is implemented, this information must be transmitted before more than 15 frames have been skipped.

NOTE – Since frames are skipped when the decoder buffer underflows, additional jitter may cause the decoder buffer to underflow more or less often than the encoder expects frame skips to happen.

additionalDecoderBuffer indicates the additional size of the video decoder buffer over and above that required by the indicated profile and level. This is defined in the same way as vbv_buffer_size H.262 [14].

7.10.4 H.223 Skew Indication

This is used to indicate to the far-end terminal the average amount of time skew between two logical channels.

logicalChannelNumber1 and logicalChannelNumber2 are logical channel numbers of opened logical channels.

skew is measured in milliseconds, and indicates the delay that must be applied to data belonging to logicalChannelNumber2 as measured at the multiplex, to achieve synchronization with logicalChannelNumber1 as measured at the multiplex. The actual delay necessary for synchronization is dependent on decoder implementation, and is a local matter for the receiver.

7.10.5 New ATM Virtual Channel Indication

This is used to indicate the parameters of an ATM virtual channel that the terminal intends to open.

resourceID is used to identify the ATM virtual channel. The means by which this parameter is associated with an ATM virtual channel is not specified in this Recommendation.

bitRate indicates the bit rate, measured at the AAL-SAP, of the virtual channel, and is measured in units of 64 kbit/s.

bitRateLockedToPCRClock indicates that the bit rate of the virtual channel is clocked to the clock used to produce H.222.0 clock reference values (program clock reference or System clock reference).

bitRateLockedToNetworkClock indicates that the bit rate of the virtual channel is clocked to the local network clock. This does not guarantee that the bit rate clock will be locked to the local network at the receiver, as common network clocks may not be available.

aal indicates which ATM Adaptation Layer will be used, and its parameters.

The sequence aal1 indicates which of the options for ATM adaptation layer 1, as specified in Recommendation I.363 [19] are supported. The codepoints are defined in Table 1.

The sequence aal5 indicates which of the options for ATM adaptation layer 5, as specified in Recommendation I.363 [19] are supported. forwardMaximumSDUSize and backwardMaximumSDUSize indicate the maximum CPCS-SDU size in the forward and reverse directions, measured in octets.

multiplex indicates the type of multiplex that will be used on the ATM virtual channel. The options are noMultiplex (no H.222.0 multiplex), H.222.0 Transport Stream and H.222.0 Program Stream.

7.10.6 User Input

This is used for User Input messages.

alphanumeric is a string of characters coded according to Recommendation T.51 [23]. This could be used for key-pad input, an equivalent to DTMF.

NOTE – Any data that is carried in Recommendation H.245, including user input messages, will not be encrypted.

8 Procedures

8.1 Introduction

This clause defines generic multimedia system control procedures that use the messages defined in this Recommendation. Recommendations using this Recommendation shall indicate which of these procedures are applicable, as well as defining any specific requirements.

Superseded by a more recent version

Procedures to perform the following functions are described in this clause:

- master-slave determination;
- terminal capability exchange;
- logical channel signalling;
- bidirectional logical channel signalling;
- receive terminal close logical channel request;
- slave terminal open bidirectional logical channel request;
- H.223 multiplex table entry modification;
- receiver to transmitter transmit mode request;
- round trip delay determination.

8.1.1 Method of specification

Procedures are generally specified in this subclause using SDLs. The SDL provides a graphical specification of the procedures, and includes specification of actions in the event of exception conditions.

8.1.2 Communication between protocol entity and protocol user

The interaction with the user of a particular function is specified in terms of primitives transferred at the interface between the protocol entity and the protocol user. Primitives are for the purpose of defining protocol procedures and are not intended to specify or constrain implementation. There may be a number of parameters associated with each primitive.

To assist in the specification, protocol states are defined. These states are conceptual and reflect general conditions of the protocol entity in the sequences of primitives exchanged between the protocol entity and the user, and the exchange of messages between the protocol entity and its peer.

For each protocol entity the allowed sequence of primitives between the user and the protocol entity is defined using a state transition diagram. The allowed sequence constrains the actions of the user, and defines the possible responses from the protocol entity.

A primitive parameter described as being null, is equivalent to the parameter not being present.

8.1.3 Peer-to-peer communication

Protocol information is transferred to the peer protocol entity via the relevant messages defined in clause 6. Some protocol entities described have state variables associated with them. A number of protocol entities described also have timers associated with them.

A timer is identified by the notation T_n , when n is a number. In the SDL diagrams setting a timer means that a timer is loaded with a specified value and the timer is started. Resetting a timer means that a timer is stopped with its value at the time of reset being retained. Timer expiry means that a timer has run for its specified time and has reached the value of zero.

A protocol entity may also have associated parameters. A parameter is identified by the notation N_n , where n is a number.

These timers and counters are listed in Appendix III.

Some protocol entities define an error primitive to report protocol error conditions to a management entity.

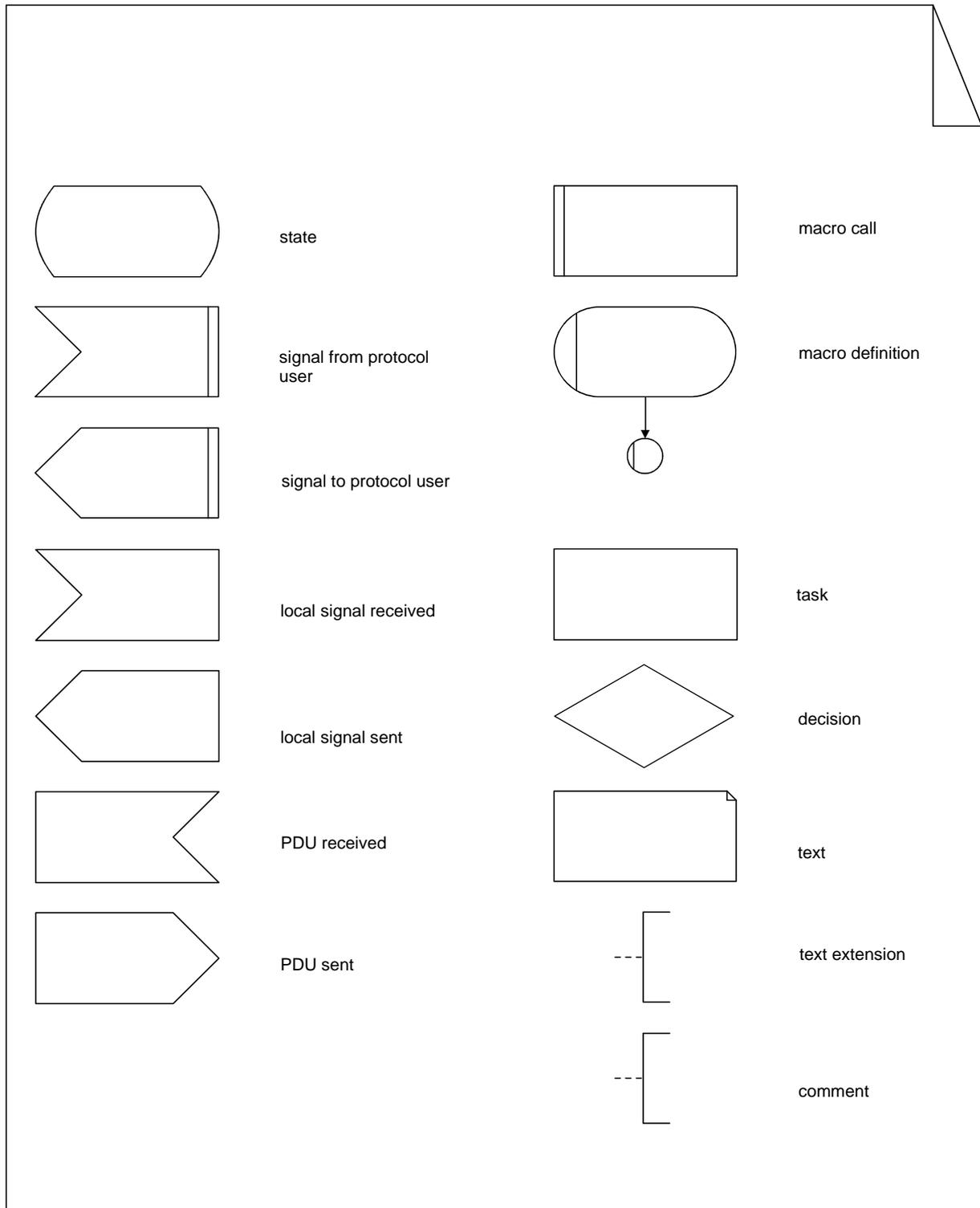
8.1.4 SDL diagrams

The SDL diagrams show actions to the allowed interactions with the protocol user, and to reception of messages from the peer protocol entity. Primitives which are not allowed for a given state, as specified by the state transition diagrams, are not shown in the SDL diagrams. However the responses to the reception of inappropriate messages are described in the SDL diagrams.

Superseded by a more recent version

8.1.5 SDL key

The SDL key is shown in Figure 1.



T1519120-95/d001

FIGURE 1/H.245

SDL key

Superseded by a more recent version

8.2 Master slave determination procedures

8.2.1 Introduction

Conflicts may arise when two or more terminals involved in a call initiate similar events simultaneously, for which resources are available for only one occurrence of the event, e.g. opening of logical channels. To resolve such conflicts, one terminal may act as a master and the other terminal(s) may act as slave terminal(s). The procedures described here allow terminals in the call to determine which is the master terminal and which is the slave terminal(s).

The protocol described here is referred to as the Master Slave Determination Signalling Entity (MSDSE). There is one instance of the MSDSE in each terminal involved in a call.

Either terminal may initiate the master slave determination process by issuing the DETERMINE.request primitive to its MSDSE. The result of the procedure is returned by the DETERMINE.indication and the DETERMINE.confirm primitives. While the DETERMINE.indication primitive indicates the result, it does not indicate that the result is known at the remote terminal. The DETERMINE.confirm primitive indicates the result and confirms that it is also known at the remote terminal.

A terminal shall respond to procedures that rely on the knowledge of the result and are initiated by the remote terminal any time after the status determination result is known as the local terminal. This may be before the local terminal has received confirmation that the remote terminal also has knowledge of the result. A terminal shall not initiate procedures that rely on knowledge of the result until it has received confirmation that the remote terminal also has knowledge of the result.

The following text provides an overview of the operation of the protocol. In the case of any discrepancy with the formal specification of the protocol that follows, the formal specification will supersede.

8.2.1.1 Protocol overview – Initiation by local user

A master slave determination procedure is initiated when the DETERMINE.request primitive is issued by the MSDSE user. A MasterSlaveDetermination message is sent to the peer MSDSE, and timer T106 is started. If a MasterSlaveDeterminationAck message is received in response to the MasterSlaveDetermination message, then timer T106 is stopped and the user is informed with the DETERMINE.confirm primitive that the master slave determination procedure was successful and a MasterSlaveDeterminationAck message is sent to the peer MSDSE. If however a MasterSlaveDeterminationReject message is received in response to the MasterSlaveDetermination message, then a new status determination number is generated, timer T106 is restarted, and another MasterSlaveDetermination message is sent. If after sending a MasterSlaveDetermination message N100 times, a MasterSlaveDeterminationAck still has not been received, then timer T106 is stopped and the user is informed with the REJECT.indication primitive that the master slave determination procedure has failed to produce a result.

If timer T106 expires, then the MSDSE user is informed with the REJECT.indication primitive and a MasterSlaveDeterminationRelease message is sent to the peer MSDSE.

8.2.1.2 Protocol overview – Initiation by remote user

When a MasterSlaveDetermination message is received at the MSDSE, a status determination procedure is initiated. If the status determination procedure returns a determinate result, then the user is informed of the master slave determination result with the DETERMINE.indication primitive, a MasterSlaveDeterminationAck message is sent to the peer MSDSE, and timer T106 is started. If the MasterSlaveDeterminationAck message is received in response to the MasterSlaveDeterminationAck message, then timer T106 is stopped and the user is informed with the DETERMINE.confirm primitive that the master slave determination procedure was successful.

If timer T106 expires, then the MSDSE user is informed with the REJECT.indication primitive.

If however the status determination procedure returns an indeterminate result, then the MasterSlaveDeterminationReject message is sent to the peer MSDSE.

8.2.1.3 Protocol overview – Simultaneous initiation

When a MasterSlaveDetermination message is received at the MSDSE that itself has already initiated a status determination procedure, and is awaiting a MasterSlaveDeterminationAck or MasterSlaveDeterminationReject message, then a status determination procedure is initiated. If the status determination procedure returns a determinate result, the

Superseded by a more recent version

MSDSE responds as if the procedure was initiated by the remote user, and the procedures, described above for this condition, apply.

If however the status determination procedure returns an indeterminate result, then a new status determination number is generated, and the MSDSE responds as if the procedure was again initiated by the local MSDSE user as described above.

8.2.1.4 Status determination procedure

The following procedure is used to determine which terminal is the master from the terminalType and statusDeterminationNumber values. Firstly, the terminalType values are compared and the terminal with the larger terminal type number is determined as the master. If the terminal type numbers are the same, the statusDeterminationNumbers are compared using modulo arithmetic to determine which is master.

If both terminals have equal terminalType field values and the difference between statusDeterminationNumber field values modulo 2^{24} is 0 or 2^{23} , an indeterminate result is obtained.

8.2.2 Communication between the MSDSE and the MSDSE user

8.2.2.1 Primitives between the MSDSE and the MSDSE user

Communication between the MSDSE and the MSDSE user is performed using the primitives shown in Table 17.

TABLE 17/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
DETERMINE	– (Note 1)	TYPE	Not defined (Note 2)	TYPE
REJECT	Not defined	–	Not defined	not defined
ERROR	Not defined	ERRCODE	Not defined	not defined

NOTE 1 – “–” means no parameters.
NOTE 2 – “not defined” means that this primitive is not defined.

8.2.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The DETERMINE primitive is used to initiate, and to return the result from, the master slave determination procedure.

The DETERMINE.request primitive is used to initiate the master slave determination procedure.

The DETERMINE.indication primitive is used to indicate the result of the master slave determination procedure. As the result of the procedure may not be known at the remote terminal, the terminal shall not initiate any procedures that rely on knowledge of the result, although it shall respond to any procedures that rely on knowledge of the result.

The DETERMINE.confirm primitive is used to indicate the result of the master slave determination procedure and that the result of the procedure is known at both terminals. The terminal may initiate, and shall respond to, any procedures that rely on knowledge of the result.

- b) The REJECT primitive indicates that the master slave determination procedure was unsuccessful.
- c) The ERROR primitive reports MSDSE errors to a management entity.

Superseded by a more recent version

8.2.2.3 Parameter definition

The definition of the primitive parameters shown in Table 17 are as follows:

- a) The TYPE parameter indicates the terminal status. It has the value of “MASTER” or “SLAVE”.
- b) The ERRCODE value indicates the type of MSDSE error. Table 21 indicates the values that the ERRCODE parameter may take.

8.2.2.4 MSDSE states

The following states are used to specify the allowed sequence of primitives between the MSDSE and the MSDSE user.

State 0: IDLE

No master slave determination procedure has been initiated.

State 1: OUTGOING AWAITING RESPONSE

The local MSDSE user has requested a master slave determination procedure. A response from the remote MSDSE is awaited.

State 2: INCOMING AWAITING RESPONSE

The remote MSDSE has initiated a master slave determination procedure in the local MSDSE. An acknowledgement was sent to the remote MSDSE and a response from the remote MSDSE is awaited.

8.2.2.5 State transition diagram

The allowed sequence of primitives between the MSDSE and the MSDSE user is defined here. The allowed sequences are shown in Figure 2.

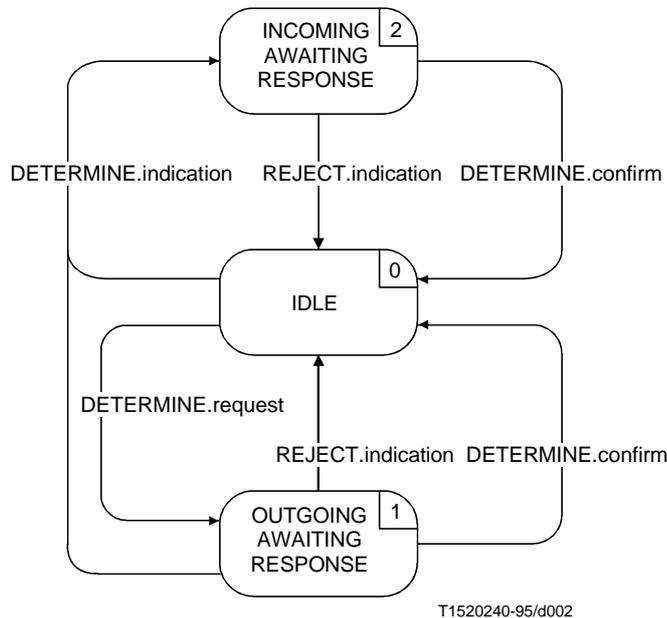


FIGURE 2/H.245

State transition diagram for sequence of primitives at MSDSE

8.2.3 Peer-to-peer MSDSE communication

8.2.3.1 MSDSE messages

Table 18 shows the MSDSE messages and fields, defined in clause 6, which are relevant to the MSDSE protocol.

Superseded by a more recent version

TABLE 18/H.245

MSDSE message names and fields

Function	Message	Field
Determination	MasterSlaveDetermination	terminalType statusDeterminationNumber
	MasterSlaveDeterminationAck	decision
	MasterSlaveDeterminationReject	cause
Error recovery	MasterSlaveDeterminationRelease	–

8.2.3.2 MSDSE state variables

The following MSDSE state variables are defined:

sv_TT

This state variable holds the terminal type number for this terminal.

sv_SDNUM

This state variable holds the status determination number for this terminal.

sv_STATUS

This state variable is used to store the result of the latest master slave determination procedure. It has values of “master”, “slave” and “indeterminate”.

sv_NCOUNT

This state variable is used to count the number of MasterSlaveDetermination messages that have been sent during the OUTGOING AWAITING RESPONSE state.

8.2.3.3 MSDSE timers

The following timer is specified for the outgoing MSDSE:

T106

This timer is used during the OUTGOING AWAITING RESPONSE state and during the INCOMING AWAITING RESPONSE state. It specifies the maximum allowed time during which no acknowledgement message may be received.

8.2.3.4 MSDSE counters

The following parameter is specified for the MSDSE:

N100

This parameter specifies the maximum value of sv_NCOUNT.

8.2.4 MSDSE procedures

8.2.4.1 Introduction

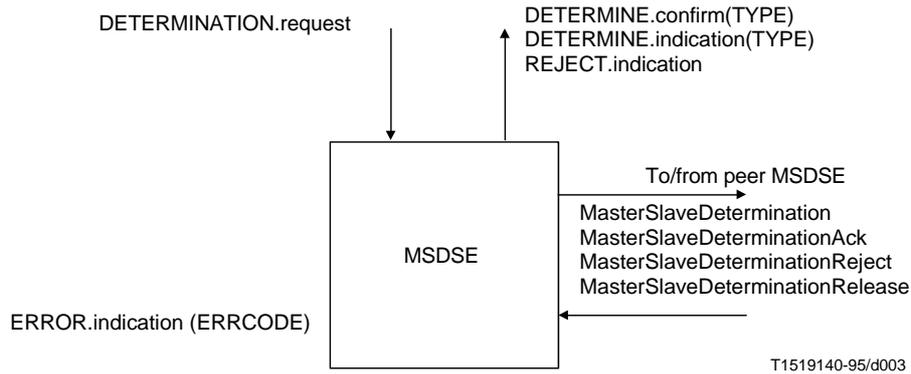
Figure 3 summarizes the MSDSE primitives and their parameters, and messages.

8.2.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams, the parameters of the indication and confirm primitives assume values as shown in Table 19.

Superseded by a more recent version

To/from MSDSE user



T1519140-95/d003

FIGURE 3/H.245

Primitives and messages in the MSDSE

TABLE 19/H.245

Default primitive parameter values

Primitive	Parameter	Default value
DETERMINE.confirm	TYPE	MasterSlaveDeterminationAck.decision
DETERMINE.indication	TYPE	sv_STATUS

8.2.4.3 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 20.

TABLE 20/H.245

Default message field values

Message	Field	Default value
MasterSlaveDetermination	terminalType statusDeterminationNumber	sv_TT sv_SDNUM
MasterSlaveDeterminationAck	decision	Opposite of sv_STATUS, i.e if (sv_STATUS = master) decision = slave if (sv_STATUS = slave) decision = master
MasterSlaveDetermination Reject	cause	identicalNumbers

8.2.4.4 ERRCODE parameter values

Table 21 shows the values that the ERRCODE parameter of the ERROR.indication primitive may take for the MSDSE.

Superseded by a more recent version

TABLE 21/H.245

ERRCODE parameter values at MSDSE

Error type	Error code	Error condition	State
No response from remote MSDSE	A	local timer T106 expiry	OUTGOING AWAITING RESPONSE INCOMING AWAITING RESPONSE
Remote sees no response from local MSDSE	B	remote timer T106 expiry	OUTGOING AWAITING RESPONSE INCOMING AWAITING RESPONSE
Inappropriate message	C	MasterSlaveDetermination	INCOMING AWAITING RESPONSE
	D	MasterSlaveDetermination Reject	INCOMING AWAITING RESPONSE
Inconsistent field value	E	MasterSlaveDetermination Ack.decision != sv_STATUS	INCOMING AWAITING RESPONSE
Maximum number of retries	F	sv_NCOUNT = N100	OUTGOING AWAITING RESPONSE

8.2.4.5 SDLs

The MSDSE procedures are expressed in SDL form in Figure 4.

terminalTypeProcess is a process that returns a number that identifies different types of terminal, such as, terminals, MCUs and gateways.

randomNumber is a process that returns a random number in the range $0..2^{24}-1$.

8.3 Capability exchange procedures

8.3.1 Introduction

These procedures are used by terminals to communicate their capabilities, and are referred to as the Capability Exchange Signalling Entity (CESE). Procedures are specified in terms of primitives and states at the interface between the CESE and the CESE user. Protocol information is transferred to the peer CESE via relevant messages defined in clause 6. There is an outgoing CESE and an incoming CESE. At each of the outgoing and incoming ends there is one instance of the CESE for each call.

All terminals intended for use in point-to-point applications or those connected to an MCU shall be able to identify a TerminalCapabilitySet and its structure, and such capability values therein that are mandatory for those applications; any unrecognized capability values shall be ignored, and no fault shall be implied.

The capability exchange may be performed at any time. The capability exchange may signal both changed and unchanged capabilities. Unchanged capabilities should not be sent repetitively without strong cause.

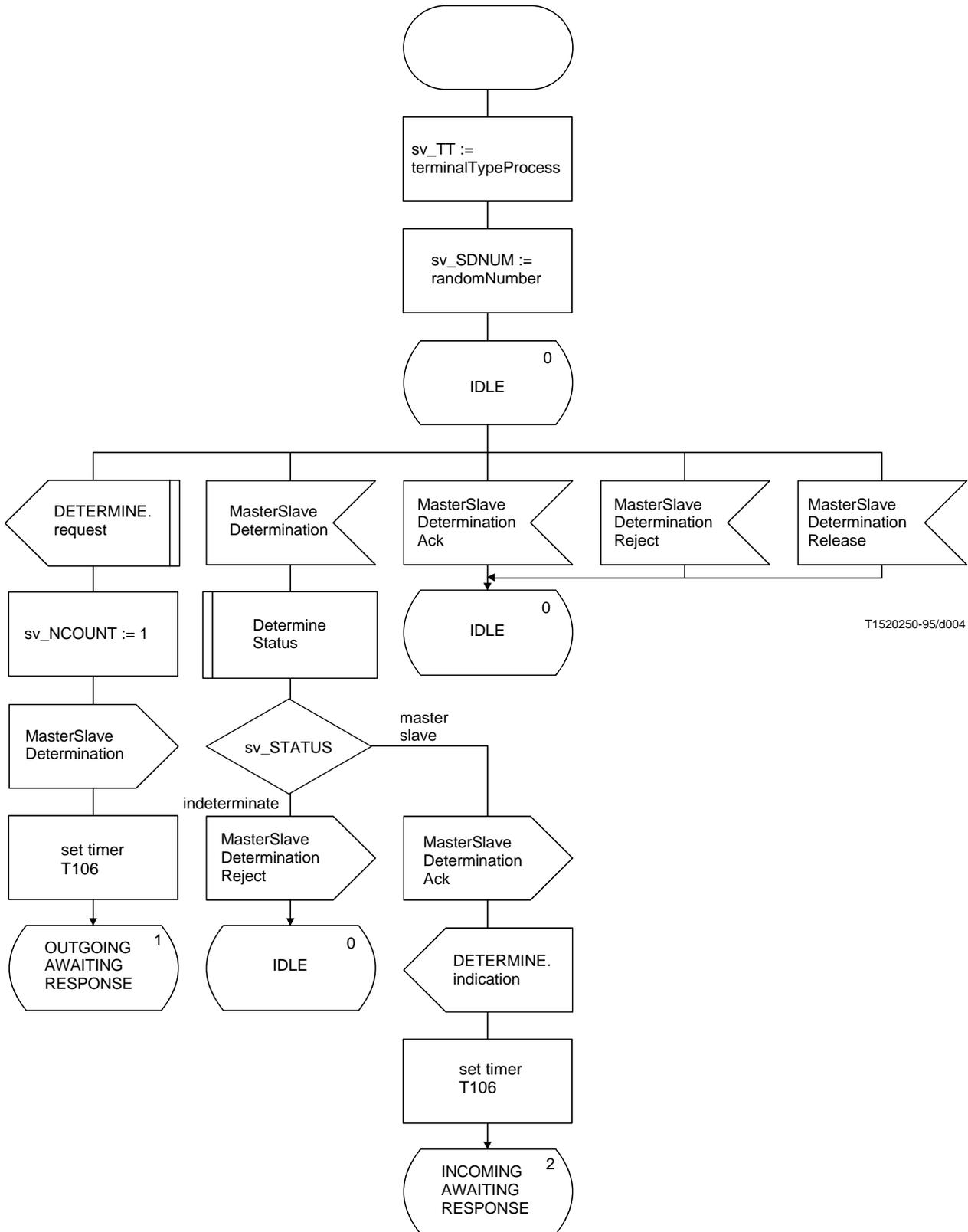
The following text provides an overview of the operation of the protocol. In the case of any discrepancy with the formal specification of the protocol that follows, the formal specification will supersede.

8.3.1.1 Protocol overview – Outgoing CESE

A capability exchange is initiated when the TRANSFER.request primitive is issued by the user at the outgoing CESE. A TerminalCapabilitySet message is sent to the peer incoming CESE, and timer T101 is started. If a TerminalCapabilitySetAck message is received in response to the TerminalCapabilitySet message, then timer T101 is stopped and the user is informed with the TRANSFER.confirm primitive that the capability exchange was successful. If however a TerminalCapabilitySetReject message is received in response to the TerminalCapabilitySet message, then timer T101 is stopped and the user is informed with the REJECT.indication primitive that the peer CESE user has refused the capability exchange.

Superseded by a more recent version

If timer T101 expires then the outgoing CESE user is informed with the REJECT.indication primitive and a TerminalCapabilitySetRelease message is sent.



T1520250-95/d004

FIGURE 4 i)/H.245
MSDSE SDL

Superseded by a more recent version

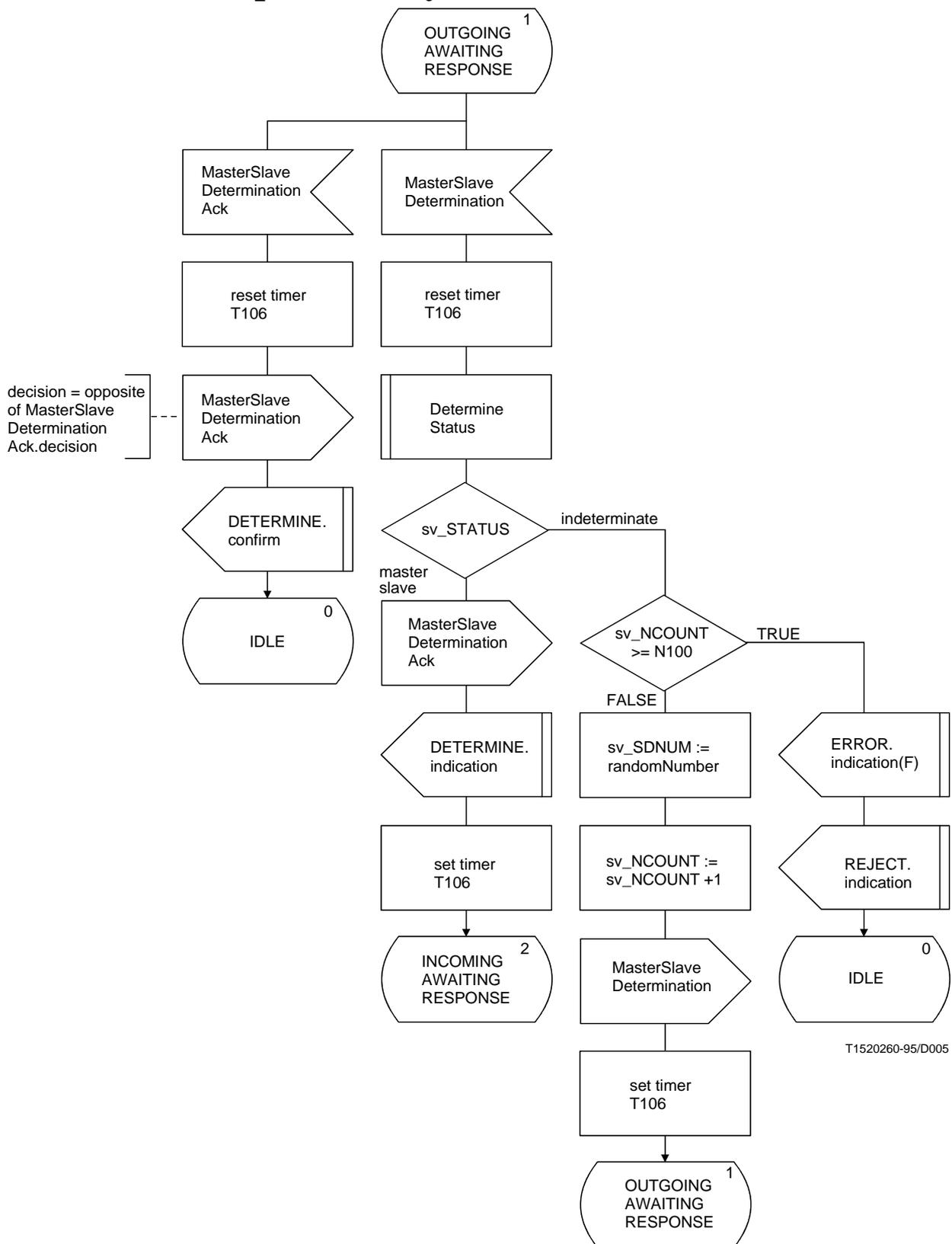
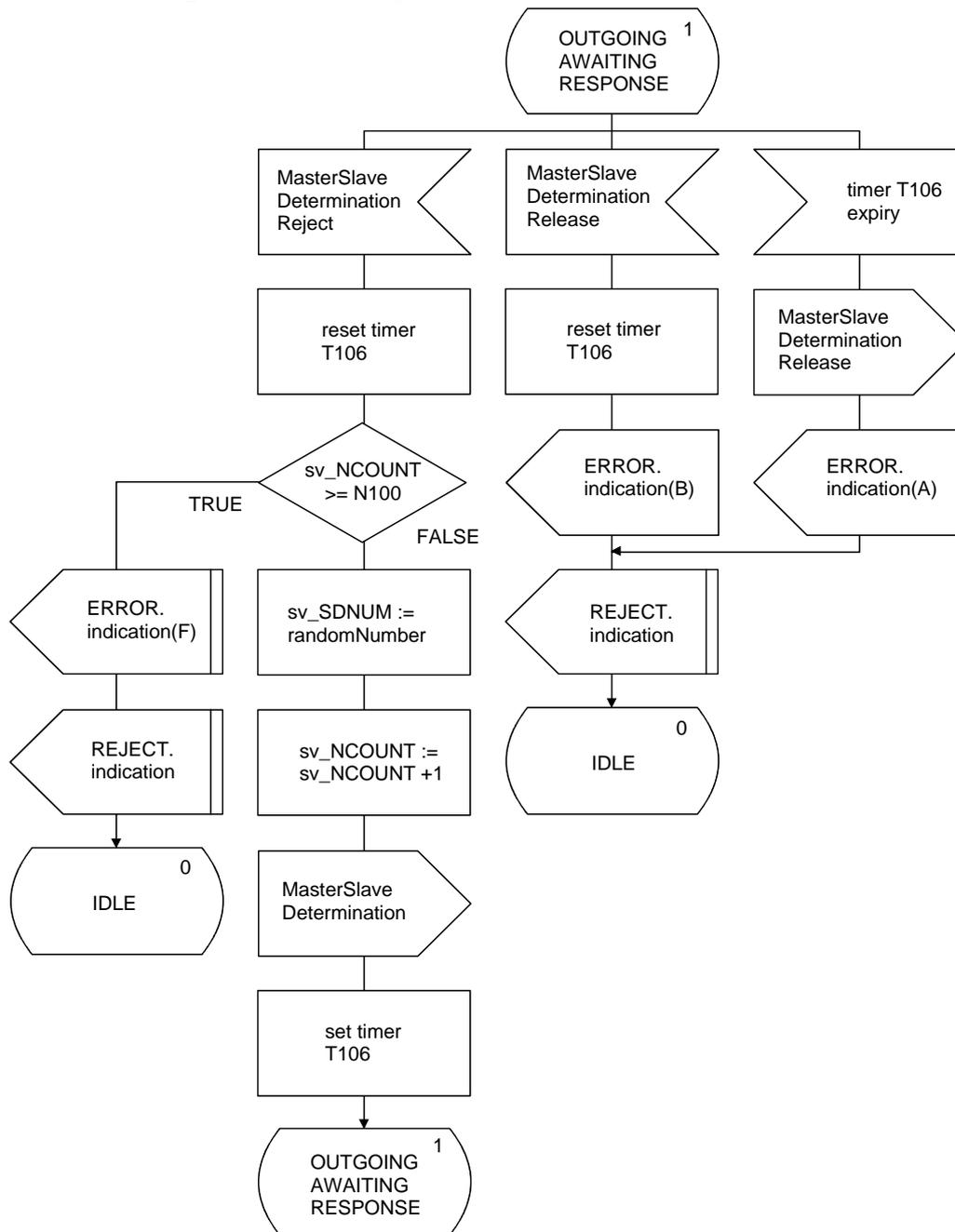


FIGURE 4 ii)/H.245
MSDSE SDL

Superseded by a more recent version

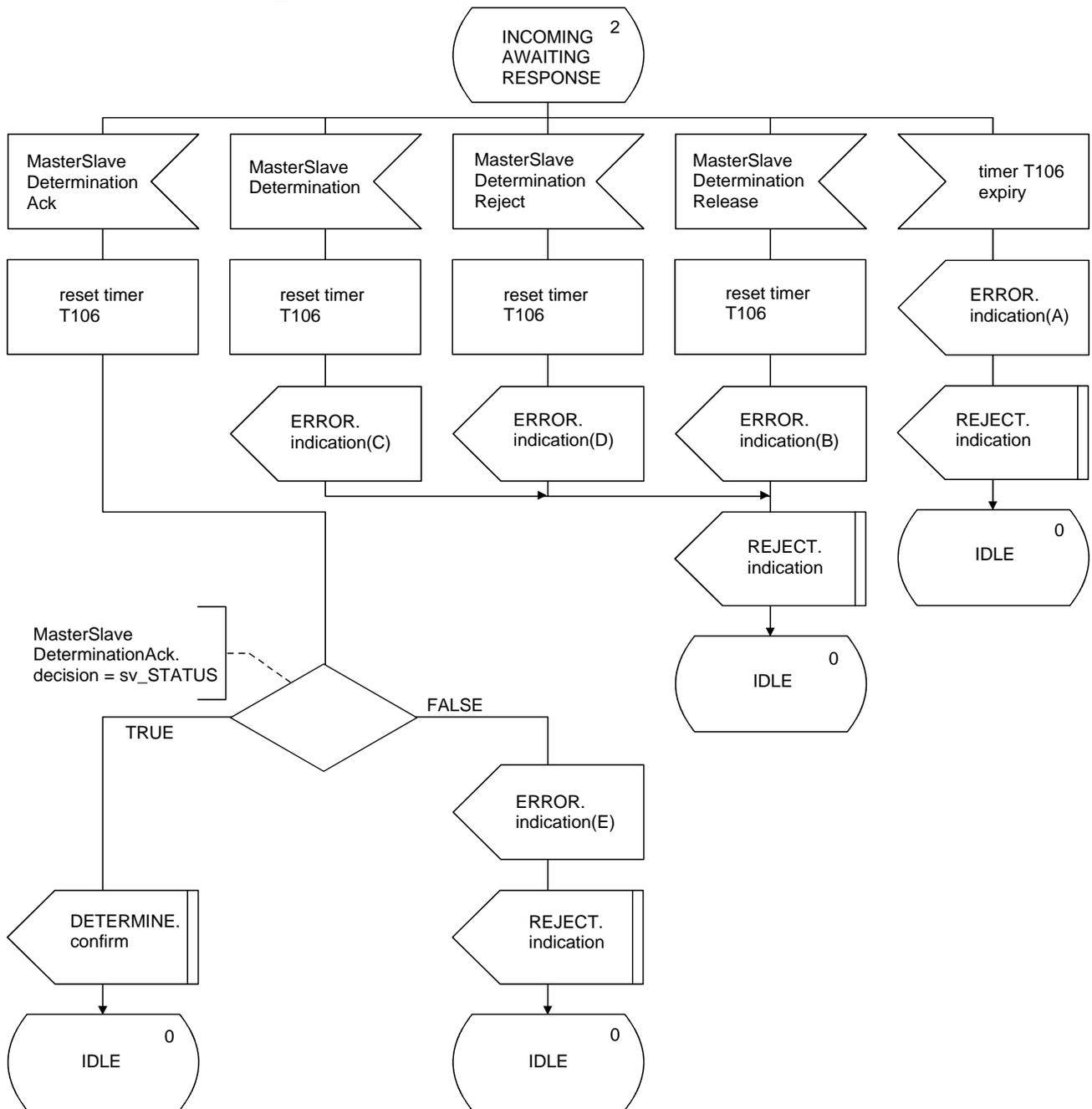


T1520270-95/d006

FIGURE 4 iii)/H.245

MSDSE SDL

Superseded by a more recent version



T1520280-95/d007

FIGURE 4 iv)/H.245

MSDSE SDL

Superseded by a more recent version

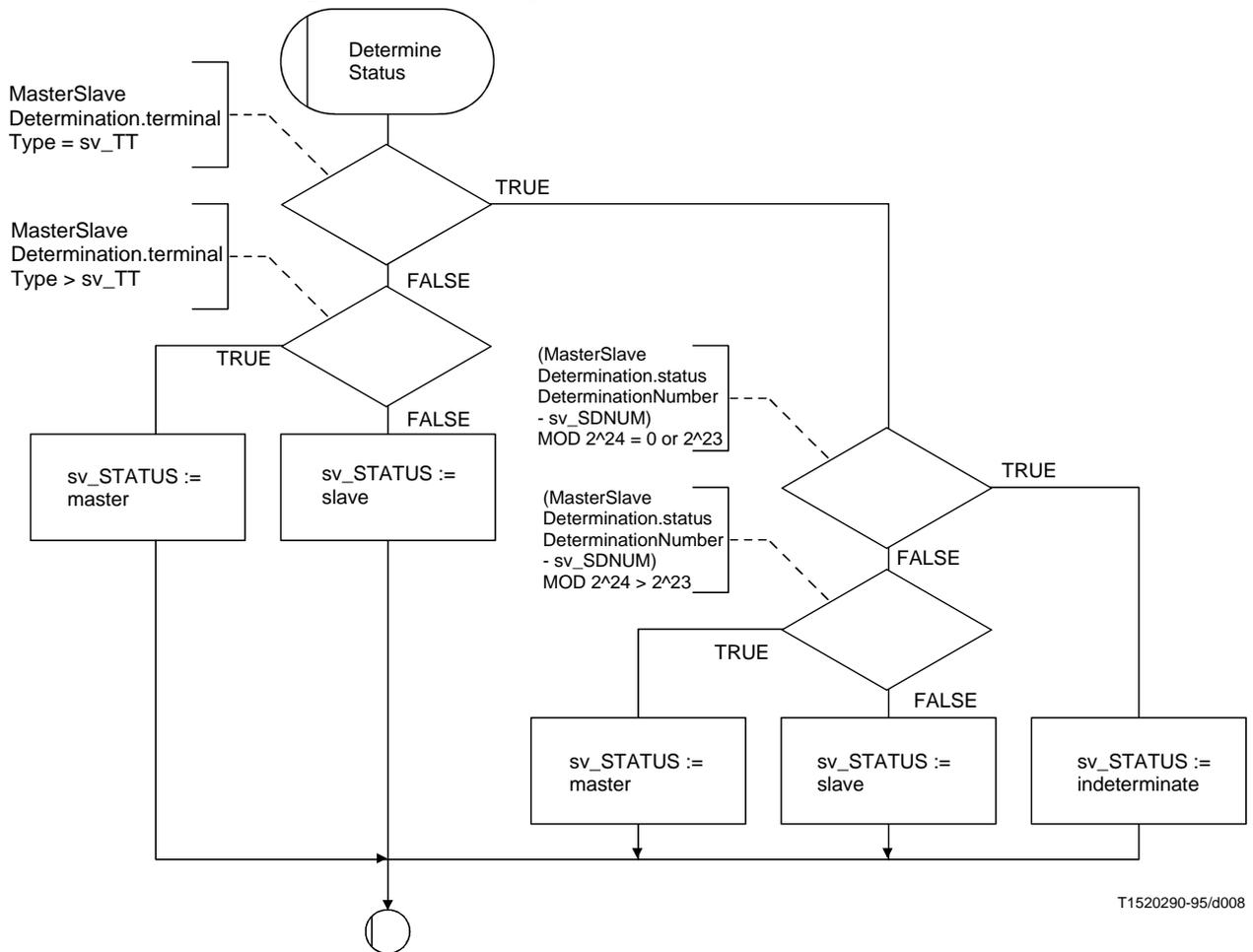


FIGURE 4 v)/H.245
MSDSE SDL

8.3.1.2 Protocol overview – Incoming CESE

When a TerminalCapabilitySet message is received at the incoming CESE, the user is informed of the capability exchange request with the TRANSFER.indication primitive. The incoming CESE user signals acceptance of the capability exchange request by issuing the TRANSFER.response primitive, and a TerminalCapabilitySetAck message are sent to the peer outgoing CESE.

8.3.2 Communication between CESE and CESE user

8.3.2.1 Primitives between CESE and CESE user

Communication between the CESE and CESE user, is performed using the primitives shown in Table 22.

8.3.2.2 Primitive definition

The definition of these primitives is as follows:

- the TRANSFER primitives are used for transfer of the capability exchange;
- the REJECT primitives are used to reject a capability descriptor entry, and to terminate a current capability transfer.

Superseded by a more recent version

TABLE 22/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
TRANSFER	PROTOID	PROTOID	– (Note 1)	–
	MUXCAP CAPTABLE CAPDESCRIPTORS	MUXCAP CAPTABLE CAPDESCRIPTORS		
REJECT	CAUSE	SOURCE CAUSE	Not defined (Note 2)	Not defined

NOTE 1 – “–” means no parameters.
NOTE 2 – “Not defined” means that this primitive is not defined.

8.3.2.3 Parameter definition

The definition of the primitive parameters shown in Table 22 are as follows:

- The PROTOID parameter is the protocol identifier parameter. This parameter is mapped to the protocol identifier field of the TerminalCapabilitySet message and carried transparently to the peer CESE user. This parameter is mandatory.
- The MUXCAP parameter is the multiplex capability parameter. This parameter is mapped to the multiplex Capability field of the TerminalCapabilitySet message and carried transparently to the peer CESE user. This parameter is optional.
- The CAPTABLE parameter is the capability table parameter. There may be zero or more capability table entries described within this parameter. This parameter is mapped to the capability Table field of the TerminalCapabilitySet message and carried transparently to the peer CESE user. This parameter is optional.
- The CAPDESCRIPTORS parameter is the capability descriptors parameter. There may be one or more capability descriptors described within in this parameter. This parameter is mapped to the capability Descriptors field of the TerminalCapabilitySet message and carried transparently to the peer CESE user. This parameter is optional.
- The SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of “USER” or “PROTOCOL”. The latter case may occur as the result of a timer expiry.
- The CAUSE parameter indicates the reason for rejection of a CAPTABLE or CAPDESCRIPTORS parameter. The CAUSE parameter is not present when the SOURCE parameter indicates “PROTOCOL”.

8.3.2.4 CESE states

The following states are used to specify the allowed sequence of primitives between the CESE and the CESE user.

The states for an outgoing CESE are:

State 0: IDLE

The CESE is idle.

State 1: AWAITING RESPONSE

The CESE is waiting for a response from the remote CESE.

Superseded by a more recent version

The states for an incoming CESE are:

State 0: IDLE

The CESE is idle.

State 1: AWAITING RESPONSE

The CESE is waiting for a response from the CESE user.

8.3.2.5 State transition diagram

The allowed sequence of primitives between the CESE and the CESE user is defined here. The allowed sequence of primitives relates to states of the CESE as viewed from the CESE user. The allowed sequences are specified separately for each of an outgoing CESE and an incoming CESE, as shown in Figures 5 and 6 respectively.

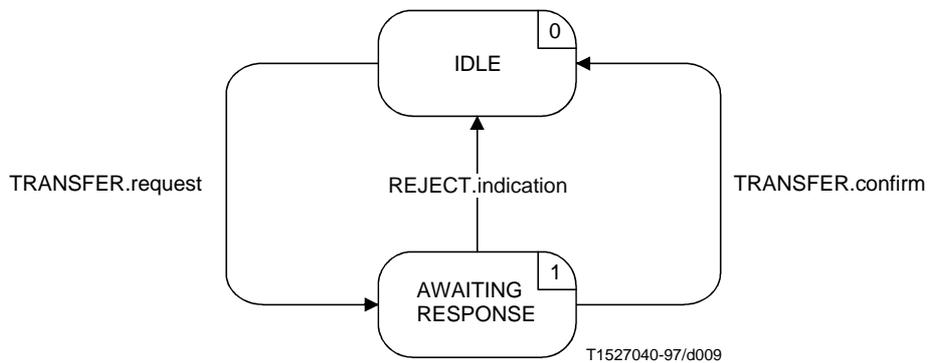


FIGURE 5/H.245

State transition diagram for sequence of primitives at CESE outgoing

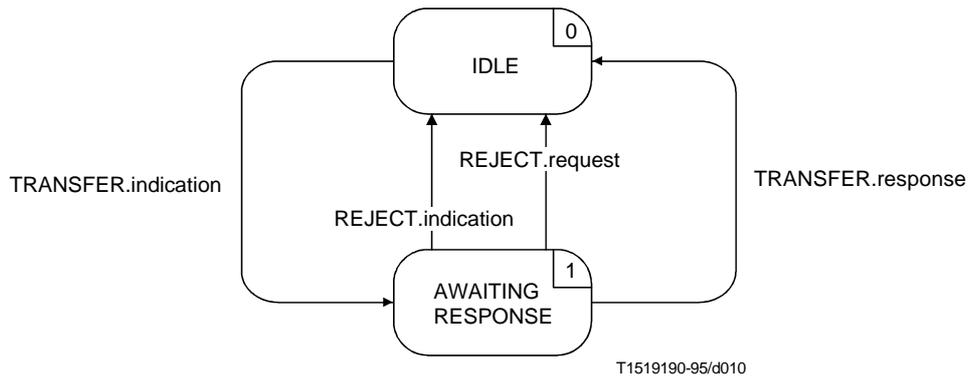


FIGURE 6/H.245

State transition diagram for sequence of primitives at CESE incoming

8.3.3 Peer-to-peer CESE communication

8.3.3.1 Messages

Table 23 shows the CESE messages and fields, defined in clause 6, which are relevant to the CESE protocol.

Superseded by a more recent version

TABLE 23/H.245

CESE message names and fields

Function	Message	Direction	Field
Transfer	TerminalCapabilitySet	O → I	sequenceNumber protocolIdentifier multiplexCapability capabilityTable capabilityDescriptors
	TerminalCapabilitySetAck	O ← I	sequenceNumber
Reject	TerminalCapabilitySetReject	O ← I	sequenceNumber cause
Reset	TerminalCapabilitySetRelease	O → I	–
O Outgoing I Incoming			

8.3.3.2 CESE state variables

The following state variables are defined at the outgoing CESE:

out_SQ

This state variable is used to indicate the most recent TerminalCapabilitySet message. It is incremented by one and mapped to the TerminalCapabilitySet message sequenceNumber field before transmission of the TerminalCapabilitySet message. Arithmetic performed on out_SQ is modulo 256.

The following state variables are defined at the incoming CESE:

in_SQ

This state variable is used to store the value of the sequenceNumber field of the most recently received TerminalCapabilitySet message. The TerminalCapabilitySetAck and TerminalCapabilitySetReject messages have their sequenceNumber fields set to the value of in_SQ, before being sent to the peer CESE.

8.3.3.3 CESE timers

The following timer is specified for the outgoing CESE:

T101

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no TerminalCapabilitySetAck or TerminalCapabilitySetReject message may be received.

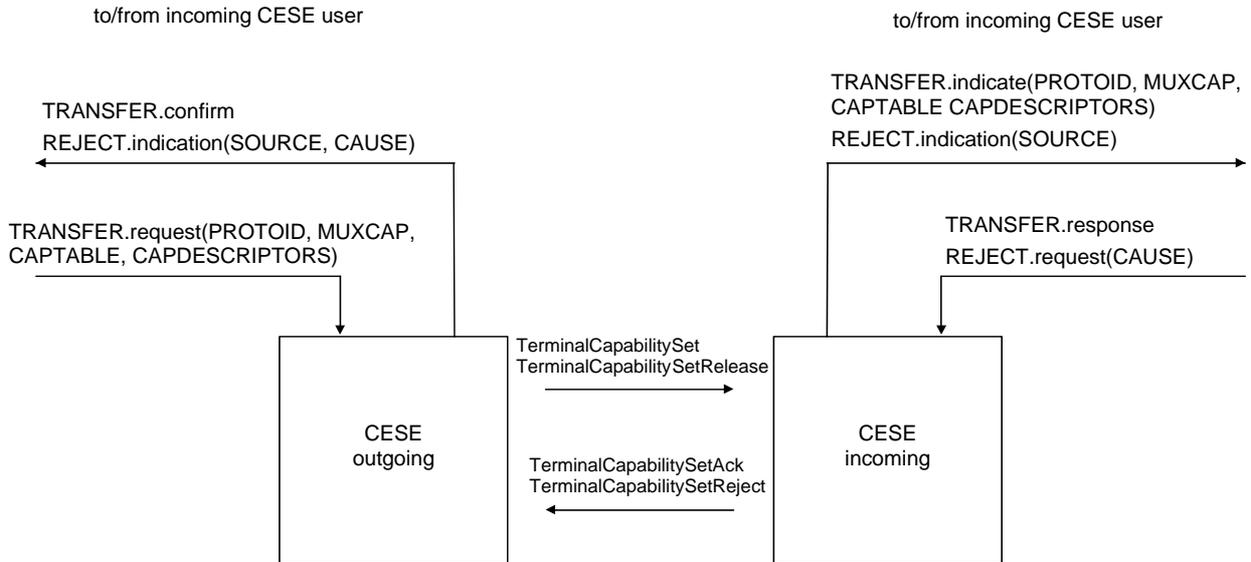
8.3.4 CESE procedures

Figure 7 summarizes the CESE primitives and their parameters, and messages, for each of the outgoing and incoming CESE.

8.3.4.1 Primitive parameter default values

Where not explicitly stated in the SDL diagrams, the parameters of the indication and confirm primitives assume values as shown in Table 24.

Superseded by a more recent version



T1519200-95/d011

FIGURE 7/H.245

Primitives and messages in the Capability Exchange Signalling Entity

TABLE 24/H.245

Default primitive parameter values

Primitive	Parameter	Default value
TRANSFER.indication	PROTOID MUXCAP CAPTABLE CAPDESCRIPTORS	TerminalCapabilitySet.protocolIdentifier TerminalCapabilitySet.multiplexCapability TerminalCapabilitySet.capabilityTable TerminalCapabilitySet.capabilityDescriptors
REJECT.indication	SOURCE CAUSE	USER null

8.3.4.2 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 25.

Superseded by a more recent version

TABLE 25/H.245

Default message field values

Message	Field	Default value (Note)
TerminalCapabilitySet	sequenceNumber protocolIdentifier multiplexCapability capabilityTable capabilityDescriptors	out_SQ TRANSFER.request(PROTOD) TRANSFER.request(MUXCAP) TRANSFER.request(CAPTABLE) TRANSFER.request(CAPDESCRIPTORS)
TerminalCapabilitySetAck	sequenceNumber	in_SQ
TerminalCapabilitySetReject	sequenceNumber cause	in_SQ REJECT.request(CAUSE)
TerminalCapabilitySetRelease	–	–

NOTE – A message field shall not be coded, if the corresponding primitive parameter is null, i.e. not present.

8.3.4.3 SDLs

The outgoing CESE and the incoming CESE procedures are expressed in SDL form in Figures 8 and 9, respectively.

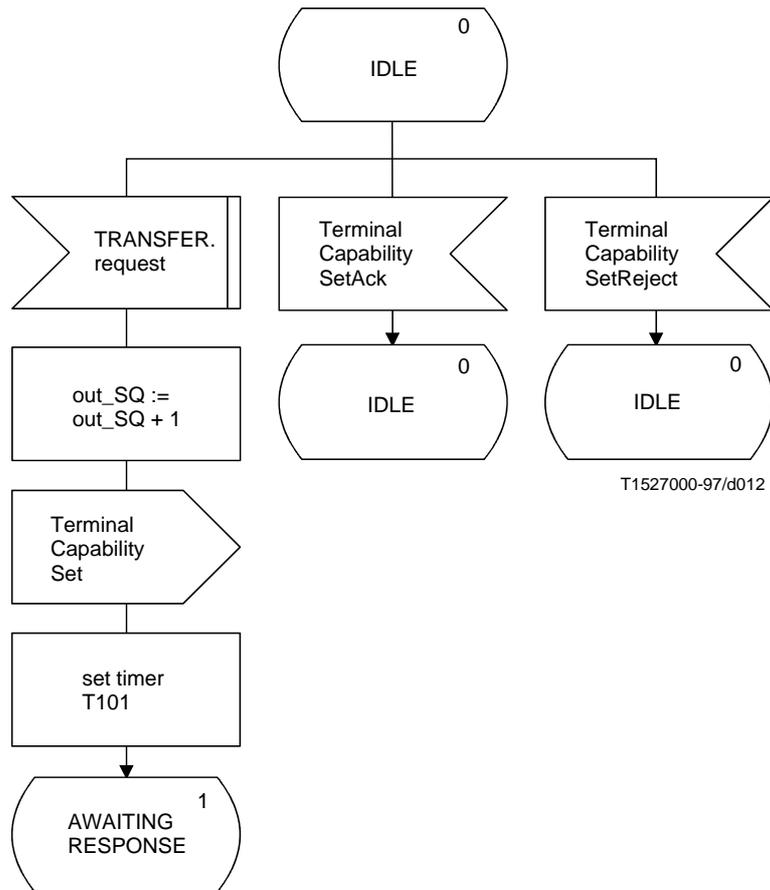
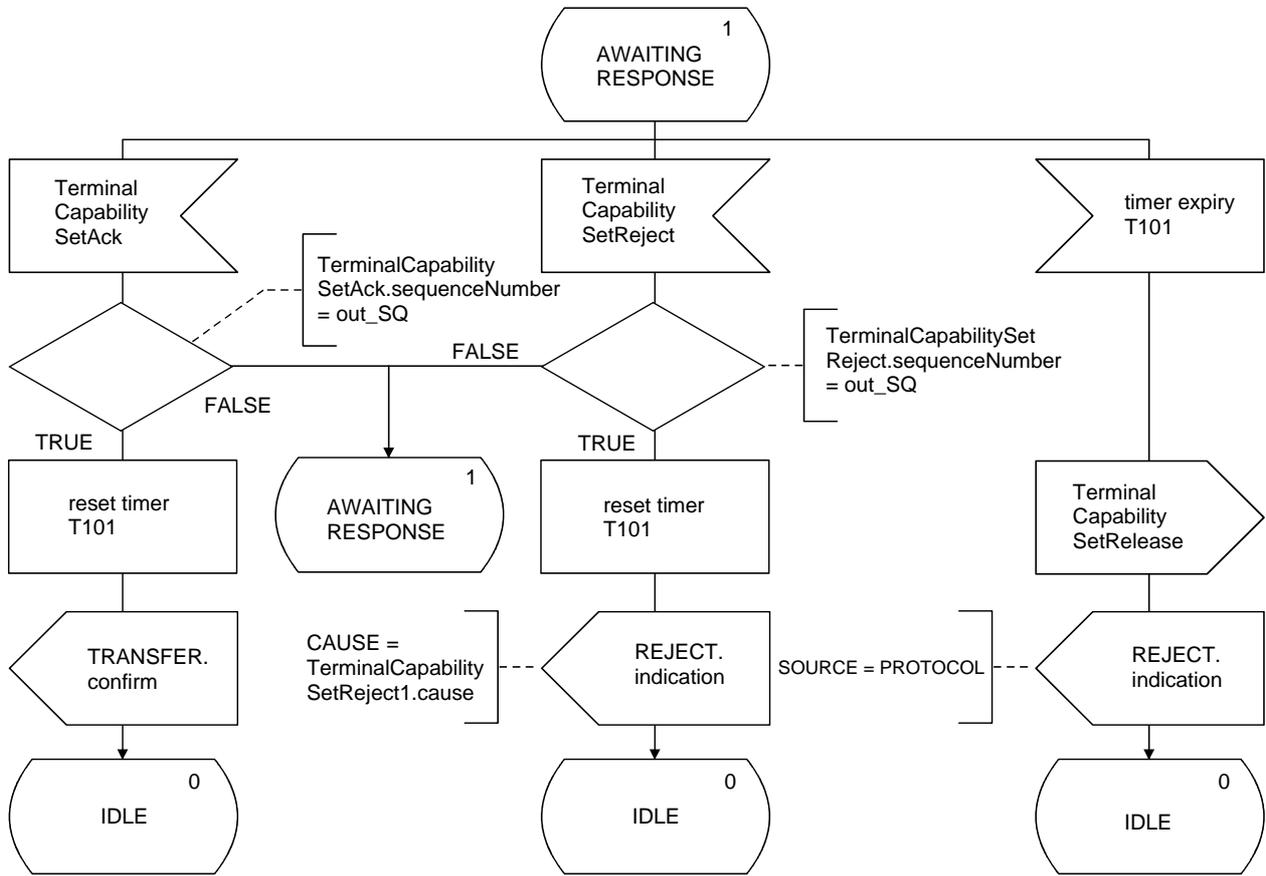


FIGURE 8 i)/H.245

Outgoing CESE SDL

Superseded by a more recent version



T1520300-95/d013

FIGURE 8 ii)/H.245
Outgoing CESE SDL

Superseded by a more recent version

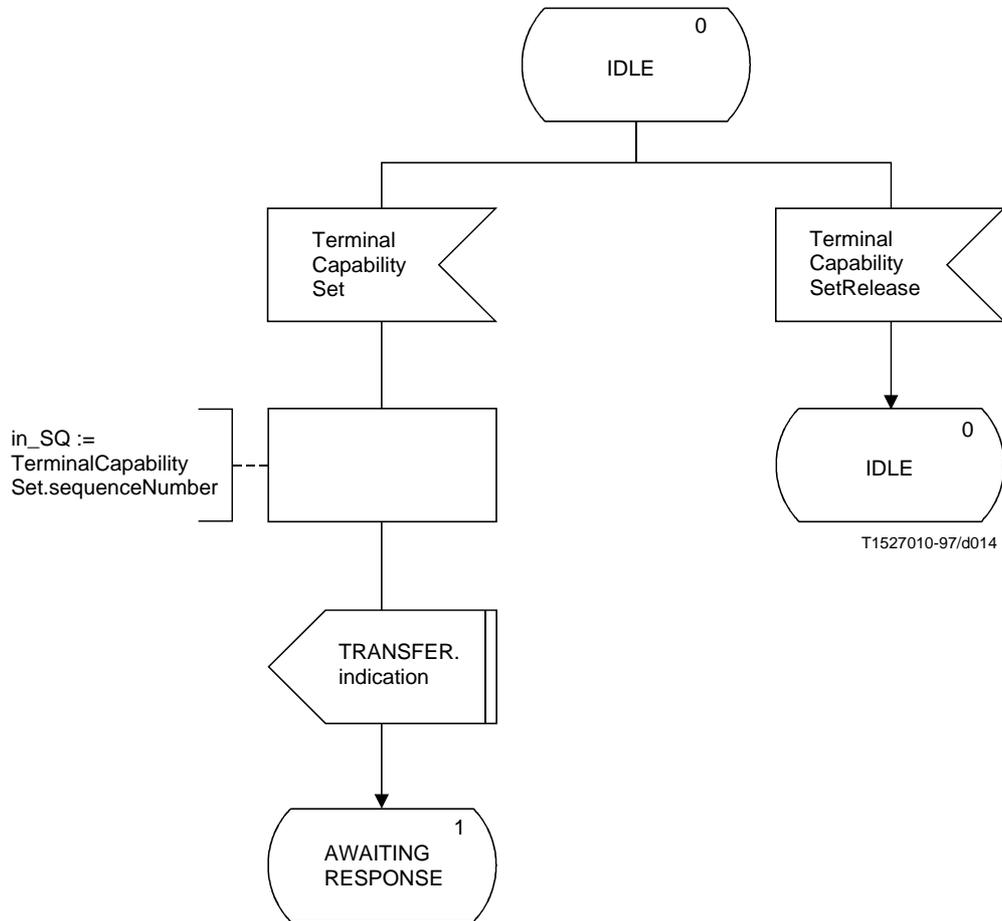
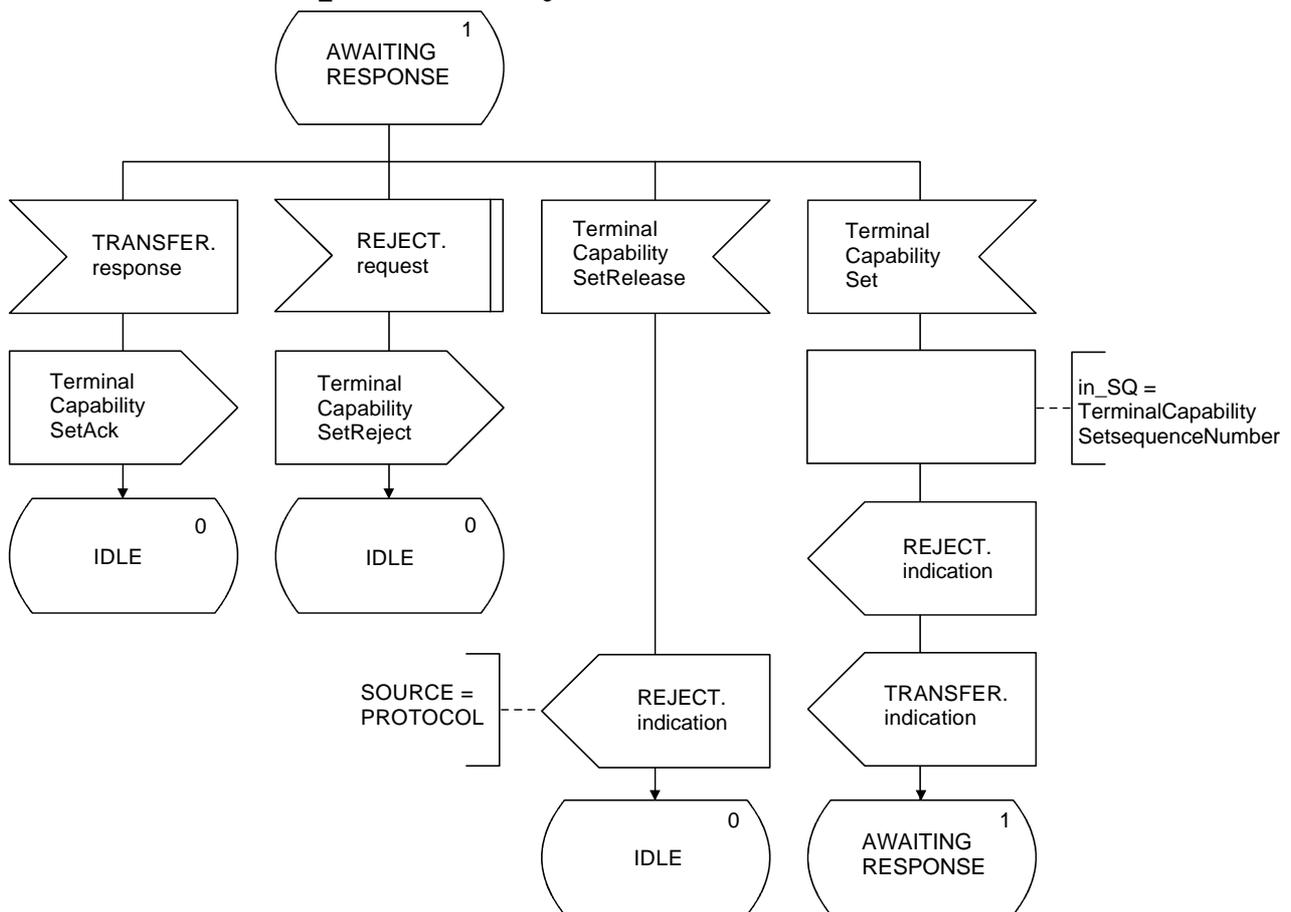


FIGURE 9 i)/H.245
Incoming CESE SDL

Superseded by a more recent version



T1519250-95/d015

FIGURE 9 ii)/H.245
Incoming CESE SDL

8.4 Unidirectional Logical Channel signalling procedures

8.4.1 Introduction

The protocol specified here provides reliable opening and closing of unidirectional logical channels using acknowledged procedures.

The protocol specified here is referred to as the Logical Channel Signalling Entity (LCSE). Procedures are specified in terms of primitives at the interface between the LCSE and the LCSE user, and LCSE states. Protocol information is transferred to the peer LCSE via relevant messages defined in clause 6.

There is an outgoing LCSE and an incoming LCSE. At each of the outgoing and incoming sides there is one instance of the LCSE for each unidirectional logical channel. There is no connection between an incoming LCSE and an outgoing LCSE at one side, other than via primitives to and from the LCSE user. LCSE error conditions are reported.

Data shall only be sent on a logical channel in the ESTABLISHED state. If data is received on a logical channel that is not in the ESTABLISHED state, the data shall be discarded and no fault shall be considered to have occurred.

Mode switching should be performed by closing and opening existing logical channels or by opening new logical channels.

Superseded by a more recent version

NOTE – Some recommendations that use this Recommendation may define some default logical channels. These shall be considered ESTABLISHED from the start of communication and shall not be opened using these procedures. They may, however, be closed by these procedures, and subsequently be reopened for the same or a different purpose.

A terminal that is no longer capable of processing the signals on a logical channel should take appropriate action; this should include closing the logical channel and transmitting the relevant (changed) capability information to the remote terminal.

The following text provides an overview of the operation of the LCSE protocol. In the case of discrepancy between this and the formal specification, the formal specification will supersede.

8.4.1.1 Protocol overview

The opening of a logical channel is initiated when the ESTABLISH.request primitive is issued by the user at the outgoing LCSE. An OpenLogicalChannel message, containing forward logical channel parameters but not including reverse logical channel parameters, is sent to the peer incoming LCSE, and timer T103 is started. If an OpenLogicalChannelAck message is received in response to the OpenLogicalChannel message, then timer T103 is stopped and the user is informed with the ESTABLISH.confirm primitive that the logical channel has been successfully opened. The logical channel may now be used to transmit user information. If however an OpenLogicalChannelReject message is received in response to the OpenLogicalChannel message, then timer T103 is stopped and the user is informed with the RELEASE.indication primitive that the peer LCSE user has refused establishment of the logical channel.

If timer T103 expires in this period, then the user is informed with the RELEASE.indication primitive, and a CloseLogicalChannel message is sent to the peer incoming LCSE.

A logical channel which has been successfully established may be closed when the RELEASE.request primitive is issued by the user at the outgoing LCSE. A CloseLogicalChannel message is sent to the peer incoming LCSE, and the timer T103 is started. When a CloseLogicalChannelAck message is received, timer T103 is stopped and the user is informed that the logical channel has been successfully closed with the RELEASE.confirm primitive.

If timer T103 expires in this period, then the user is informed with the RELEASE.indication primitive.

Before either of the OpenLogicalChannelAck or OpenLogicalChannelReject messages have been received in response to a previously sent OpenLogicalChannel message, the user at the outgoing LCSE may close the logical channel using the RELEASE.request primitive.

Before the CloseLogicalChannelAck message is received in response to a previously sent CloseLogicalChannel message, the user at the outgoing LCSE may establish a new logical channel by issuing the ESTABLISH.request primitive.

8.4.1.2 Protocol overview – Incoming LCSE

When an OpenLogicalChannel message is received at the incoming LCSE, the user is informed of the request to open a new logical channel with the ESTABLISH.indication primitive. The incoming LCSE user signals acceptance of the request to establish the logical channel by issuing the ESTABLISH.response primitive, and an OpenLogicalChannelAck message is sent to the peer outgoing LCSE. The logical channel may now be used to receive user information. The incoming LCSE user signals rejection of the request to establish the logical channel by issuing the RELEASE.request primitive, and an OpenLogicalChannelReject message is sent to the peer outgoing LCSE.

A logical channel which has been successfully established may be closed when the CloseLogicalChannel message is received at the incoming LCSE. The incoming LCSE user is informed with the RELEASE.indication primitive, and the CloseLogicalChannelAck message is sent to the peer outgoing LCSE.

8.4.1.3 Conflict resolution

Conflicts may arise when requests to open logical channels are initiated at the same time. It may be possible to determine that there is a conflict from knowledge of exchanged capabilities.

Terminals shall be capable of detecting when conflict has arisen, or might arise, and shall act as follows.

Before logical channels can be opened, one terminal must be determined as the master terminal, and the other as the slave. The protocol defined in 8.2 provides one means to make this decision. The master terminal shall reject immediately any request from the slave that it identifies as a conflicting request. The slave terminal may identify such conflicts, but shall respond to the request from the master terminal, with the knowledge that its earlier request will be rejected.

Superseded by a more recent version

NOTE – Such conflicts might be caused by limited terminal resources, for example, when receive and transmission capabilities are dependent, as in the case of a terminal that can support a number of audio algorithms, but can only decode the same algorithm as it is encoding.

8.4.2 Communication between the LCSE and the LCSE user

8.4.2.1 Primitives between the LCSE and the LCSE user

Communication between the LCSE and the LCSE user is performed using the primitives shown in Table 26.

TABLE 26/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
ESTABLISH	FORWARD_PARAM	FORWARD_PARAM	– (Note 1)	
RELEASE	CAUSE	SOURCE CAUSE	Not defined (Note 2)	–
ERROR	Not defined	ERRCODE	Not defined	Not defined
NOTE 1 – “–” means no parameters.				
NOTE 2 – “Not defined” means that this primitive does not exist.				

8.4.2.2 Primitive definition

The definition of these primitives is as follows:

- The ESTABLISH primitives are used to establish a logical channel for audiovisual and data communication;
- the RELEASE primitives are used to release a logical channel;
- the ERROR primitive reports LCSE errors to some management entity.

8.4.2.3 Parameter definition

The definition of the primitive parameters shown in Table 26 are as follows:

- The FORWARD_PARAM parameter specifies the parameters associated with the logical channel. This parameter is mapped to the forwardLogicalChannelParameters field of the OpenLogicalChannel message and is carried transparently to the peer LCSE user.
- The SOURCE parameter indicates to the LCSE user the source of the logical channel release. The SOURCE parameter has the value of “USER” or “LCSE”, indicating either the LCSE user, or the LCSE. The latter may occur as the result of a protocol error.
- The CAUSE parameter indicates the reason as to why the peer LCSE user rejected a request to establish a logical channel. The CAUSE parameter is not present when the SOURCE parameter indicates “LCSE”.
- The ERRCODE parameter indicates the type of LCSE error. Table 30 shows the allowed values of the ERRCODE parameter.

8.4.2.4 LCSE states

The following states are used to specify the allowed sequence of primitives between the LCSE and the LCSE user, and the exchange of messages between peer LCSEs. The states are specified separately for each of an outgoing LCSE and an incoming LCSE. The states for an outgoing LCSE are:

State 0: RELEASED

Superseded by a more recent version

The logical channel is released. The logical channel shall not be used to send outgoing data.

State 1: AWAITING ESTABLISHMENT

The outgoing LCSE is waiting to establish a logical channel with a peer incoming LCSE. The logical channel shall not be used to send outgoing data.

State 2: ESTABLISHED

The LCSE peer-to-peer logical channel connection has been established. The logical channel may be used to send outgoing data.

State 3: AWAITING RELEASE

The outgoing LCSE is waiting to release a logical channel with the peer incoming LCSE. The logical channel shall not be used to send outgoing data.

The states for an incoming LCSE are:

State 0: RELEASED

The logical channel is released. The logical channel shall not be used to receive incoming data.

State 1: AWAITING ESTABLISHMENT

The incoming LCSE is waiting to establish a logical channel with a peer outgoing LCSE. The logical channel shall not be used to receive incoming data.

State 2: ESTABLISHED

An LCSE peer-to-peer logical channel connection has been established. The logical channel may be used to receive incoming data.

8.4.2.5 State transition diagram

The allowed sequence of primitives between the LCSE and the LCSE user is defined here. The allowed sequence of primitives relates to states of the LCSE as viewed from the LCSE user. The allowed sequences are specified separately for each of an outgoing LCSE and an incoming LCSE, as shown in Figures 10 and 11 respectively.

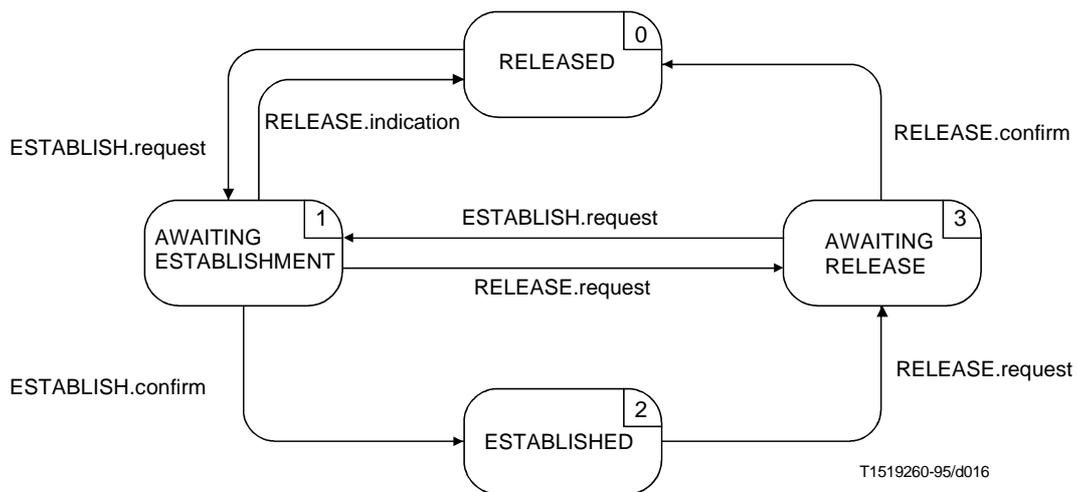


FIGURE 10/H.245

State transition diagram for sequence of primitives at outgoing LCSE

Superseded by a more recent version

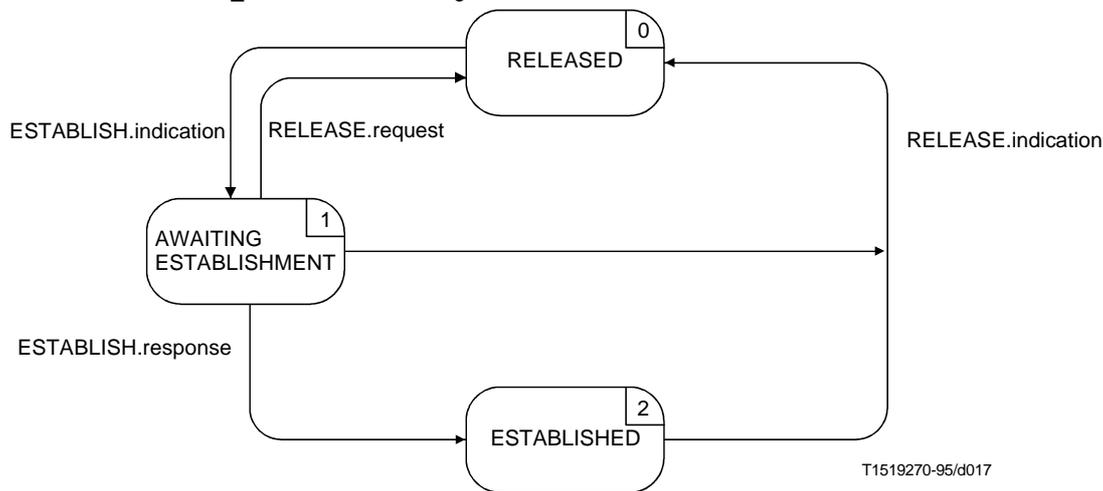


FIGURE 11/H.245

State transition diagram for sequence of primitives at incoming LCSE

8.4.3 Peer-to-peer LCSE communication

8.4.3.1 LCSE messages

Table 27 shows the LCSE messages and fields, defined in clause 6, which are relevant to the LCSE protocol.

TABLE 27/H.245

LCSE message names and fields

Function	Message	Direction	Field
Establishment	OpenLogicalChannel	O → I	forwardLogicalChannelNumber forwardLogicalChannelParameters
	OpenLogicalChannelAck	O ← I	forwardLogicalChannelNumber
	OpenLogicalChannelReject	O ← I	forwardLogicalChannelNumber cause
Release	CloseLogicalChannel	O → I	forwardLogicalChannelNumber source
	CloseLogicalChannelAck	O ← I	forwardLogicalChannelNumber
O Outgoing I Incoming			

8.4.3.2 LCSE state variables

The following state variable is defined at the outgoing LCSE:

out_LCN

This state variable distinguishes between outgoing LCSEs. It is initialized at outgoing LCSE initialization. The value of out_LCN is used to set the forwardLogicalChannelNumber field of LCSE messages sent from an outgoing LCSE. For LCSE messages received at an outgoing LCSE, the message forwardLogicalChannelNumber field value is identical to the value of out_LCN.

Superseded by a more recent version

The following state variable is defined at the incoming LCSE:

in_LCN

This state variable distinguishes between incoming LCSEs. It is initialized at incoming LCSE initialization. The value of in_LCN is used to set the forwardLogicalChannelNumber field of LCSE messages sent from an incoming LCSE. For LCSE messages received at an incoming LCSE, the message forwardLogicalChannelNumber field value is identical to the value of in_LCN.

8.4.3.3 LCSE timers

The following timer is specified for the outgoing LCSE:

T103

This timer is used during the AWAITING ESTABLISHMENT and AWAITING RELEASE states. It specifies the maximum allowed time during which no OpenLogicalChannelAck, OpenLogicalChannelReject or CloseLogicalChannelAck message may be received.

8.4.4 LCSE procedures

8.4.4.1 Introduction

Figure 12 summarizes the primitives and their parameters, and the messages, for each of the outgoing and incoming LCSE.

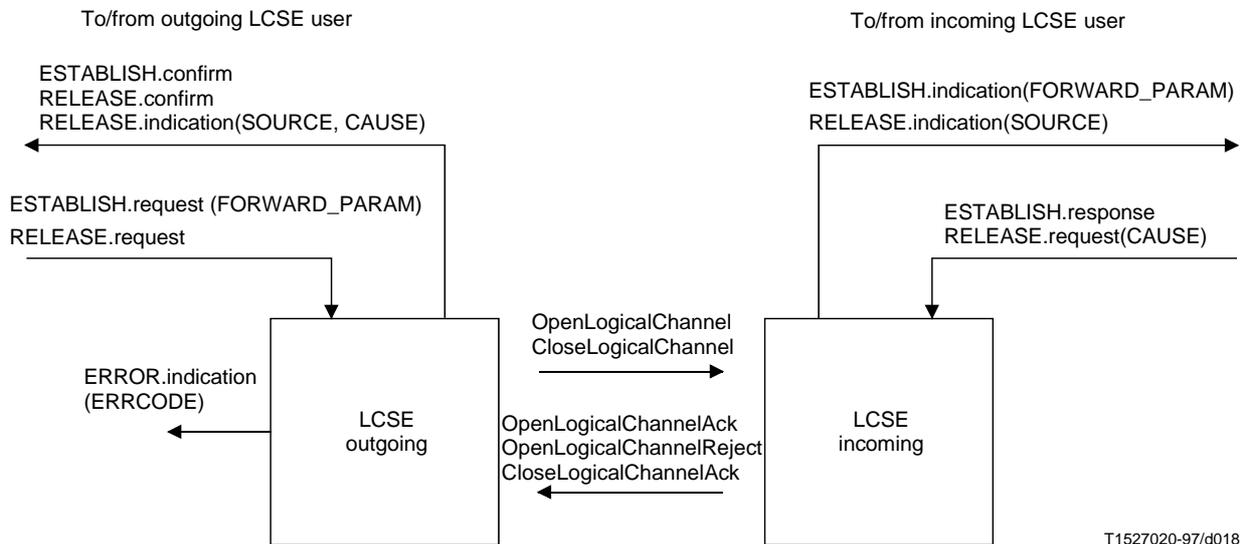


FIGURE 12/H.245

Primitives and messages in the Logical Channel Signalling Entity

8.4.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams, the parameters of the indication and confirm primitives assume values as shown in Table 28.

Superseded by a more recent version

TABLE 28/H.245

Default primitive parameter values

Primitive	Parameter	Default value (Note)
ESTABLISH.indication	FORWARD_PARAM	OpenLogicalChannel.forwardLogicalChannelParameters
RELEASE.indication	SOURCE CAUSE	CloseLogicalChannel.source null
NOTE – A primitive parameter shall be coded as null if an indicated message field is not present in the message.		

8.4.4.3 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 29.

TABLE 29/H.245

Default message field values

Message	Field	Default value (Note 1)
OpenLogicalChannel (Note 2)	forwardLogicalChannelNumber forwardLogicalChannelParameters	out_LCN ESTABLISH.request(FORWARD_PARAM)
OpenLogicalChannelAck	forwardLogicalChannelNumber	in_LCN
OpenLogicalChannelReject	forwardLogicalChannelNumber cause	in_LCN RELEASE.request(CAUSE)
CloseLogicalChannel	forwardLogicalChannelNumber source	out_LCN user
CloseLogicalChannelAck	forwardLogicalChannelNumber	in_LCN
NOTE 1 – A message field shall not be coded if the corresponding primitive parameter is null, i.e. not present.		
NOTE 2 – reverseLogicalChannelParameters are not coded in unidirectional logical channel signalling procedures.		

8.4.4.4 ERRCODE parameter values

The ERRCODE parameter of the ERROR.indication primitive indicates a particular error condition. Table 30 shows the values that the ERRCODE parameter may take at the outgoing LCSE. There is no ERROR.indication primitive associated with the incoming LCSE.

TABLE 30/H.245

ERRCODE parameter values at outgoing LCSE

Error type	Error code	Error condition	State
Inappropriate message	A	OpenLogicalChannelAck	RELEASED
	B	OpenLogicalChannelReject	RELEASED ESTABLISHED
	C	CloseLogicalChannelAck	ESTABLISHED
No response from peer LCSE	D	timer T103 expiry	AWAITING ESTABLISHMENT AWAITING RELEASE

Superseded by a more recent version

8.4.4.5 SDLs

The outgoing LCSE and the incoming LCSE procedures are expressed in SDL form in Figures 13 and 14 respectively.

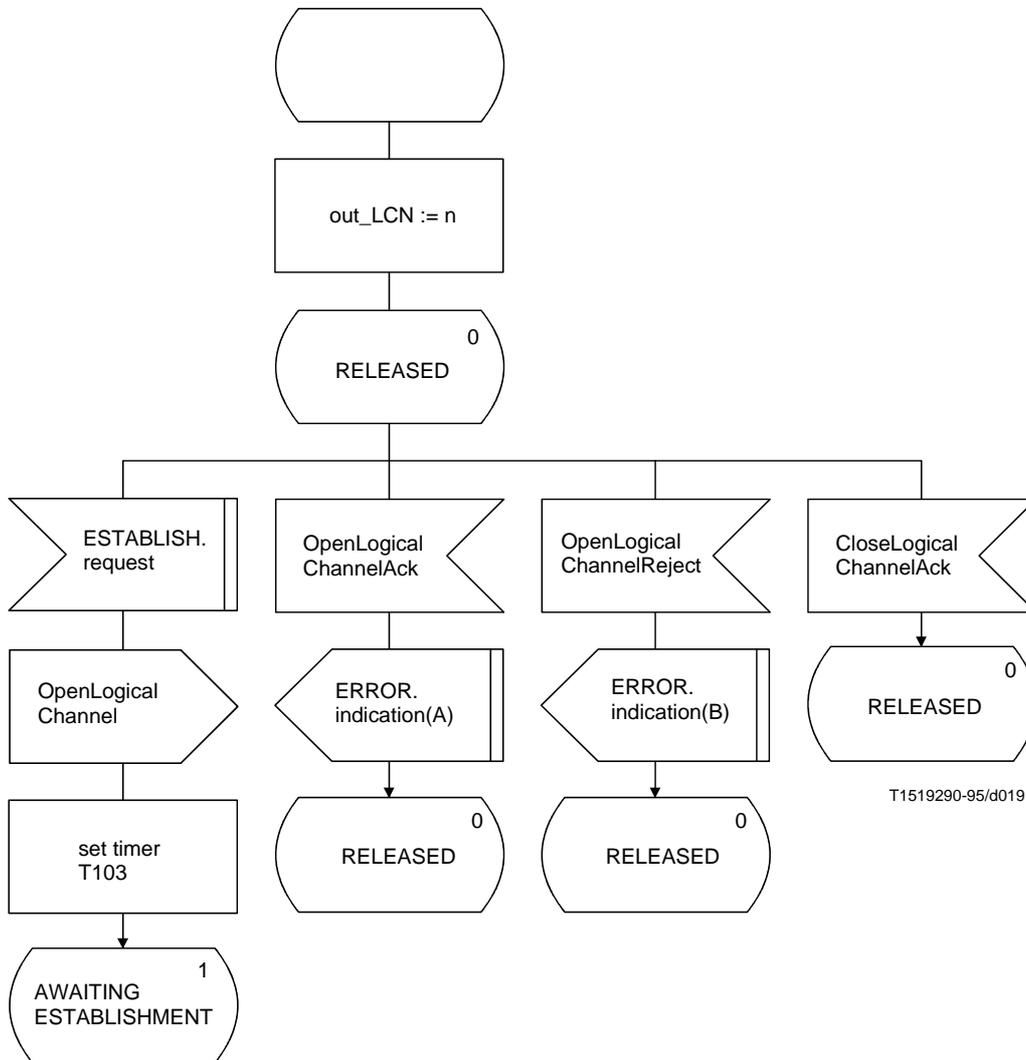
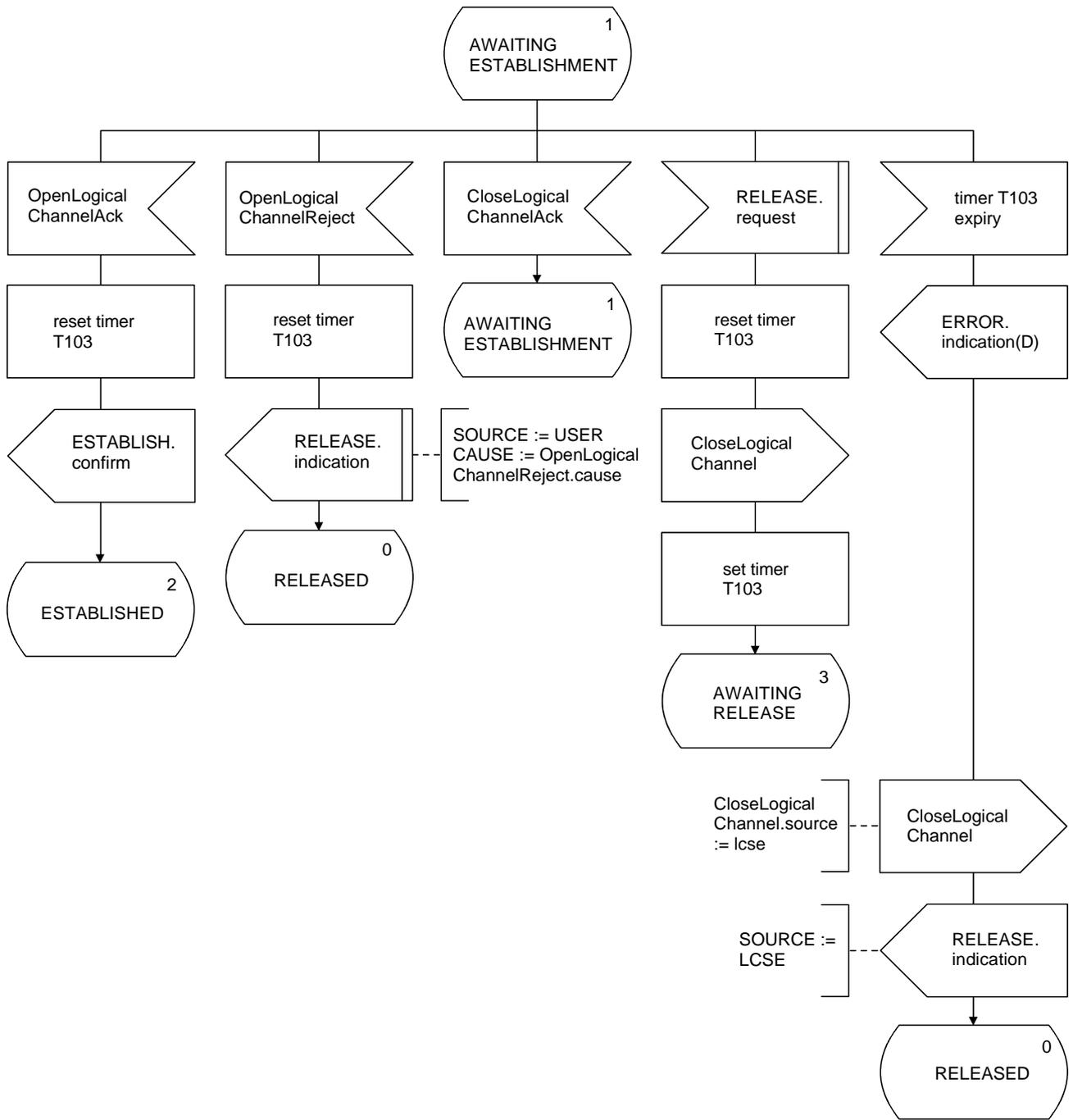


FIGURE 13 i)/H.245

Outgoing LCSE SDL

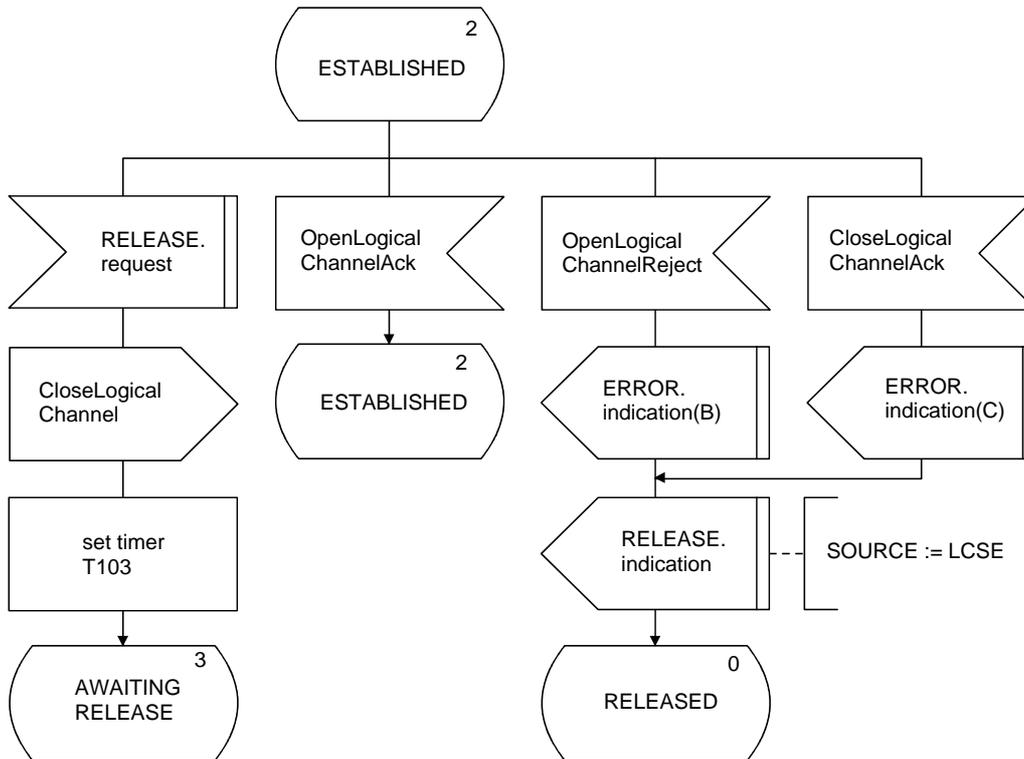
Superseded by a more recent version



T1519300-95/d020

FIGURE 13 ii)/H.245
Outgoing LCSE SDL

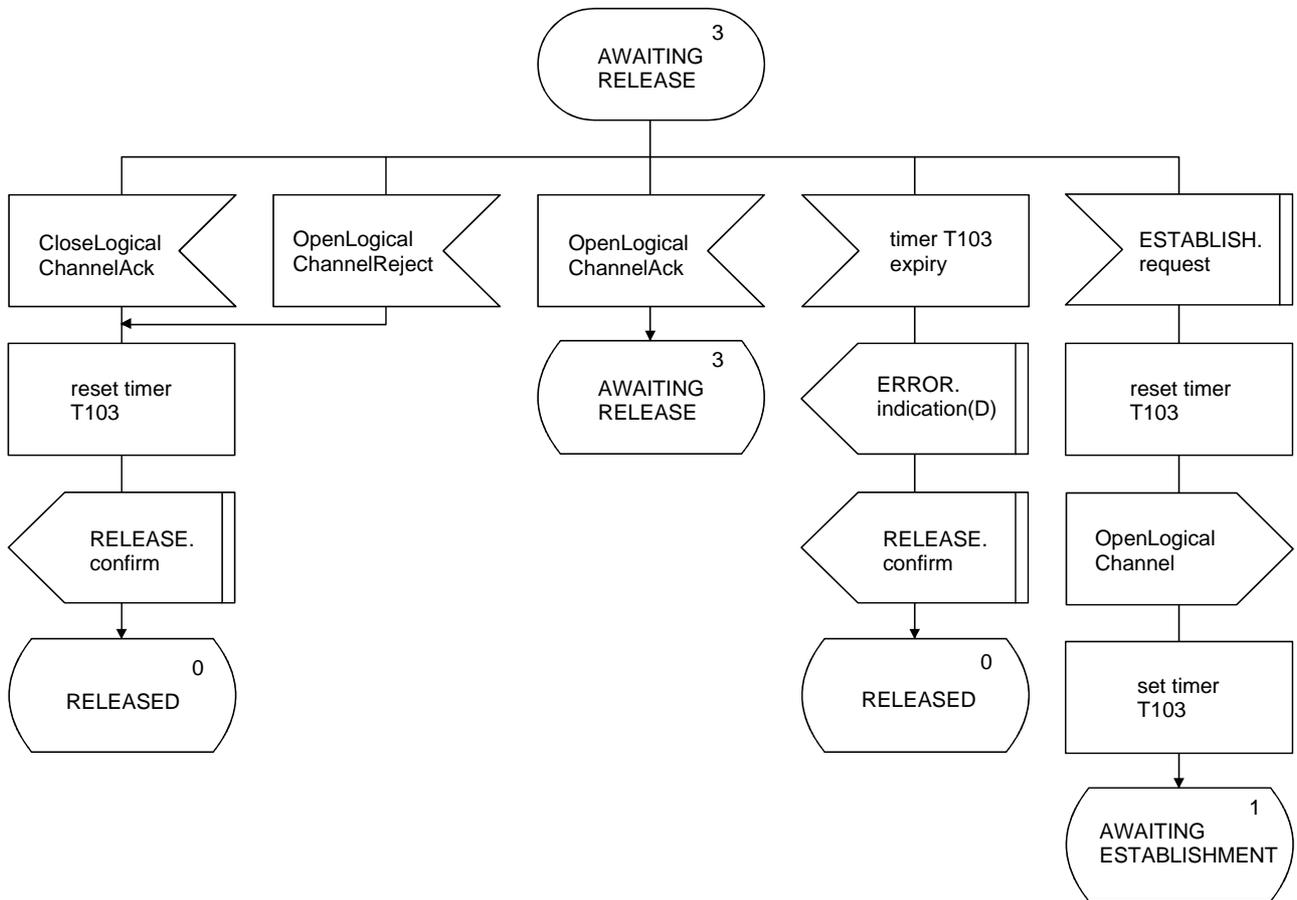
Superseded by a more recent version



T1519310-95/d021

FIGURE 13 iii)/H.245
Outgoing LCSE SDL

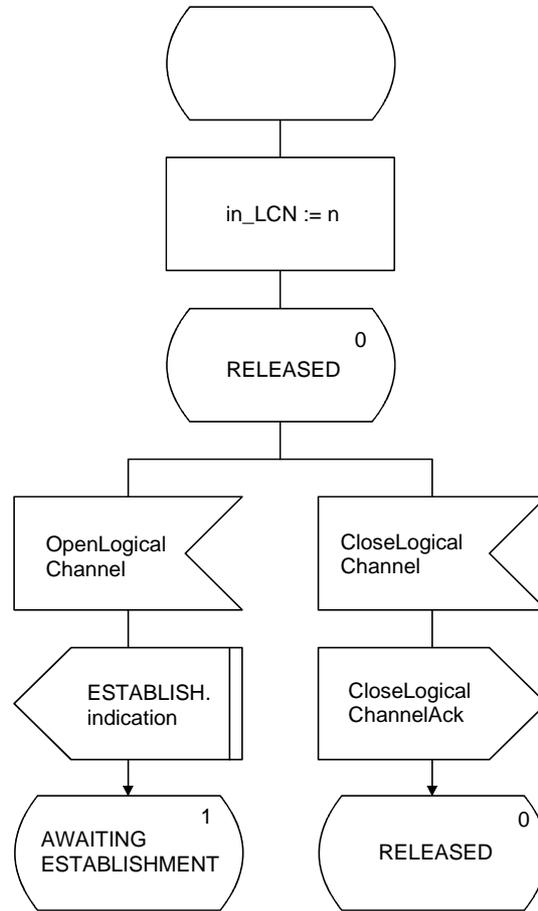
Superseded by a more recent version



T1519320-95/d022

FIGURE 13 iv)/H.245
Outgoing LCSE SDL

Superseded by a more recent version



T1519330-95/d023

FIGURE 14 i)/H.245
Incoming LCSE SDL

Superseded by a more recent version

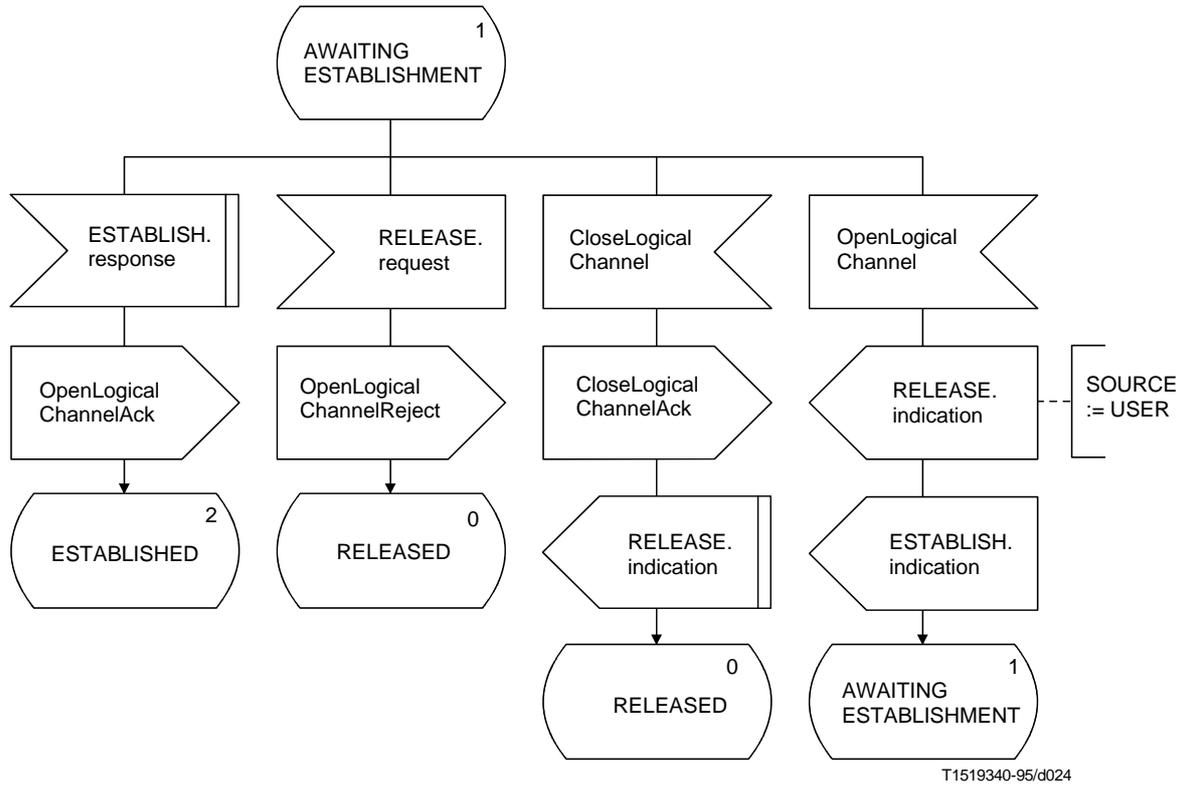
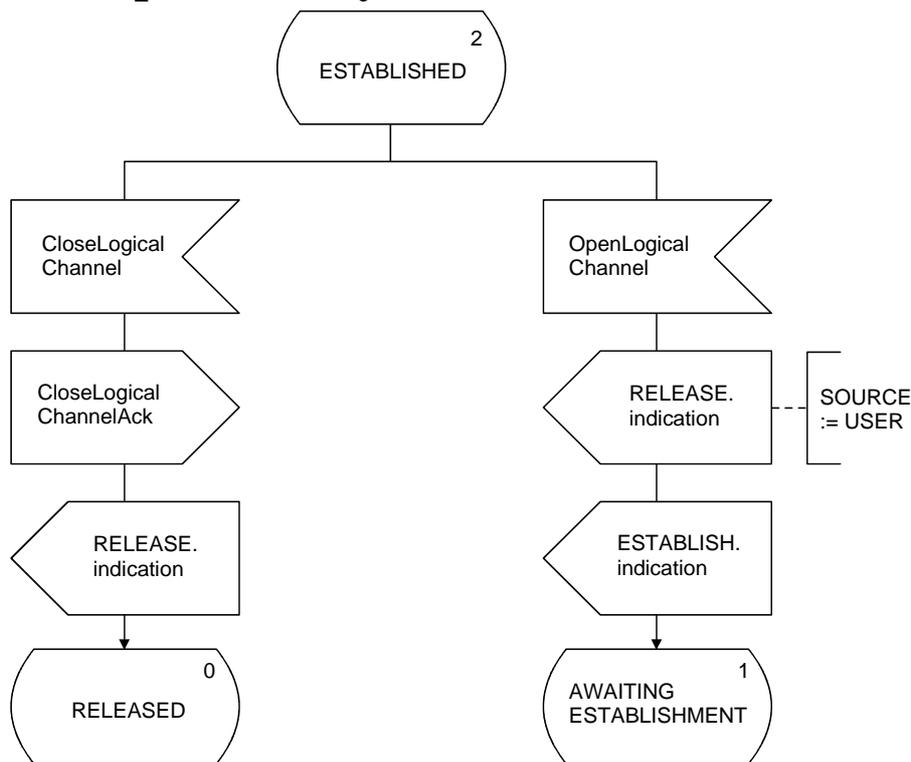


FIGURE 14 ii)/H.245
Incoming LCSE SDL

Superseded by a more recent version



T1519350-95/d025

FIGURE 14 iii)/H.245
Incoming LCSE SDL

8.5 Bidirectional Logical Channel signalling procedures

8.5.1 Introduction

The protocol specified here provides reliable opening and closing of bidirectional logical channels using acknowledged procedures.

The protocol specified here is referred to as the Bidirectional Logical Channel Signalling Entity (B-LCSE). Procedures are specified in terms of primitives at the interface between the B-LCSE and the B-LCSE user, and B-LCSE states. Protocol information is transferred to the peer B-LCSE via relevant messages defined in clause 6.

There is an outgoing B-LCSE and an incoming B-LCSE. At each of the outgoing and incoming sides there is one instance of the B-LCSE for each bidirectional logical channel.

A bidirectional logical channel consists of a pair of associated unidirectional channels. “Forward” is used to refer to transmission in the direction from the terminal making the request for a bidirectional logical channel to the other terminal and “reverse” is used to refer to the opposite direction of transmission.

Data shall only be sent on a bidirectional logical channel in the ESTABLISHED state. However, data may be received on the forward channel when the incoming B-LCSE is in the AWAITING CONFIRMATION state. Data that is received while in other states than the ESTABLISHED state and the AWAITING CONFIRMATION state shall be discarded and no fault shall be considered to have occurred.

A terminal may reject a request to open a bidirectional logical channel solely because it cannot support the requested reverse channel parameters. In this case it shall reject the request with cause equal to unsuitableReverseParameters, and shall immediately initiate procedures to establish a bidirectional logical channel as requested by the remote terminal, in which the reverse parameters are identical to the forward parameters of the remote terminal’s failed request, and with forward parameters that the terminal can support and which the remote terminal is known to be able to support.

Superseded by a more recent version

Mode switching should be performed by closing and opening existing logical channels or by opening new logical channels.

NOTE – Some recommendations that use this Recommendation may define some default logical channels. These shall be considered ESTABLISHED from the start of communication and shall not be opened using these procedures. They may, however, be closed by these procedures, and subsequently, be reopened for the same or a different purpose.

A terminal that is no longer capable of processing the signals on a logical channel should take appropriate action: this should include closing the logical channel and transmitting the relevant (changed) capability information to the remote terminal.

The following text provides an overview of the operation of the B-LCSE protocol. In the case of discrepancy between this and the formal specification, the formal specification will supersede.

8.5.1.1 Protocol overview

The opening of a logical channel is initiated when the ESTABLISH.request primitive is issued by the user at the outgoing B-LCSE. An OpenLogicalChannel message, containing both forward and reverse logical channel parameters, is sent to the peer incoming B-LCSE, and timer T103 is started. If an OpenLogicalChannelAck message is received in response to the OpenLogicalChannel message, then timer T103 is stopped, an OpenLogicalChannelConfirm message is sent to the peer incoming B-LCSE, and the user is informed with the ESTABLISH.confirm primitive that the logical channel has been successfully opened. The logical channel may not be used to transmit and receive user information. If, however, an OpenLogicalChannelReject message is received in response to the OpenLogicalChannel message, then timer T103 is stopped and the user is informed with the RELEASE.indication primitive that the peer B-LCSE user has refused establishment of the logical channel.

If timer T103 expires in this period, then the user is informed with the RELEASE.indication primitive, and a CloseLogicalChannel message is sent to the peer incoming B-LCSE.

A logical channel which has been successfully established may be closed when the RELEASE.request primitive is issued by the user at the outgoing B-LCSE. A CloseLogicalChannel message is sent to the peer incoming B-LCSE, and the timer T103 is started. When a CloseLogicalChannelAck message is received, timer T103 is stopped and the user is informed that the logical channel has been successfully closed with the RELEASE.confirm primitive.

If timer T103 expires in this period, then the user is informed with the RELEASE.indication primitive.

Before either of the OpenLogicalChannelAck or OpenLogicalChannelReject messages have been received in response to a previously sent OpenLogicalChannel message, the user at the outgoing B-LCSE may close the logical channel using the RELEASE.request primitive.

Before the CloseLogicalChannelAck message is received in response to a previously sent CloseLogicalChannel message, the user at the outgoing B-LCSE may establish a new logical channel by issuing the ESTABLISH.request primitive.

8.5.1.2 Protocol overview – Incoming B-LCSE

When an OpenLogicalChannel message is received at the incoming B-LCSE, the user is informed of the request to open a new logical channel with the ESTABLISH.indication primitive. The incoming B-LCSE user signals acceptance of the request to establish the logical channel by issuing the ESTABLISH.response primitive, and an OpenLogicalChannelAck message is sent to the peer outgoing B-LCSE. The forward channel of the bidirectional logical channel may now be used to receive user information. The incoming B-LCSE user signals rejection of the request to establish the logical channel by issuing the RELEASE.request primitive, and an OpenLogicalChannelReject message is sent to the peer outgoing B-LCSE.

When an OpenLogicalChannelConfirm message is received at the incoming B-LCSE, the user is informed that the bidirectional logical channel is established with the ESTABLISH.confirm primitive. The reverse channel of the bidirectional logical channel may now be used to transmit user information.

A logical channel which has been successfully established may be closed when the CloseLogicalChannel message is received at the incoming B-LCSE. The incoming B-LCSE user is informed with the RELEASE.indication primitive and the CloseLogicalChannelAck message is sent to the peer outgoing B-LCSE.

Superseded by a more recent version

8.5.1.3 Conflict resolution

Conflicts may arise when requests to open logical channels are initiated at the same time. It may be possible to determine that there is conflict from knowledge of exchanged capabilities. On other occasions, both terminals may initiate the opening of a bidirectional logical channel for the same purpose, even though the exact parameters requested may be different, and both terminals have sufficient capability for both requests. Terminals shall be capable of detecting when both of these situations have arisen, any shall act as follows.

Before logical channels can be opened, one terminal must be determined as the master terminal, and the other as the slave. The protocol defined in 8.2 provides one means to make this decision. The master terminal shall reject immediately any request from the slave that it identifies as a conflicting request. The slave terminal may identify such conflicts, but shall respond to the request from the master terminal, with the knowledge that its earlier request will be rejected.

In the second type of conflict defined above, it is impossible to distinguish when two bidirectional channels are actually wanted from the case when only one is wanted. Terminals shall respond assuming that only one is wanted, but a terminal may subsequently repeat its request if the assumption was incorrect.

8.5.2 Communication between the B-LCSE and the B-LCSE user

8.5.2.1 Primitives between the B-LCSE and the B-LCSE user

Communication between the B-LCSE and the B-LCSE user is performed using the primitives shown in Table 31.

TABLE 31/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
ESTABLISH	FORWARD_PARAM REVERSE_PARAM	FORWARD_PARAM REVERSE_PARAM	REVERSE_DATA	REVERSE_DATA
RELEASE	CAUSE	SOURCE CAUSE	Not defined (Note 2)	– (Note 1)
ERROR	Not defined	ERRCODE	Not defined	Not defined

NOTE 1 – “–” means no parameters.
NOTE 2 – “Not defined” means that this primitive does not exist.

8.5.2.2 Primitive definition

The definition of these primitives is as follows:

- The ESTABLISH primitives are used to establish a logical channel for audiovisual and data communication.
- The RELEASE primitives are used to release a logical channel.
- The ERROR primitive reports B-LCSE errors to a management entity.

8.5.2.3 Parameter definition

The definition of the primitive parameters shown in Table 31 are as follows:

- The FORWARD_PARAM parameter specifies the parameters associated with the forward channel, that is, from the terminal containing the outgoing B-LCSE to the terminal containing the incoming B-LCSE. This parameter is mapped to the forwardLogicalChannelParameters field of the OpenLogicalChannel message and is carried transparently to the peer LCSE user.

Superseded by a more recent version

- b) The REVERSE_PARAM parameter specifies the parameters associated with the reverse channel, that is, from the terminal containing the incoming B-LCSE to the terminal containing the outgoing B-LCSE. This parameter is mapped to the reverseLogicalChannelParameters field of the OpenLogicalChannel message and is carried transparently to the peer LCSE user.
- c) The REVERSE_DATA parameter specifies some parameters associated with the reverse channel, that is, from the terminal containing the incoming B-LCSE to the terminal containing the outgoing B-LCSE. This parameter is mapped to reverseLogicalChannelParameters field of the OpenLogicalChannelAck message and is carried transparently to the peer B-LCSE user.
- d) The SOURCE parameter indicates to the B-LCSE user the source of the logical channel release. The SOURCE parameter has the value of "USER" or "B-LCSE", indicating either the B-LCSE user, or the B-LCSE. The latter may occur as the result of a protocol error.
- e) The CAUSE parameter indicates the reason as to why the peer B-LCSE user rejected a request to establish a logical channel. The CAUSE parameter is not present when the SOURCE parameter indicates "B-LCSE".
- f) The ERRCODE parameter indicates the type of B-LCSE error. Table 35 shows the allowed values of the ERRCODE parameter.

8.5.2.4 B-LCSE states

The following states are used to specify the allowed sequence of primitives between the B-LCSE and the B-LCSE user, and the exchange of messages between peer B-LCSEs. The states are specified separately for each of an outgoing B-LCSE and an incoming B-LCSE. The states for an outgoing B-LCSE are:

State 0: RELEASED

The logical channel is released. The logical channel shall not be used to send or receive data.

State 1: AWAITING ESTABLISHMENT

The outgoing B-LCSE is waiting to establish a logical channel with a peer incoming B-LCSE. The logical channel shall not be used to send or receive data.

State 2: ESTABLISHED

The B-LCSE peer-to-peer logical channel connection has been established. The logical channel may be used to send and receive data.

State 3: AWAITING RELEASE

The outgoing B-LCSE is waiting to release a logical channel with the peer incoming B-LCSE. The logical channel shall not be used to send data, but may continue to be received.

The states for an incoming B-LCSE are:

State 0: RELEASED

The logical channel is released. The logical channel shall not be used to receive or send data.

State 1: AWAITING ESTABLISHMENT

The incoming B-LCSE is waiting to establish a logical channel with a peer outgoing B-LCSE. The logical channel shall not be used to receive or send data.

State 2: AWAITING CONFIRMATION

The incoming B-LCSE is awaiting confirmation that the logical channel is established with a peer outgoing B-LCSE. The logical channel shall not be used to send data, but data may be received.

State 3: ESTABLISHED

A B-LCSE peer-to-peer logical channel connection has been established. The logical channel may be used to receive and send data.

8.5.2.5 State transition diagram

The allowed sequence of primitives between the B-LCSE and the B-LCSE user is defined here. The allowed sequence of primitives relates to states of the B-LCSE as viewed from the B-LCSE user. The allowed sequences are specified separately for each of an outgoing B-LCSE and an incoming B-LCSE, as shown in Figures 15 and 16 respectively.

Superseded by a more recent version

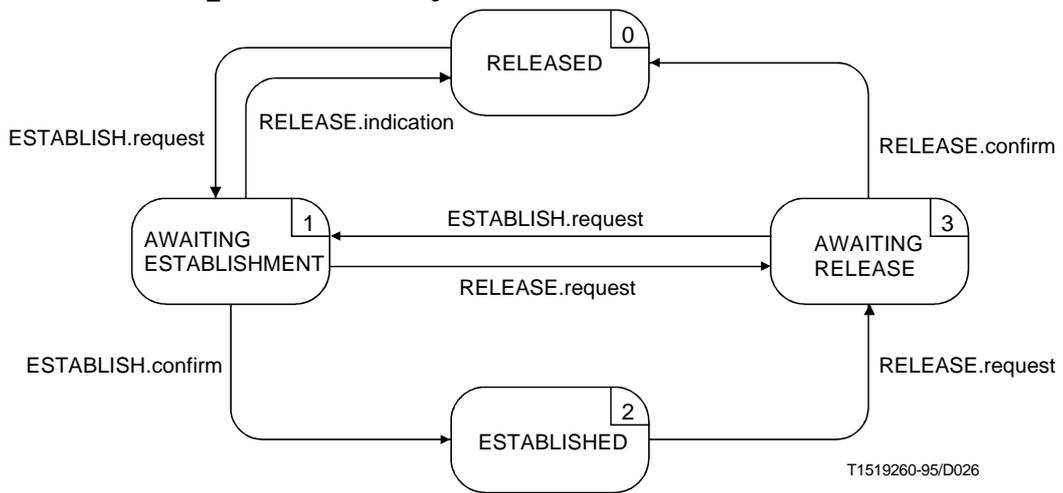


FIGURE 15/H.245

State transition diagram for sequence of primitives at outgoing B-LCSE

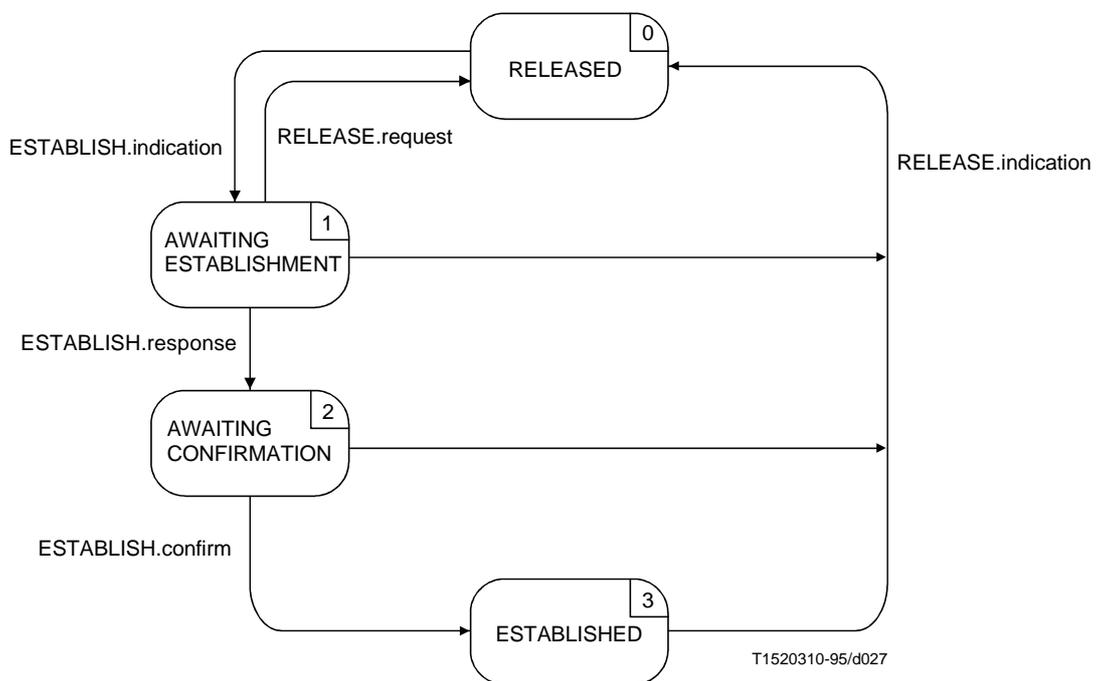


FIGURE 16/H.245

State transition diagram for sequence of primitives at incoming B-LCSE

Superseded by a more recent version

8.5.3 Peer-to-peer B-LCSE communication

8.5.3.1 B-LCSE messages

Table 32 shows the B-LCSE messages and fields, defined in clause 6, which are relevant to the B-LCSE protocol.

TABLE 32/H.245

B-LCSE message names and fields

Function	Message	Direction	Field
Establishment	OpenLogicalChannel	O → I	forwardLogicalChannelNumber forwardLogicalChannelParameters reverseLogicalChannelParameters
	OpenLogicalChannelAck	O ← I	forwardLogicalChannelNumber reverseLogicalChannelParameters
	OpenLogicalChannelReject	O ← I	forwardLogicalChannelNumber cause
	OpenLogicalChannelConfirm	O ← I	forwardLogicalChannelNumber
Release	CloseLogicalChannel	O → I	forwardLogicalChannelNumber
	CloseLogicalChannelAck	O ← I	source forwardLogicalChannelNumber
O Outgoing I Incoming			

8.5.3.2 B-LCSE state variables

The following state variable is defined at the outgoing B-LCSE:

out_LCN

This state variable distinguishes between outgoing B-LCSEs. It is initialized at outgoing B-LCSE initialization. The value of out_LCN is used to set the forwardLogicalChannelNumber field of B-LCSE messages sent from an outgoing B-LCSE. For B-LCSE messages received at an outgoing B-LCSE, the message forwardLogicalChannelNumber field value is identical to the value of out_LCN.

The following state variable is defined at the incoming B-LCSE:

in_LCN

This state variable distinguishes between incoming B-LCSEs. It is initialized at incoming B-LCSE initialization. The value of in_LCN is used to set the forwardLogicalChannelNumber field of B-LCSE messages sent from an incoming B-LCSE. For B-LCSE messages received at an incoming B-LCSE, the message forwardLogicalChannelNumber field value is identical to the value of in_LCN.

8.5.3.3 B-LCSE timers

The following timer is specified for the outgoing and incoming B-LCSE:

T103

At the outgoing B-LCSE this timer is used during the AWAITING ESTABLISHMENT and AWAITING RELEASE states. It specifies the maximum time during which no OpenLogicalChannelAck, OpenLogicalChannelReject or CloseLogicalChannelAck message may be received.

At the incoming B-LCSE, this timer is used during the AWAITING CONFIRMATION state. It specifies the maximum time during which no OpenLogicalChannelConfirm message may be received.

Superseded by a more recent version

8.5.4 B-LCSE procedures

8.5.4.1 Introduction

Figure 17 summarizes the primitives and their parameters, and the messages, for each of the outgoing and incoming B-LCSEs.

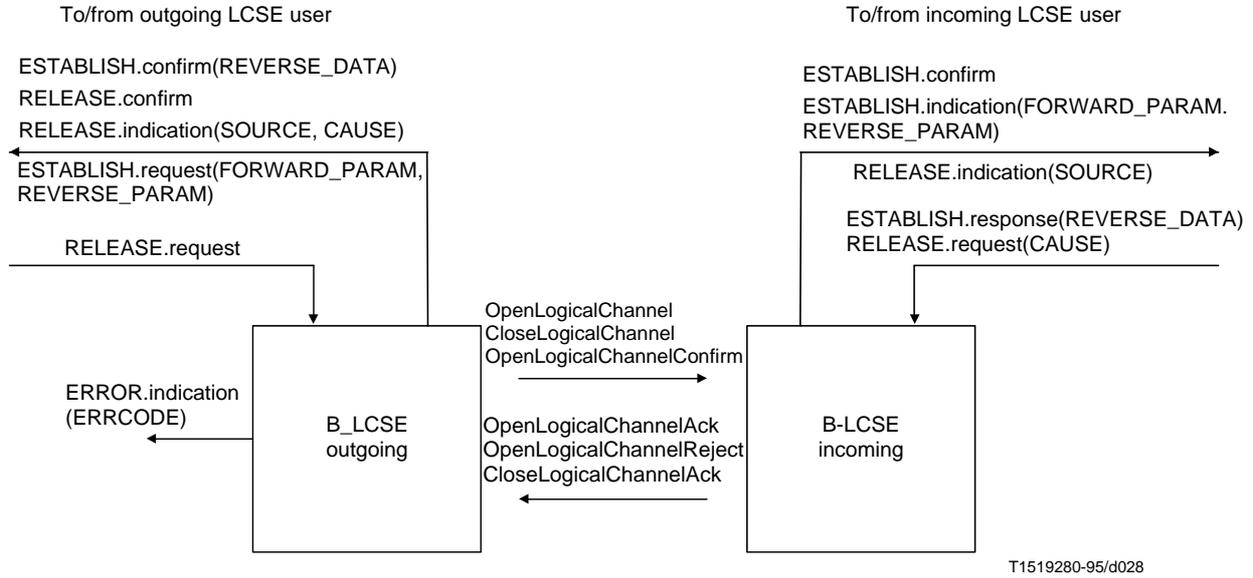


FIGURE 17/H.245

Primitives and messages in the Bidirectional Logical Channel Signalling Entity

8.5.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table 33.

TABLE 33/H.245

Default primitive parameter values

Primitive	Parameter	Default value (Note)
ESTABLISH.indication	FORWARD_PARAM REVERSE_PARAM	OpenLogicalChannel.forwardLogicalChannelParameters OpenLogicalChannel.reverseLogicalChannelParameters
RELEASE.indication	SOURCE CAUSE	CloseLogicalChannel.source null

NOTE – A primitive parameter shall be coded as null, if an indicated message field is not present in the message.

Superseded by a more recent version

8.5.4.3 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 34.

TABLE 34/H.245

Default message field values

Message	Field	Default value (Note)
OpenLogicalChannel	forwardLogicalChannelNumber	out_LCN
	forwardLogicalChannelParameters	ESTABLISH.request(FORWARD_PARAM)
	reverseLogicalChannelParameters	ESTABLISH.request(REVERSE_PARAM)
OpenLogicalChannelAck	forwardLogicalChannelNumber	in_LCN
	reverseLogicalChannelParameters	ESTABLISH.response(REVERSE_DATA)
OpenLogicalChannelReject	forwardLogicalChannelNumber cause	in_LCN RELEASE.request(CAUSE)
OpenLogicalChannelConfirm	forwardLogicalChannelNumber	out_LCN
CloseLogicalChannel	forwardLogicalChannelNumber source	out_LCN user
CloseLogicalChannelAck	forwardLogicalChannelNumber	in_LCN
NOTE – A message field shall not be coded, if the corresponding primitive parameter is null, i.e. not present.		

8.5.4.4 ERRCODE parameter values

The ERRCODE parameter of the ERROR.indication primitive indicates a particular error condition. Table 35 shows the values that the ERRCODE parameter may take at the outgoing B-LCSE and Table 36 shows the values that the ERRCODE parameter may take at the incoming B-LCSE.

TABLE 35/H.245

ERRCODE parameter values at outgoing B-LCSE

Error type	Error code	Error condition	State
Inappropriate message	A	OpenLogicalChannelAck	RELEASED
	B	OpenLogicalChannelReject	RELEASED ESTABLISHED
	C	CloseLogicalChannelAck	ESTABLISHED
No response from peer B-LCSE	D	timer T103 expiry	AWAITING ESTABLISHMENT AWAITING RELEASE

Superseded by a more recent version

TABLE 36/H.245

ERRCODE parameter values at incoming B-LCSE

Error type	Error code	Error condition	State
Inappropriate message	E	OpenLogicalChannelConfirm	AWAITING ESTABLISHMENT
No response from peer B-LCSE	F	timer T103 expiry	AWAITING CONFIRMATION

8.5.4.5 SDLs

The outgoing B-LCSE and the incoming B-LCSE procedures are expressed in SDL form in Figures 18 and 19 respectively.

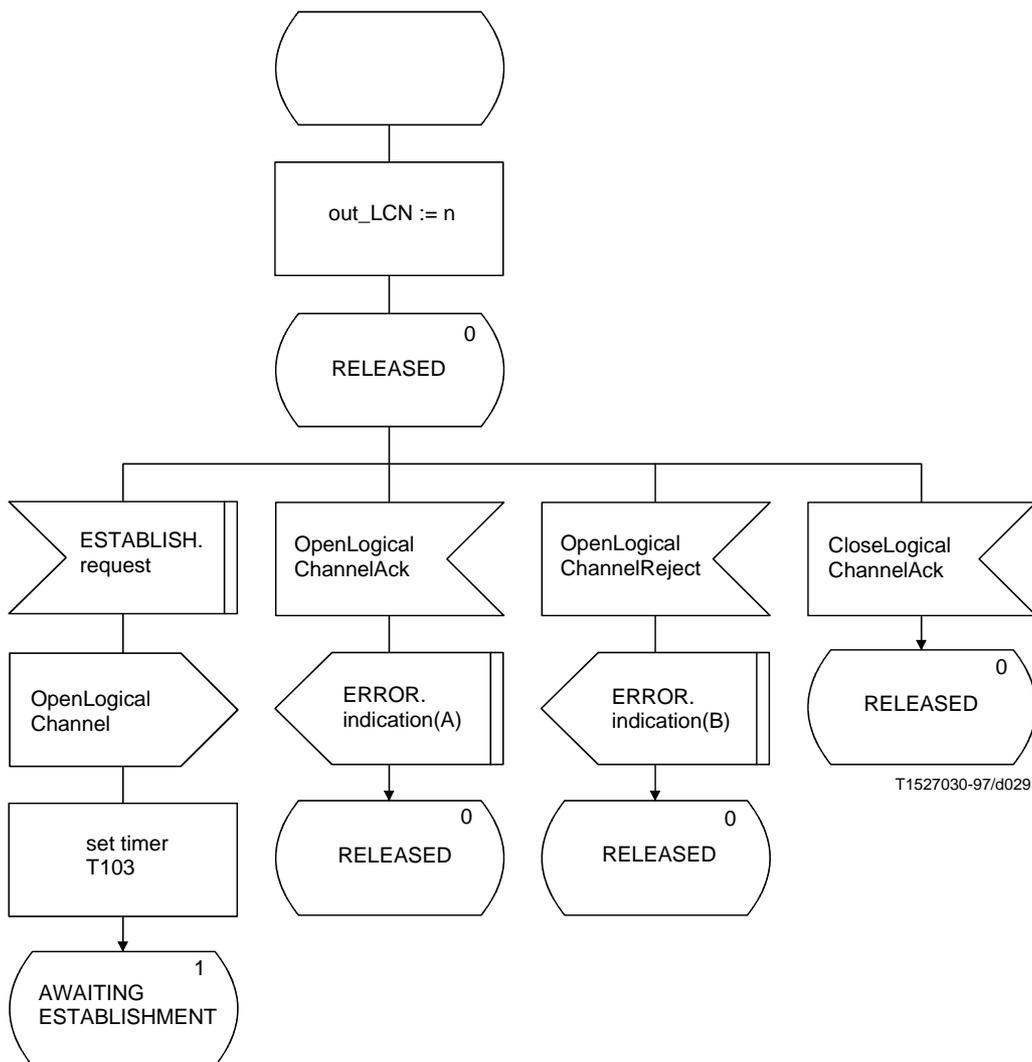
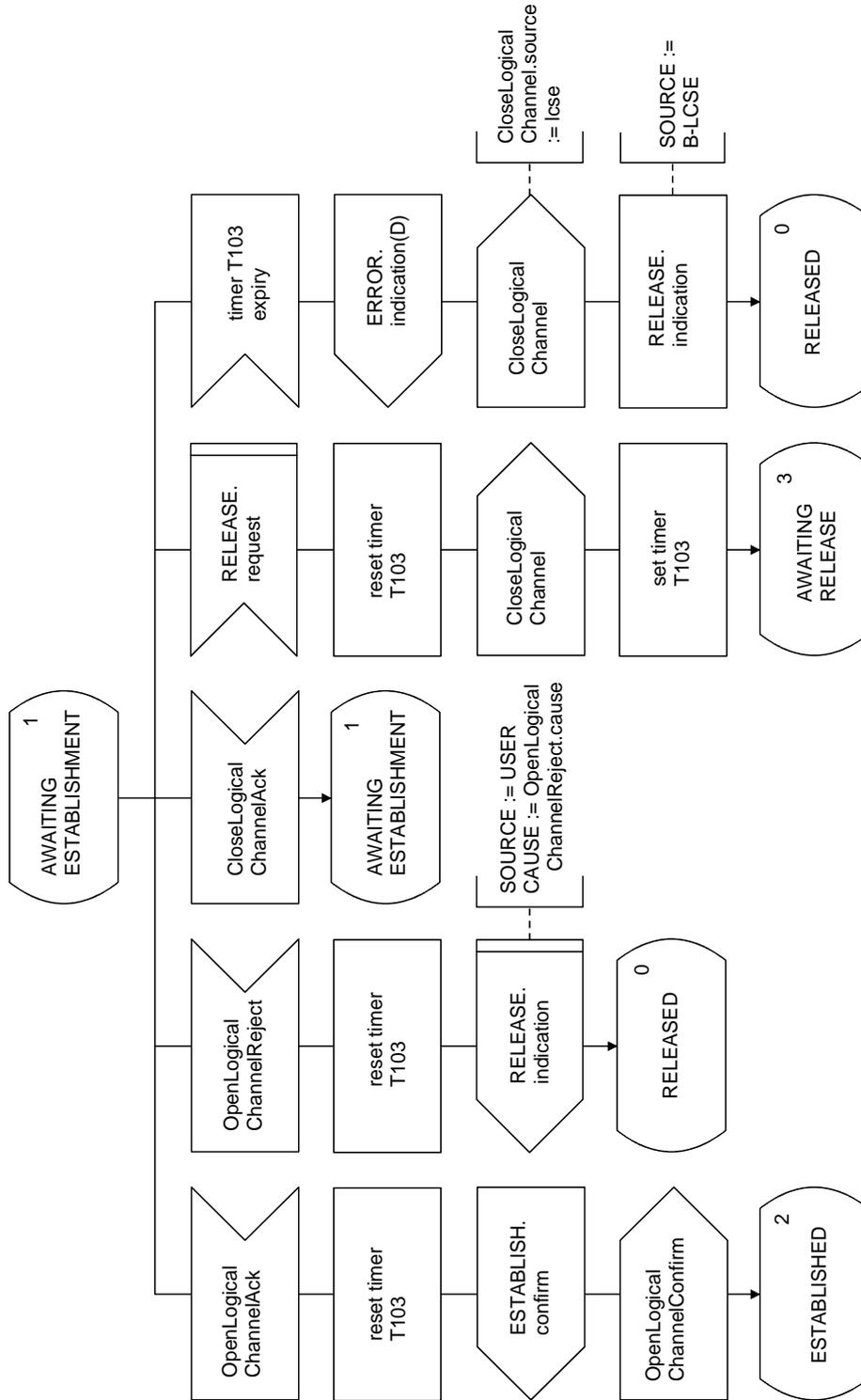


FIGURE 18 i)/H.245

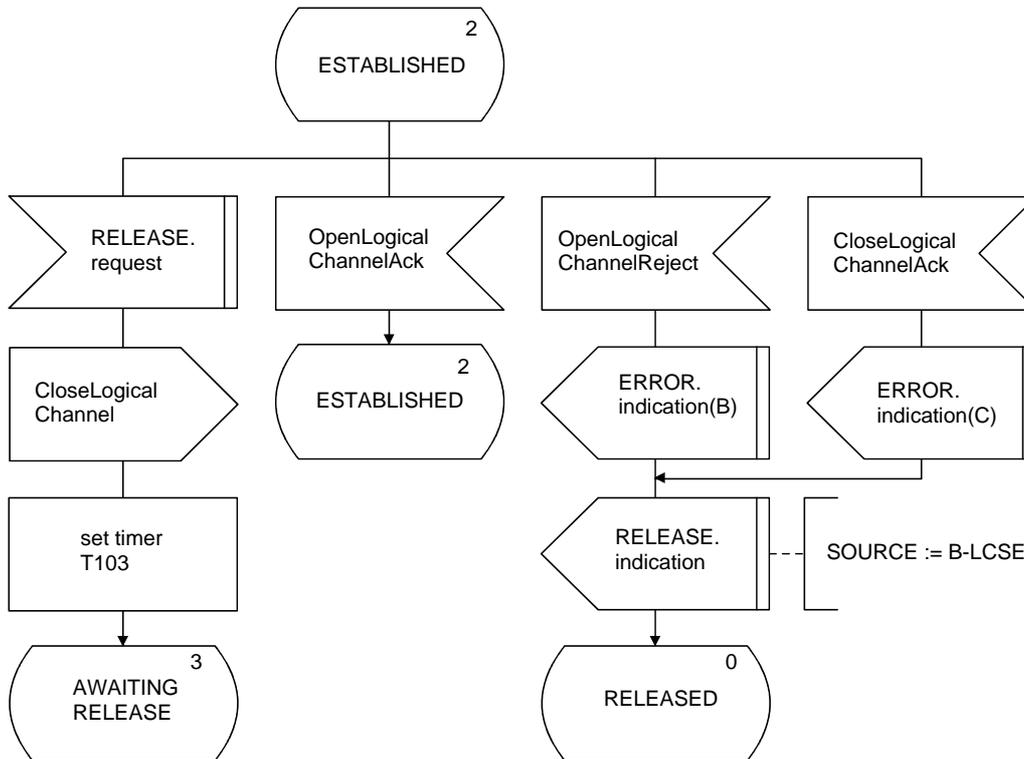
Outgoing B-LCSE SDL



T1520320-95/d030

FIGURE 18 ii)/H.245
Outgoing B-LCSE SDL

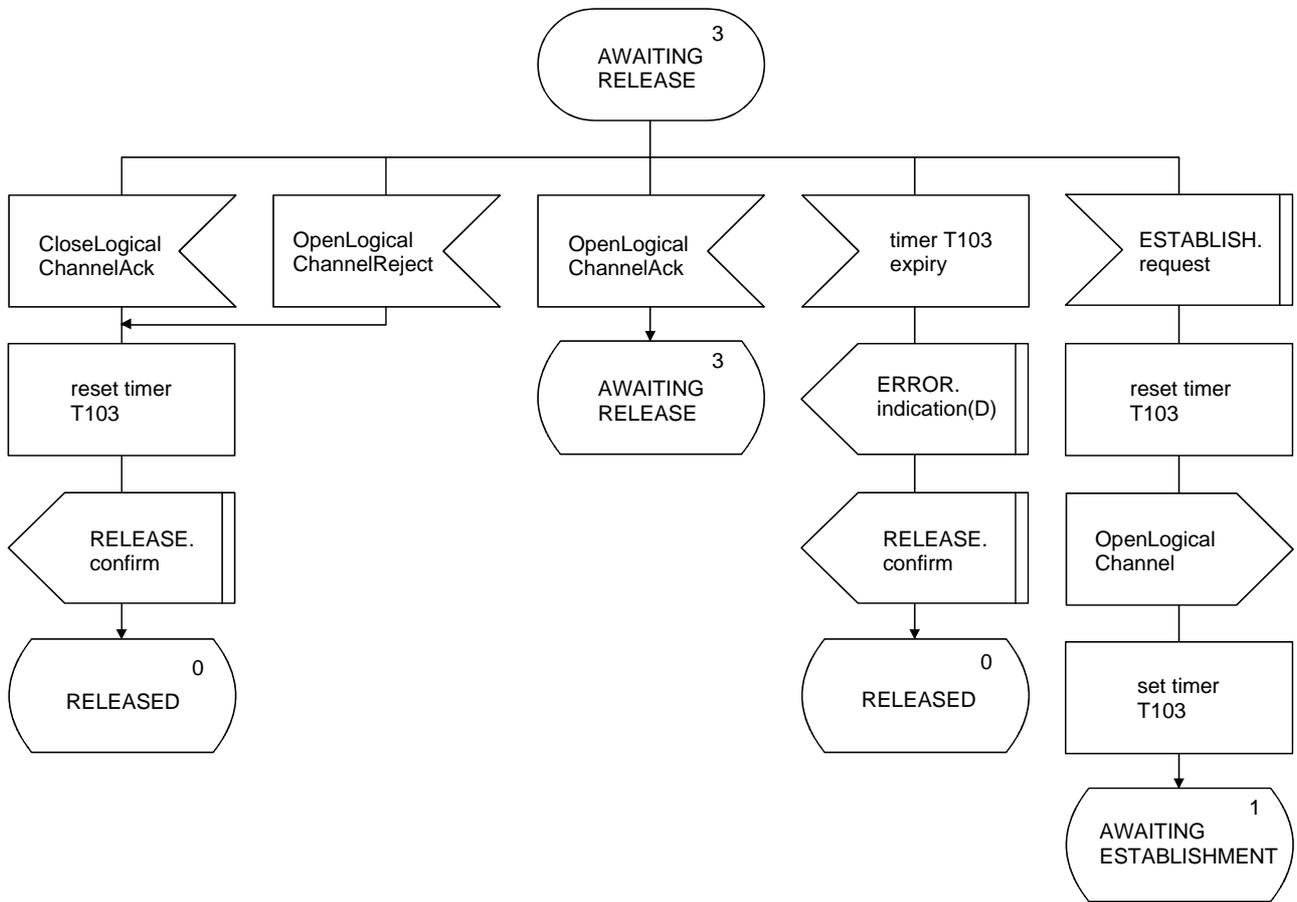
Superseded by a more recent version



T1519310-95/d031

FIGURE 18 iii)/H.245
Outgoing B-LCSE SDL

Superseded by a more recent version



T1519320-95/d032

FIGURE 18 iv)/H.245
Outgoing B-LCSE SDL

Superseded by a more recent version

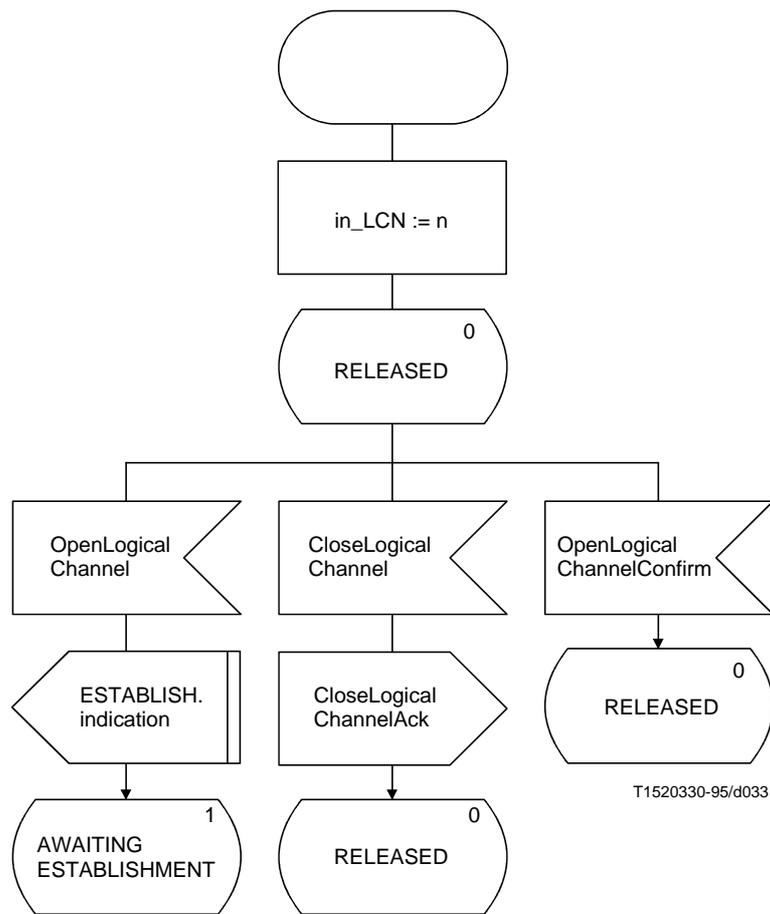
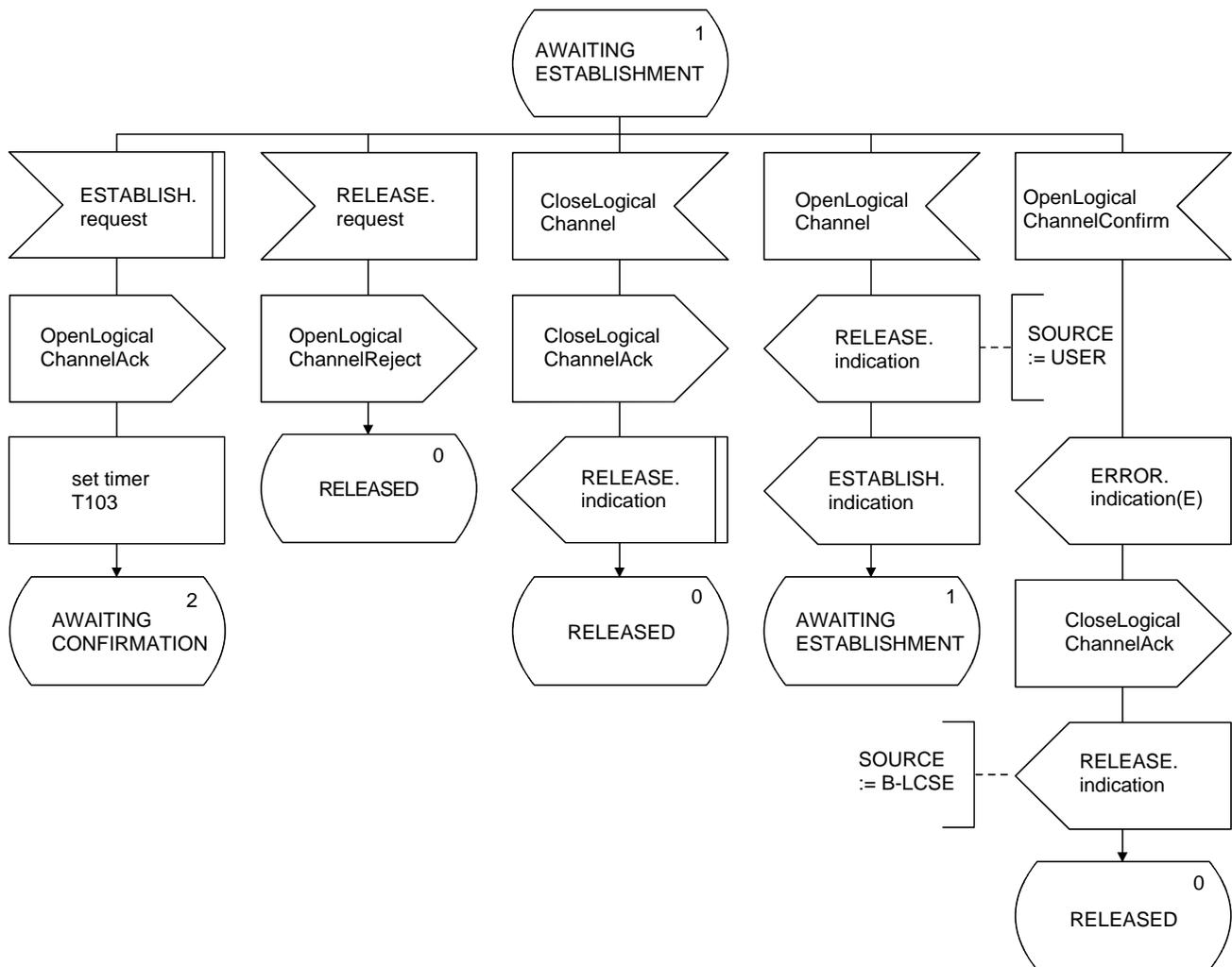


FIGURE 19 i)/H.245
Incoming B-LCSE SDL

Superseded by a more recent version



T1520340-95/d034

FIGURE 19 ii)/H.245
Incoming B-LCSE SDL

Superseded by a more recent version

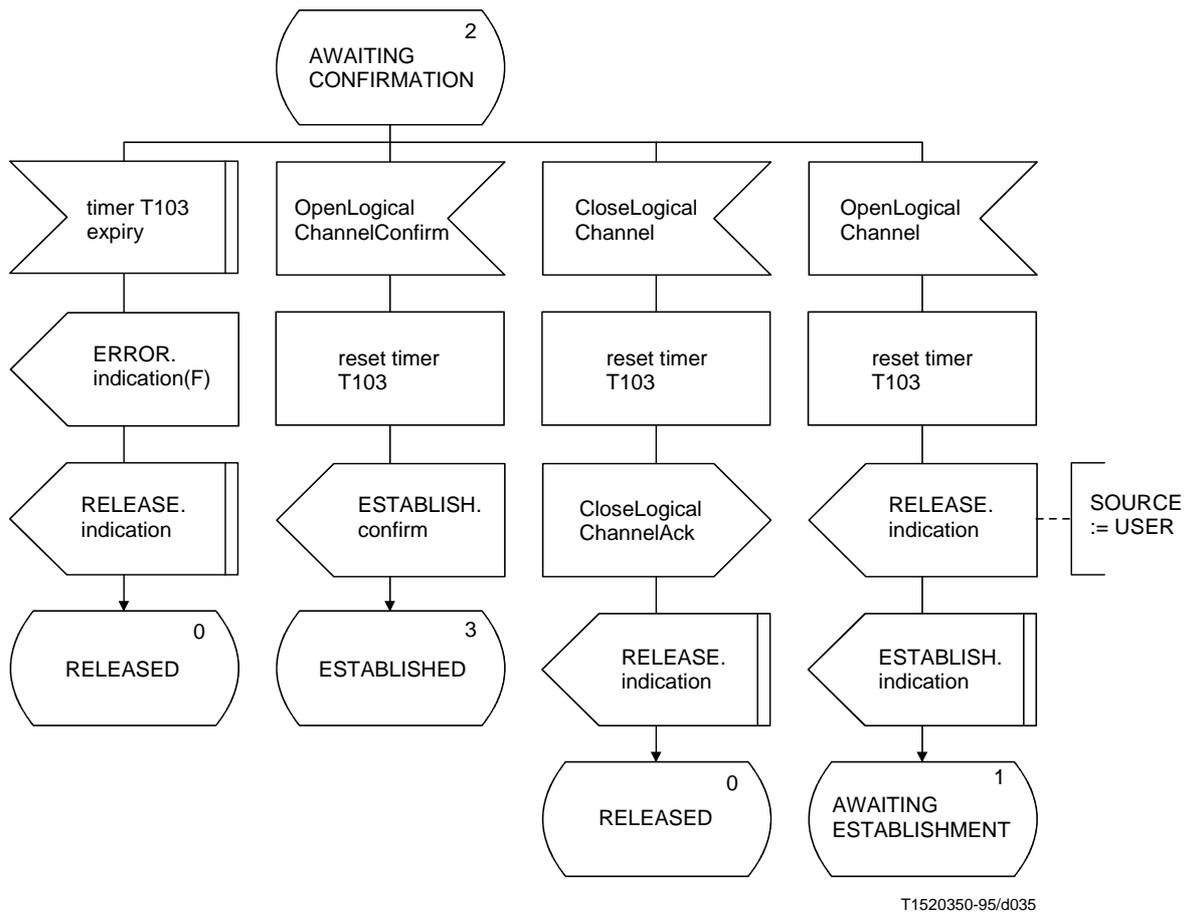


FIGURE 19 iii)/H.245
Incoming B-LCSE SDL

Superseded by a more recent version

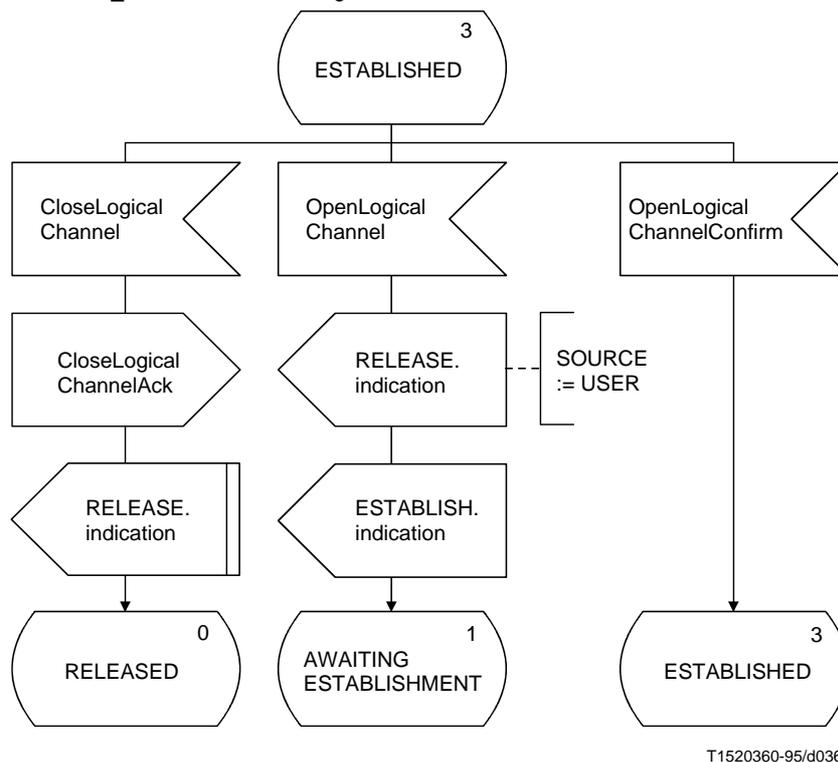


FIGURE 19 iv)/H.245
Incoming B-LCSE SDL

8.6 Close Logical Channel procedures

8.6.1 Introduction

These procedures are used by a terminal to request the remote terminal to close a logical channel. Note that these are only close request procedures; the actual logical channel close occurs using the LCSE and B-LCSE procedures. The procedures are referred to here as the Close Logical Channel Signalling Entity (CLCSE). Procedures are specified in terms of primitives and states at the interface between the CLCSE user. Protocol information is transferred to the peer CLCSE via relevant messages defined in clause 6. There is an outgoing CLCSE and an incoming CLCSE. At each of the outgoing and incoming ends there is one instance of the CLCSE for each logical channel.

If a terminal is incapable of processing the incoming signals, it may use these procedures to request the closing of the relevant logical channels. The following text provides an overview of the operation of the protocol. In the case of any discrepancy with the formal specification of the protocol that follows, the formal specification will supersede.

A terminal that answers such a response positively, that is, by issuing the CLOSE.response primitive, shall initiate the closing of the logical channel by sending the RELEASE.request primitive to the appropriate LCSE or B-LCSE as soon as possible.

8.6.1.1 Protocol overview – Outgoing CLCSE

A close logical channel request procedure is initiated when the CLOSE.request primitive is issued by the user at the outgoing CLCSE. A RequestChannelClose message is sent to the peer incoming CLCSE, and timer T108 is started. If a RequestChannelCloseAck message is received in response to the RequestChannelClose message, then timer T108 is stopped and the user is informed with the CLOSE.confirm primitive that the close logical channel request procedure was successful. If however a RequestChannelCloseReject message is received in response to the RequestChannelClose

Superseded by a more recent version

message, then timer T108 is stopped and the user is informed with the REJECT.indication primitive that the peer CLCSE user has refused to close the logical channel.

If timer T108 expires, then the outgoing CLCSE user is informed with the REJECT.indication primitive and a RequestChannelCloseRelease message is sent.

8.6.1.2 Protocol overview – Incoming CLCSE

When a RequestChannelClose message is received at the incoming CLCSE, the user is informed of the close logical channel request with the CLOSE.indication primitive. The incoming CLCSE user signals acceptance of the close logical channel request by issuing the CLOSE.response primitive, and a RequestChannelCloseAck message is sent to the peer outgoing CLCSE. The incoming CLCSE user signals rejection of the close logical channel request by issuing the REJECT.request primitive, and a RequestChannelCloseReject message is sent to the peer outgoing CLCSE.

8.6.2 Communication between CLCSE and CLCSE user

8.6.2.1 Primitives between CLCSE and CLCSE user

Communication between the CLCSE and CLCSE user, is performed using the primitives shown in Table 37.

TABLE 37/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
CLOSE	– (Note 1)	–	–	–
REJECT	CAUSE	SOURCE CAUSE	Not defined (Note 2)	Not defined

NOTE 1 – “–” means no parameters.
NOTE 2 – “Not defined” means that this primitive is not defined.

8.6.2.2 Primitive definition

The definition of these primitives is as follows:

- the CLOSE primitives are used to request closure of a logical channel;
- the REJECT primitives are used to reject the closing of a logical channel.

8.6.2.3 Parameter definition

The definition of the primitive parameters shown in Table 37 are as follows:

- The SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of “USER” or “PROTOCOL”. The latter case may occur as the result of a timer expiry.
- The CAUSE parameter indicates the reason for refusal to close a logical channel. The CAUSE parameter is not present when the SOURCE parameter indicates “PROTOCOL”.

8.6.2.4 CLCSE states

The following states are used to specify the allowed sequence of primitives between the CLCSE and the CLCSE user.

The states for an outgoing CLCSE are:

State 0: IDLE

The CLCSE is idle.

Superseded by a more recent version

State 1: AWAITING RESPONSE

The CLCSE is waiting for a response from the remote CLCSE.

The states for an incoming CLCSE are:

State 0: IDLE

The CLCSE is idle.

State 1: AWAITING RESPONSE

The CLCSE is waiting for a response from the CLCSE user.

8.6.2.5 State transition diagram

The allowed sequence of primitives between the CLCSE and the CLCSE user is defined here. The allowed sequences are specified separately for each of an outgoing CLCSE and an incoming CLCSE, as shown in Figure 20 and Figure 21 respectively.

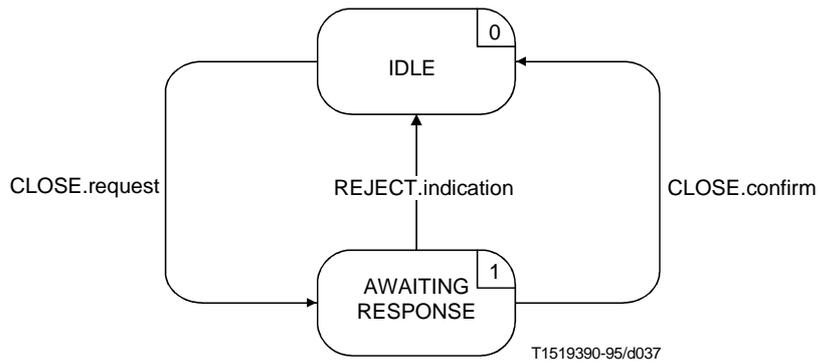


FIGURE 20/H.245

State transition diagram for sequence of primitives at CLCSE outgoing

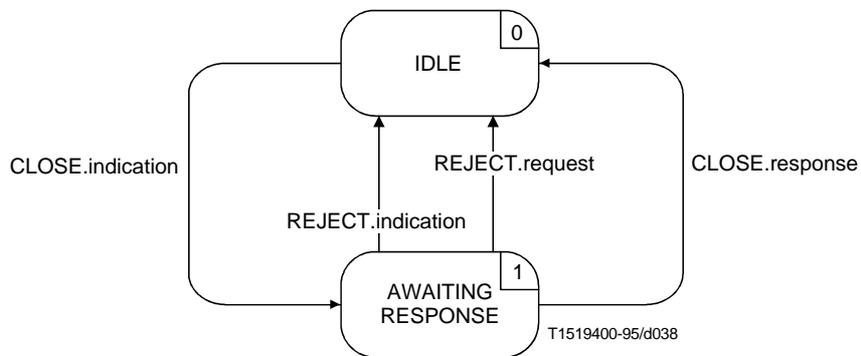


FIGURE 21/H.245

State transition diagram for sequence of primitives at CLCSE incoming

Superseded by a more recent version

8.6.3 Peer-to-peer CLCSE communication

8.6.3.1 Messages

Table 38 shows the CLCSE messages and fields, defined in clause 6, which are relevant to the CLCSE protocol.

TABLE 38/H.245

CLCSE message names and fields

Function	Message	Direction	Field
Transfer	RequestChannelClose	O → I	forwardLogicalChannelNumber
	RequestChannelCloseAck	O ← I	forwardLogicalChannelNumber
	RequestChannelCloseReject	O ← I	forwardLogicalChannelNumber
Reset	RequestChannelCloseRelease	O → I	forwardLogicalChannelNumber
O	Outgoing		
I	Incoming		

8.6.3.2 CLCSE state variables

The following state variable is defined at the outgoing CLCSE:

out_LCN

This state variable distinguishes between outgoing CLCSEs. It is initialized at outgoing CLCSE initialization. The value of out_LCN is used to set the forwardLogicalChannelNumber field of CLCSE messages sent from an outgoing CLCSE. For CLCSE messages received at an outgoing CLCSE, the message forwardLogicalChannelNumber field value is identical to the value of out_LCN.

The following state variable is defined at the incoming CLCSE:

in_LCN

This state variable distinguishes between incoming CLCSEs. It is initialized at incoming CLCSE initialization. The value of in_LCN is used to set the forwardLogicalChannelNumber field of CLCSE messages sent from an incoming CLCSE. For CLCSE messages received at an incoming CLCSE, the message forwardLogicalChannelNumber field value is identical to the value of in_LCN.

8.6.3.3 CLCSE timers

The following timer is specified for the outgoing CLCSE:

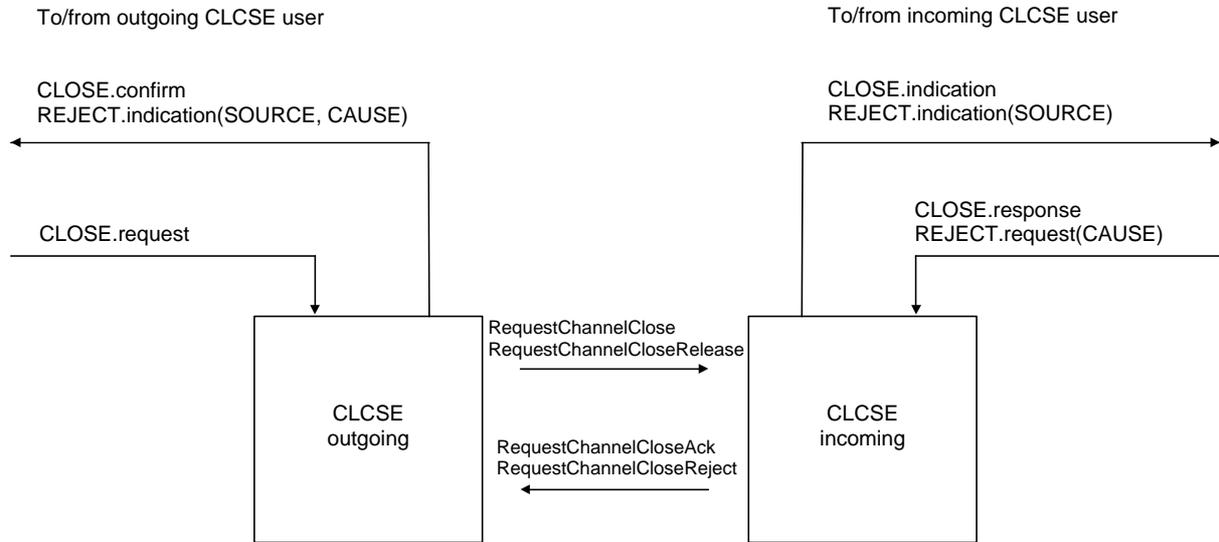
T108

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no RequestChannelCloseAck or RequestChannelCloseReject message maybe received.

8.6.4 CLCSE procedures

Figure 22 summarizes the CLCSE primitives and their parameters, and messages, for each of the outgoing and incoming CLCSE.

Superseded by a more recent version



T1519410-95/d039

FIGURE 22/H.245

Primitives and messages in the Close Logical Channel Signalling Entity

8.6.4.1 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table 39.

TABLE 39/H.245

Default primitive parameter values

Primitive	Parameter	Default value
REJECT.indication	SOURCE	USER
	CAUSE	null

8.6.4.2. Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 40.

TABLE 40/H.245

Default message field values

Message	Field	Default value
RequestChannelClose	forwardLogicalChannelNumber	out_LCN
RequestChannelCloseAck	forwardLogicalChannelNumber	in_LCN
RequestChannelCloseReject	forwardLogicalChannelNumber	in_LCN
	cause	REJECT.request(CAUSE)
RequestChannelCloseRelease	forwardLogicalChannelNumber	out_LCN

Superseded by a more recent version

8.6.4.3 SDLs

The outgoing CLCSE and the incoming CLCSE procedures are expressed in SDL form in Figures 23 and 24 respectively.

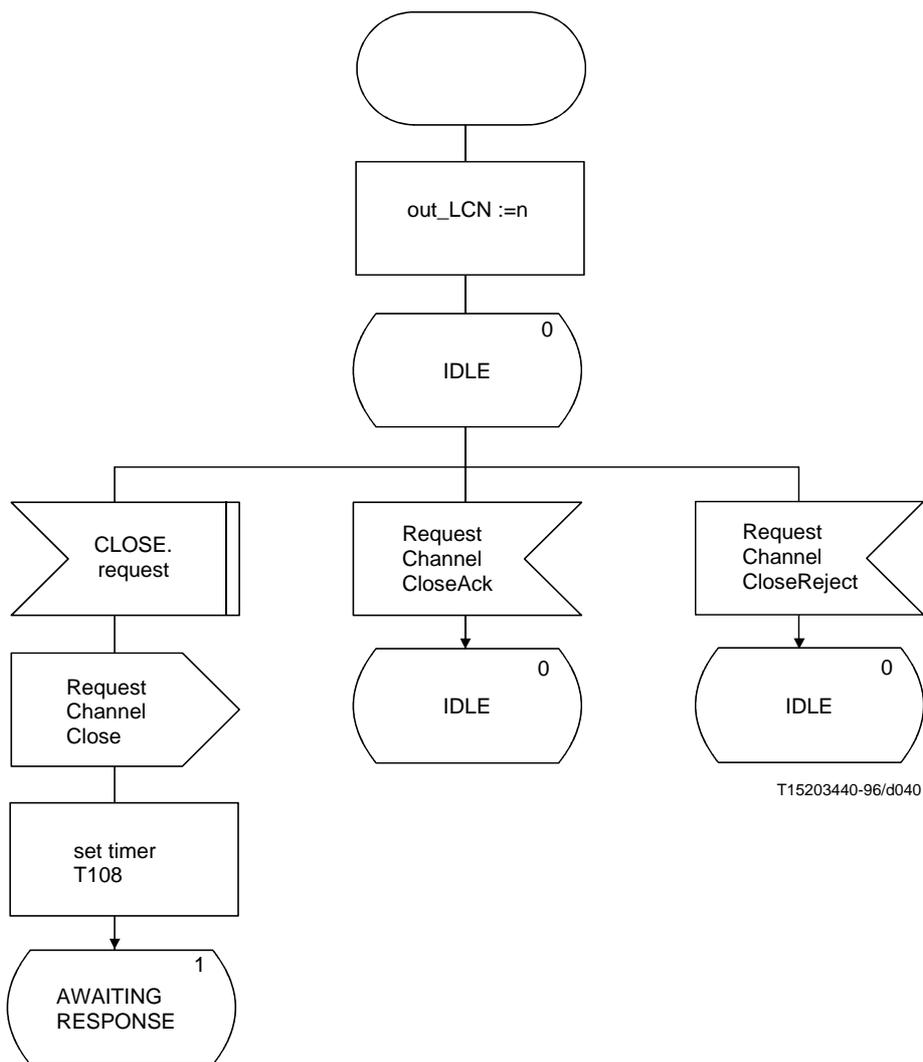
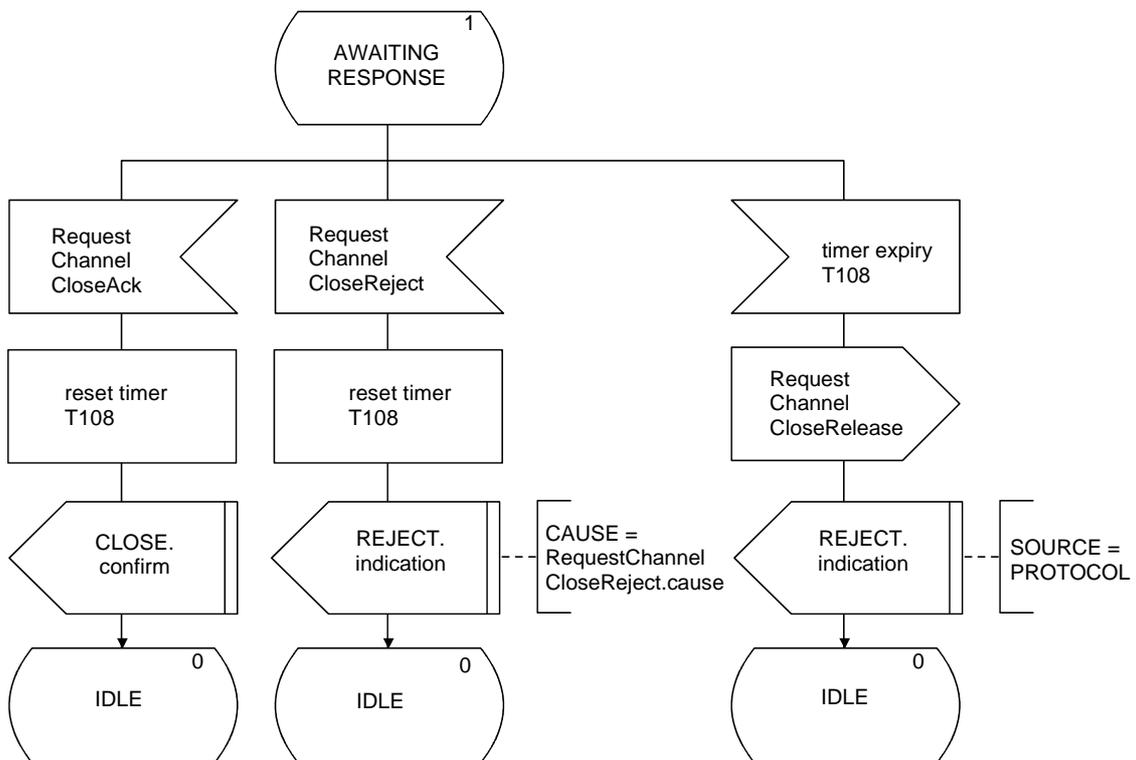


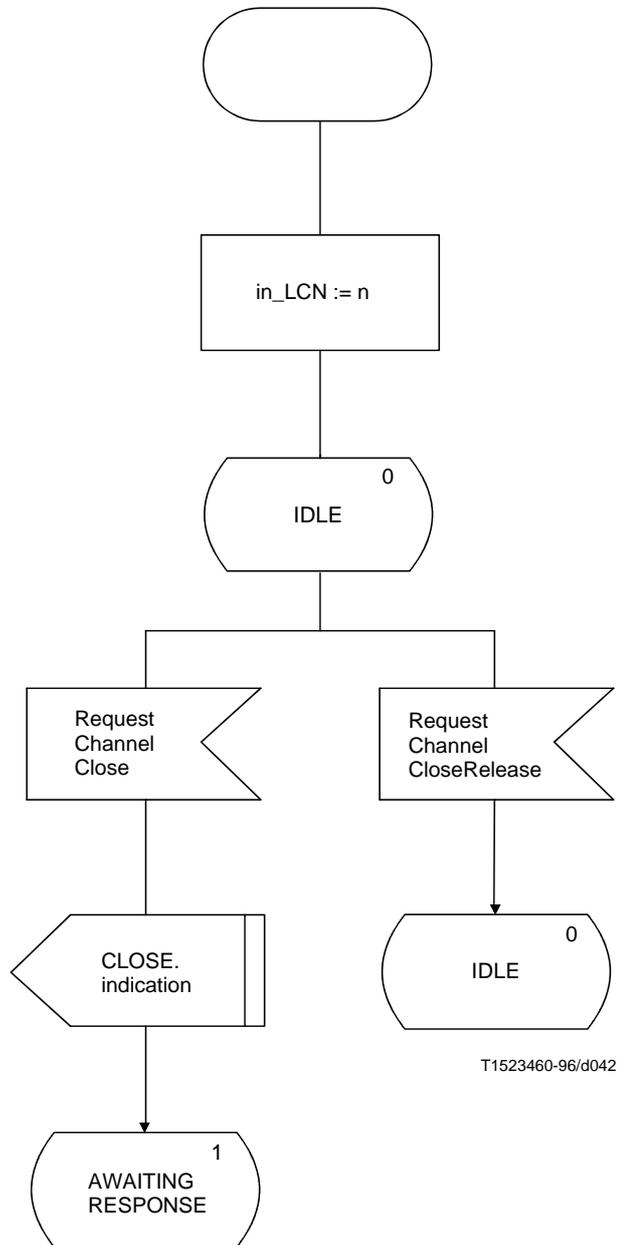
FIGURE 23 i)/H.245
Outgoing CLCSE SDL

Superseded by a more recent version



T1523450-96/d041

FIGURE 23 ii)/H.245
Outgoing CLCSE SDL



T1523460-96/d042

FIGURE 24 i)/H.245
Incoming CLCSE SDL

Superseded by a more recent version

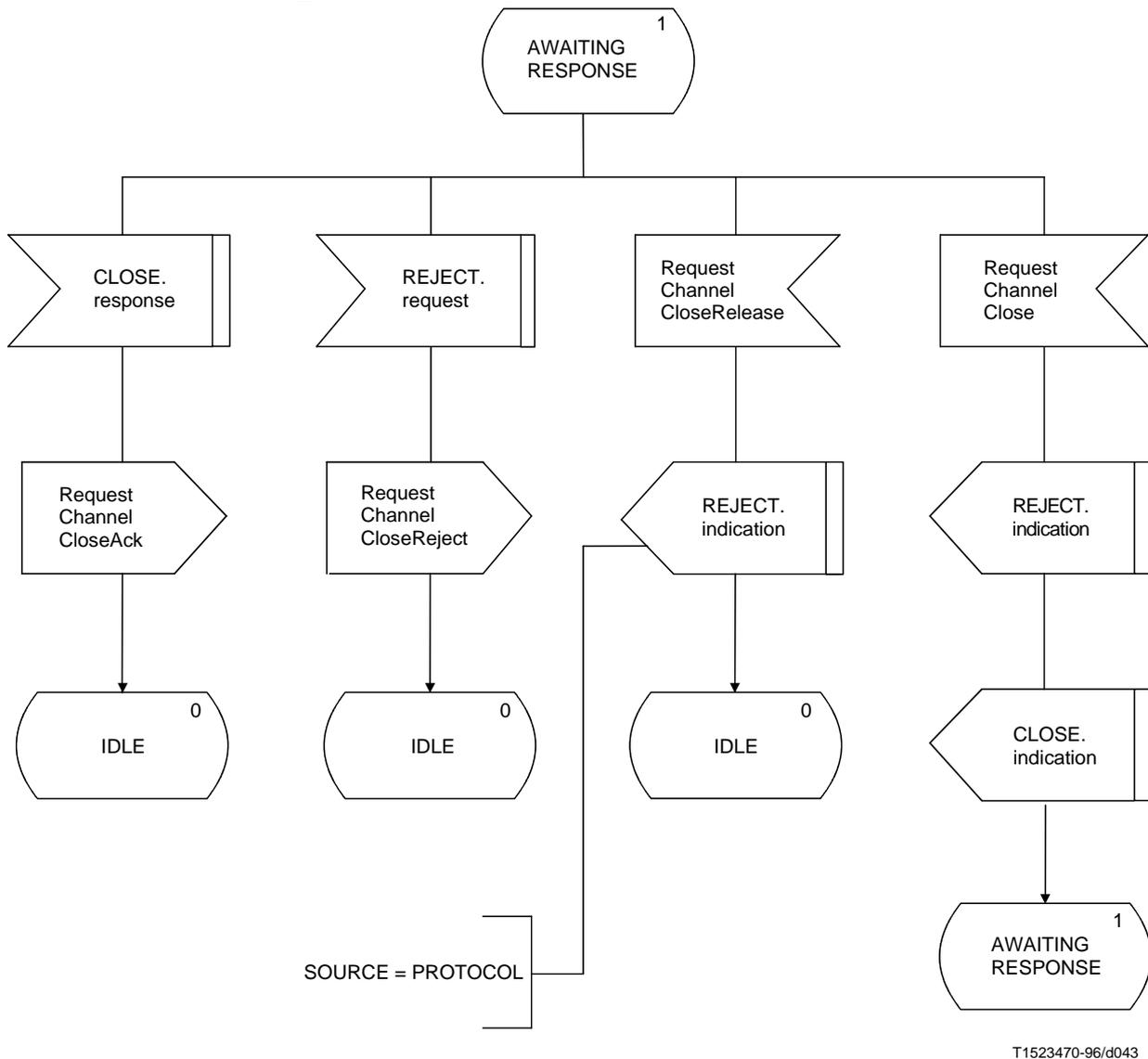


FIGURE 24 ii)/H.245
Incoming CLCSE SDL

8.7 H.223 Multiplex Table procedures

8.7.1 Introduction

The multiplex table serves to associate each octet within a H.223 MUX-message [8] with a particular logical channel number. The H.223 multiplex table may have up to 16 entries, numbered from 0 to 15. Table entries 1 to 15 shall be sent from transmitters to receivers as specified in the following procedures.

The procedures described here are referred to as the Multiplex Table Signalling Entity (MTSE). Procedures are specified in terms of primitives and states at the interface between the MTSE and the MTSE user. Protocol information is transferred to the peer MTSE via relevant messages defined in clause 6.

There is an outgoing MTSE and an incoming MTSE. There is one instance of the MTSE for each multiplex table entry.

Superseded by a more recent version

A transmit terminal uses this protocol to signal to a remote terminal one or more new multiplex table entries. The remote terminal may accept or reject the new multiplex table entries. If the remote terminal accepts a multiplex table entry, the previous entry at the given entry number is replaced with the new entry.

The transmitter may deactivate a multiplex table entry by sending a MultiplexEntryDescriptor with no elementList. The transmitter shall at no time use a multiplex table entry that is deactivated. Before transmitting a MultiplexEntrySend, the transmitter shall stop using the entries that are described by it. It shall not restart using those entries until it has received a MultiplexEntrySendAck. This procedure is used because if the use of these multiplex table entries is not stopped before sending the MultiplexEntrySend, errors may cause an ambiguity in the receiver.

The transmitter shall stop using deactivated entries before sending the MultiplexEntrySend indicating that they have been deactivated. Deactivated entries may be used again at any time by transmitting a MultiplexEntrySend message for activating that entry. Deactivating entries that are no longer required by the transmitter may increase the probability of detecting errors in the H.223 Multiplex Code field.

NOTE – While some multiplex table entries are being updated, other (active) entries may continue to be used. Also, a multiplex table entry may be deleted in the same MultiplexEntry send that is used to modify other multiplex table entries.

At the start of communication, unless specified otherwise in an appropriate recommendation, only table entry 0 is available for transmission, and table entries 1 to 15 are deactivated.

A Request Multiplex Entry procedure may be used at any time to elicit retransmission of specified multiplex table entries from the remote terminal, for example, following an interruption or other cause for uncertainty.

The following text provides an overview of the operation of the protocol. In the case of any discrepancy with the formal specification of the protocol that follows, the formal specification will supersede.

8.7.1.1 Protocol overview – Outgoing MTSE

A multiplex table entry send request procedure is initiated when the TRANSFER.request primitive is issued by the user at the outgoing MTSE. A MultiplexEntrySend message is sent to the peer incoming MTSE, and timer T104 is started. If a MultiplexEntrySendAck message is received in response to the MultiplexEntrySend message, then timer T104 is stopped and the user is informed with the TRANSFER.confirm primitive that the multiplex table entry send request was successful. If however, a MultiplexEntrySendReject message is received in response to the MultiplexEntrySend message, then timer T104 is stopped and the user is informed with the REJECT.indication primitive that the peer MTSE user has refused to accept the multiplex table entry.

If timer T104 expires, then the outgoing MTSE user is informed with the REJECT.indication primitive and a MultiplexEntrySendRelease message is sent.

Only MultiplexEntrySendAck and MultiplexEntrySendReject messages which are in response to the most recent MultiplexEntrySend message are accepted. Responses to earlier MultiplexEntrySend messages are ignored.

A new multiplex table entry send request procedure may be initiated with the TRANSFER.request primitive by the user at the outgoing MTSE before a MultiplexEntrySendAck or a MultiplexEntrySendReject message has been received.

8.7.1.2 Protocol overview – Incoming MTSE

When a MultiplexEntrySend message is received at the incoming MTSE, the user is informed of the multiplex table entry send request with the TRANSFER.indication primitive. The incoming MTSE user signals acceptance of the multiplex table entry by issuing the TRANSFER.response primitive, and a MultiplexEntrySendAck message is sent to the peer outgoing MTSE. The incoming MTSE user signals rejection of the multiplex table entry by issuing the REJECT.request primitive, and a MultiplexEntrySendReject message is sent to the peer outgoing MTSE.

A new MultiplexEntrySend message may be received before the incoming MTSE user has responded to an earlier MultiplexEntrySend message. The incoming MTSE user is informed with the REJECT.indication primitive, followed by the TRANSFER.indication primitive, and the incoming MTSE user responds to the new multiplex table entry.

If a MultiplexEntrySendRelease message is received before the incoming MTSE user has responded to an earlier MultiplexEntrySend message, then the incoming MTSE user is informed with the REJECT.indication, and the earlier multiplex table entry is discarded.

Superseded by a more recent version

8.7.2 Communication between MTSE and MTSE user

8.7.2.1 Primitives between MTSE and MTSE user

Communication between the MTSE and MTSE user, is performed using the primitives shown in Table 41.

TABLE 41/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
TRANSFER	MUX-DESCRIPTOR	MUX-DESCRIPTOR	– (Note 1)	–
REJECT	CAUSE	SOURCE CAUSE	Not defined (Note 2)	Not defined

NOTE 1 – “–” means no parameters.
NOTE 2 – “Not defined” means that this primitive is not defined.

8.7.2.2 Primitive definition

The definition of these primitives is as follows:

- a) the TRANSFER primitives are used to transfer multiplex table entries;
- b) the REJECT primitives are used to reject a multiplex table entry, and to terminate a multiplex table entry transfer

8.7.2.3 Parameter definition

The definition of the primitive parameters shown in Table 41 are as follows:

- a) The MUX-DESCRIPTOR parameter is a multiplex table entry. This parameter is mapped to the MultiplexEntryDescriptor field of the MultiplexEntrySend message and carried transparently from the MTSE user at the outgoing MTSE to the MTSE user at the incoming MTSE. There may be multiple MUX-DESCRIPTORS associated with the TRANSFER primitive.
- b) The SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of “USER” or “PROTOCOL”. The latter case may occur as the result of a timer expiry.
- c) The CAUSE parameter indicates the reason for rejection of a multiplex table entry. The CAUSE parameter is not present when the SOURCE parameter indicates “PROTOCOL”.

8.7.2.4 MTSE states

The following states are used to specify the allowed sequence of primitives between the MTSE and the MTSE user. The states are specified separately for each of an outgoing MTSE and incoming MTSE.

The states for an outgoing MTSE are:

State 0: IDLE

There is no MTSE transfer in progress. The multiplex table entry may be used by the transmitter.

State 1: AWAITING RESPONSE

The MTSE user has requested the transfer of a multiplex table entry, and a response from the peer MTSE is awaited. The multiplex table entry shall not be used by the transmitter.

Superseded by a more recent version

The states for an incoming MTSE are:

State 0: IDLE

There is no MTSE transfer in progress. The multiplex table entry may be used by the transmitter.

State 1: AWAITING RESPONSE

The peer MTSE has transferred a multiplex table entry, and a response from the peer MTSE is awaited. The multiplex table entry may not be in use by the transmitter.

8.7.2.5 State transition diagram

The allowed sequence of primitives between the MTSE and the MTSE user is defined here. The allowed sequences are specified separately for each of an outgoing MTSE and an incoming MTSE, as shown in Figures 25 and 26 respectively.

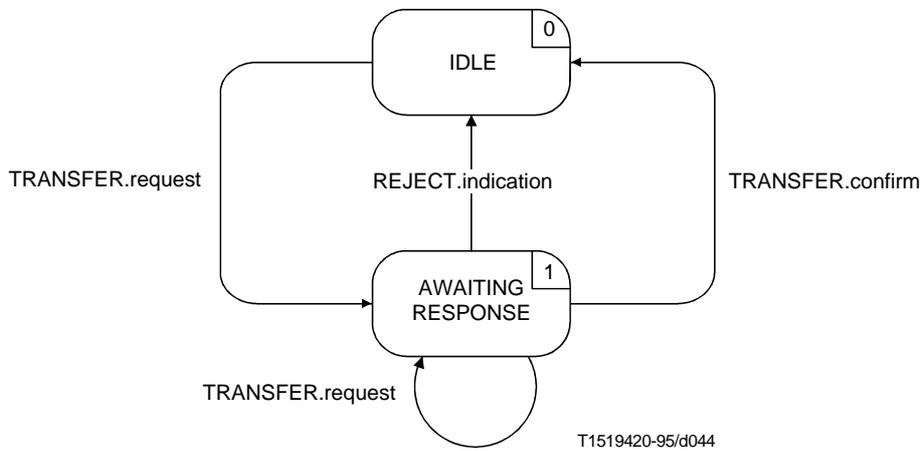


FIGURE 25/H.245

State transition diagram for sequence of primitives at outgoing MTSE

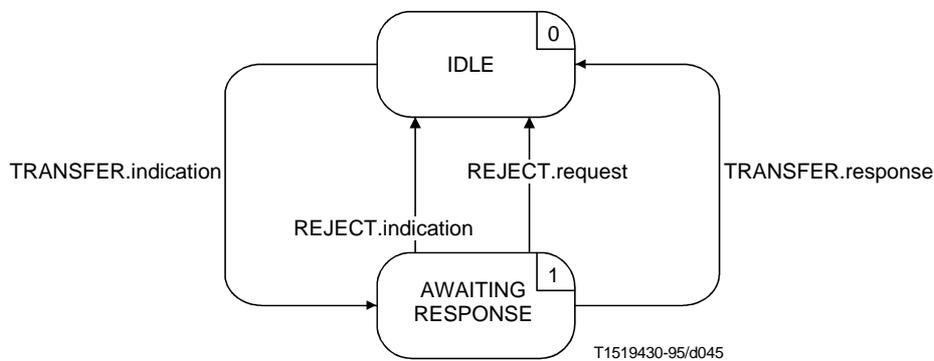


FIGURE 26/H.245

State transition diagram for sequence of primitives at incoming MTSE

Superseded by a more recent version

8.7.3 Peer-to-peer MTSE communication

8.7.3.1 Messages

Table 42 shows the MTSE messages and fields, defined in clause 6, which are relevant to the MTSE protocol.

TABLE 42/H.245

MTSE message names and fields

Function	Message	Direction	Field
Transfer	MultiplexEntrySend	O → I	sequenceNumber multiplexTableEntryNumber MultiplexElement
	MultiplexEntrySendAck	O ← I	sequenceNumber multiplexTableEntryNumber
Reject	MultiplexEntrySendReject	O ← I	sequenceNumber multiplexTableEntryNumber rejectionDescriptions.Cause
Reset	MultiplexEntrySendRelease	O → I	MultiplexTableEntryNumber
O	Outgoing		
I	Incoming		

8.7.3.2 MTSE state variables

The following state variables are defined at the outgoing MTSE:

out_ENUM

This state variable distinguishes between outgoing MTSEs. It is initialized at outgoing MTSE initialization. The value of out_ENUM is used to set the multiplexTableEntryNumber field of MTSE messages sent from an outgoing MTSE. For MTSE messages received at an outgoing MTSE, the message multiplexTableEntryNumber field value is identical to the value of out_ENUM:

out_SQ

This state variable is used to indicate the most recently sent MultiplexEntrySend message. It is incremented by one and mapped to the MultiplexEntrySend message sequenceNumber field before transmission of a MultiplexEntrySend message. Arithmetic performed on out_SQ is modulo 256.

The following state variables are defined at the incoming MTSE:

in_ENUM

This state variable distinguishes between incoming MTSEs. It is initialized at incoming MTSE initialization. The value of in_ENUM is used to set the multiplexTableEntryNumber field of MTSE messages sent from an incoming MTSE. For MTSE messages received at an incoming MTSE, the message multiplexTableEntryNumber field value is identical to the value of in_ENUM:

in_SQ

This state variable is used to store the value of the sequenceNumber field of the most recently received MultiplexEntrySend message. The MultiplexEntrySendAck and MultiplexEntrySendReject messages have their sequenceNumber fields set to the value of in_SQ, before being sent to the peer MTSE.

Superseded by a more recent version

8.7.3.3 MTSE timers

The following timer is specified for the outgoing MTSE:

T107

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no MultiplexEntrySendAck or MultiplexEntrySendReject message may be received.

8.7.4 MTSE procedures

8.7.4.1 Introduction

Figure 27 summarizes the MTSE primitives and their parameters, and the messages and relevant fields, for each of the outgoing and incoming MTSE.

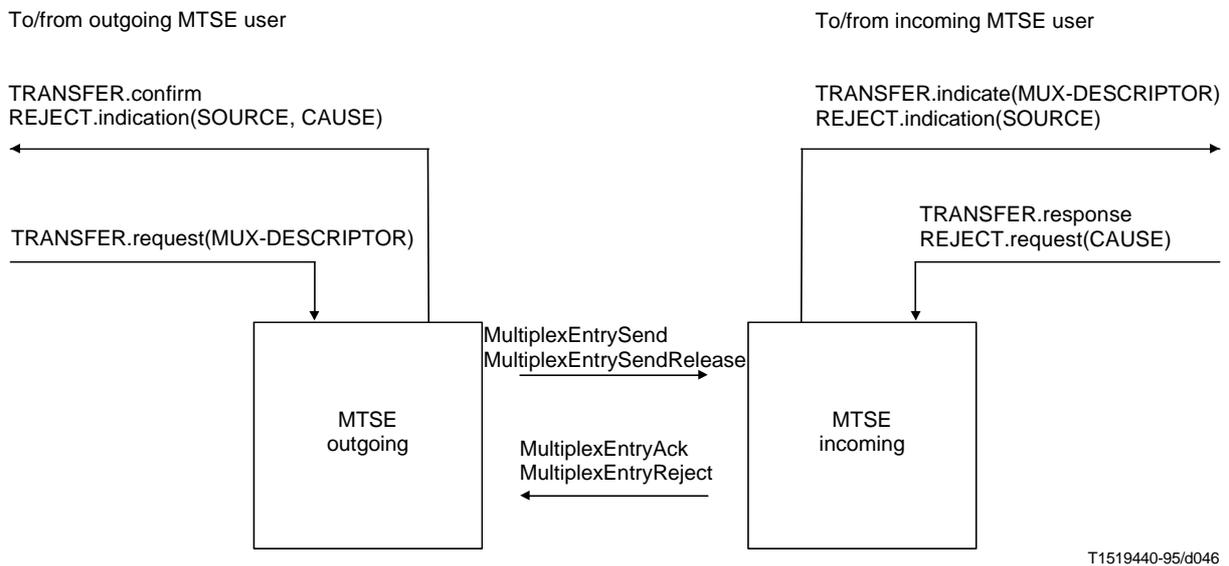


FIGURE 27/H.245

Primitives and messages in the MultiplexTableSignalling Entity

8.7.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table 43.

TABLE 43/H.245

Default primitive parameter values

Primitive	Parameter	Default value
TRANSFER.indication	MUX-DESCRIPTOR	MultiplexEntrySend.MultiplexEntryDescriptor
REJECT.indication	SOURCE	USER
	CAUSE	null

Superseded by a more recent version

8.7.4.3 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 44.

TABLE 44/H.245

Default message field values

Message	Field	Default value (Note)
MultiplexEntrySend	sequenceNumber	out_SQ
	multiplexTableEntryNumber	out_ENUM
	MultiplexElement	TRANSFER.request(MUX-DESCRIPTOR)
MultiplexEntrySendAck	sequenceNumber	in_SQ
	multiplexTableEntryNumber	in_ENUM
MultiplexEntrySendReject	sequenceNumber	in_SQ
	multiplexTableEntryNumber	in_ENUM
	cause	REJECT.request(CAUSE)
MultiplexEntrySendRelease	multiplexTableEntryNumber	out_ENUM

NOTE – A message field shall not be coded, if the corresponding primitive parameter is not null i.e. not present.

8.7.4.4 SDLs

The outgoing MTSE and the incoming MTSE procedures are expressed in SDL form in Figures 28 and 29 respectively.

8.8 Request Multiplex Entry procedures

8.8.1 Introduction

These procedures are used by a terminal to request the retransmission of one or more MultiplexEntryDescriptors. The procedures are referred to here as the Request Multiplex Entry Signalling Entity (RMESE). Procedures are specified in terms of primitives and states at the interface between the RMESE and the RMESE user. Protocol information is transferred to the peer RMESE via relevant messages defined in clause 6. There is an outgoing RMESE and an incoming RMESE. There is one instance of the RMESE for each multiplex table entry.

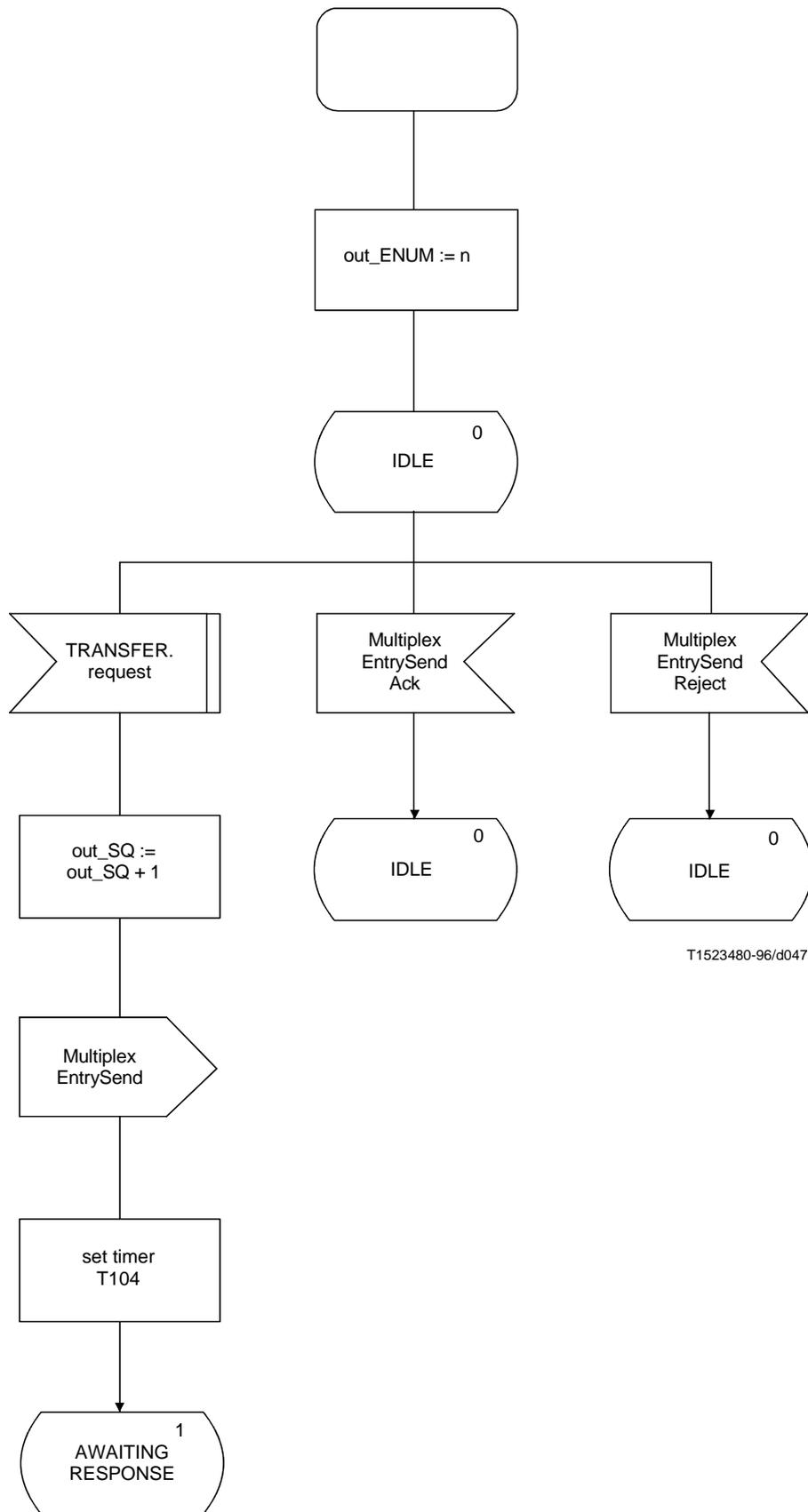
A terminal that answers such a response positively, that is, by issuing the SEND.response primitive, shall initiate the Multiplex Table procedures to send the multiplex table entry as soon as possible.

8.8.1.1 Protocol overview – Outgoing RMESE

A request multiplex entry procedure is initiated when the SEND.request primitive is issued by the user at the outgoing RMESE. A RequestMultiplexEntry message is sent to the peer incoming RMESE, and timer T107 is started. If a RequestMultiplexEntryAck message is received in response to the RequestMultiplexEntry message, then timer T107 is stopped and the user is informed with the SEND.confirm primitive that the request multiplex entry procedure was successful. If however, a RequestMultiplexEntryReject message is received in response to the RequestMultiplexEntry message, then timer T107 is stopped and the user is informed with the REJECT.indication primitive that the peer RMESE user has refused to send the multiplex entry.

If timer T107 expires, then the outgoing RMESE user is informed with the REJECT.indication primitive and a RequestMultiplexEntryRelease message is sent.

Superseded by a more recent version



T1523480-96/d047

FIGURE 28 i)/H.245
Outgoing MTSE SDL

Superseded by a more recent version

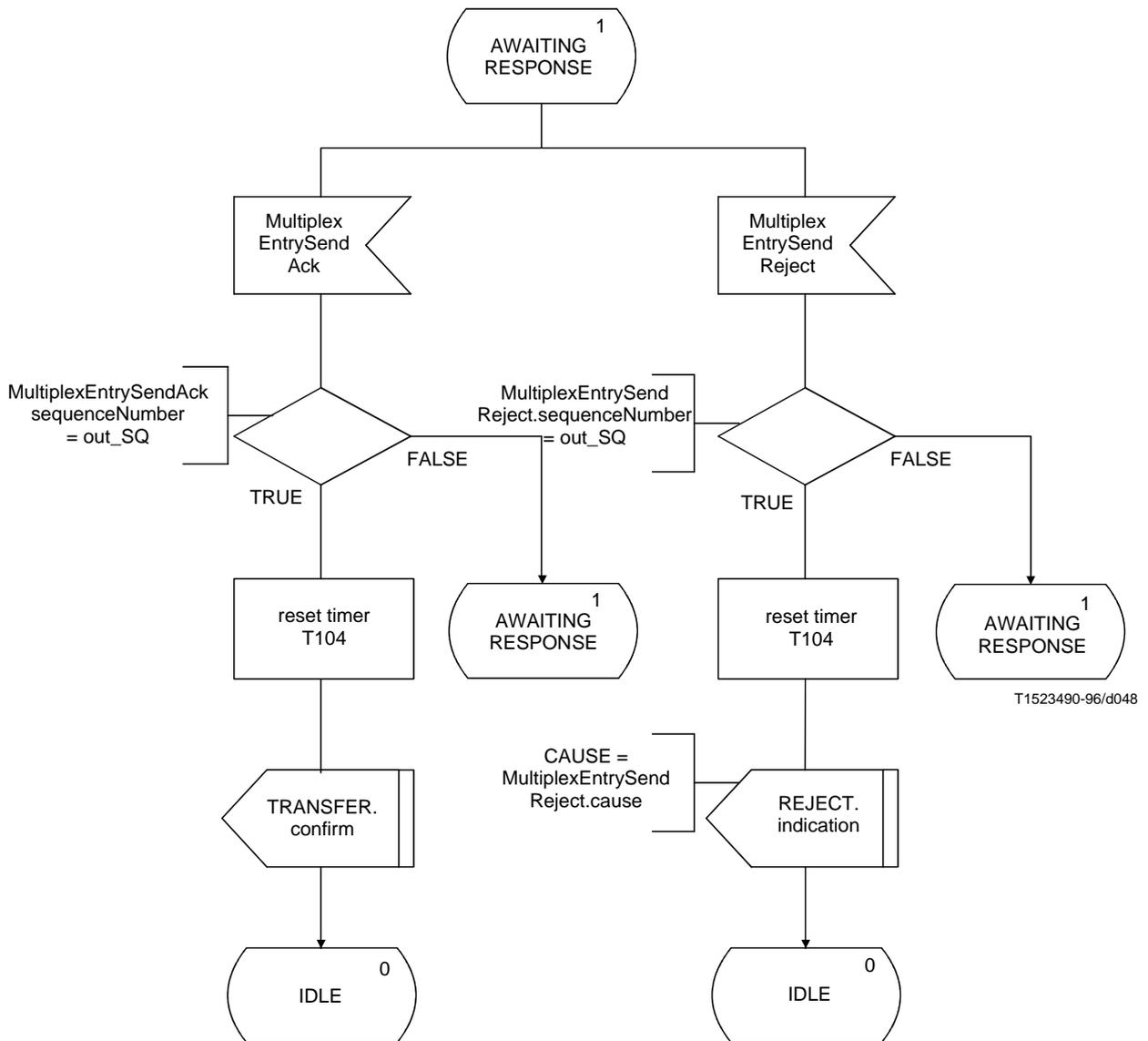
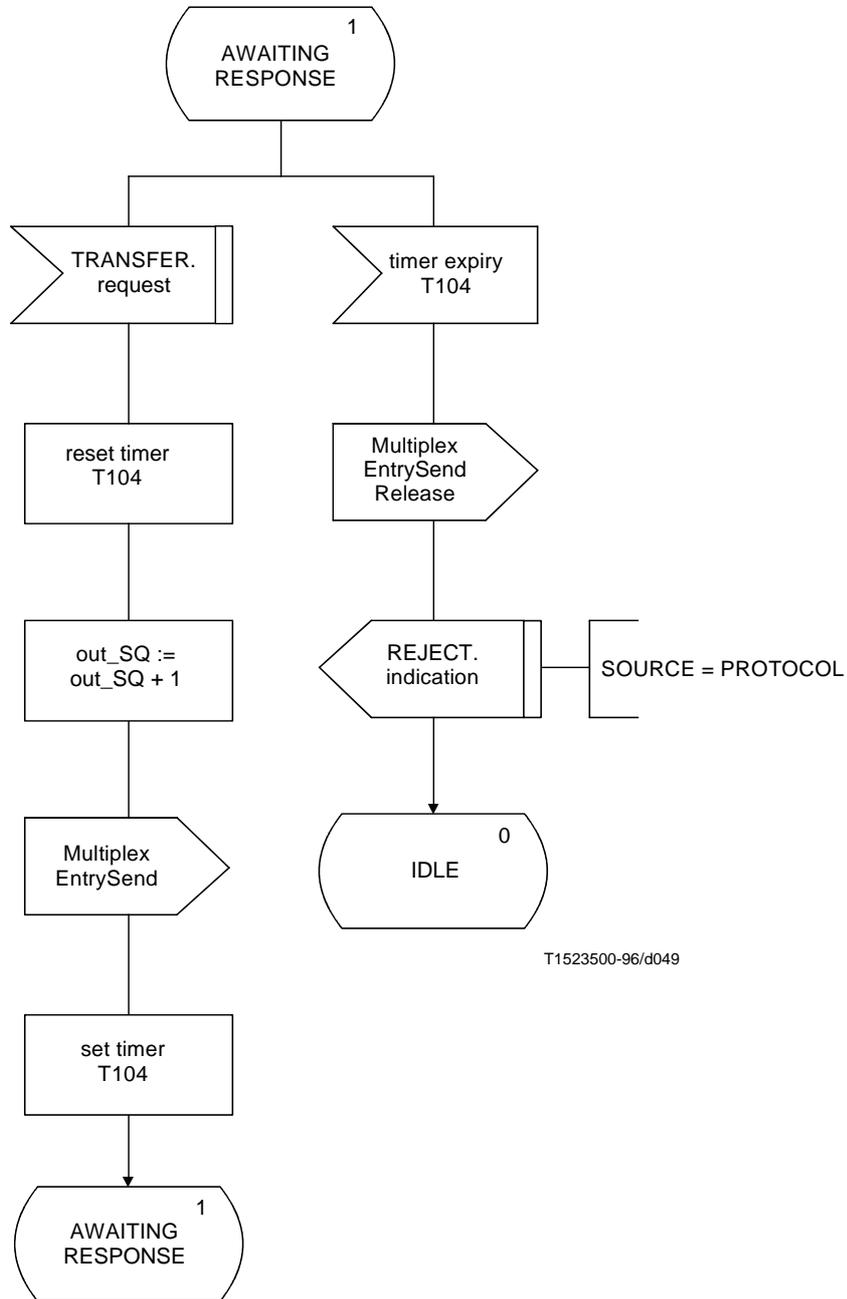


FIGURE 28 ii)/H.245

Outgoing MTSE SDL

Superseded by a more recent version



T1523500-96/d049

FIGURE 28 iii)/H.245
Outgoing MTSE SDL

Superseded by a more recent version

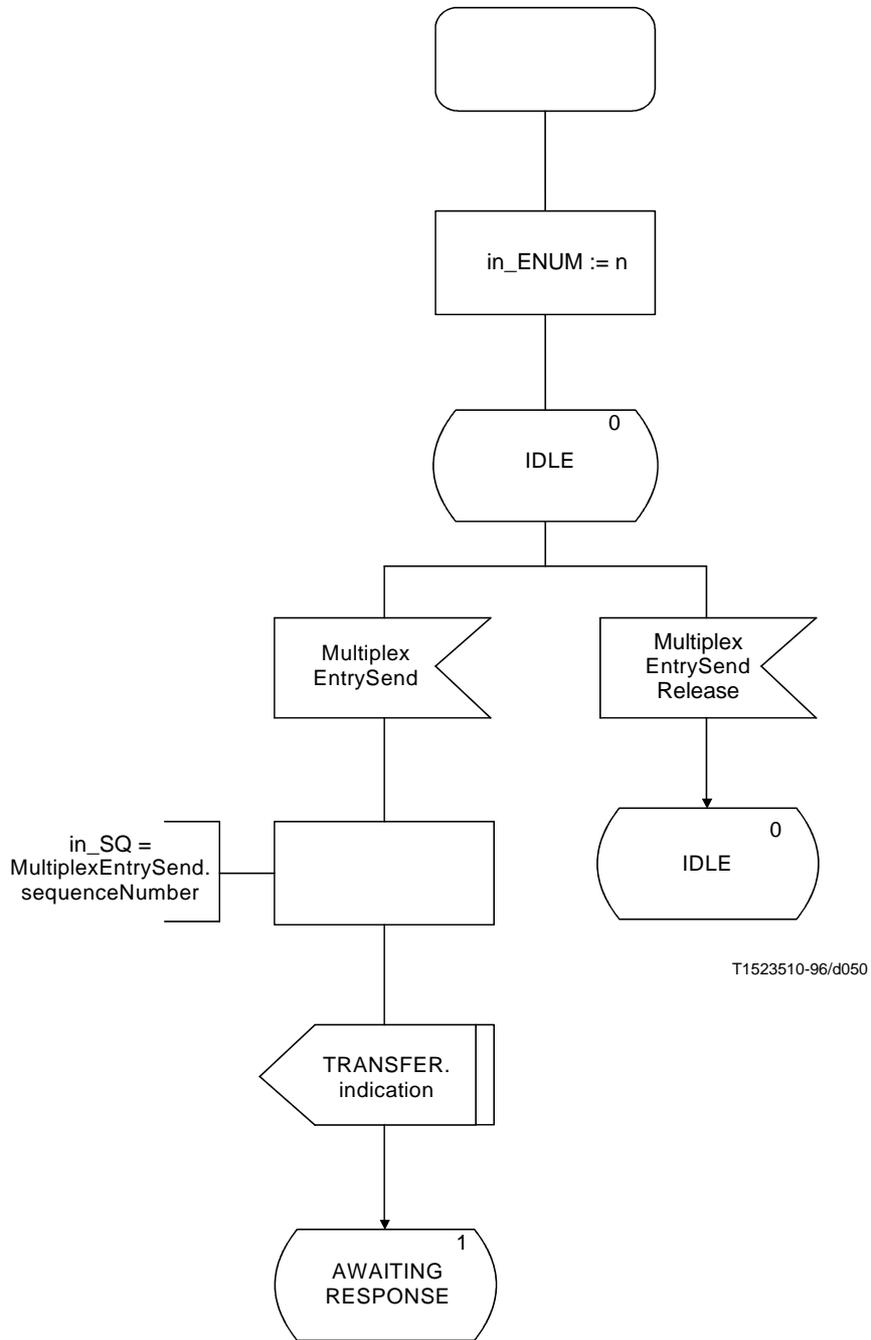
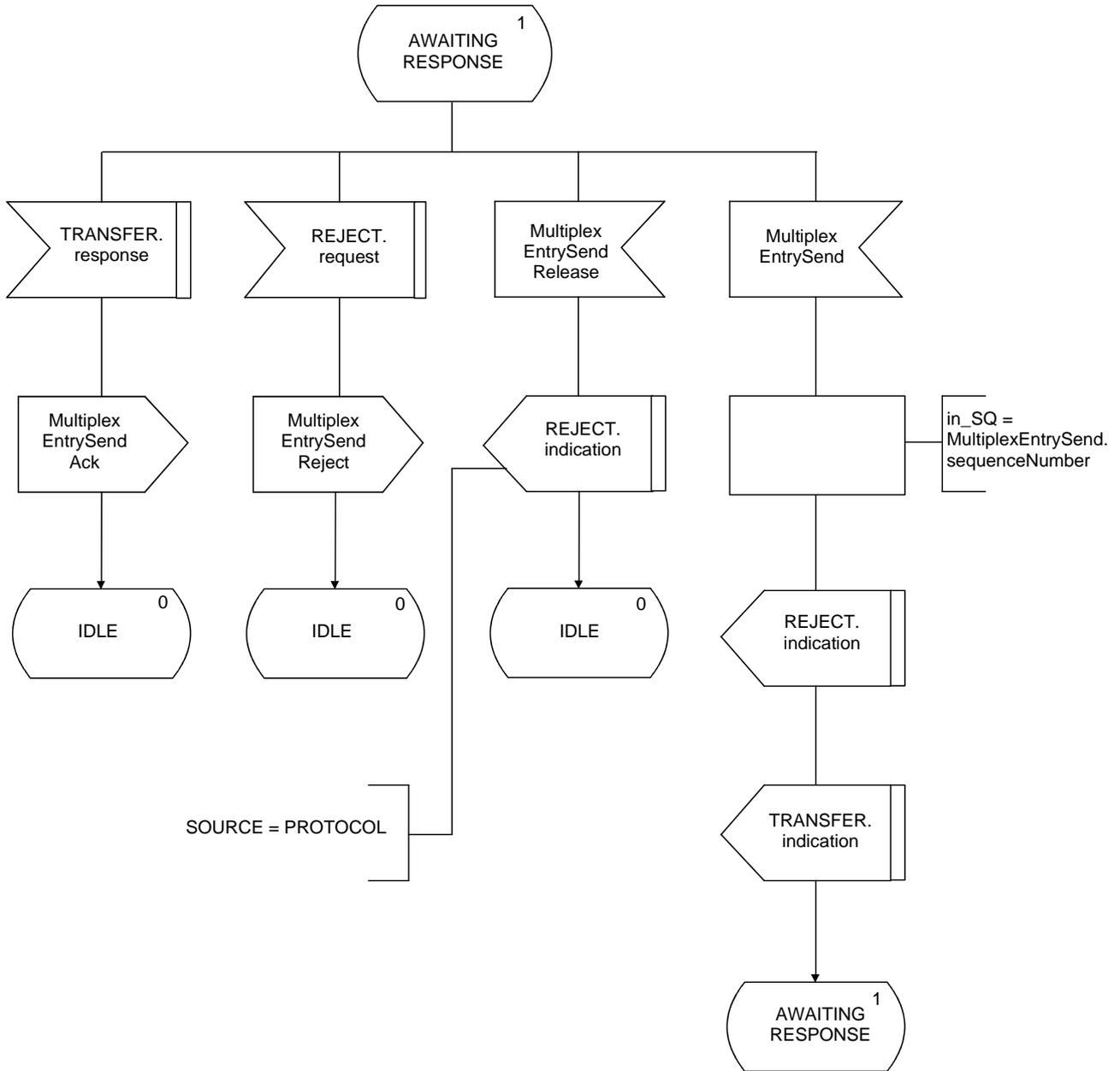


FIGURE 29 i)/H.245
Incoming MTSE SDL

Superseded by a more recent version



T1523520-96/d051

FIGURE 29 ii)/H.245
Incoming MTSE SDL

Superseded by a more recent version

8.8.1.2 Protocol overview – Incoming RMESE

When a RequestMultiplexEntry message is received at the incoming RMESE, the user is informed of the multiplex entry request with the SEND.indication primitive. The incoming RMESE user signals acceptance of the multiplex entry request by issuing the SEND.response primitive, and a RequestMultiplexEntryAck message is sent to the peer outgoing RMESE. The incoming RMESE user signals rejection of the multiplex entry request by issuing the REJECT.request primitive, and a RequestMultiplexEntryReject message is sent to the peer outgoing RMESE.

8.8.2 Communication between RMESE and RMESE user

8.8.2.1 Primitives between RMESE and RMESE user

Communication between the RMESE and RMESE user is performed using the primitives shown in Table 45.

TABLE 45/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
SEND	– (Note 1)	–	–	–
REJECT	CAUSE	SOURCE CAUSE	Not defined (Note 2)	Not defined

NOTE 1 – “–” means no parameters.
NOTE 2 – “Not defined” means that this primitive is not defined.

8.8.2.2 Primitive definition

The definition of these primitives is as follows:

- a) the SEND primitives are used to request the transmission of a multiplex entry.
- b) the REJECT primitives are used to reject the request for transmission of a multiplex entry.

8.8.2.3 Parameter definition

The definition of the primitive parameters shown in Table 45 are as follows:

- a) the SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of “USER” or “PROTOCOL”. The latter case may occur as the result of a timer expiry.
- b) the CAUSE parameter indicates the reason for refusal to send a multiplex table entry. The CAUSE parameter is not present when the SOURCE parameter indicates “PROTOCOL”.

8.8.2.4 RMESE states

The following states are used to specify the allowed sequence of primitives between the RMESE and the RMESE user.

The states for an outgoing RMESE are:

State 0: IDLE

Superseded by a more recent version

The RMESE is idle.

State 1: AWAITING RESPONSE

The RMESE is waiting for a response from the remote RMESE.

The states for an incoming RMESE are:

State 0: IDLE

The RMESE is idle.

State 1: AWAITING RESPONSE

The RMESE is waiting for a response from the RMESE user.

8.8.2.5 State transition diagram

The allowed sequence of primitives between the RMESE and the RMESE user is defined here. The allowed sequences are specified separately for each of an outgoing RMESE and an incoming RMESE, as shown in Figures 30 and 31.

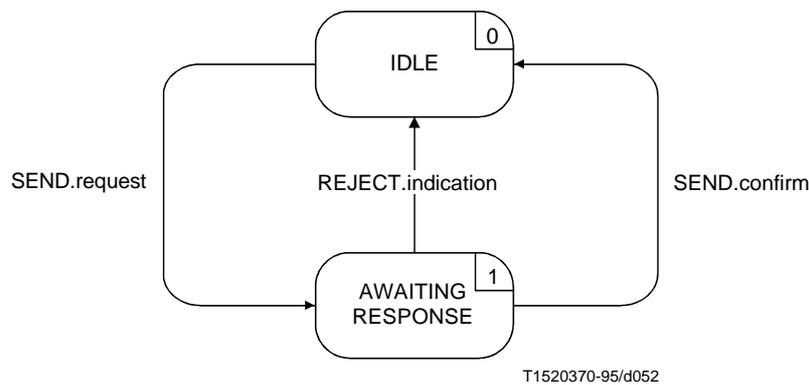


FIGURE 30/H.245

State transition diagram for sequence of primitives at RMESE outgoing

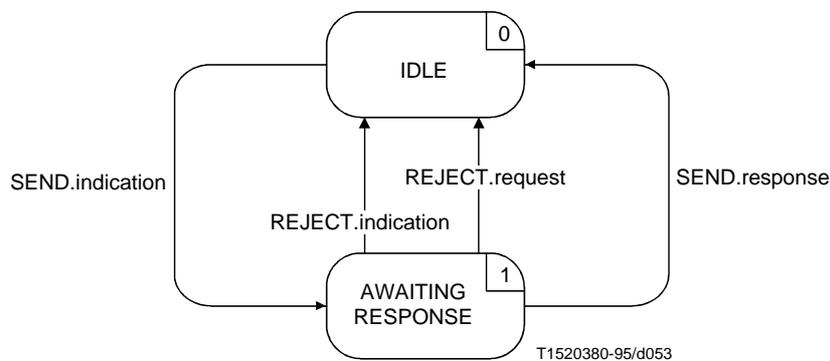


FIGURE 31/H.245

State transition diagram for sequence of primitives at RMESE incoming

Superseded by a more recent version

8.8.3 Peer-to-peer RMESE communication

8.8.3.1 Messages

Table 46 shows the RMESE messages and fields defined in clause 6, which are relevant to the RMESE protocol.

TABLE 46/H.245

RMESE message names and fields

Function	Message	Direction	Field
Transfer	RequestMultiplexEntry	O → I	multiplexTableEntryNumber
	RequestMultiplexEntryAck	O ← I	multiplexTableEntryNumber
	RequestMultiplexEntryReject	O ← I	multiplexTableEntryNumber rejectionDescriptions.cause
Reset	RequestMultiplexEntryRelease	O → I	–
O	Outgoing		
I	Incoming		

8.8.3.2 RMESE state variables

The following state variable is defined at the outgoing RMESE:

out_ENUM

This state variable distinguishes between outgoing RMESEs. It is initialized at outgoing RMESE initialization. The value of out_ENUM is used to set the multiplexTableEntryNumber field of RMESE messages sent from an outgoing RMESE. For RMESE messages received at an outgoing RMESE, the message multiplexTableEntryNumber field value is identical to the value of out_ENUM.

The following state variable is defined at the incoming RMESE:

in_ENUM

This state variable distinguishes between incoming RMESEs. It is initialized at incoming RMESE initialization. The value of in_ENUM is used to set the multiplexTableEntryNumber field of RMESE messages sent from an incoming RMESE. For RMESE messages received at an incoming RMESE, the message multiplexTableEntryNumber field value is identical to the value of in_ENUM.

8.8.3.3 RMESE timers

The following timer is specified for the outgoing RMESE:

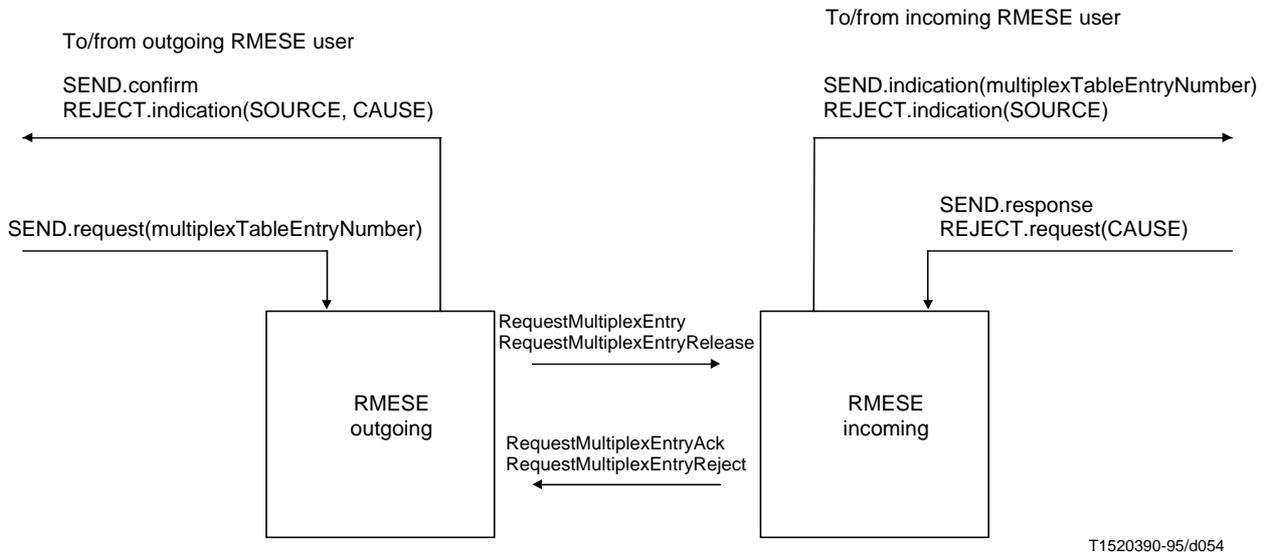
T107

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no RequestMultiplexEntryAck or RequestMultiplexEntryReject message may be received.

8.8.4 RMESE procedures

Figure 32 summarizes the RMESE primitives and their parameters and messages for each of the outgoing and incoming RMESEs.

Superseded by a more recent version



T1520390-95/d054

FIGURE 32/H.245

Primitives and messages in the Request Multiplex Entry Signalling Entity

8.8.4.1 Primitive parameter default values

Where not explicitly stated in the SDL diagrams, the parameters of the indication and confirm primitives assume values as shown in Table 47.

TABLE 47/H.245

Default primitive parameter values

Primitive	Parameter	Default value
REJECT.indication	SOURCE	USER
	CAUSE	null

8.8.4.2 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 48.

TABLE 48/H.245

Default message field values

Message	Field	Default value
RequestMultiplexEntry	multiplexTableEntryNumber	out_ENUM
RequestMultiplexEntryAck	multiplexTableEntryNumber	in_ENUM
RequestMultiplexEntryReject	multiplexTableEntryNumber cause	in_ENUM REJECT.request(CAUSE)
RequestMultiplexEntryRelease	multiplexTableEntryNumber	out_ENUM

Superseded by a more recent version

8.8.4.3 SDLs

The outgoing RMESE and the incoming RMESE procedures are expressed in SDL form in Figures 33 and 34 respectively.

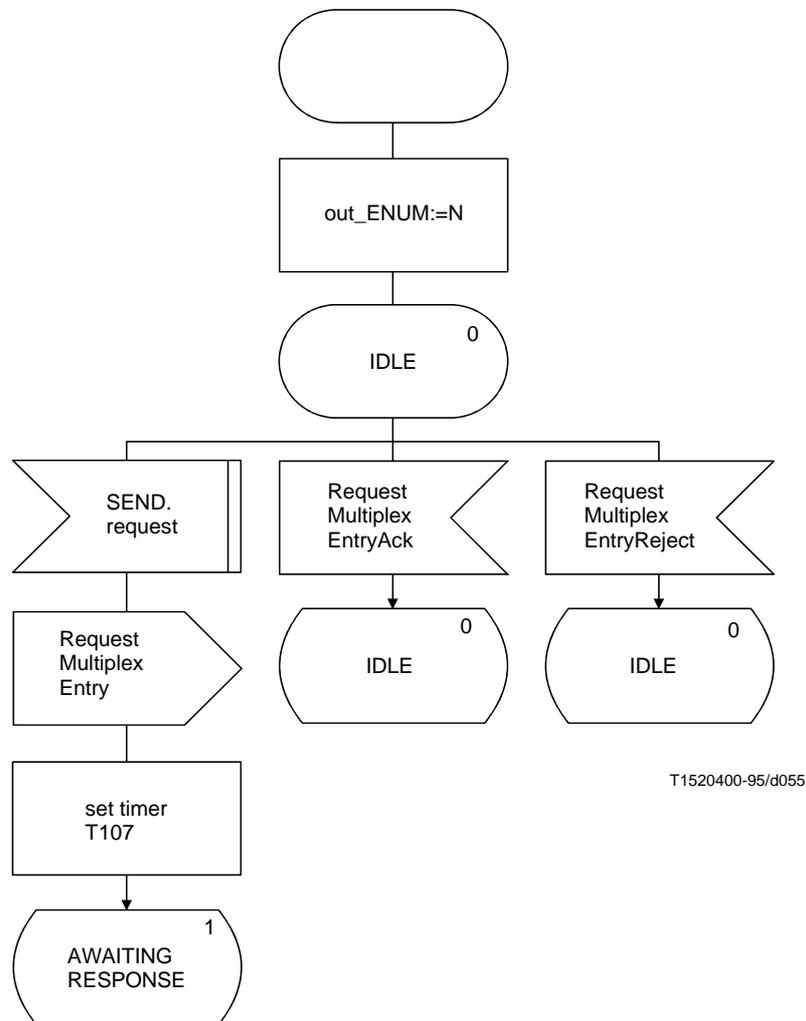


FIGURE 33 i)/H.245
Outgoing RMSE SDL

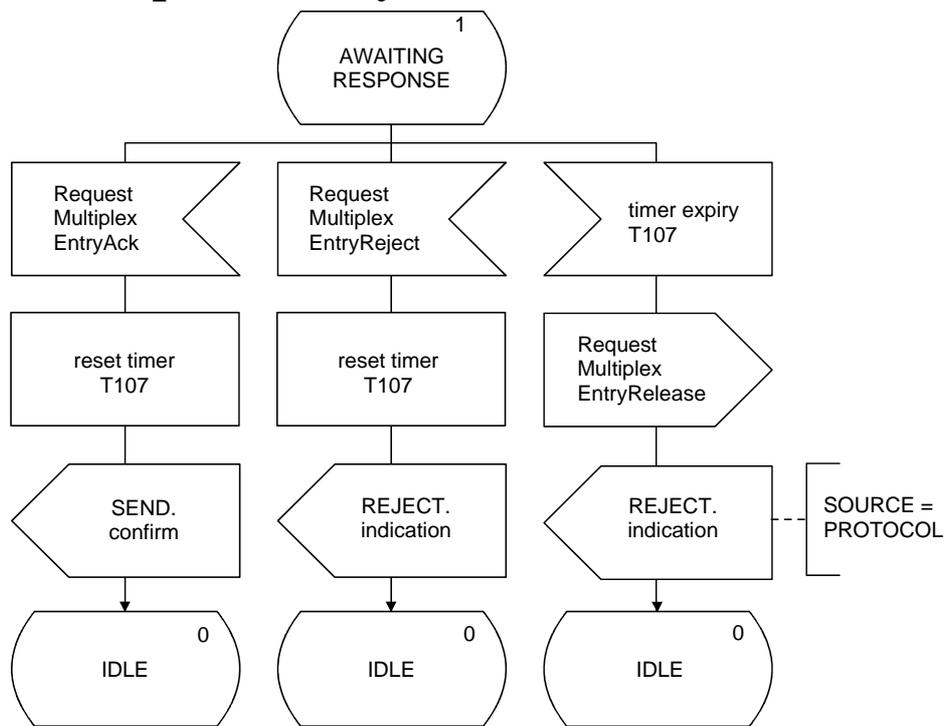
8.9 Mode Request procedures

8.9.1 Introduction

The procedures described here allow a terminal to request a remote terminal to use a particular mode of operation in its transmit direction. The procedures are referred to here as the Mode Request Signalling Entity (MRSE). Procedures are specified in terms of primitives and states at the interface between the MRSE and the MRSE user. Protocol information is transferred to the peer MRSE via relevant messages defined in clause 6. There is an outgoing MRSE and an incoming MRSE. At each of the outgoing and incoming ends there is one instance of the MRSE per call.

A terminal that answers such a response positively, that is, by issuing the TRANSFER.response primitive, shall initiate the logical channel signalling procedures to establish the appropriate mode of transmission as soon as possible.

Superseded by a more recent version



T1520410-95/d056

FIGURE 33 ii)/H.245
Outgoing RMESE SDL

If the currently valid capabilities received from the remote terminal contain one or more transmission capabilities, a terminal may select a mode that it prefers to have transmitted to it by performing the Mode Request procedures. A terminal whose currently valid capabilities contain one or more transmission capabilities and which is in receipt of such a request, should comply with the request.

A mode request shall not be sent to a terminal whose currently valid capabilities contain no transmission capabilities, that is, the terminal does not wish to, and shall not, be remotely controlled. If such a terminal does however receive a mode request, it may comply.

A terminal that receives multipointModeCommand shall comply with all received mode requests, until the command is cancelled by receipt of cancelMultipointModeCommand. A mode request may be sent to a terminal whose currently valid capabilities contain no transmission capabilities when multipointModeCommand has previously been sent.

The requested mode may include channels which are already open. For example, if a channel for G.723 was currently open and a terminal wished to receive an additional G.728 channel, it would send a mode request containing both the G.723 and the G.728 channel. If the G.723 channel request were absent, this would indicate that G.723 was no longer desired.

NOTE – The request mode description specifies a complete mode. If, for example, video is currently being transmitted and a mode request is received that does not include any specification for video, then this requests video transmission to stop.

Where one source is feeding several receivers, it may be unable to respond to any received signals such as requests to transmit in a particular mode.

The following text provides an overview of the operation of the MRSE protocol. In the case of any discrepancy between this and the formal specification, the formal specification will supersede.

Superseded by a more recent version

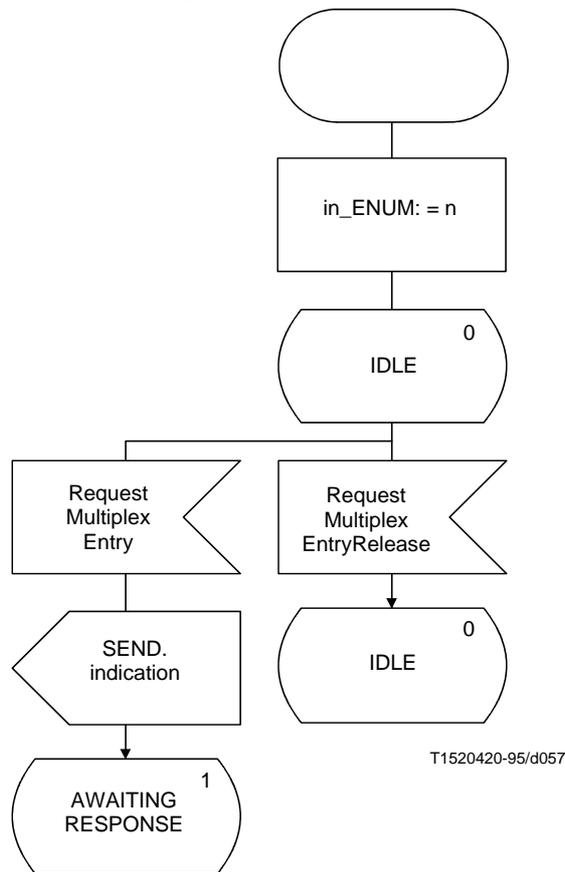


FIGURE 34 i)/H.245
Incoming RMESE SDL

8.9.1.1 Protocol overview – Outgoing MRSE

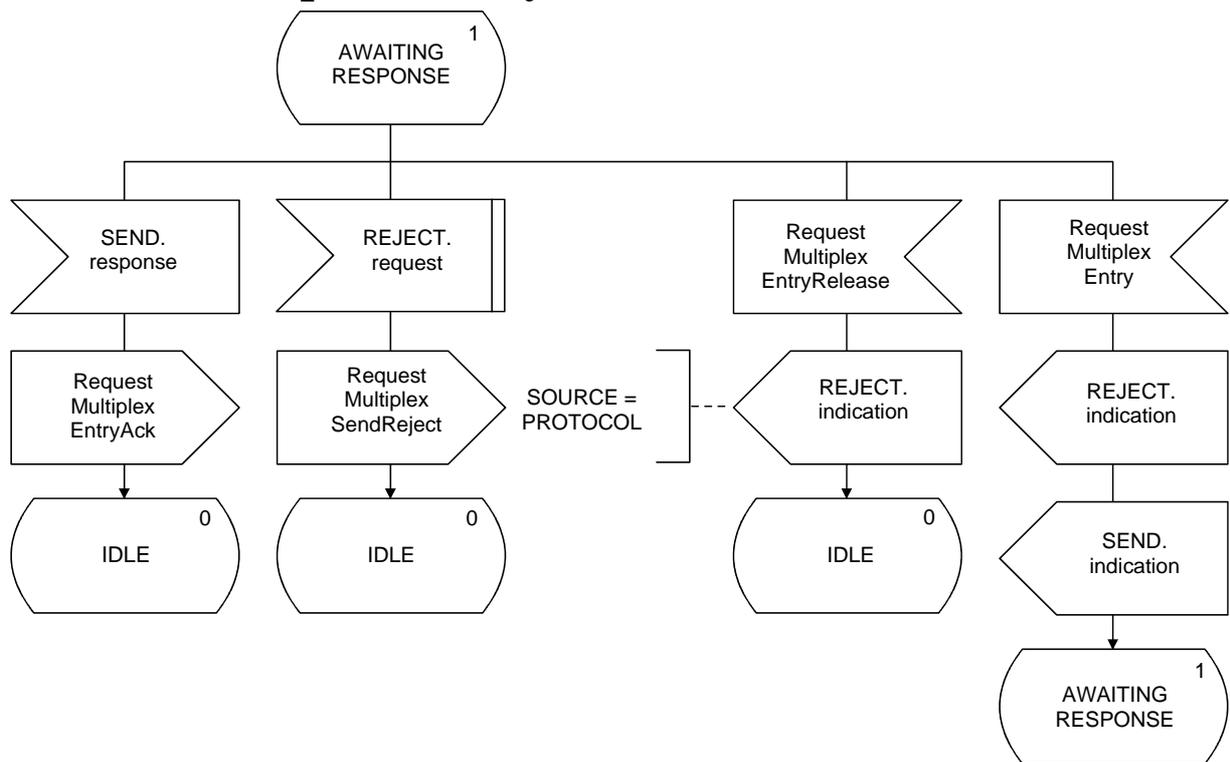
A mode request procedure is initiated when the TRANSFER.request primitive is issued by the user at the outgoing MRSE. A RequestMode message is sent to the peer incoming MRSE, and timer T109 is started. If a RequestModeAck message is received in response to the RequestMode message, then timer T109 is stopped and the user is informed with the TRANSFER.confirm primitive that the mode request was successful. If however a RequestModeReject message is received in response to the RequestMode message, then timer T109 is stopped and the user is informed with the REJECT.indication primitive that the peer MRSE user has refused to accept the mode request.

If timer T109 expires, then the outgoing MRSE user is informed with the REJECT.indication primitive and a RequestModeRelease message is sent.

Only RequestModeAck and RequestModeReject messages which are in response to the most recent RequestMode message are accepted. Messages in response to earlier RequestMode messages are ignored.

A new mode request procedure may be initiated with the TRANSFER.request primitive by the user at the outgoing MRSE before a RequestModeAck or a RequestModeReject message has been received.

Superseded by a more recent version



T1520430-95/d058

FIGURE 34 ii)/H.245
Incoming RMESE SDL

8.9.1.2 Protocol overview – Incoming MRSE

When a RequestMode message is received at the incoming MRSE, the user is informed of the mode request with the TRANSFER.indication primitive. The incoming MRSE user signals acceptance of the mode request by issuing the TRANSFER.response primitive, and a RequestModeAck message is sent to the peer outgoing MRSE. The incoming MRSE user signals rejection of the mode request by issuing the REJECT.request primitive, and a RequestModeReject message is sent to the peer outgoing MRSE.

A new RequestMode message may be received before the incoming MRSE user has responded to an earlier RequestMode message. The incoming MRSE user is informed with the REJECT.indication primitive, followed by the TRANSFER.indication primitive, and the incoming MRSE user responds to the new multiplex table entry.

If a RequestModeRelease message is received before the incoming MRSE user has responded to an earlier RequestMode message, then the incoming MRSE user is informed with the REJECT.indication, and the earlier mode request is discarded.

8.9.2 Communication between MRSE and MRSE user

8.9.2.1 Primitives between MRSE and MRSE user

Communication between the MRSE and MRSE user, is performed using the primitives shown in Table 49.

Superseded by a more recent version

TABLE 49/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
TRANSFER	MODE-ELEMENT	MODE-ELEMENT	MODE-PREF	MODE-PREF
REJECT	CAUSE	SOURCE CAUSE	Not defined (Note)	Not defined

NOTE – “Not defined” means that this primitive is not defined.

8.9.2.2 Primitive definition

The definition of these primitives is as follows:

- the TRANSFER primitives are used for the transfer of the mode request;
- the REJECT primitives are used to reject a mode request.

8.9.2.3 Parameter definition

The definition of the primitive parameters shown in Table 49 are as follows:

- The MODE-ELEMENT parameter specifies a mode element. This parameter is mapped to the requestedModes field of the RequestMode message and is carried transparently from the outgoing MRSE user to the incoming MRSE user. This parameter is mandatory. There may be multiple MODE-ELEMENTS associated with the TRANSFER primitives.
- The MODE-PREF parameter informs the user as to whether the most preferred mode requested will be used or not. This parameter is mapped to the response field of the RequestModeAck message and carried transparently from the incoming RMSE user to the outgoing RMSE user. It has two values being “MOST-PREFERRED” and “LESS-PREFERRED”.
- The SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of “USER” or “PROTOCOL”. The latter case may occur as the result of a timer expiry.
- The CAUSE parameter indicates the reason for refusal to close a logical channel. The CAUSE parameter is not present when the SOURCE parameter indicates “PROTOCOL”.

8.9.2.4 MRSE states

The following states are used to specify the allowed sequence of primitives between the MRSE and the MRSE user. The states for an outgoing MRSE are:

State 0: IDLE

The MRSE is idle.

State 1: AWAITING RESPONSE

The MRSE is waiting for a response from the remote MRSE.

The states for an incoming MRSE are:

State 0: IDLE

The MRSE is idle.

State 1: AWAITING RESPONSE

The MRSE is waiting for a response from the MRSE user.

Superseded by a more recent version

8.9.2.5 State transition diagram

The allowed sequence of primitives between the MRSE and the MRSE user is defined here. The allowed sequences are specified separately for each of an outgoing MRSE and an incoming MRSE, as shown in Figures 35 and 36 respectively.

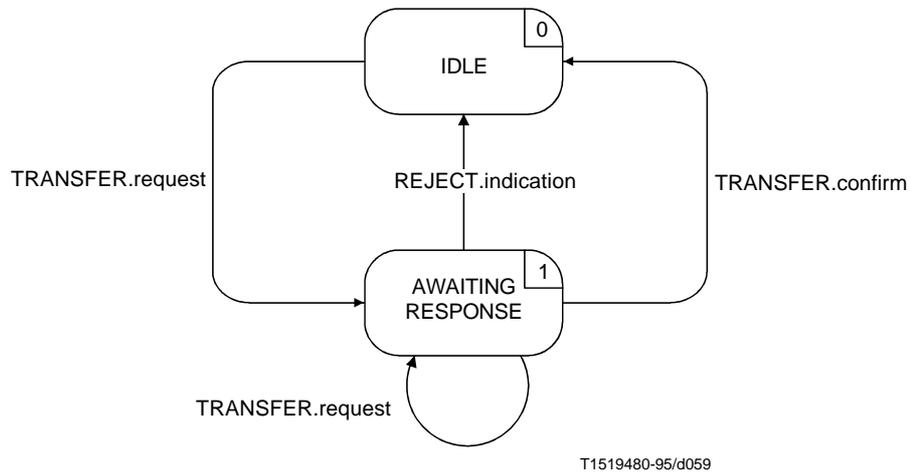


FIGURE 35/H.245

State transition diagram for sequence of primitives at MRSE outgoing

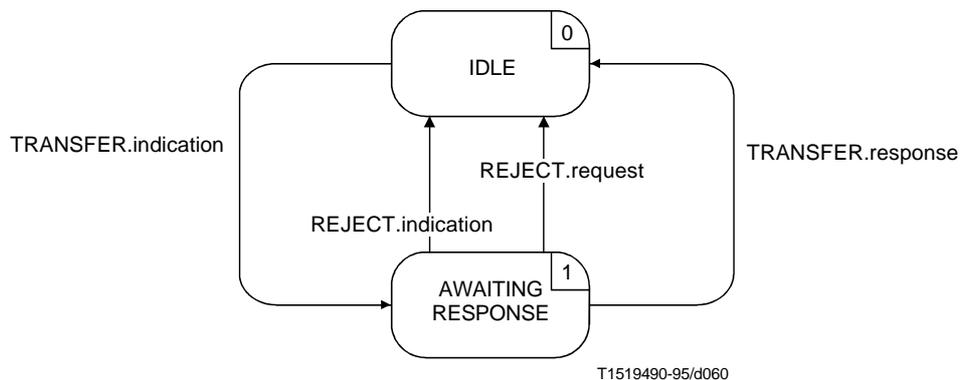


FIGURE 36/H.245

State transition diagram for sequence of primitives at MRSE incoming

Superseded by a more recent version

8.9.3 Peer-to-peer MRSE communication

8.9.3.1 Messages

Table 50 shows the MRSE messages and fields, defined in clause 6, which are relevant to the MRSE protocol.

TABLE 50/H.245

MRSE message names and fields

Function	Message	Direction	Field
Mode request	RequestMode	O → I	sequenceNumber requestedModes
	RequestModeAck	O ← I	sequenceNumber response
	RequestModeReject	O ← I	sequenceNumber cause
Reset	RequestModeRelease	O → I	–
O Outgoing I Incoming			

8.9.3.2 MRSE state variables

The following state variables are defined at the outgoing MRSE:

out_SQ

This state variable is used to indicate the most recent RequestMode message. It is incremented by one and mapped to the RequestMode message sequenceNumber field before transmission of the RequestMode message. Arithmetic performed on out_SQ is modulo 256.

The following state variables are defined at the incoming MRSE:

in_SQ

This state variable is used to store the value of the sequenceNumber field of the most recently received RequestMode message. The RequestModeAck and RequestModeReject messages have their sequenceNumber fields set to the value of in_SQ, before being sent to the peer MRSE.

8.9.3.3 MRSE timers

The following timer is specified for the outgoing MRSE:

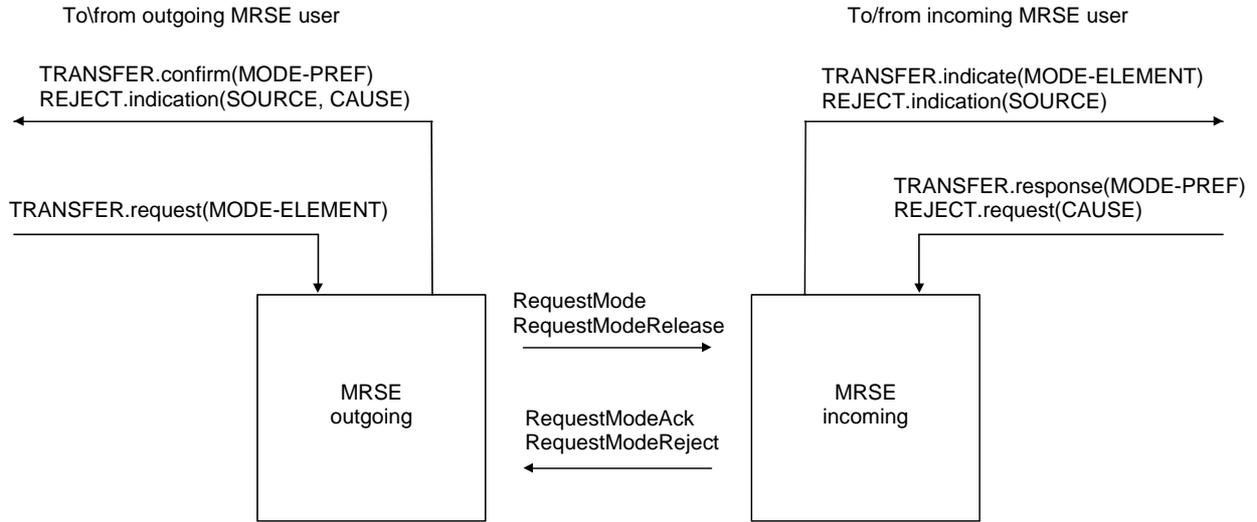
T109

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no RequestModeAck or RequestModeRejectMessage may be received.

8.9.4 MRSE procedures

Figure 37 summarizes the MRSE primitives and their parameters, and messages, for each of the outgoing and incoming MRSE.

Superseded by a more recent version



T1519500-95/d061

FIGURE 37/H.245

Primitives and messages in the Mode Request Signalling Entity

8.9.4.1 Primitive parameter default values

Where not explicitly stated in the SDL diagrams, the parameters of the indication and confirm primitives assume values as shown in Table 51.

TABLE 51/H.245

Default primitive parameter values

Primitive	Parameter	Default value
TRANSFER.indication	MODE-ELEMENT	RequestMode.requestedModes
TRANSFER.confirm	MODE-PREF	RequestModeAck.response
REJECT.indication	SOURCE CAUSE	USER null

8.9.4.2 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 52.

TABLE 52/H.245

Default message field values

Message	Field	Default value
RequestMode	sequenceNumber	out_SQ
	requestedModes	TRANSFER.request(MODE-ELEMENT)
RequestModeAck	sequenceNumber	in_SQ
	response	TRANSFER.response(MODE-PREF)
RequestModeReject	sequenceNumber cause	in_SQ REJECT.request(CAUSE)
RequestModeRelease	–	–

Superseded by a more recent version

8.9.4.3 SDLs

The outgoing MRSE and the incoming MRSE procedures are expressed in SDL form in Figures 38 and 39 respectively.

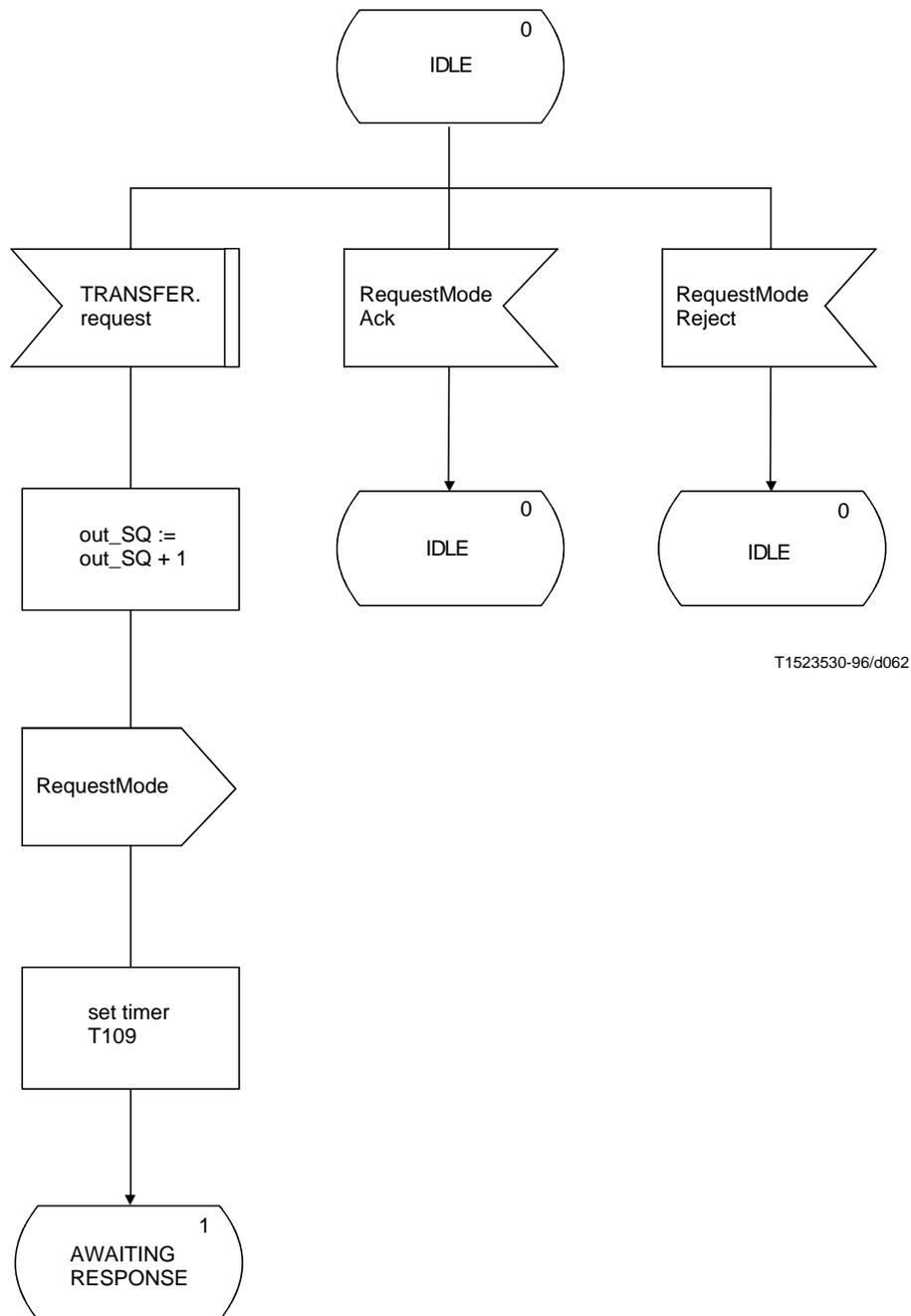


FIGURE 38 i)/H.245
Outgoing MRSE SDL

Superseded by a more recent version

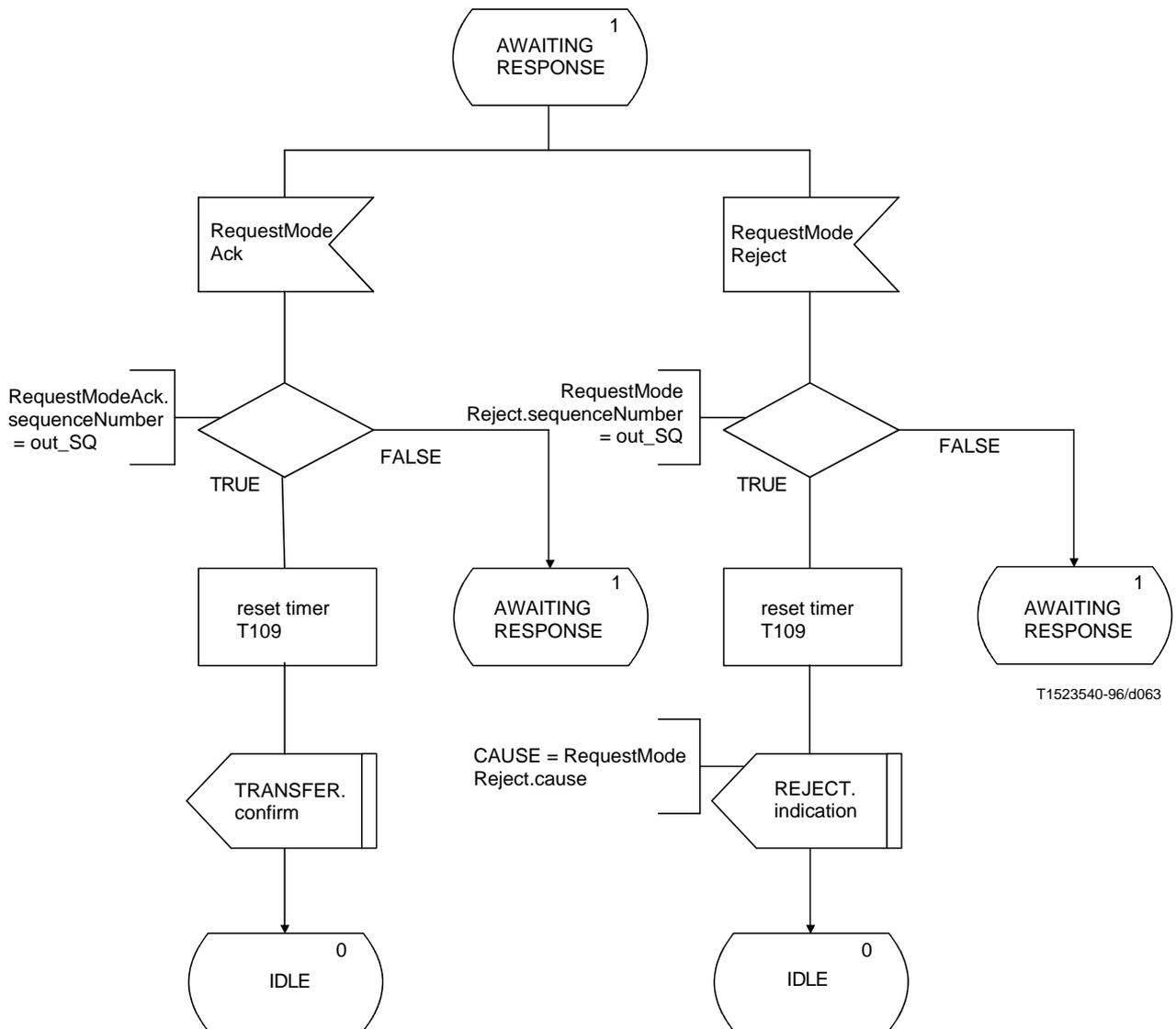
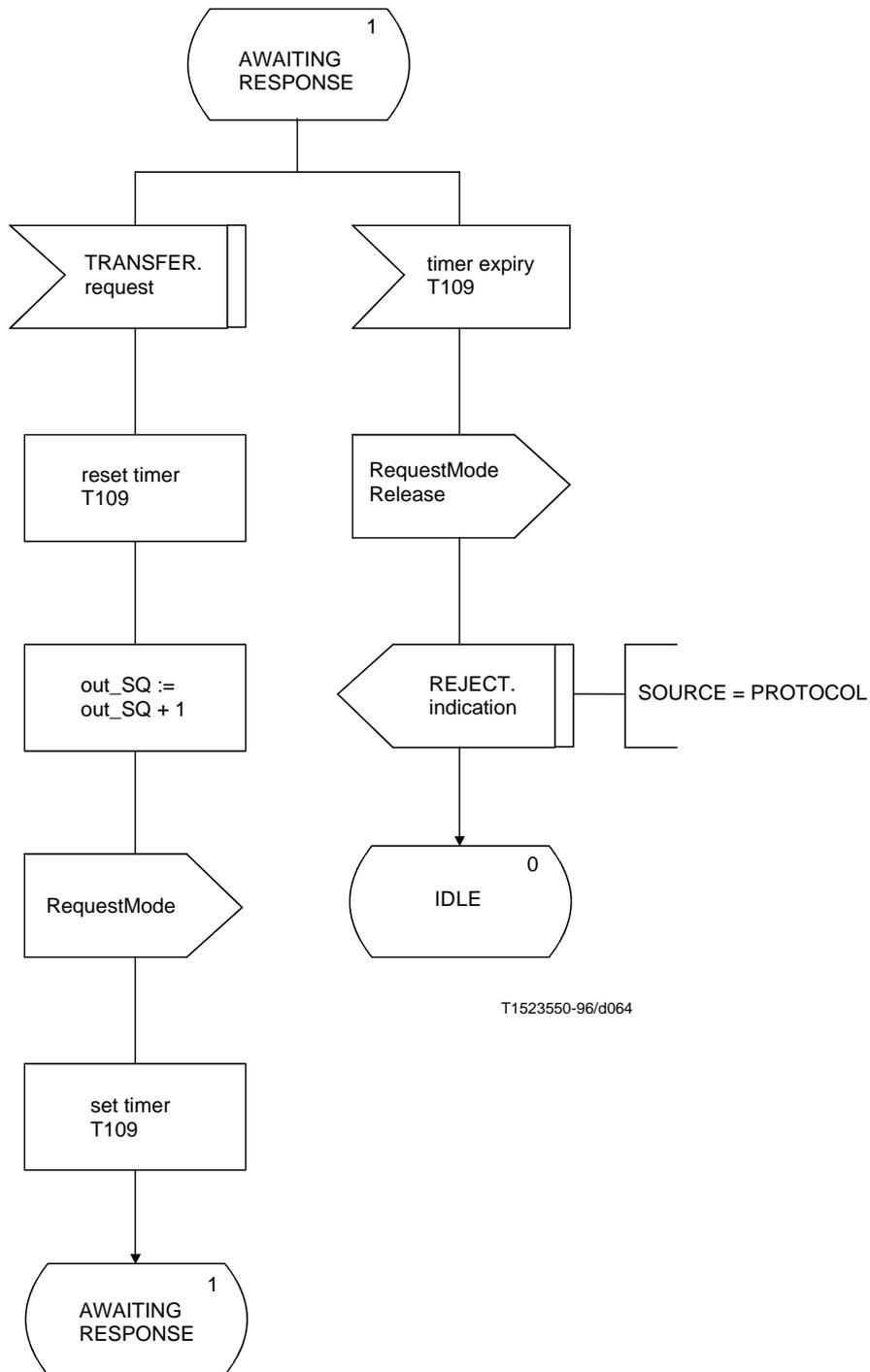


FIGURE 38 ii)/H.245
Outgoing MRSE SDL

Superseded by a more recent version



T1523550-96/d064

FIGURE 38 iii)/H.245
Outgoing MRSE SDL

Superseded by a more recent version

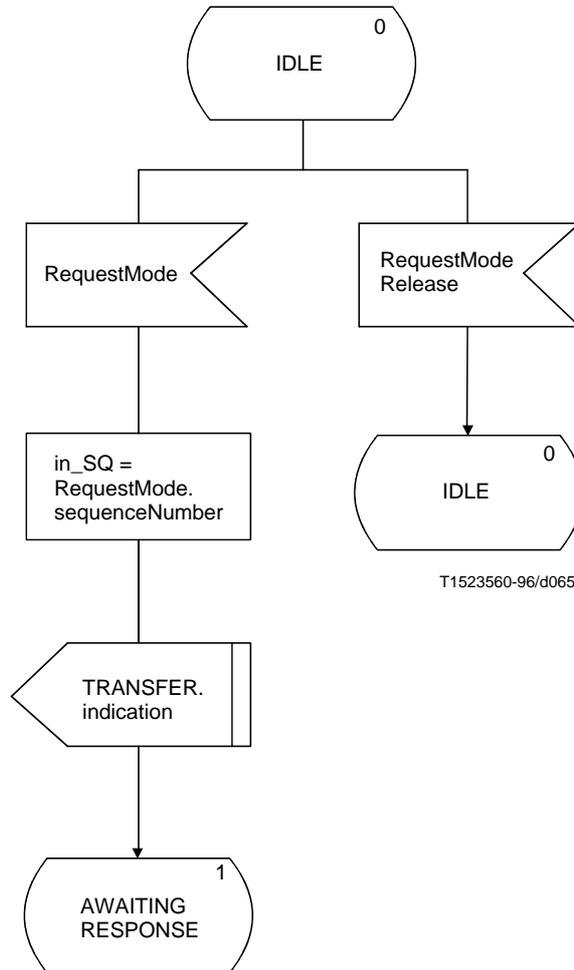
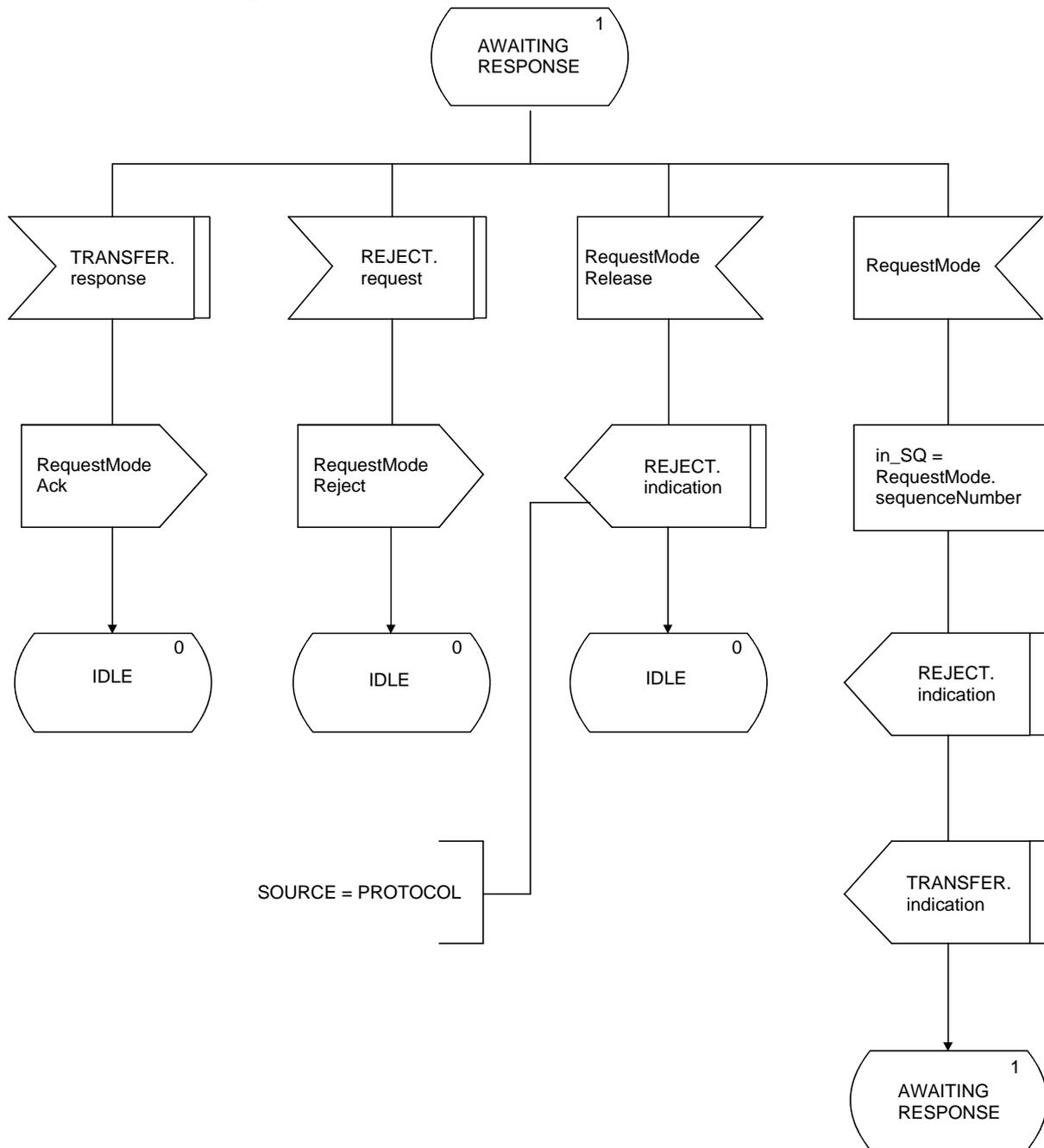


FIGURE 39 i)/H.245
Incoming MRSE SDL

Superseded by a more recent version



T1523570-96/D066

FIGURE 39 ii)/H.245
Incoming MRSE SDL

8.10 Round trip delay procedures

8.10.1 Introduction

Procedures are described here that allow the determination of the round trip delay between two communicating terminals. This function also enables a H.245 user to determine if the peer H.245 protocol entity is still alive.

Superseded by a more recent version

The function described here is referred to as the Round Trip Delay Signalling Entity (RTDSE). Procedures are specified in terms of primitives and states at the interface between the RTDSE and the RTDSE user. There is one instance of the RTDSE in each terminal. Any terminal may perform the round trip delay determination.

The following text provides an overview of the operation of the RTDSE protocol. In the case of any discrepancy between this and the formal specification, the formal specification will supersede.

8.10.1.1 Protocol overview – RTDSE

A round trip determination delay procedure is initiated when the TRANSFER.request primitive is issued by the RTDSE user. A RoundTripDelayRequest message is sent to the peer RTDSE, and timer T105 is started. If a RoundTripDelayResponse message is received in response to the RoundTripDelayRequest message, then timer T105 is stopped and the user is informed with the TRANSFER.confirm primitive of the round trip delay, which is the value of timer T105.

If a RoundTripDelayRequest message is at any time received from the peer RTDSE, a RoundTripDelayResponse message is immediately sent to the peer RTDSE.

If timer T105 expires, then the RTDSE user is informed with the EXPIRY.indication primitive

Only the RoundTripDelayResponse message which is in response to the most recent RoundTripDelayRequest message is accepted. Messages in response to earlier RoundTripDelayRequest messages are ignored.

A new round trip determination delay procedure may be initiated with the TRANSFER.request primitive by the RTDSE user before a RoundTripDelayResponse message has been received.

8.10.2 Communication between the RTDSE and the RTDSE user

8.10.2.1 Primitives between the RTDSE and the RTDSE user

Communication between the RTDSE, and RTDSE user, is performed using the primitives shown in Table 53. These primitives are for the purpose of defining RTDSE procedures and are not meant to specify or constrain implementation.

TABLE 53/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
TRANSFER	– (Note 1)	Not defined (Note 2)	Not defined	DELAY
EXPIRY	Not defined	–	Not defined	Not defined

NOTE 1 – “–” means no parameters.
 NOTE 2 – “Not defined” means that this primitive is not defined.

8.10.2.2 Primitive definition

The definition of these primitives is as follows:

- a) the TRANSFER primitive is used to request, and report upon, the round trip delay determination;
- b) The EXPIRY primitive indicates that no response has been received from the peer terminal.

8.10.2.3 Parameter definition

The definition of the primitive parameters shown in Table 53 are as follows:

- The DELAY parameter returns the measured round trip delay.

Superseded by a more recent version

8.10.2.4 RTDSE states

The following states are used to specify the allowed sequence of primitives between the RTDSE and the RTDSE user:

State 0: IDLE

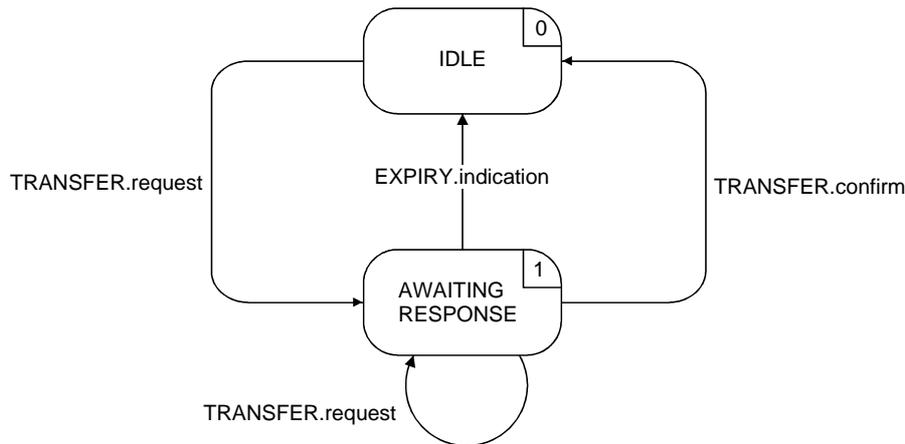
There is no RTDSE transfer in progress.

State 1: AWAITING RESPONSE

The RTDSE user has requested the measurement of the round trip delay. A response from the peer RTDSE is awaited.

8.10.2.5 State transition diagram

The allowed sequence of primitives between the RTDSE and the RTDSE user is defined here. The allowed sequences are shown in Figure 40.



T1519510-95/d067

FIGURE 40/H.245

State transition diagram for sequence of primitives at RTDSE

8.10.3 Peer-to-peer RTDSE communication

8.10.3.1 Messages

Table 54 shows the RTDSE messages and fields, defined in clause 6, which are relevant to the RTDSE protocol.

TABLE 54/H.245

RTDSE message names and fields

Function	Message	Field
Transfer	RoundTripDelayRequest	sequenceNumber
	RoundTripDelayResponse	sequenceNumber

8.10.3.2 RTDSE state variables

The following RTDSE state variables are defined:

out_SQ

This state variable is used to indicate the most recent RoundTripDelayRequest message. It is incremented by one and mapped to the RoundTripDelayRequest message sequenceNumber field before transmission of a RoundTripDelayRequest message. Arithmetic performed on out_SQ is modulo 256.

Superseded by a more recent version

8.10.3.3 RTDSE timers

The following timer is specified for the outgoing RTDSE:

T105

This timer is used during the Awaiting Response state. It specifies the maximum time during which no RoundTripDelayResponse message may be received.

8.10.4 RTDSE procedures

8.10.4.1 Introduction

Figure 41 summarizes the RTDSE primitives and their parameters, and messages.

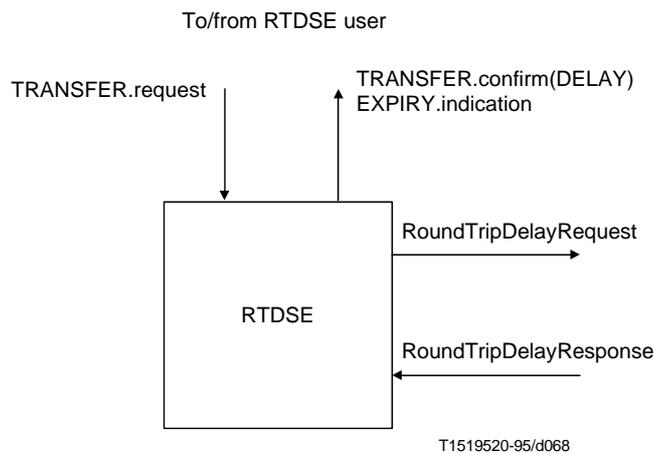


FIGURE 41/H.245

Primitives and messages in the RTDSE

8.10.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams, the parameters of the indication and confirm primitives assume values as shown in Table 55.

TABLE 55/H.245

Default primitive parameter values

Primitive	Parameter	Default value
TRANSFER.confirm	DELAY	Initial value of timer T105 minus value of timer T105
EXPIRY.indication	–	–

NOTE – Timers are defined to count down to zero. The DELAY parameter indicates the time that the timer has been running, and so has the value of the difference between the initial setting and the retained value of the timer.

8.10.4.3 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 56.

Superseded by a more recent version

TABLE 56/H.245

Default message field values

Message	Field	Default value
RoundTripDelayRequest	sequenceNumber	out_SQ
RoundTripDelayResponse	sequenceNumber	RoundTripDelayRequest.sequenceNumber

8.10.4.4 SDLs

The RTDSE procedures are expressed in SDL form in Figure 42.

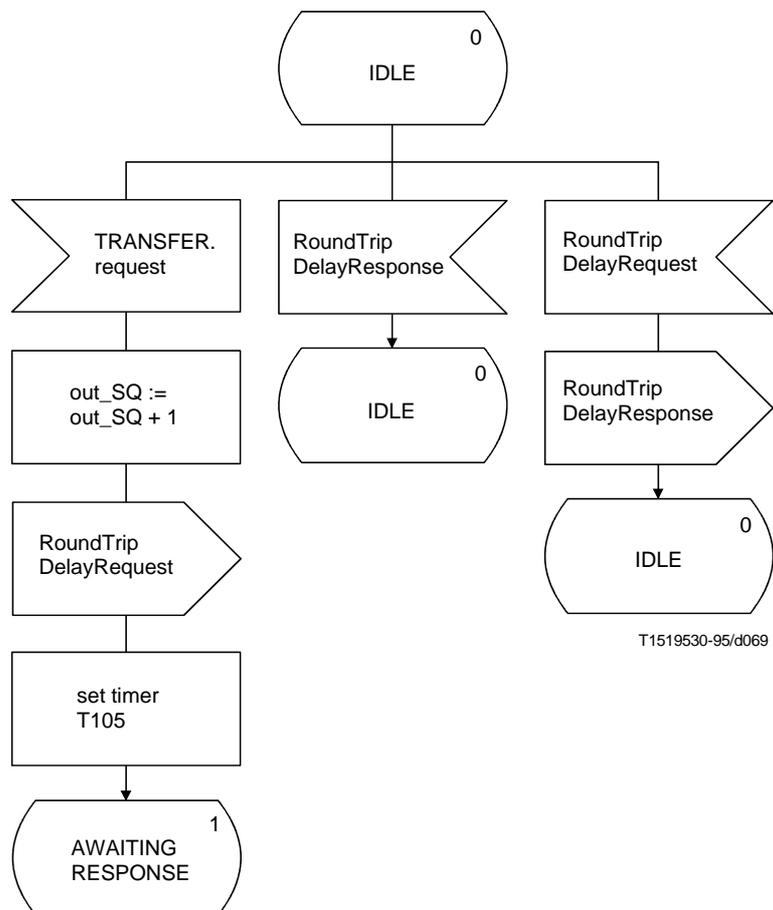


FIGURE 42 i)/H.245

RTDSE SDL

Superseded by a more recent version

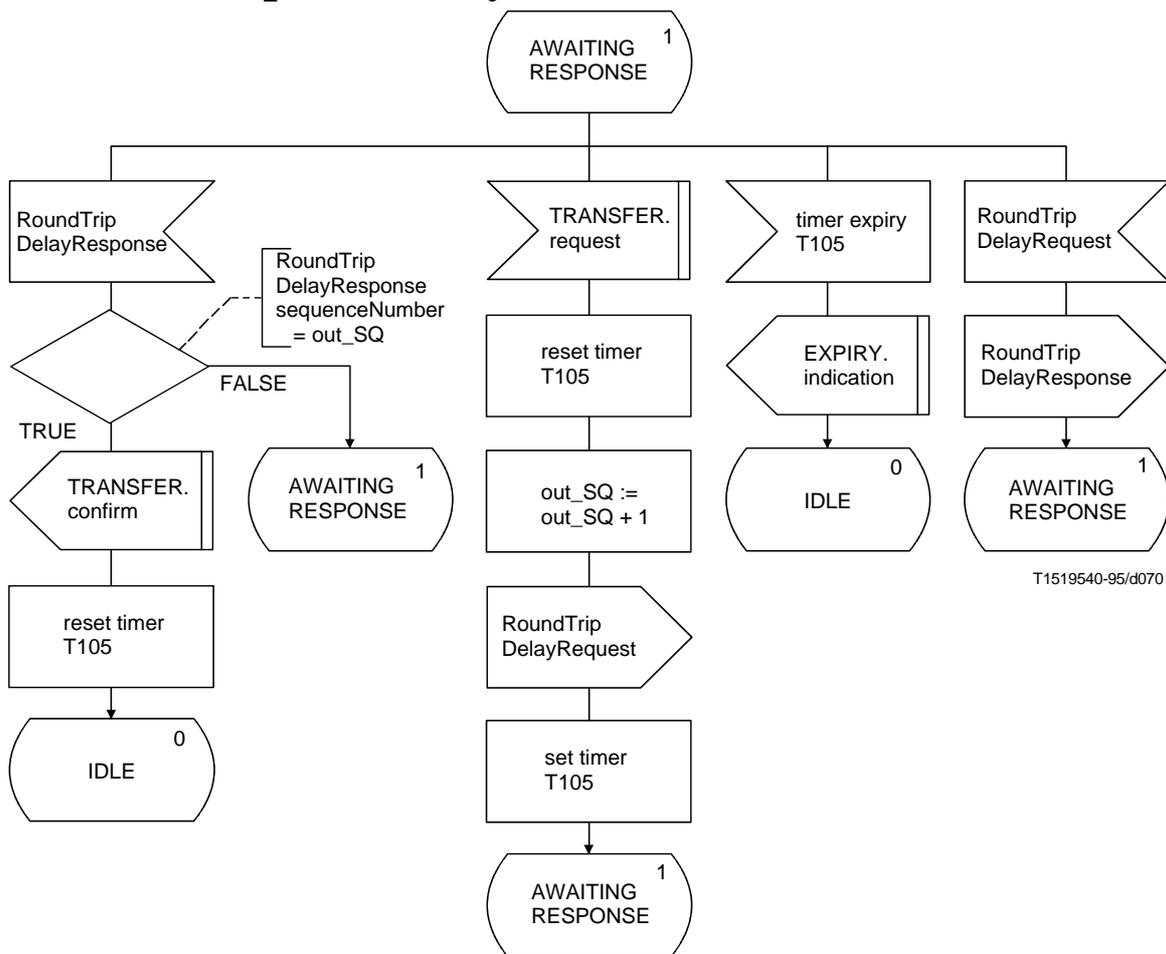


FIGURE 42 ii)/H.245
RTDSE SDL

8.11 Maintenance Loop procedures

8.11.1 Introduction

The protocol specified here provides reliable operation of maintenance loops using acknowledged procedures.

The protocol specified here is referred to as the Maintenance Loop Signalling Entity (MLSE). Procedures are specified in terms of primitives at the interface between the MLSE and the MLSE user, and MLSE states. Protocol information is transferred to the peer MLSE via relevant messages defined in clause 6.

There is an outgoing MLSE and an incoming MLSE. At each of the outgoing and incoming sides there is one instance of the MLSE for each bidirectional logical channel and one for the system loop. There is no connection between an incoming MLSE and an outgoing MLSE at one side, other than via primitives to and from the MLSE user. MLSE error conditions are reported.

The terminal that contains the incoming MLSE shall loop the appropriate data while it is in the LOOPED state, and not at any other time. The terminal that contains the outgoing MLSE shall be capable of receiving looped data while in any state, but while in the LOOPED state, should receive looped data only.

NOTE – The MaintenanceLoopOffCommand message applies to all MLSEs. It is always used to stop all maintenance loops.

Superseded by a more recent version

The following text provides an overview of the operation of the MLSE protocol. In the case of discrepancy between this and the formal specification, the formal specification will supersede.

8.11.1.1 Protocol overview

The establishment of a maintenance loop is initiated when the LOOP.request primitive is issued by the user at the outgoing MLSE. A MaintenanceLoopRequest message is sent to the peer incoming MLSE, and timer T102 is started. If a MaintenanceLoopAck message is received in response to the MaintenanceLoopRequest message, then timer T102 is stopped and the user is informed with the LOOP.confirm primitive that the maintenance loop has been successfully established. If however, a MaintenanceLoopReject message is received in response to the MaintenanceLoopRequest message, then timer T102 is stopped and the user is informed with the RELEASE.indication primitive that the peer MLSE user has refused establishment of the maintenance loop.

If timer T102 expires in this period, then the user is informed with the RELEASE.indication primitive and a MaintenanceLoopOffCommand message is sent to the peer incoming MLSE. This will cancel all maintenance loops, and not just the one concerned with the particular MLSE.

A maintenance loop that has been successfully established may be cancelled when the RELEASE.request primitive is issued by the user at the outgoing MLSE. A MaintenanceLoopOffCommand message is sent to the peer incoming MLSE.

Before either of the MaintenanceLoopAck or MaintenanceLoopReject messages have been received in response to a previously sent MaintenanceLoopRequest message, the user at the outgoing MLSE may cancel the maintenance loop using the RELEASE.request primitive.

8.11.1.2 Protocol overview – Incoming

When a MaintenanceLoopRequest message is received at the incoming MLSE, the user is informed of the request to establish a maintenance loop with the LOOP.indication primitive. The incoming MLSE user signals acceptance of the request to establish the maintenance loop by issuing the LOOP.response primitive and a MaintenanceLoopAck message is sent to the peer outgoing MLSE. The maintenance loop shall now be performed. The incoming MLSE user signals rejection of the request to establish the maintenance loop by issuing the RELEASE.request primitive and a MaintenanceLoopReject message is sent to the peer outgoing MLSE.

A maintenance loop that has been successfully established may be cancelled when the MaintenanceLoopOffCommand message is received at the incoming MLSE. The incoming MLSE user is informed with the RELEASE.indication primitive.

8.11.2 Communication between the MLSE and the MLSE user

8.11.2.1 Primitives between the MLSE and the MLSE user

Communication between the MLSE and the MLSE user is performed using the primitives shown in Table 57.

TABLE 57/H.245

Primitives and parameters

Generic name	Type			
	Request	Indication	Response	Confirm
LOOP	LOOP_TYPE	LOOP_TYPE	– (Note 1)	–
RELEASE	CAUSE	SOURCE CAUSE	Not defined (Note 2)	Not defined
ERROR	Not defined	ERRCODE	Not defined	Not defined
NOTE 1 – “–” means no parameters.				
NOTE 2 – “Not defined” means that this primitive does not exist.				

Superseded by a more recent version

8.11.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The LOOP primitives are used to establish a maintenance loop.
- b) The RELEASE primitives are used to cancel a maintenance loop.
- c) The ERROR primitive reports MLSE errors to a management entity.

8.11.2.3 Parameter definition

The definition of the primitive parameters shown in Table 57 are as follows:

- a) The LOOP_TYPE parameter specifies the parameters associated with the maintenance loop. It has values of "SYSTEM", "MEDIA" and "LOGICAL CHANNEL". This parameter, and the logical channel number, determine the value of the type field of the MaintenanceLoopRequest message which is then carried transparently to the peer MLSE user.
- b) The SOURCE parameter indicates to the MLSE user the source of the maintenance loop release. The SOURCE parameter has the value of "USER" or "MLSE", indicating either the MLSE user, or the MLSE. The latter may occur as the result of a protocol error.
- c) The CAUSE parameter indicates the reason as to why the peer MLSE user rejected a request to establish a maintenance loop. The CAUSE parameter is not present when the SOURCE parameter indicates "MLSE".
- d) The ERRCODE parameter indicates the type of MLSE error. Table 61 shows the allowed values of the ERRCODE parameter.

8.11.2.4 MLSE states

The following states are used to specify the allowed sequence of primitives between the MLSE and the MLSE user, and the exchange of messages between peer MLSEs. The states are specified separately for each of an outgoing MLSE and an incoming MLSE. The states for an outgoing MLSE are:

State 0: NOT LOOPED

There is no maintenance loop.

State 1: AWAITING RESPONSE

The outgoing MLSE is waiting to establish a maintenance loop with a peer incoming MLSE.

State 2: LOOPED

The MLSE peer-to-peer maintenance loop has been established. All data received on the appropriate channel should be looped data.

The states for an incoming MLSE are:

State 0: NOT LOOPED

There is no maintenance loop.

State 1: AWAITING RESPONSE

The incoming MLSE is waiting to establish a maintenance loop with a peer outgoing MLSE. The appropriate data shall not be looped.

State 2: LOOPED

An MLSE peer-to-peer maintenance loop has been established. All data received on the appropriate channel shall be looped.

8.11.2.5 State transition diagram

The allowed sequence of primitives between the MLSE and the MLSE user is defined here. The allowed sequence of primitives relates to states of the MLSE as viewed from the MLSE user. The allowed sequences are specified separately for each of an outgoing MLSE and an incoming MLSE, as shown in Figures 43 and 44 respectively.

Superseded by a more recent version

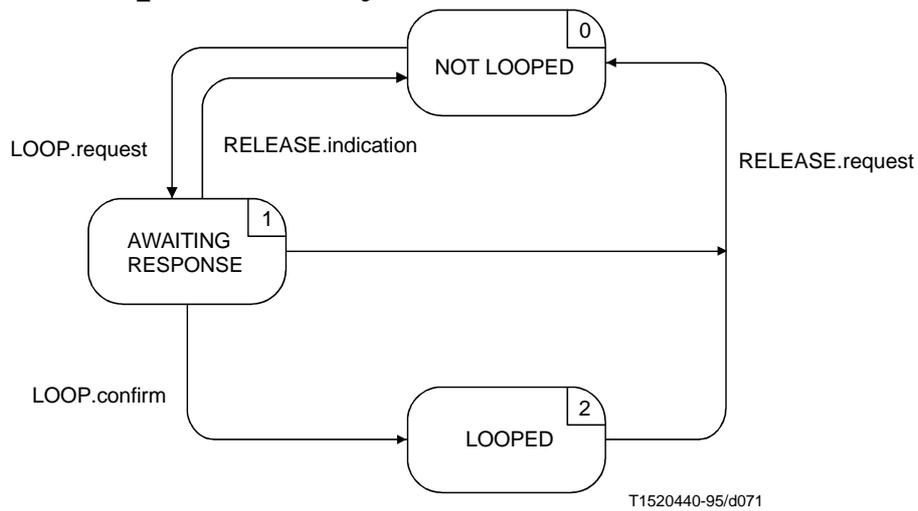


FIGURE 43/H.245

State transition diagram for sequence of primitives at outgoing MLSE

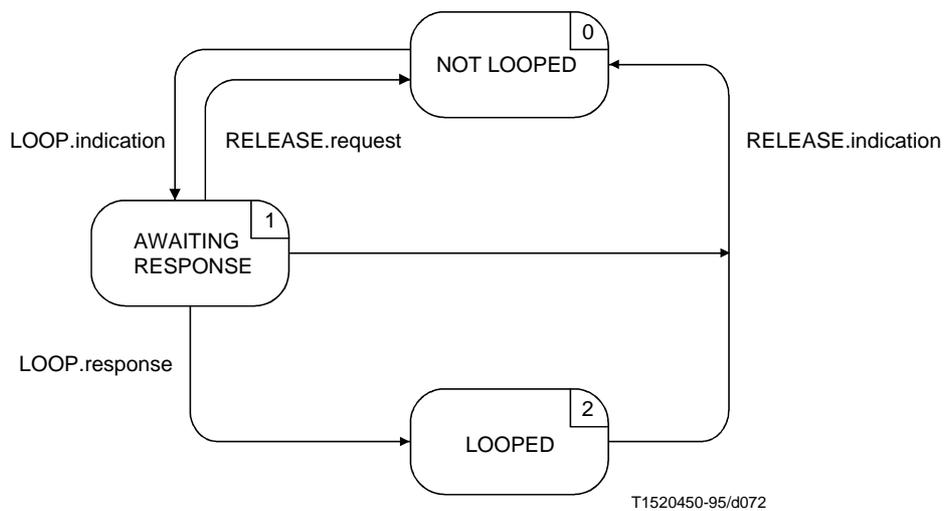


FIGURE 44/H.245

State transition diagram for sequence of primitives at incoming MLSE

8.11.3 Peer-to-peer MLSE communication

8.11.3.1 MLSE messages

Table 58 shows the MLSE messages and fields defined in clause 6 which are relevant to the MLSE protocol.

Superseded by a more recent version

TABLE 58/H.245

MLSE message names and fields

Function	Message	Direction	Field
Establish	MaintenanceLoopRequest	O → I	type
	MaintenanceLoopAck	O ← I	type
	MaintenanceLoopReject	O ← I	type cause
Release	MaintenanceLoopOffCommand	O → I	–
O	Outgoing		
I	Incoming		

8.11.3.2 MLSE state variables

The following state variable is defined at the outgoing MLSE:

out_MLN

This state variable distinguishes between outgoing MLSEs. It is initialized at outgoing MLSE initialization. The value of out_MLN is used to set the type field of MaintenanceLoopRequest messages sent from an outgoing MLSE.

The following state variable is defined at the incoming MLSE:

in_MLN

This state variable distinguishes between incoming MLSEs. It is initialized at incoming MLSE initialization. For MaintenanceLoopRequest messages received at an incoming MLSE, the message type field value is consistent with the value of in_MLN.

In_TYPE

This state variable stores the value of LOOP_TYPE when the MaintenanceLoopRequest is received. This state variable assists in setting the value of the type field in the MaintenanceLoopAck message.

8.11.3.3 MLSE timers

The following timer is specified for the outgoing MLSE:

T102

This timer is used during the AWAITING RESPONSE state. It specifies the maximum allowed time during which no MaintenanceLoopAck or MaintenanceLoopReject message may be received.

8.11.4 MLSE procedures

8.11.4.1 Introduction

Figure 45 summarizes the primitives and their parameters, and the messages, for each of the outgoing and incoming MLSEs.

8.11.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table 59.

8.11.4.3 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table 60.

Superseded by a more recent version

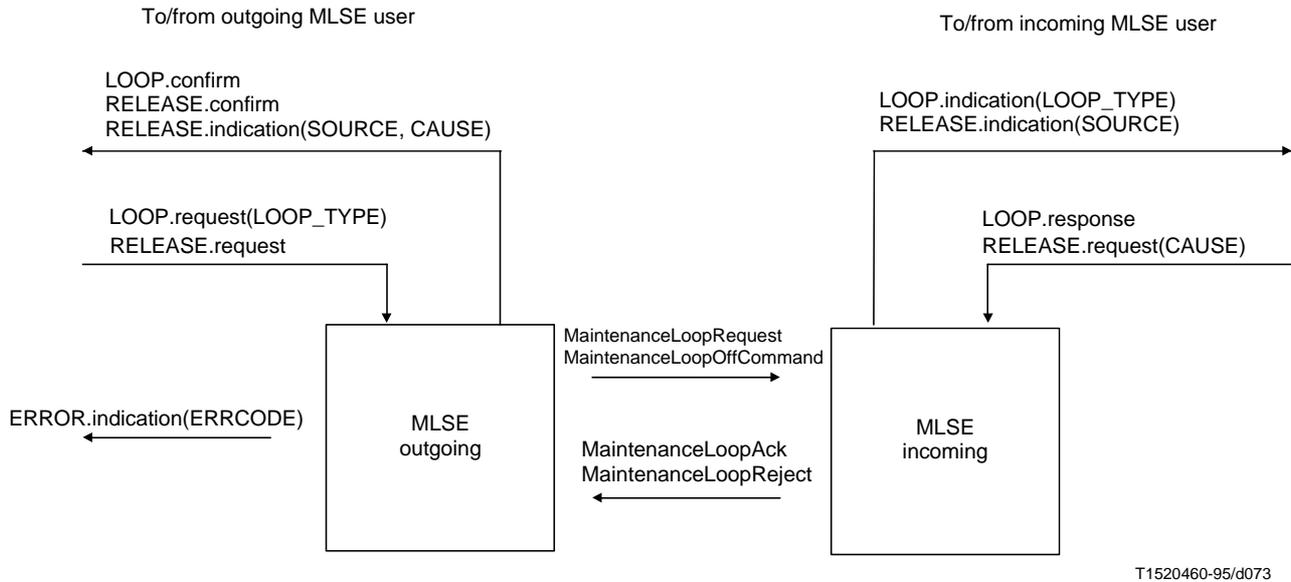


FIGURE 45/H.245

Primitives and messages in the Maintenance Loop Signalling Entity

TABLE 59/H.245

Default primitive parameter values

Primitive	Parameter	Default value (Note)
LOOP.indication	LOOP_TYPE	MaintenanceLoopRequest.type
RELEASE.indication	SOURCE CAUSE	USER MaintenanceLoopReject.cause

NOTE – A primitive parameter shall be coded as null, if an indicated message field is not present in the message.

TABLE 60/H.245

Default message field values

Message	Field	Default value (Note 1)
MaintenanceLoopRequest	type	LOOP.request(LOOP_TYPE) and out_MLN (Note 2)
MaintenanceLoopAck	type	in_LOOP and in_MLN (Note 3)
MaintenanceLoopReject	type cause	in_LOOP and in_MLN (Note 3) RELEASE.request(CAUSE)
MaintenanceLoopOffCommand	–	–

NOTE 1 – A message field shall not be coded, if the corresponding primitive parameter is null i.e. not present.
 NOTE 2 – The value of the type field is derived from the LOOP_TYPE parameter and the logical channel number.
 NOTE 3 – The value of the type field is derived from the in_LOOP and in_MLN state variables.

Superseded by a more recent version

8.11.4.4 ERRCODE parameter values

The ERRCODE parameter of the ERROR.indication primitive indicates a particular error condition. Table 61 shows the values that the ERRCODE parameter may take at the outgoing MLSE. There is no ERROR.indication primitive associated with the incoming MLSE.

TABLE 61/H.245

ERRCODE parameter values at outgoing MLSE

Error type	Error code	Error condition	State
Inappropriate message	A	MaintenanceLoopAck	LOOPEd
No response from peer MLSE	B	timer T102 expiry	AWAITING RESPONSE

8.11.4.5 SDLs

The outgoing MLSE and the incoming MLSE procedures are expressed in SDL form in Figures 46 and 47 respectively.

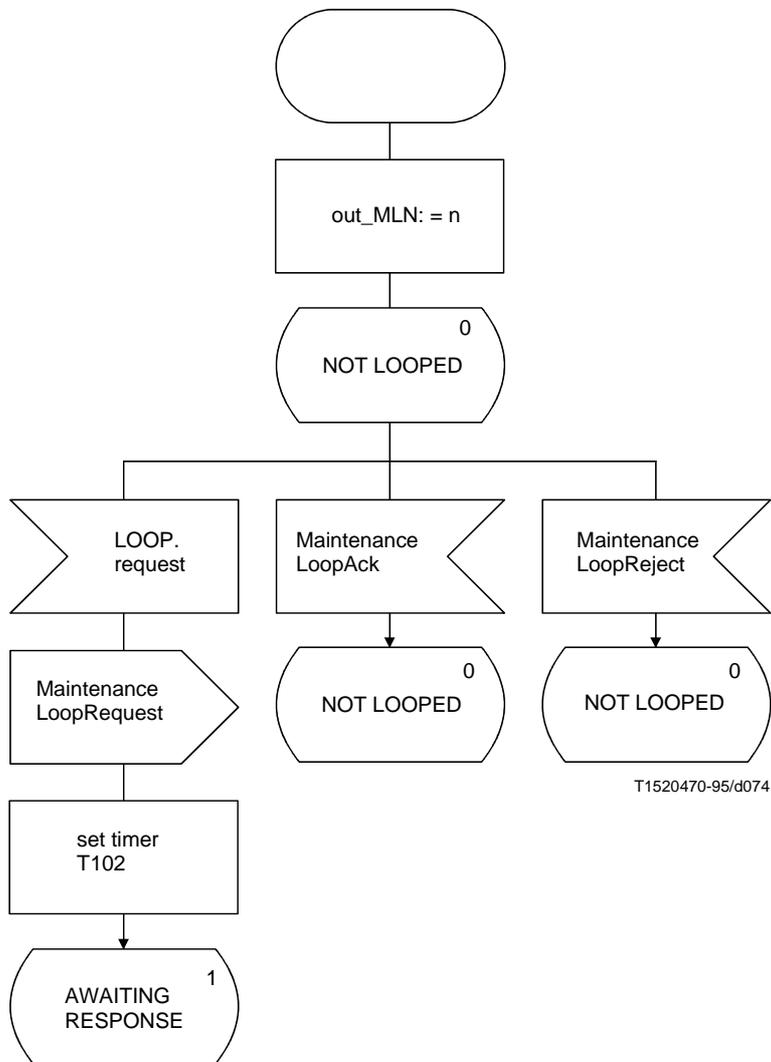
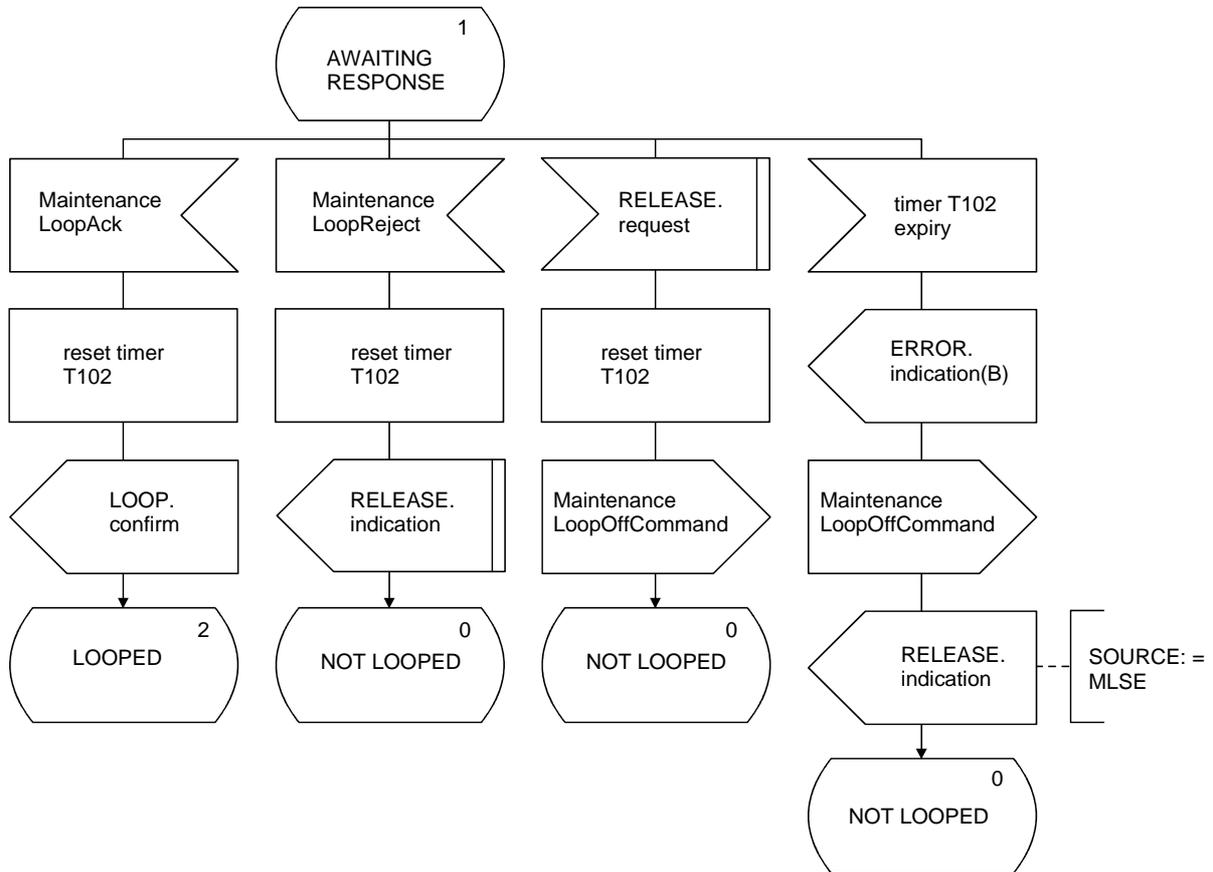


FIGURE 46 i)/H.245

Outgoing MLSE SDL

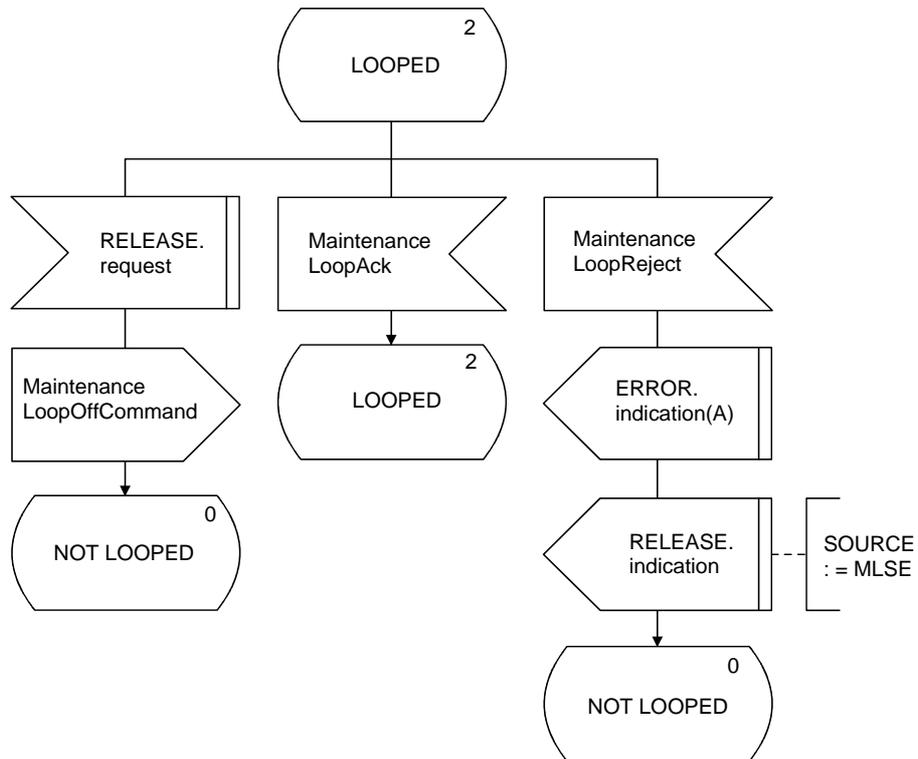
Superseded by a more recent version



T1520480-95/d075

FIGURE 46 ii)/H.245
Outgoing MLSE SDL

Superseded by a more recent version



T1520490-95/d076

FIGURE 46 iii)/H.245
Outgoing MLSE SDL

Superseded by a more recent version

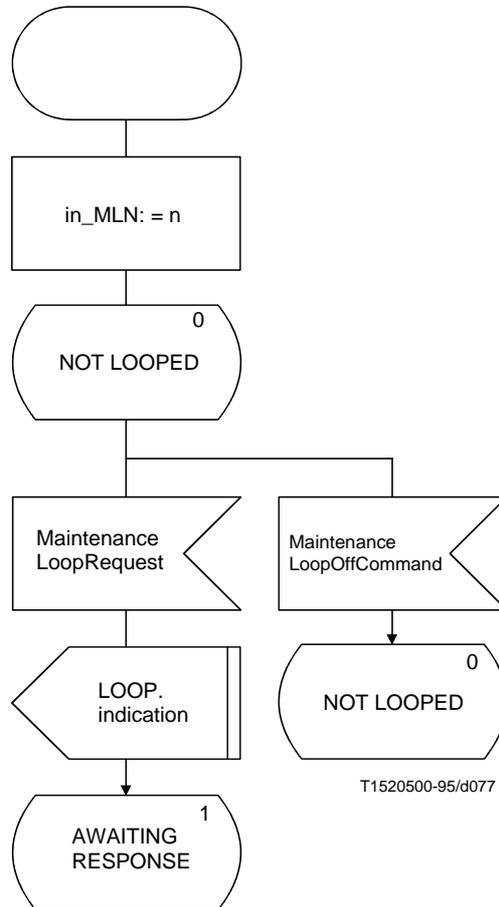
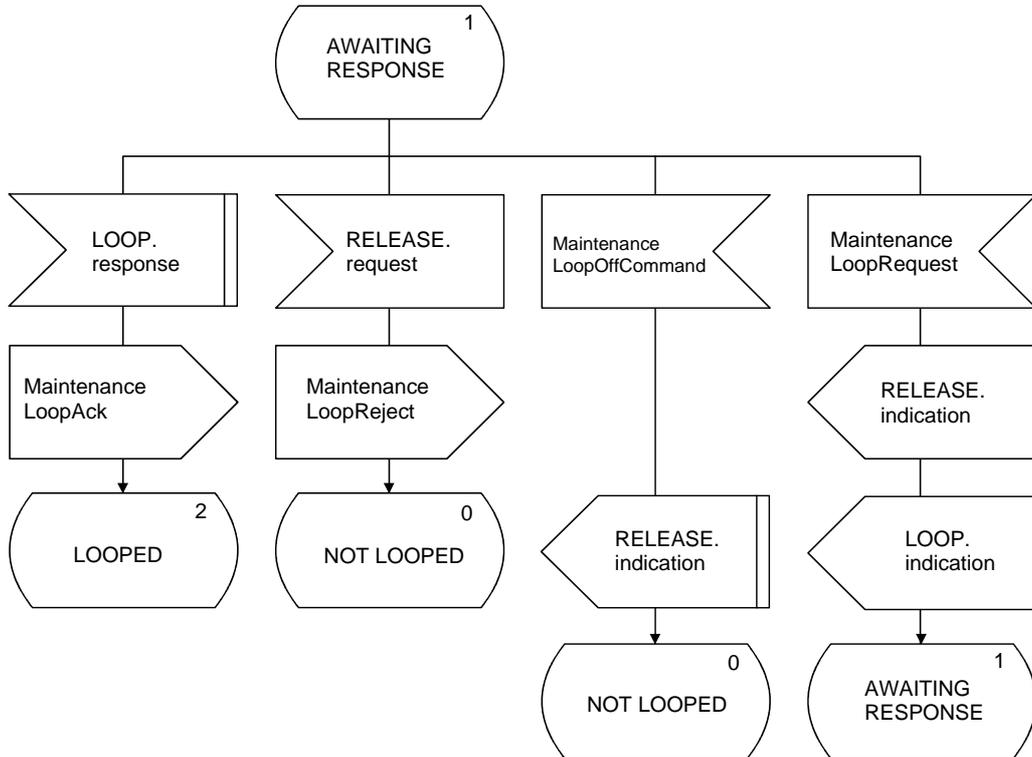


FIGURE 47 i)/H.245
Incoming MLSE SDL

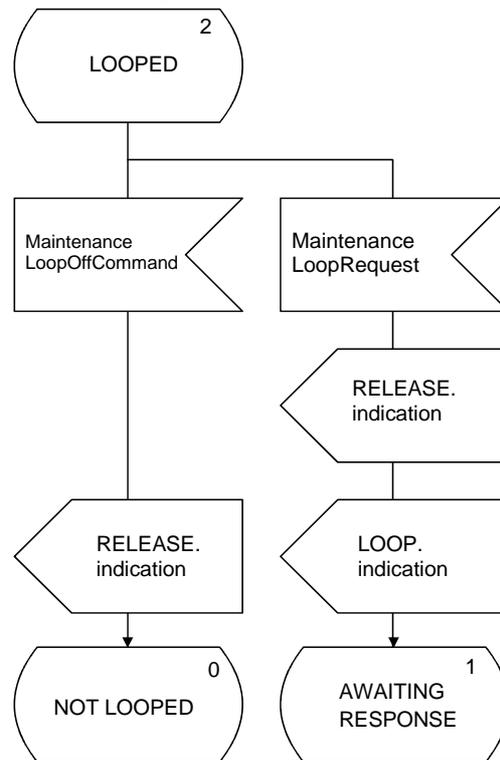
Superseded by a more recent version



T1520510-95/d078

FIGURE 47 ii)/H.245
Incoming MLSE SDL

Superseded by a more recent version



T1520520-95/d079

FIGURE 47 iii)/H.245
Incoming MLSE SDL

Annex A

Object identifier assignments

Table A.1 lists the assignment of Object Identifiers defined for use by this Recommendation.

TABLE A.1/H.245

Object Identifier Value	Description
{itu-t (0) recommendation (0) h (8) 245 version_(0) 1}	This Object Identifier is used to indicate the version of this Recommendation in use as a multimedia system control protocol. At this time there is a single standardized version defined.

Superseded by a more recent version

Appendix I

Overview of ASN.1 syntax

I.1 Introduction to ASN.1

Abstract Syntax Notation One (ASN.1) is a data specification language. It was originally standardized as part of the X.400 electronic mail series as X.409. This evolved to X.208 and most recently X.680. ASN.1 allows unambiguous specification of complex data structures including those with variable-length fields, optional fields and recursion.

The above Recommendations deal only with the syntax and semantics of ASN.1 specifications. The binary encoding of data structures is covered in other Recommendations, notably Recommendations X.690 (Basic Encoding Rules or BER) and X.691 (Packed Encoding Rules or PER). BER allows data to be deciphered by systems that have general knowledge of ASN.1 but do not know the details of the specification used to form the data. In other words, the data types are encoded along with the data values. PER is much more efficient since only data values are encoded and the coding is designed with very little redundancy. This method can be used when both the transmitter and the receiver expect data to adhere to a known structure.

This Recommendation is implemented using the packed encoding rules. Since both sides of a call know that messages will conform to the H.245 specification, it is not necessary to encode that specification into the messages. For decoding simplicity, the aligned variant of PER is used. This forces fields that require eight or more bits to be aligned on octet boundaries and to consume an integral number of octets. Alignment is done by padding the data with zeros before large fields.

I.2 Basic ASN.1 data types

The simplest data type is **BOOLEAN**, which represents the values **FALSE** and **TRUE**. These are encoded in a single bit as 0 and 1 respectively. For example, **segmentableFlag BOOLEAN** is coded:

Value	Encoding
FALSE	0
TRUE	1

The most fundamental data type is **INTEGER**, which represents whole number values. Integers can be unconstrained as in:

bitRate

INTEGER

or they can be constrained to a range of values, for example:

maximumAI2SDUSize

INTEGER (0..65535)

Constrained integers are encoded differently depending on the size of the range. Suppose N is the number of integers in the range, i.e. the upper limit minus the lower limit plus one. Depending on N, the constrained integer will be encoded in one of five ways:

N	Encoding
1	No bits needed
2-255	An unaligned field of 1 to 8 bits
256	An aligned 8-bit field
257-65536	An aligned 16-bit field
Larger	As the minimum number of aligned octets preceded by the above encoding of the number of octets

Superseded by a more recent version

In all cases, the number that is actually used is the value to be encoded minus the lower limit of the range. In these examples “pad” represents zero to seven 0 bits that are added to the encoding so that the following field will start on a 8-bit boundary.

firstGOB

INTEGER (0..17)

Value	Encoding
0	00000
3	00011

h233IVResponseTime

INTEGER (0..255)

Value	Encoding
3	pad 00000011
254	pad 11111110

skew

INTEGER (0..4095)

Value	Encoding
3	pad 00000000 00000011
4095	pad 00001111 11111111

Unconstrained (2’s complement) integer values that can be represented in 127 octets or less, are encoded in the minimum number of octets needed. The number of octets (the length) is encoded as an aligned octet that precedes the number itself. For example:

```

-1      pad 00000001 11111111
0       pad 00000001 00000000
128    pad 00000010 00000000 10000000
1000000 pad 00000011 00001111 01000010 01000000
    
```

ASN.1 supports a variety of string data types. These are variable-length lists of bits, octets or other short data types. They are typically encoded as a length followed by the data. The length can be encoded as an unconstrained integer or as a constrained integer if the SIZE of the string is specified. For example:

data

OCTET STRING

Since the length of the octet string is not bounded, it will have to be encoded as a *semi-constrained whole number* (has a lower bound, but no upper bound). First, the data is padded so that the encoding will be aligned. The rest of the code is as follows:

Length	Encoding
0 to 127	8-bit length followed by the data
128 to 16K-1	16-bit length with the MSB set, then the data
16K to 32K-1	11000001, 16K octets of data, then code the rest
32K to 48K-1	11000010, 32K octets of data, then code the rest
48K to 64K-1	11000011, 48K octets of data, then code the rest
64K or more	11000100, 64K octets of data, then code the rest

This method is called “fragmentation”. Note that if the length is a multiple of 16K, then the representation will end with an octet of zero indicating a zero-length string.

Superseded by a more recent version

I.3 Aggregate data types

ASN.1 includes several aggregate or container data types that are similar in concept to C's union, struct and array types. These are, respectively, CHOICE, SEQUENCE and SEQUENCE OF. In all cases they are encoded with some bits specific to the container followed by the normal encodings of the contents.

CHOICE is used to select exactly one of a group of data types. For example:

```
VideoCapability ::= CHOICE
{
    nonStandard          NonStandardParameter,
    h261VideoCapability  H261VideoCapability,
    h262VideoCapability  H262VideoCapability,
    h263VideoCapability  H263VideoCapability,
    is11172VideoCapability IS11172VideoCapability
    ...
}
```

An index number is assigned to each choice, starting with zero. The index of the actual choice is encoded as a constrained integer. The index is followed by the encoding of the actual selection or nothing if the selection is **NULL**. If the extension marker is present (as above), the index is preceded by a bit that is zero if the actual choice is from the original list.

SEQUENCE is simply a grouping of dissimilar data types. Individual elements of the sequence may be **OPTIONAL**. The encoding is very simple. If there is an extension marker, the first bit indicates the presence of additional elements. This is followed by a series of bits, one for each optional element that indicates if that data is present. This is followed by the encodings of the components of the sequence. For example:

```
H261VideoCapability ::= SEQUENCE
{
    qcifMPI          INTEGER (1..4) OPTIONAL, -- units 1/29.97 Hz
    cifMPI           INTEGER (1..4) OPTIONAL, -- units 1/29.97 Hz,
    temporalSpatialTradeOffCapability  BOOLEAN,
    ...
}
```

The encoding has one bit for the extension marker, two bits for the optional fields, two bits each for any optional field that is present, one bit for the boolean and then any extension data. Note that this sequence has no padding for octet alignment.

The SEQUENCE OF and SET OF types describe a collection of similar components (an array). SEQUENCE OF implies that the order of the elements is significant, with SET OF the element order is arbitrary. The PER encoding is the same for both types.

These types can have a **SIZE** constraint or an unconstrained number of elements. If the number is known *a priori* and is less than 64K, it is not encoded. Otherwise the actual number of components is encoded as a constrained or semi-constrained length. This is followed by the encoding of the data. If the length is at least 16K and is encoded, then the list of data will be broken into fragments like the octet string. In this case the fragments are broken after some number of component fields (16K, 32K, etc.), not after some number of octets.

I.4 Object Identifier type

Normally the type of a value is given in the ASN.1 specification so that the only information that needs to be coded and transmitted is the data itself. Occasionally, however, it is desirable to encode the data type as well as the data value. For example, **NonStandardIdentifier** contains

```
object          OBJECT IDENTIFIER
```

This is encoded as the data encoded with the BER (see Recommendation X.690) preceded by the length of that encoding in octets. The length is encoded as a semi-constrained whole number (see the OCTET STRING example above).

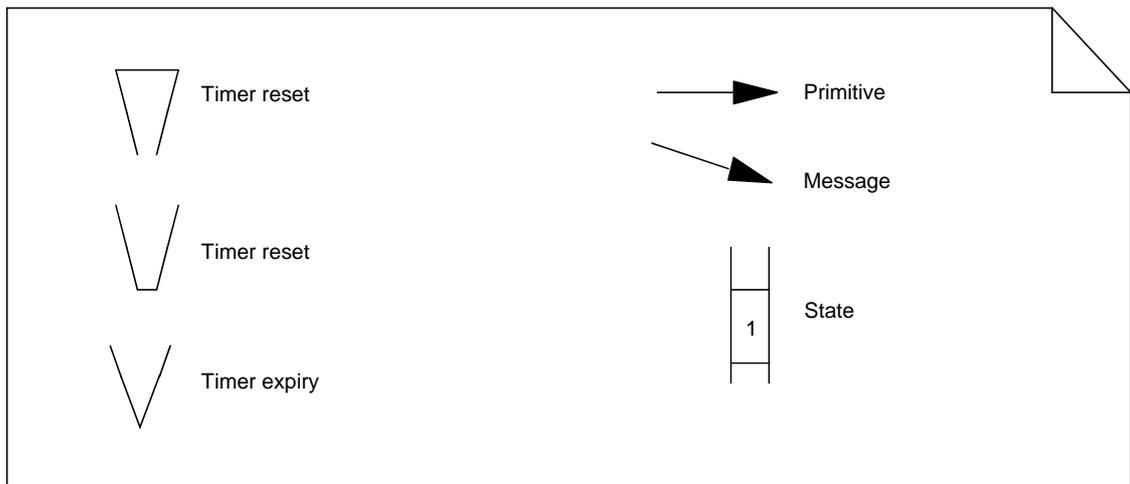
Superseded by a more recent version

Appendix II

Examples of H.245 procedures

II.1 Introduction

This appendix illustrates examples of the procedures defined in clause 8. Figure II.1-1 shows the key to diagrams used in this appendix.



T1523580-96/d080

FIGURE II.1-1/H.245

Key to figures

II.2 Master Slave Determination Signalling Entity

In Figures II.2-1 to II.2-5, messages are represented by the shortened names given in Table II.2-1.

TABLE II.2-1/H.245

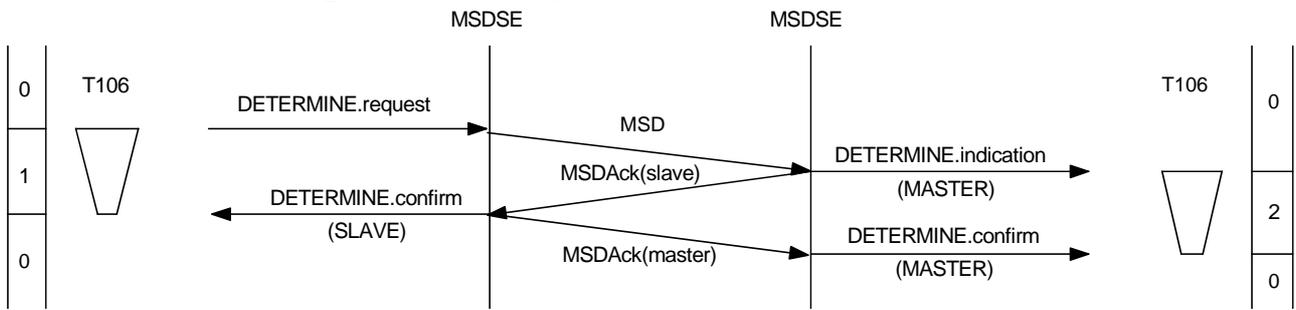
Master slave determination shortened names

Message	Name in examples
MasterSlaveDetermination	MSD
MasterSlaveDeterminationAck	MSDAck
MasterSlaveDeterminationReject	MSDReject
MasterSlaveDeterminationRelease	MSDRelease

In Figures II.2-1 to II.2-5, IDLE, OUTGOING AWAITING RESPONSE, and INCOMING AWAITING RESPONSE states are labelled as “0”, “1”, and “2” respectively.

In Figures II.2-1 to II.2-5, the parameter value associated with the DETERMINE.indication and DETERMINE.confirm primitives is that of the TYPE parameter. The field value associated with the MasterSlaveDeterminationAck message is that of the decision field.

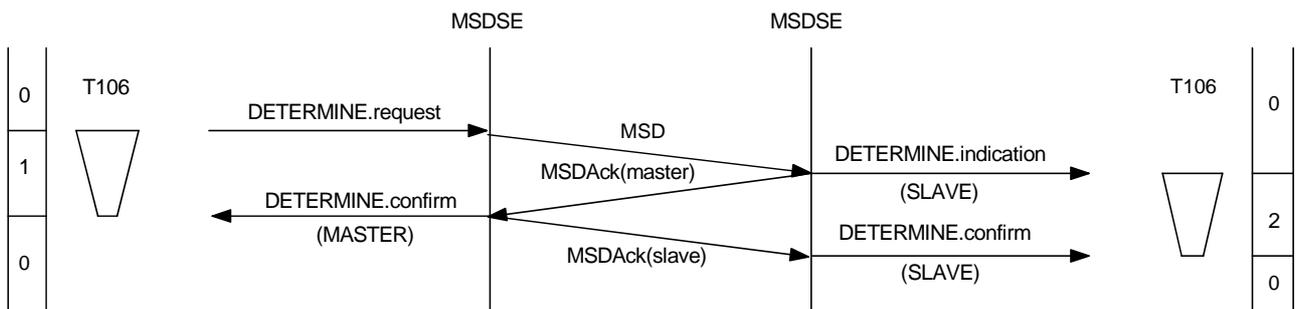
Superseded by a more recent version



T1523590-96/d081

FIGURE II.2-1/H.245

Master slave determination – Master at remote MSDSE

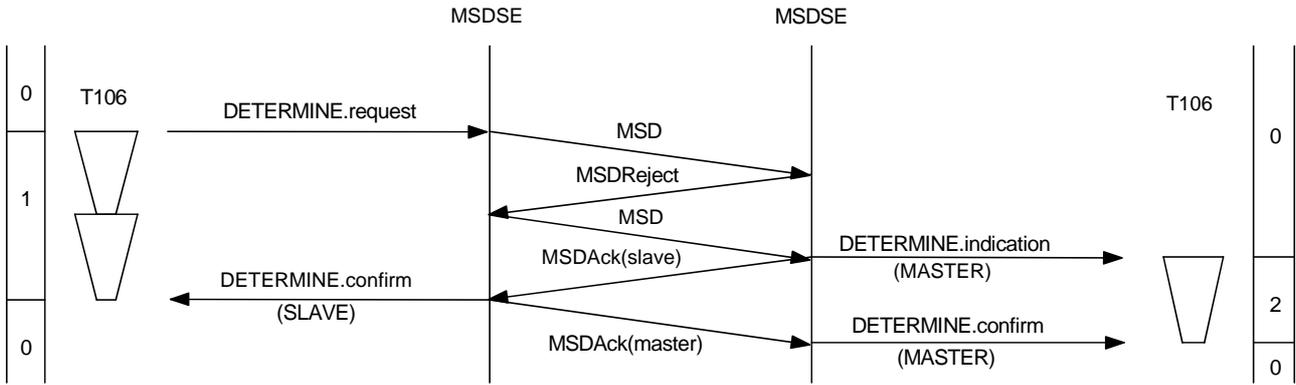


T1523600-96/d082

FIGURE II.2-2/H.245

Master slave determination – Slave at remote MSDSE

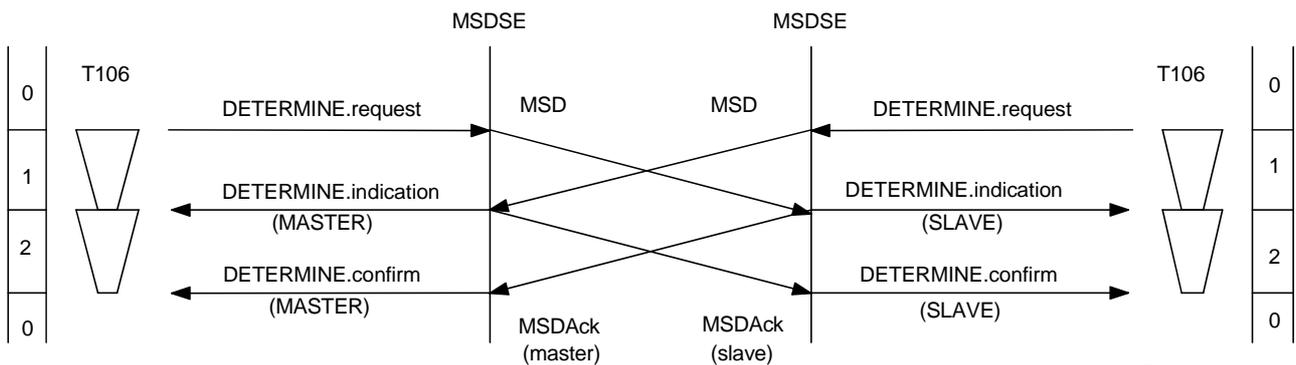
Superseded by a more recent version



T1523610-96/d083

FIGURE II.2-3/H.245

Master slave determination – First attempt produced an indeterminate result – The second attempt was successful



T1523620-96/d084

FIGURE II.2-4/H.245

Master slave determination – Simultaneous determination

Superseded by a more recent version

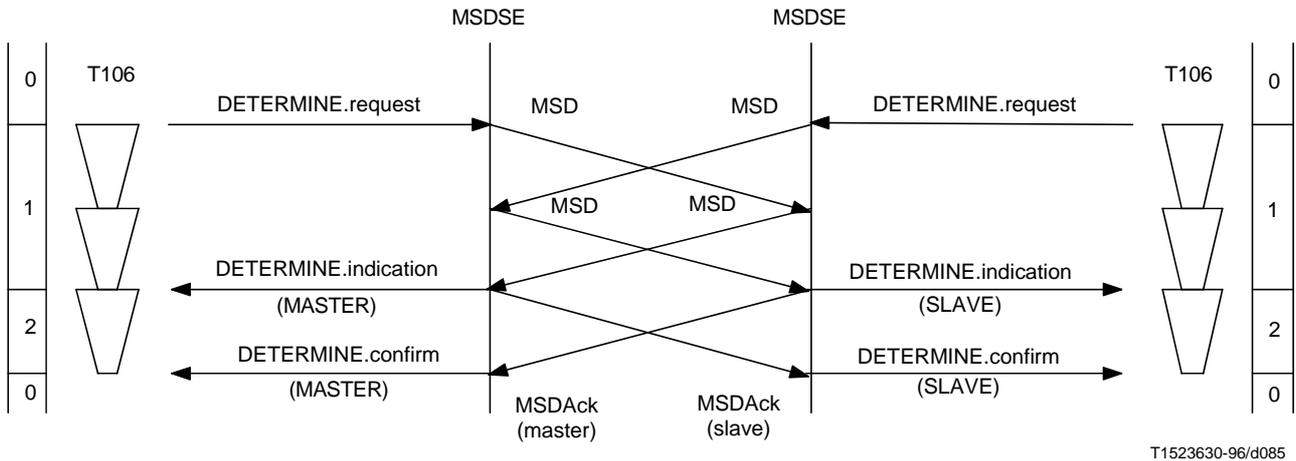


FIGURE II.2-5/H.245

Master slave determination – Simultaneous determination but with the first attempt returning an indeterminate result

In Figure II.2-6 local timer T106 has expired. Only the terminal on the right knows its status. The terminal on the right is able to receive new commands but may not request anything of the other terminal that relies on knowledge of the status determination result. The terminal on the left can neither accept nor initiate new procedures. A second status determination procedure should be initiated.

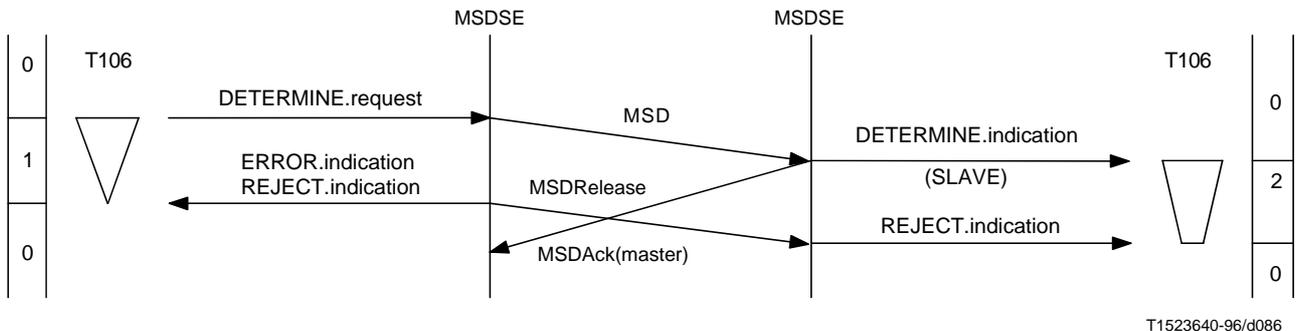


FIGURE II.2-6/H.245

Master slave determination – Local timer T106 expiry with slave at remote end

In Figure II.2-7 remote timer T106 has expired during the INCOMING AWAITING ACKNOWLEDGEMENT state. Both terminals know their status. The terminal on the left may receive and issue commands. However, the remote terminal does not know if the local terminal is ready to receive, and can not issue commands that rely on knowledge of the status determination result. A second status determination procedure should be initiated.

Superseded by a more recent version

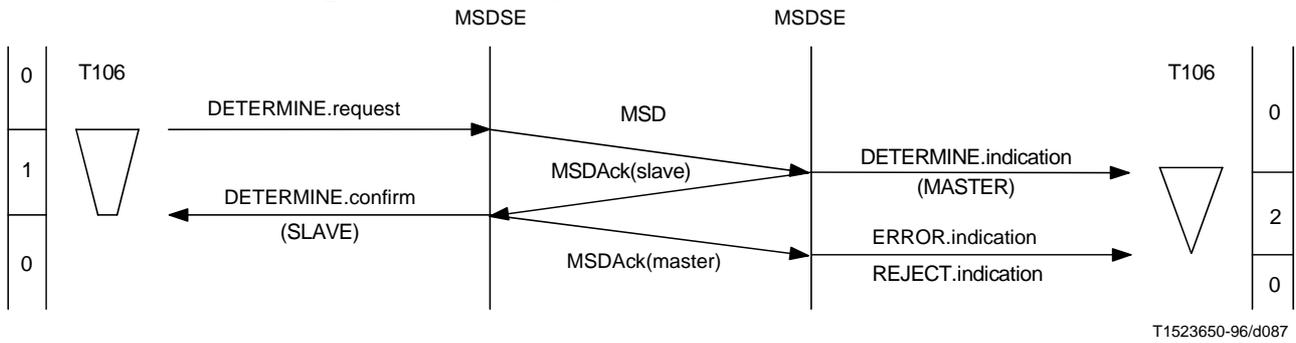


FIGURE II.2-7/H.245

Master slave determination – Remote timer T106 expiry with master at remote end

In Figure II.2-8 remote timer T106 has expired during the OUTGOING AWAITING ACKNOWLEDGEMENT state during a simultaneous determination procedure. Both terminals know their status. The terminal on the right can receive and issue commands. However, the terminal on the left does not know if the other terminal is ready to receive, and can not issue commands that rely on knowledge of the status determination result. It may receive such commands. A second status determination procedure should be initiated.

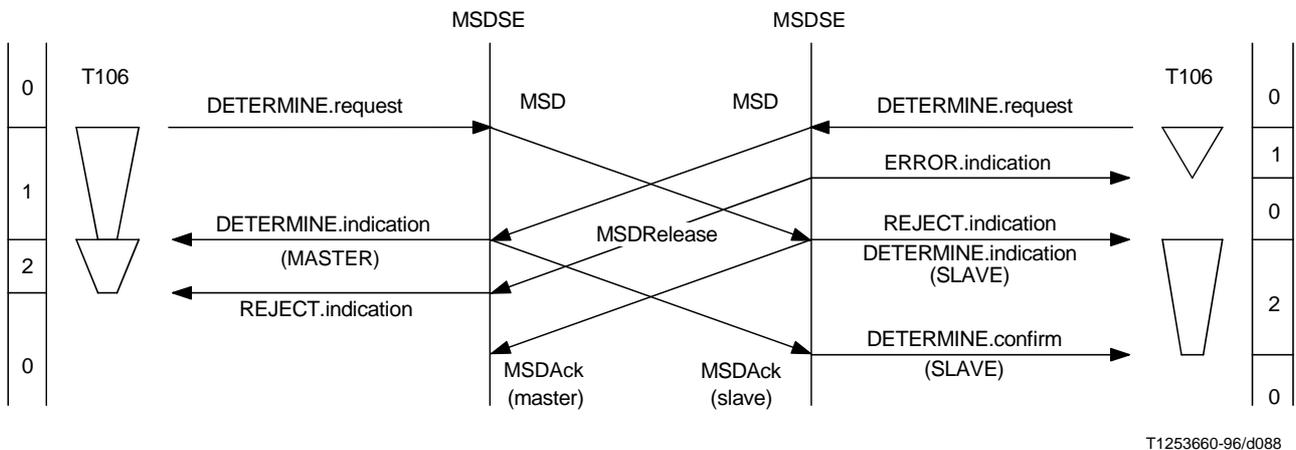


FIGURE II.2-8/H.245

Master slave determination – Simultaneous determination procedures with timer T106 expiry at slave

In Figure II.2-9 remote timer T106 has expired during the INCOMING AWAITING ACKNOWLEDGEMENT state, during a simultaneous determination procedure. Both terminals know their status. The terminal on the left can receive and issue commands. However the terminal on the right does not know if the other terminal is ready to receive, and can not issue commands that rely on knowledge of the status determination result. It may receive such commands. A second status determination procedure should be initiated.

Superseded by a more recent version

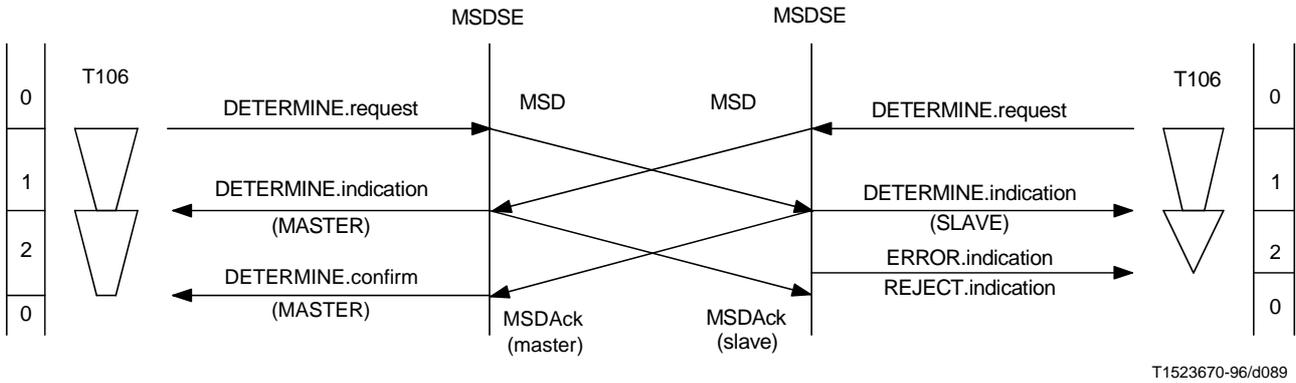


FIGURE II.2-9/H.245

Master slave determination – Simultaneous determination procedures with timer T106 expiry during INCOMING AWAITING ACKNOWLEDGEMENT state

In Figure II.2-10 an indeterminate result was obtained N100 times. In this case N100 = 3.

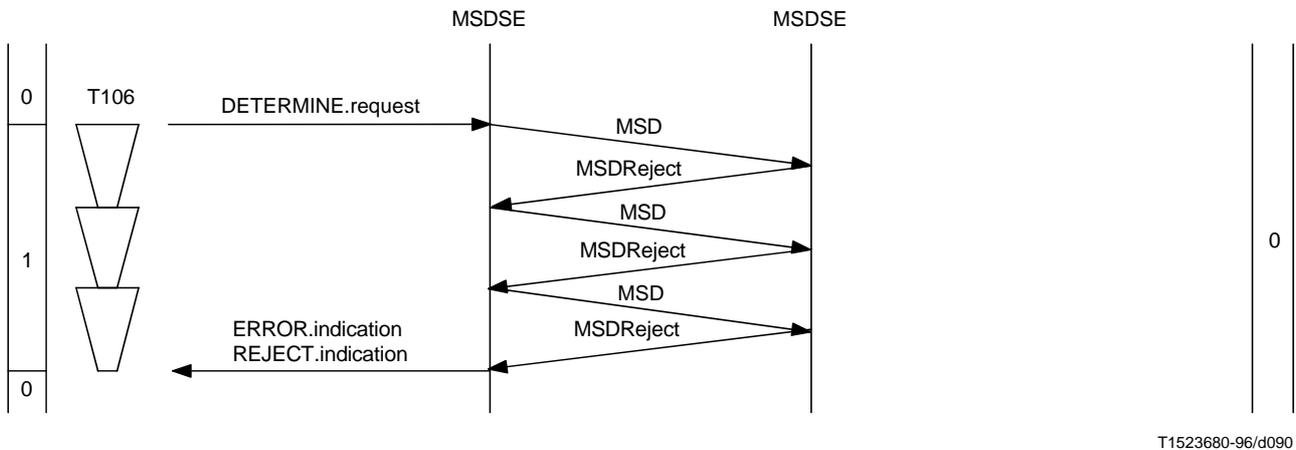


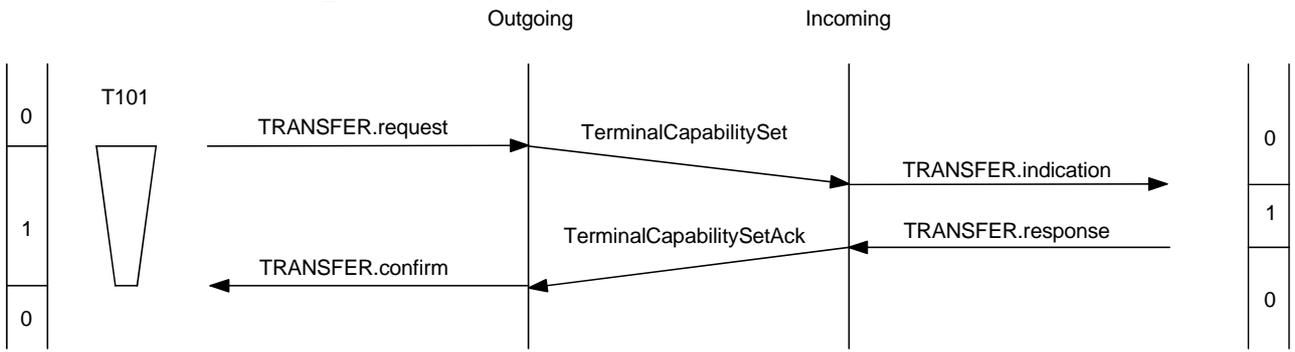
FIGURE II.2-10/H.245

Master slave determination – Indeterminate result with N100 = 3

II.3 Capability Exchange Signalling Entity

Figures II.3-1 to II.3-4 illustrate CESE procedures. The IDLE and AWAITING RESPONSE states are labelled as “0” and “1” respectively.

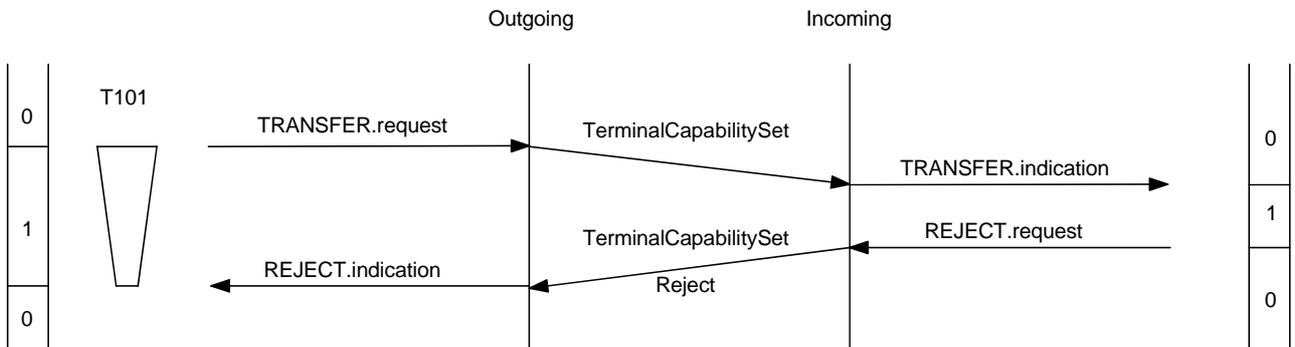
Superseded by a more recent version



T1523690-96/d091

FIGURE II.3-1/H.245

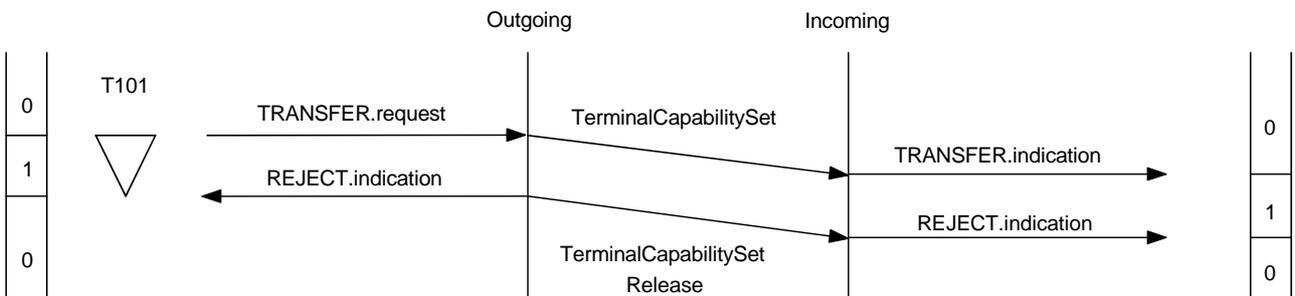
Capability exchange with acceptance from the peer incoming CESE user



T1523700-96/d092

FIGURE II.3-2/H.245

Capability exchange with rejection from peer incoming CESE user

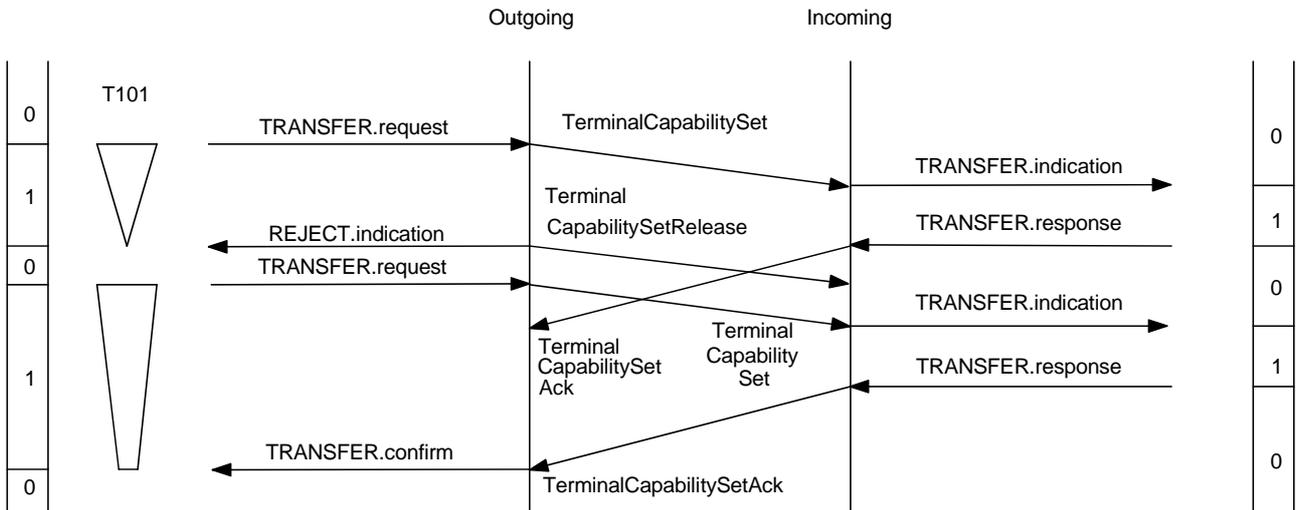


T1523710-96/d093

FIGURE II.3-3/H.245

Capability exchange with timer T101 expiry – The TerminalCapabilitySetRelease message arrives at the incoming CESE before response from the incoming CESE user

Superseded by a more recent version



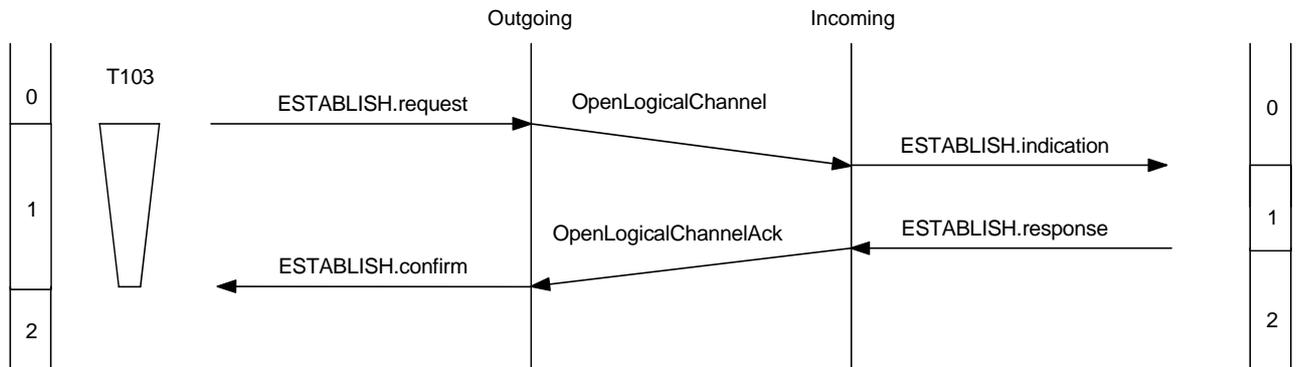
T1523720-96/d094

FIGURE II.3-4/H.245

Capability exchange with timer T101 expiry followed by a second capability exchange – The TerminalCapabilitySetRelease message arrives at the incoming CESE after response from the incoming CESE user – At the outgoing CESE the TerminalCapabilitySetAck message in response to the first TerminalCapabilitySet message is ignored – Only the second capability exchange is successful

II.4 Logical Channel Signalling Entity

Figures II.4-1 to II.4-7 illustrate LCSE procedures. The outgoing LCSE states of RELEASED, AWAITING ESTABLISHMENT, ESTABLISHED, and AWAITING RELEASE are labelled as “0”, “1”, “2”, and “3” respectively. The incoming LCSE states of RELEASED, AWAITING ESTABLISHMENT, and ESTABLISHED, are labelled as “0”, “1”, and “2” respectively.

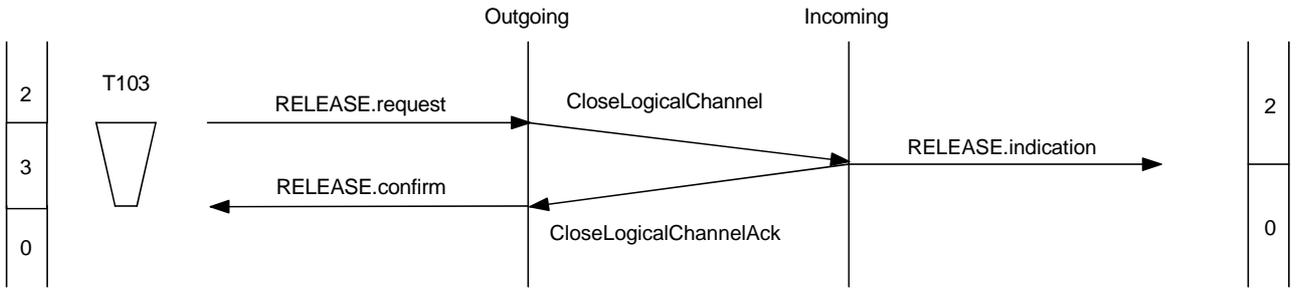


T1523730-96/d095

FIGURE II.4-1/H.245

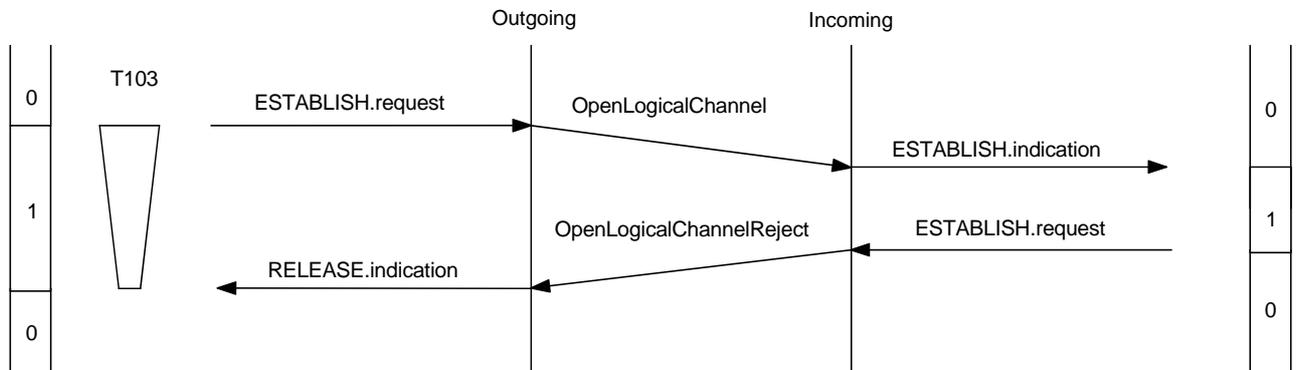
Logical channel establishment

Superseded by a more recent version



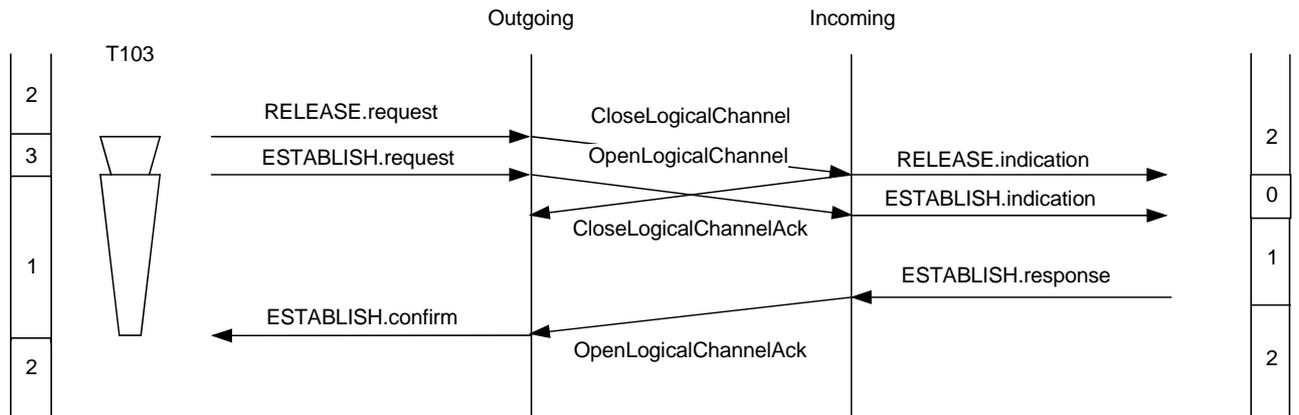
T1523740-96/d096

FIGURE II.4-2/H.245
Logical channel release



T1523750-96/d097

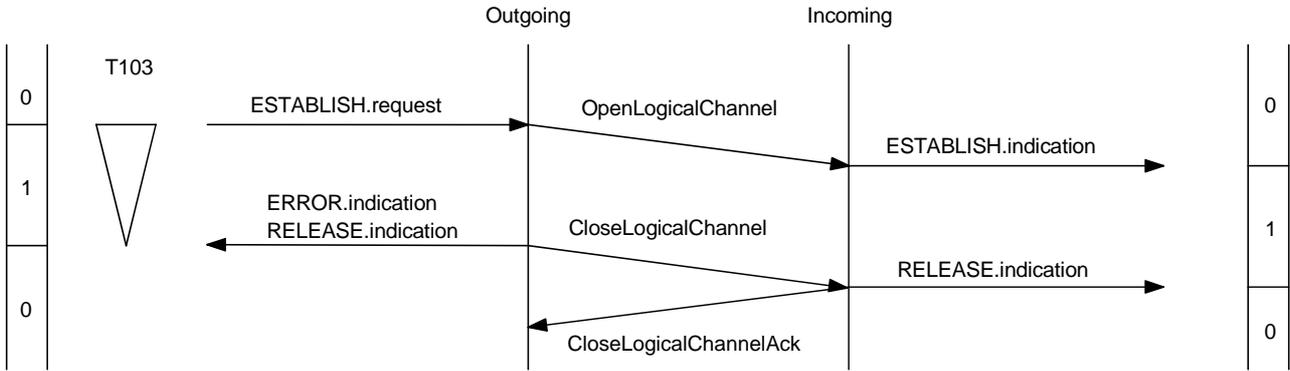
FIGURE II.4-3/H.245
Logical channel establishment rejection by peer LCSE user



T1523760-96/d098

FIGURE II.4-4/H.245
Logical channel release followed by immediate re-establishment

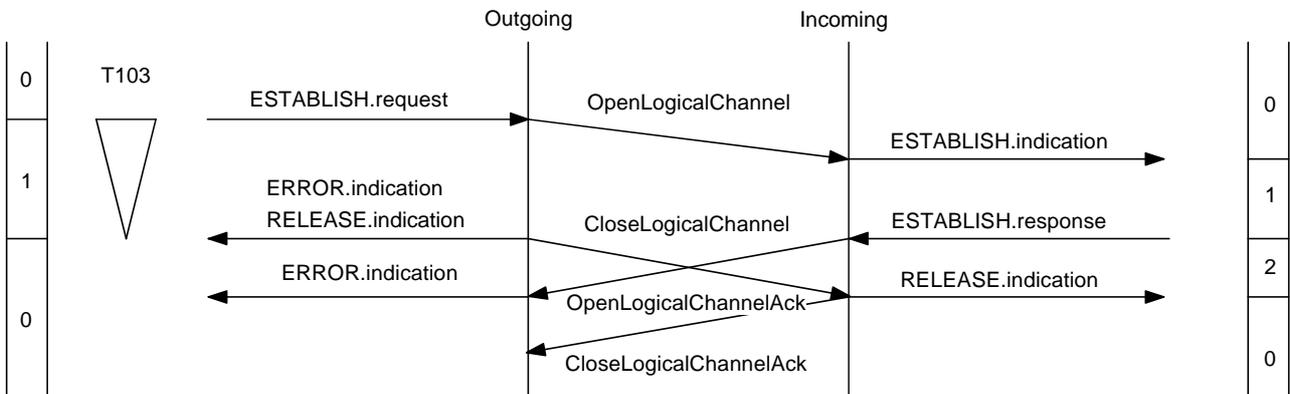
Superseded by a more recent version



T1523770-96/d099

FIGURE II.4-5/H.245

Logical channel establishment request with expiry of timer T103 due to slow response from peer incoming LCSE user

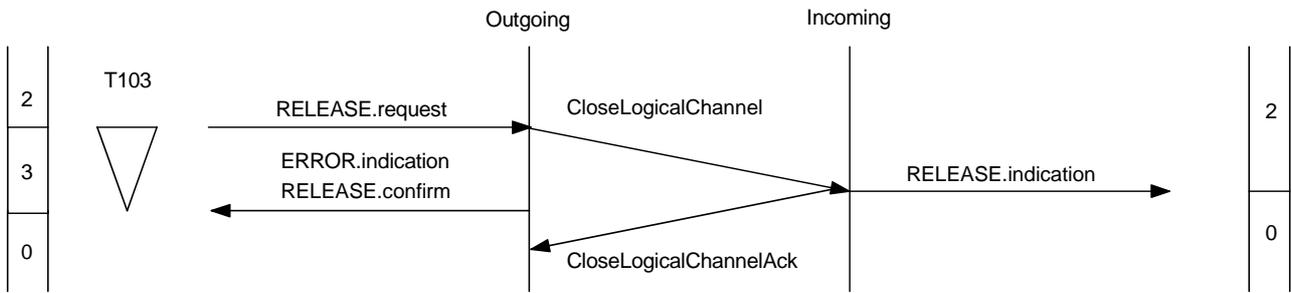


T1523780-96/d100

FIGURE II.4-6/H.245

Logical channel establishment request with expiry of timer T103 – Timer T103 has expired after transmission of the OpenLogicalChannelAck message at the incoming LCSE, but before reception of the OpenLogicalChannelAck message at the outgoing LCSE

Superseded by a more recent version



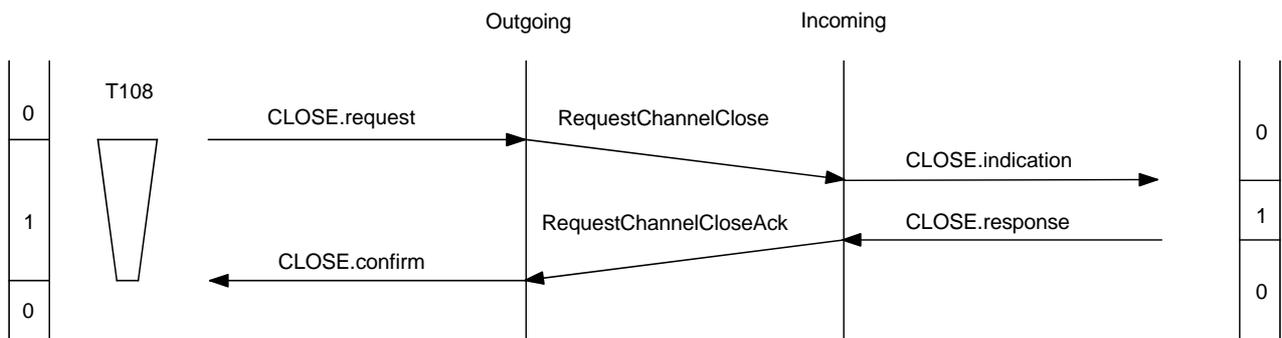
T1523790-96/d101

FIGURE II.4-7/H.245

Logical channel release request with expiry of timer T103

II.5 Close Logical Channel Signalling Entity

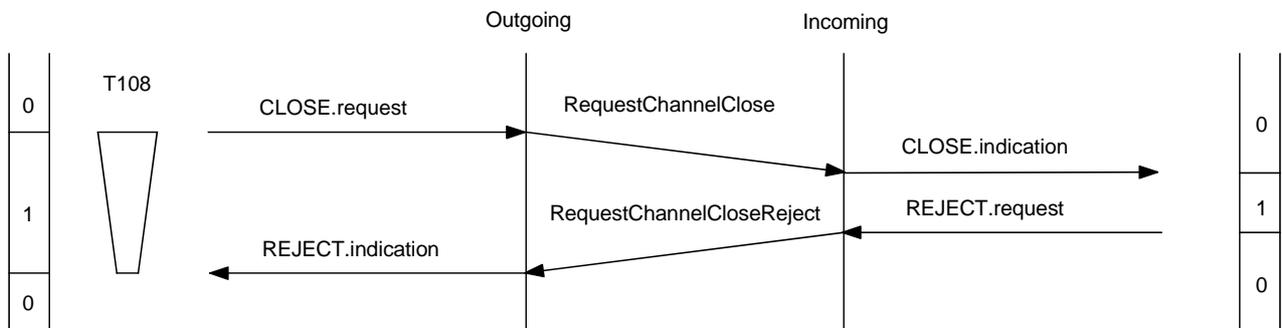
Figures II.5-1 to II.5-4 illustrate CLCSE procedures. The IDLE and AWAITING RESPONSE states are labelled as “0” and “1” respectively.



T1523800-96/d102

FIGURE II.5-1/H.245

Close logical channel request



T1523810-96/d103

FIGURE II.5-2/H.245

Close logical channel request with rejection from peer incoming CLCSE user

Superseded by a more recent version

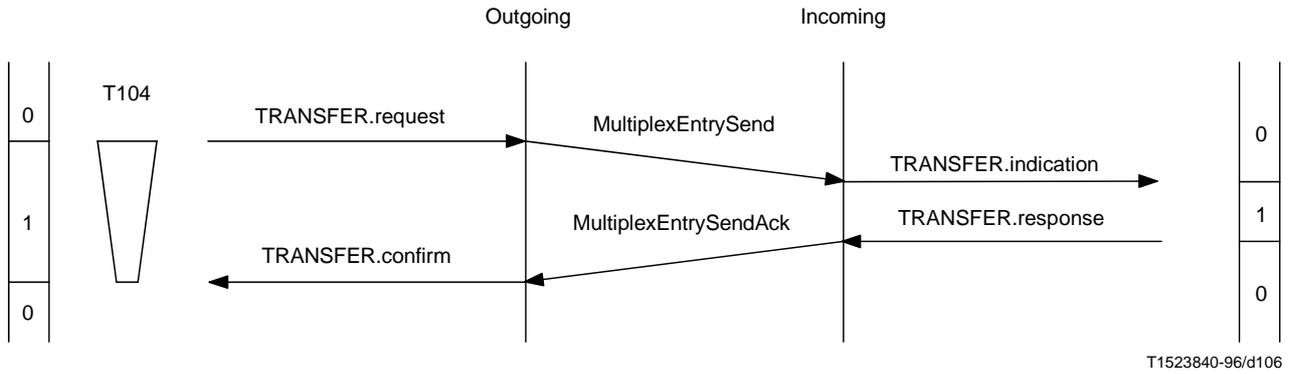


FIGURE II.6-1/H.245
Successful multiplex table send request

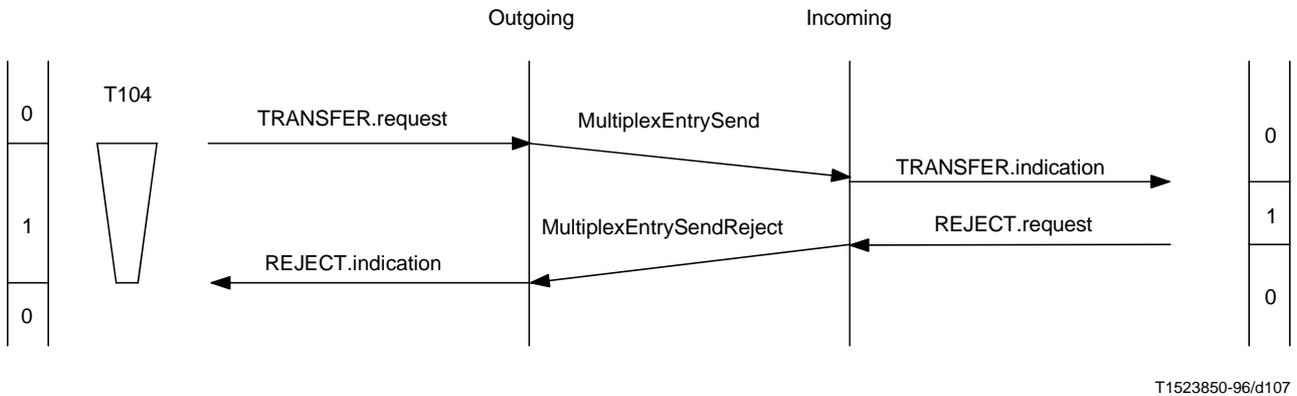


FIGURE II.6-2/H.245
Multiplex table send request with rejection from the peer MTSE user

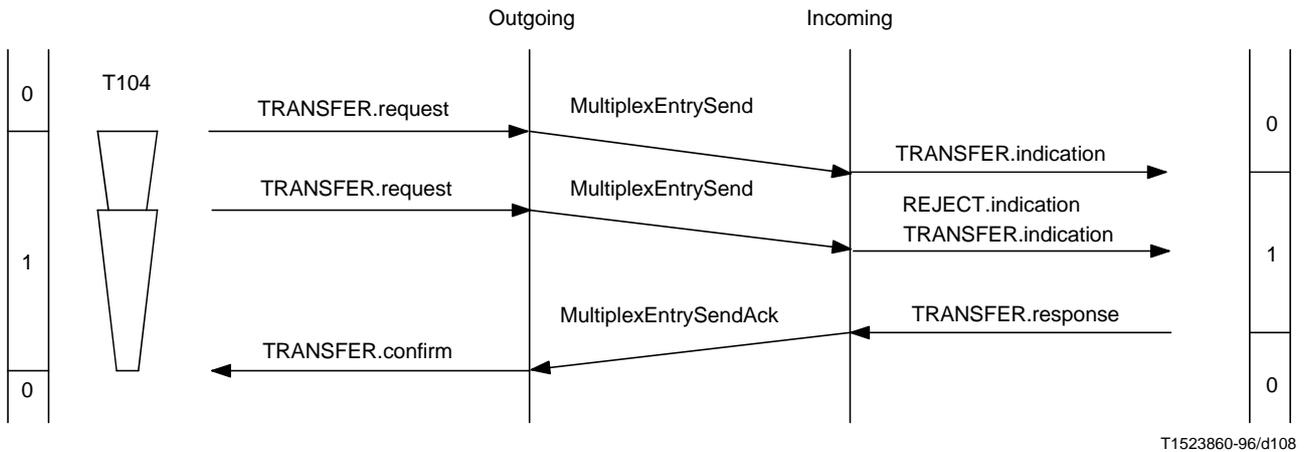


FIGURE II.6-3/H.245
Multiplex table send request with a second multiplex table send request before acknowledgement of the first request – The first request was unsuccessful

Superseded by a more recent version

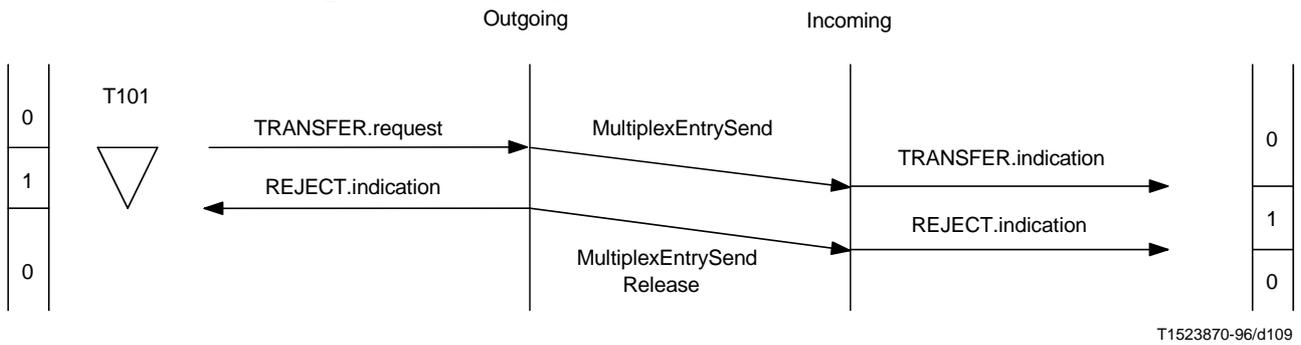


FIGURE II.6-4/H.245

Multiplex table send request with timer T104 expiry

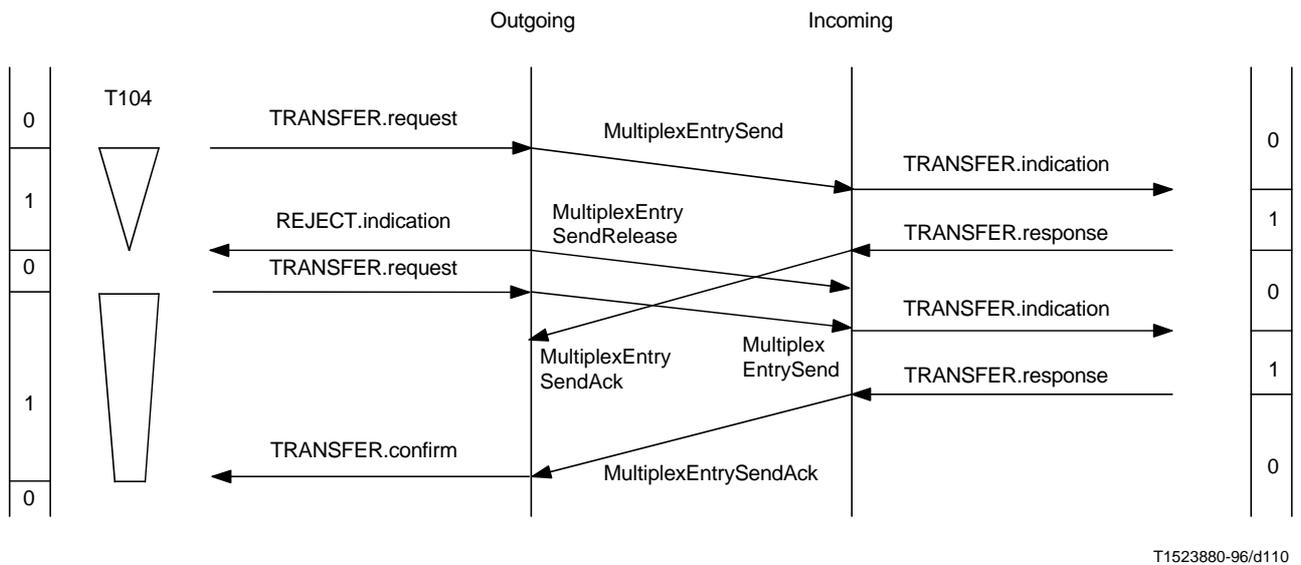


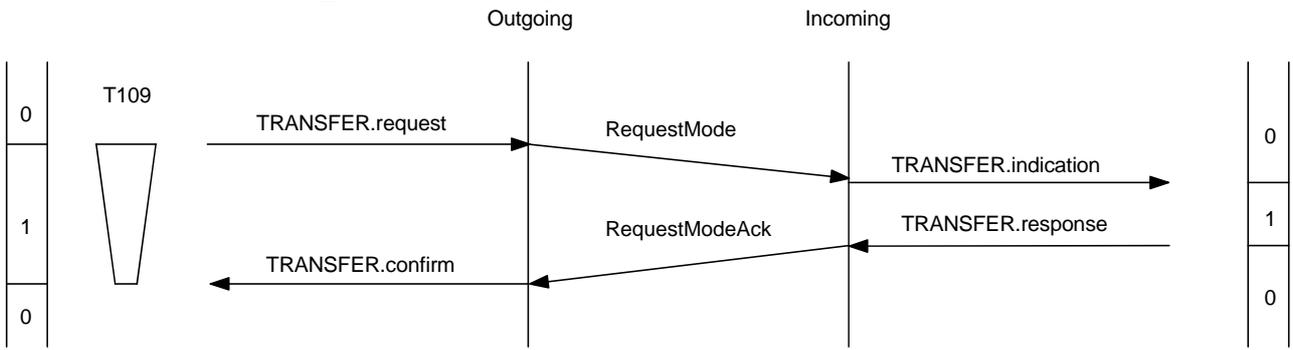
FIGURE II.6-5/H.245

Multiplex table send request with timer T104 expiry followed by a second multiplex table send request – The first MultiplexEntrySendAck message is ignored at the outgoing MTSE – Only the second request was successful

II.7 Mode Request Signalling Entity

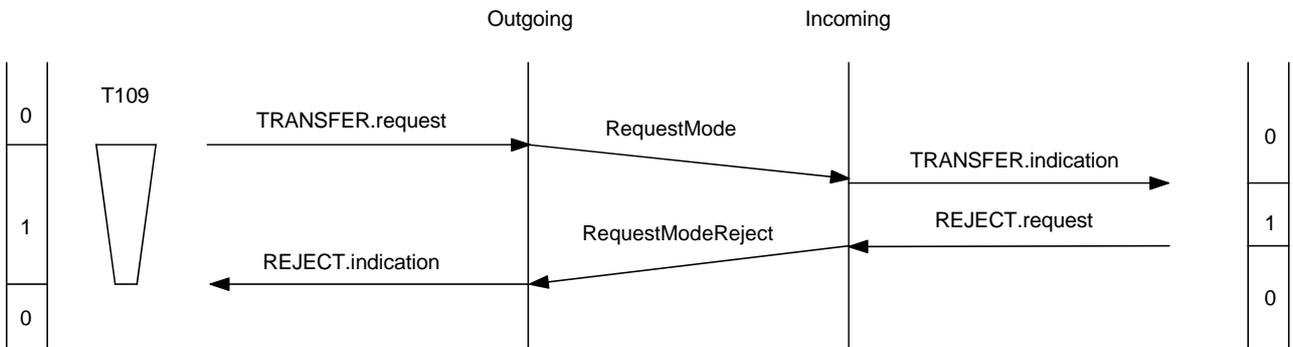
Figures II.7-1 to II.7-5 illustrate MTSE exchanges. The IDLE and AWAITING RESPONSE states are labelled as “0” and “1” respectively.

Superseded by a more recent version



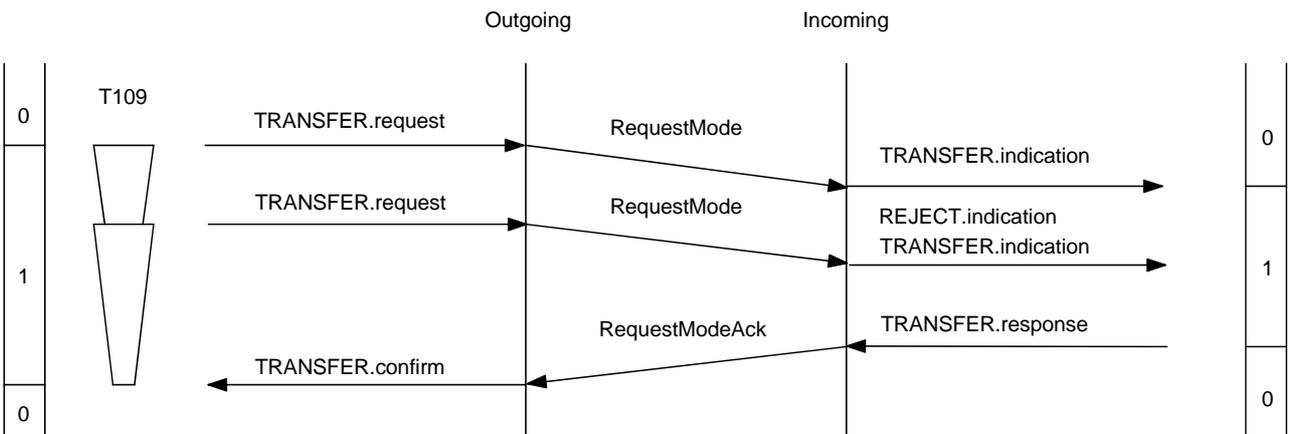
T1523890-96/d111

FIGURE II.7-1/H.245
Successful mode request



T1523900-96/d112

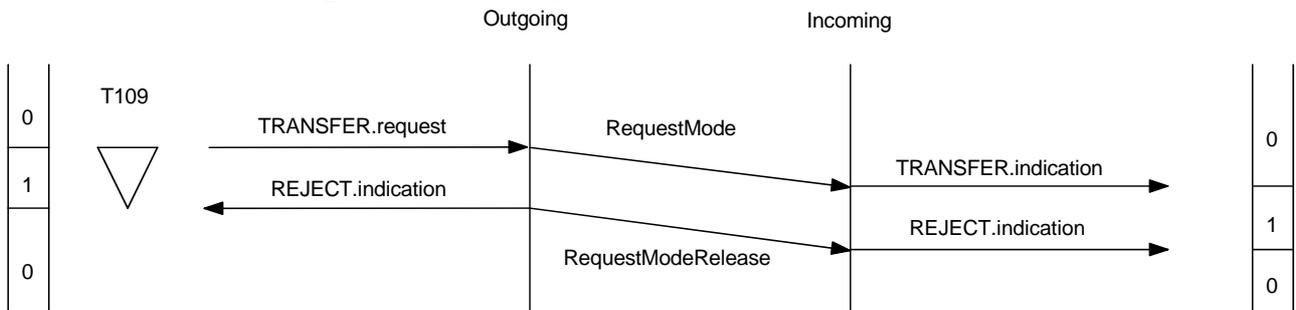
FIGURE II.7-2/H.245
Mode request with rejection from the peer MTSE user



T1523910-96/d113

FIGURE II.7-3/H.245
Mode request with a second mode request before acknowledgement of the first request –
The first request was unsuccessful

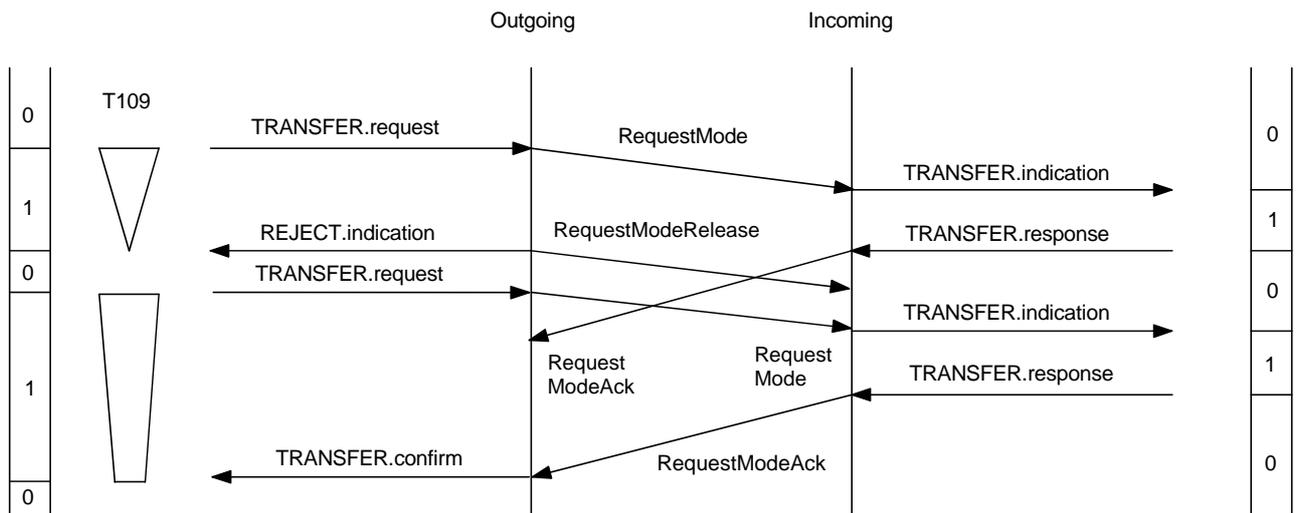
Superseded by a more recent version



T1523920-96/d114

FIGURE II.7-4/H.245

Mode request with timer T109 expiry – The mode request was unsuccessful



T1523930-96/d115

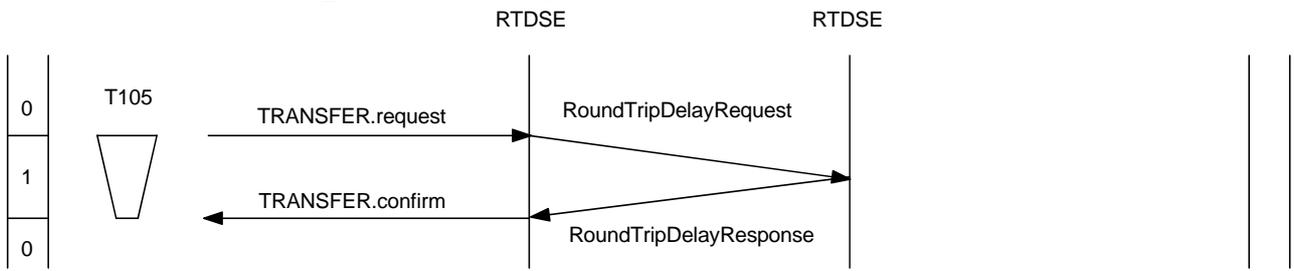
FIGURE II.7-5/H.245

**Mode request with timer T109 expiry followed by a second mode request –
The first RequestModeAck message is ignored at the outgoing MRSE –
Only the second request was successful**

II.8 Round Trip Delay Signalling Entity

Figures II.8-1 to II.8-4 illustrate RTDSE procedures. The RTDSE states of IDLE and AWAITING RESPONSE are labelled as “0” and “1” respectively.

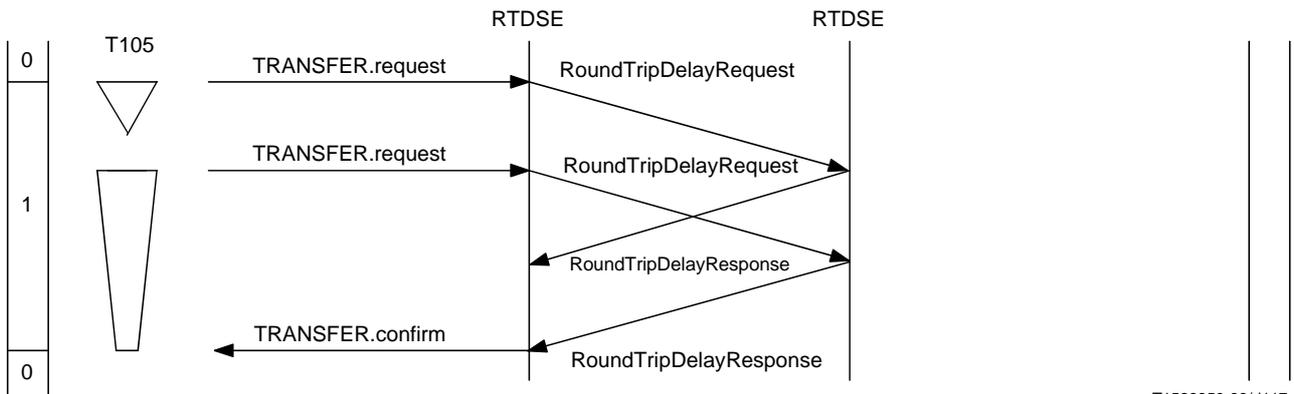
Superseded by a more recent version



T1523940-96/d116

FIGURE II.8-1/H.245

Round trip delay determination procedure

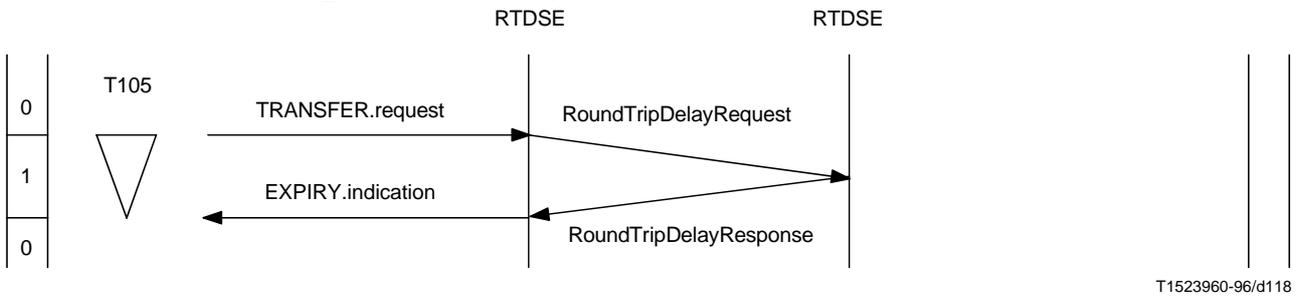


T1523950-96/d117

FIGURE II.8-2/H.245

**Round trip delay determination procedure
with an earlier unacknowledged round trip delay procedure outstanding**

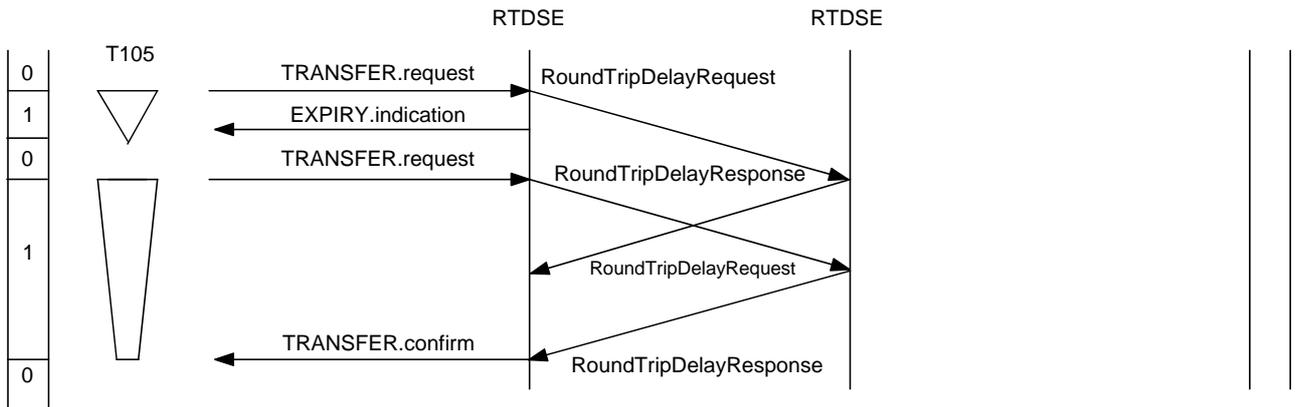
Superseded by a more recent version



T1523960-96/d118

FIGURE II.8-3/H.245

Round trip delay determination procedure with timer T105 expiry



T1523970-96/d119

FIGURE II.8-4/H.245

Round trip delay determination procedure with timer T105 expiry, followed by a second round trip delay determination procedure – The RoundTripDelayResponse message from the first procedure arrives during the second procedure and is ignored

Superseded by a more recent version

Appendix III

Summary of procedure timers and counters

This appendix provides a list of the timers and counters specified in clause 8.

III.1 Timers

Table III.1-1 shows the timers specified in this Recommendation.

TABLE III.1-1/ H.245

Procedure timers

Timer	Procedure	Definition
T106	Master Slave Determination	This timer is used in the OUTGOING AWAITING RESPONSE state and during the INCOMING AWAITING RESPONSE state. It specifies the maximum time during which no acknowledgement message may be received.
T101	Capability Exchange	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no TerminalCapabilitySetAck or TerminalCapabilitySetReject message may be received.
T103	Unidirectional and Bidirectional Logical Channel Signalling	This timer is used in the AWAITING ESTABLISHMENT and AWAITING RELEASE states. It specifies the maximum time during which no OpenLogicalChannelAck or OpenLogicalChannelReject or CloseLogicalChannelAck message may be received.
T108	SEND Logical Channel	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no RequestMultiplexEntryAck or RequestMultiplexEntryReject message may be received.
T104	H.223 Multiplex Table	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no MultiplexEntrySendAck or MultiplexEntrySendReject message may be received.
T109	Mode Request	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no RequestModeAck or RequestModeReject message may be received.
T105	Round Trip Delay	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no RoundTripDelayResponse message may be received.
T107	Request Multiplex Entry	This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no RequestMultiplexEntryAck or RequestMultiplexEntryReject message may be received.
T102	Maintenance Loop	This timer is used during the AWAITING RESPONSE state. It specifies the maximum allowed time during which no MaintenanceLoopAck or MaintenanceLoopReject message may be received.

III.2 Counters

Table III.2-1 shows the counters specified in this Recommendation.

TABLE III.2-1/ H.245

Procedure counters

Timer	Procedure	Definition
N100	Master slave determination	This counter specifies the maximum number of times that MasterSlaveDetermination messages will be sent during the OUTGOING AWAITING RESPONSE state.

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: definitions, symbols, classification
- Series C General telecommunication statistics
- Series D General tariff principles
- Series E Overall network operation, telephone service, service operation and human factors
- Series F Non-telephone telecommunication services
- Series G Transmission systems and media, digital systems and networks
- Series H Audiovisual and multimedia systems**
- Series I Integrated services digital network
- Series J Transmission of television, sound programme and other multimedia signals
- Series K Protection against interference
- Series L Construction, installation and protection of cables and other elements of outside plant
- Series M Maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
- Series N Maintenance: international sound programme and television transmission circuits
- Series O Specifications of measuring equipment
- Series P Telephone transmission quality, telephone installations, local line networks
- Series Q Switching and signalling
- Series R Telegraph transmission
- Series S Telegraph services terminal equipment
- Series T Terminals for telematic services
- Series U Telegraph switching
- Series V Data communication over the telephone network
- Series X Data networks and open system communication
- Series Z Programming languages