ITU Telecommunications Standardisation Sector **LBC-95-261 / AVC-824**
Study Group 15, Working Party 15/1
Experts' Group on Very Low Bitrate Video Telephony **Darmstadt, 17-20 October, 1995**
Experts' Group for Video Coding and Systems in **Yokosuka, 24-27 October, 1995**
ATM and Other Network Environments

Title: **Report of the H.245 Ad Hoc Committee**
Source*: **Chairperson H.245 Ad Hoc Committee**

A considerable amount of e-mail was sent to the reflector both before the submission of the White Paper H.245 to Geneva in July and afterwards. Modifications were made by Mike Nilsson to the Boston Draft based on agreements made at the Boston meeting and after discussion on the reflector. These are summarised below:
- Addition of Stuart Dunstan's text for Section 8 on procedures with some modifications after discussion on the reflector.
- Addition of Q.2931 parameters.
- Addition of Dave Lindberg's explanatory text for capsets.
- Removal of original Appendix I.
- Addition of extra audio codepoints.
- Addition of codepoints into end session according to proposal of Chris Hansen.

The modified draft was submitted as a White Paper to Geneva in time to meet the deadline.

The following summarise the comments and suggested additions sent to the reflector after submission of the White Paper:

1) Suggested text was sent by Michael Patrick regarding the NLPID data application. In subsequent discussion it was suggested that the current text is enough to properly support NLPID but that it might require clarification.

2) Problem in Appendix II relating to the coding of 'larger' constrained integers pointed out.

3) DataProtocolCapability is defined as a sequence and the first element in the sequence is nonStandard. Suggested fixes are: make nonStandard OPTIONAL; make DataProtocolCapability a CHOICE and change the BOOLEANs to NULLs; keep DataProtocolCapability as a SEQUENCE and make nonStandard a SET SIZE (0..x) of NonStandardParameter.

4) A query was raised regarding the FunctionNotSupportedIndication which currently includes the erroneous Request, Response or Command PDU. It might not be possible to find the length of a PDU that is from a version that the receiving terminal does not support. Also it was thought that it might be necessary for the receiver to bit-shift the erroneous PDU in order to form the IndicationPDU. However, it was demonstrated by Mike Nilsson that it is always possible to find the length of the PDU and also that since the encoding of the unknown PDU is an open type it is octet aligned and can be copied back without the burden of bit shifting.

5) A question regarding the actions to be taken in the event of timer expiry arose and whether these actions should be specified in H.324 or H.245. As far as default values are concerned, it was agreed in Boston that these should not be specified in H.245.

6) The semantics of Open bi-directional channel request were found to be confusing, and incorrect in places. Modification of the syntax of Open bi-directional channel request was proposed to overcome the problem of having to specify how OpenLogicalChannel is used in this case and which parameters are arbitrary.

7) Figure 8 is missing a 'set timer Tn' task box.

8) Agreement to add note saying that multiplexCapability is not optional in the first TerminalCapabilitySet sent.

9) Comment that defining DataType in OpenLogicalChannel in terms of capabilities is unsound. dataType in OpenLogicalChannel has problems for particular types - those with sequences - such as MPEG audio and data, which have sequences of booleans. Alternative data type syntax was suggested by Mike Nilsson (see Annex 1), which is simple and avoids the need to send all the capability details again when opening logical channels, which would be unnecessary if we say that channels should only be opened for data that the far-end terminal is known to be capable of receiving.

* Contact	Bill Welsh	Mike Nilsson
	Tel: +44 1473 643810	Tel: +44 1473 645413
	Fax: +44 1473 643791	Fax: +44 1473 643791
	e-mail: welsh_w_j@bt-web.bt.co.uk	e-mail: nilsson_m_e@bt-web.bt.co.uk

10) Suggestion that a method be provided in a SendTerminalCapabilitySetCommandPDU to ask for all capabilityTableEntries and all capabilityDescriptors. SendTerminalCapabilitySetCommandPDU does not have an associated mechanism by which an erroneous command can be rejected. Mike Nilsson suggested the text and syntax in Annex 2.

11) Possible problem with the the procedure specified in the third part of figure 8. If while awaiting response to one capability transfer, a second capability transfer is made, the timer will be reset and sequence number incremented. So if an ack or reject is eventually received corresponding to the first transfer it will be ignored. Mike Nilsson offers two solutions:

1. Define a separate CESE for each capabilityDescriptorNumber and multiplexCapability. This would make the implementation very complex.
2. Define a capability exchange protocol which is slightly different to the others. In the outgoing CESE, when a TRANSFER request is received in the AWAITING RESPONSE state, an error message could be returned to the CESE user, no PDU sent and no state changed. In the incoming CESE when a SEND PDU is received in the AWAITING RESPONSE state, a REJECT indication is sent to the incoming CESE user, a REJECT PDU is sent for the received SEND PDU (and possibly for the previous SEND PDU) and the state is changed to IDLE.

12) Suggestion to add means to exchange LAN-type addresses between terminals which might be useful to allow routing of data over the internet while audio and video goes the PSTN or ISDN.

13) Some questions were raised about whether H.245 should mandate a capability exchange and whether it should not state that other master/slave determinations might be used by a particular system recommendation. It was pointed out that Mike Nilsson that since H.245 is just a collection of generic procedures and should not mandate any procedure. This should be done in the appropriate system recommendation. Also if a system recommendation wished to use an alternative master/slave determination procedure to the one defined in H.245, that is its prerogative.

14) Comment that H.245 does not specify which bit of an encoding will be the first bit to be transmitted

15) Request made for 'dataTypeALCombinationNotSupported' to be added to causes in OpenLogicalChannelReject.

16) For reverseLogicalChannelParameters, in both ASN.1 comment and in the semantics, it is stated that H.222 parameters are not present, but the syntax does not provide a means of leaving them out. In reverseLogicalChannelParameters, should portNumber be set?

17) There is dispute concerning the current text in which a subset of commands corresponding to H.230 commands has been defined and described as opposed to having all the commands in H.230 defined by reference. Arguments for both sides of the argument have been presented at great length but without any final conclusions being made.

18) There was much discussion of Bidirectional Logical Channel signalling. Stuart Dunstan proposed modifications to this in (LBC-95-265, AVC-817), and Dave Lindbergh and Mike Nilsson proposed simplifying the process, which would also have the benefit of reducing the number of round trip delays incurred. Mike Nilsson also proposed alternative syntax, as given in Annex 3.

19) Stuart Dunstan in (LBC-95-266, AVC-818) proposed an informative appendix on H.245 procedures in response to the people who had experienced difficulty understanding section 8 of H.245.

20) Stuart Dunstan in (LBC-95-248, AVC-msd) proposed modifications to the master slave determination procedures in response to a request to clarify the procedures for Master-Slave Determination so that it is made clear that a terminal shall not change its status determination number on timer expiry while waiting for an MSDACK PDU. Mike Nilsson has written a contribution supporting Stuart's document with some modifications and gives some possible text and syntax, as given in Annex 4.

21) Mr. Okubo questioned whether there is a need for separate MPEG audio request modes '2/0 stereo' and '1/0 + 1/0 monoaural' in place of the existing 'twoChannels' mode.

An addition to the above, Chris Hansen suggested that a group be started up to discuss the implementation issues related to getting interworking H.324 terminals using H.245 control PDUs and procedures.

- END -

Annex 1
Alternative Syntax for DataType

```
DataType                        ::=CHOICE
{
    nonStandard                 NonStandardParameter,
    nullData                    NULL,
    videoData                   VideoDataType,
    audioData                   AudioDataType,
    data                        DataApplicationType,
    encryptionData              EncryptionMode,
    ...
}

VideoDataType                   ::=CHOICE
{
    nonStandard                 NonStandardParameter ,
    h261                        NULL,
    h262                        NULL,
    h263                        NULL,
    ...
}

AudioDataType                   ::=CHOICE
{
    nonStandard                 NonStandardParameter,
    g711Alaw                    NULL,
    g711Ulaw                    NULL,
    g722                        NULL,
    g723                        NULL,
    g728                        NULL,
    g729                        NULL,
    g-dsvd                      NULL,
    is11172                     NULL,
    is13818                     NULL,
     ...
}

DataApplicationType             ::=CHOICE
{
    nonStandard                 NonStandardParameter,
    t120                        DataModeProtocol,
    dsm-cc                      DataModeProtocol,
    userData                    DataModeProtocol,
    t84                         DataModeProtocol,
    t434                        DataModeProtocol,
    h224                        DataModeProtocol,
    nlpid                       SEQUENCE
    {
        nlpidProtocol           DataModeProtocol,
        nlpidData               OCTET STRING
    },
    ...
}
```

Annex 2
Alternative Text and Syntax for SendTerminalCapabilitySet

SendTerminalCapabilitySet was originally included as a means to recover from error conditions, or some 'cause for uncertainty', and was originally a simple request for the far end to send its capability set again.

Since then, the capability messages have got more complex, and this command has tried to follow the changes in the definition of the capabilities.

There is a minor problem with the current definition of SendTerminalCapabilitySet as Hardish points out, but I think this can be fixed by adding some words to section 7.8.1, such as the following.

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.

It may also be beneficial to retain the original meaning of this message as well: allow a terminal to request the far-end to send its capability set again. It is quite likely that if some 'cause for uncertainty' has occurred, that the receiver may not know what capabilityTableEntryNumbers and capabilityDescriptorNumbers have been sent and which ones to request retransmission of. The following syntax illustrates this.

```
SendTerminalCapabilitySet               ::=CHOICE
{
    specificRequest                     SEQUENCE
    {
        multiplexCapability             BOOLEAN,
        capabilityTableEntryNumbers     SET SIZE (1..256) OF CapabilityTableEntryNumber OPTIONAL,
        capabilityDescriptorNumbers     SET SIZE (1..256) OF CapabilityDescriptorNumber OPTIONAL,
        ...
    },
    genericRequest                      NULL,
    ...
}
```

Annex 3
Alternative Text and Syntax for Logical Channel Signalling

```
================================================================================
-- 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 bi-directional channel request.

OpenLogicalChannel                  ::=SEQUENCE
{
    forwardLogicalChannelNumber     LogicalChannelNumber,

    forwardLogicalChannelParameters SEQUENCE
    {
        portNumber                  INTEGER (0..65535),
        dataType                    DataType,
        multiplexParameters         CHOICE
        {
            h222LogicalChannelParameters H222LogicalChannelParameters,
            h223LogicalChannelParameters H223LogicalChannelParameters,
            ...
        },
        ...
    },

    -- Used to specify the reverse channel for bi-directional open request

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

    ...
}

LogicalChannelNumber                ::=INTEGER (1..65535)

DataType                            Defined as appropriate - see proposal below

H222LogicalChannelParameters        ::=SEQUENCE
{
    virtualChannelID                INTEGER (0..65535),
    subChannelID                    INTEGER (0..8191),
    pcr-pid                         INTEGER (0..8191) OPTIONAL,
    programDescriptors              OCTET STRING OPTIONAL,
    streamDescriptors               OCTET STRING OPTIONAL,
    ...
}

H223LogicalChannelParameters        ::=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,
    ...
}


OpenLogicalChannelAck                   ::=SEQUENCE
{
    forwardLogicalChannelNumber         LogicalChannelNumber,

    reverseLogicalChannelParameters     SEQUENCE
    {
        reverseLogicalChannelNumber     LogicalChannelNumber,
        portNumber                      INTEGER (0..65535),
        multiplexParameters             CHOICE
        {
            h222LogicalChannelParameters H222LogicalChannelParameters,
            -- H.223 parameters are never present in reverse direction
            ...
        } OPTIONAL,                     -- Not present for H.223
        ...

    } OPTIONAL,                         -- Not present for uni-directional channel request

    ...
}

OpenLogicalChannelReject                ::=SEQUENCE
{
    forwardLogicalChannelNumber         LogicalChannelNumber,
    cause                               CHOICE
    {
        unspecified                     NULL,
        dataTypeNotSupported            NULL,
        dataTypeNotAvailable            NULL,
        unknownDataType                 NULL,
        dataTypeALCombinationNotSupported                       NULL,
        ...
    },
    ...
}

CloseLogicalChannel                     ::=SEQUENCE
{
    forwardLogicalChannelNumber         LogicalChannelNumber,
    source                              CHOICE
    {
        user                            NULL,
        lcse                            NULL
```

```
    },
    ...
}

CloseLogicalChannelAck                  ::=SEQUENCE
{
    forwardLogicalChannelNumber         LogicalChannelNumber,
    ...
}

-- OpenBiDirectionalChannelRequest is no longer needed

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

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

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

RequestChannelCloseRelease              ::=SEQUENCE
{
    forwardLogicalChannelNumber         LogicalChannelNumber,
    ...
}
```

Annex 4
Alternative Text and Syntax for Master-Slave Determination

Master slave determination procedures - Introduction
The text in 8.2.1/H.245 could be improved to reflect more accurately the formal procedure. The following is possible text, assuming that the protocol makes automatic retries.

Some procedures, such as bi-directional logical channel signalling, require one of the terminals involved in a communications call to act as a master terminal and the other terminal(s) involved in the call to act as slave terminals. The protocol defined here allows terminals in the call to determine which is the master terminal and which is the slave terminal.

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 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 not initiate procedures that require knowledge of the result until it has received confirmation that the remote terminal also has knowledge of the result. It shall, however, respond to procedures initiated at the remote terminal before it has 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.

Either terminal may initiate the master slave determination process by transmitting a random number, called statusDeterminationNumber. When a terminal receives such a number, it compares it with its own number, and the terminal with the larger number is determined as the master. The terminal that receives the statusDeterminationNumber acknowledges by sending the decision, that is, the status of the terminal that sent the statusDeterminationNumber. The initiating terminal, unless it has already sent an acknowledgement, responds to the receipt of an acknowledgement by sending one itself, to confirm to that remote terminal that it knows the result.

If both terminals have equal statusDeterminationNumbers, the terminal that receives the other's number responds with the rejection message indicating that the numbers were identical. In this case, the initiating terminal(s) shall transmit a new statusDeterminationNumber. This process shall be repeated until the two numbers are different or one or other terminal ends the connection.

Reference to master slave determination procedures in H.324
The text in 7.4/H.324 relating to the master slave determination procedures should be improved. It is preferable for system recommendations to refer to the protocols and primitives of H.245 rather than the messages. The following is possible text, assuming that the protocol makes automatic retries.

The H.245 master slave determination procedure shall be initiated at this time. H.324 terminals shall be capable of operating in both master and slave modes. H.324 terminals shall choose statusDeterminationNumbers in the range 2^{30} to $2^{31}-1$.

NOTE - *as at present.*

After this protocol has confirmed that the far-end terminal knows the result of the master slave determination procedure, and the far-end's capabilities have been received, the H.245 logical channel signalling procedures may be used to open logical channels for various information streams.

Master slave determination syntax

The following is the master slave determination ASN.1 syntax that would be needed for Stuart's revised protocol and the new determination process.

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

MasterSlaveDetermination                ::=SEQUENCE
{
    sequenceNumber                      SequenceNumber,
    terminalTypeNumber                  INTEGER (0..255),
    statusDeterminationNumber           INTEGER (0..16777215),
    ...
}

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

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

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

END