Telecommunication Standardisation Sector
Study Group 15
Experts Group for Video Coding and Systems in
ATM and Other Network Environments

Document AVC-881

8 Jan 1996

SOURCE : Stuart Dunstan, Siemens Ltd
TITLE : Some editorial comments on H.245
PURPOSE : Proposal

Introduction

H.245 was decided according to Resolution 1 at the Study Group 15 meeting in November. The work done on H.245 leading up to that meeting appears to be quite sound. In particular the section on bi-directional logical channel procedures has captured the required logic using one additional message, and one additional state with an associated timer, but without further complicating the procedures. The logic of the maintenance loop procedures is quite straight forward.

This document lists some editorial comments on H.245 (h245ncm4.ww6). While it may or may not be possible to incorporate these comments in a final version of H.245, I hope that these comments will at least be of use to other interested H.245 users.

1. Section 8.5, B-LCSE

1.1. reverseLogicalChannelNumber

A bi-directional logical channel is specified by one number, though it has forward and reverse transmission channels and parameters. This number is the forwardLogicalChannelNumber which discriminates between all logical channel signalling messages.

reverseLogicalChannelNumber is transparent to the B-LCSE. It is assigned by the user at the incoming B-LCSE and carried to the outgoing B-LCSE in the field of the **OpenLogicalChannelAck.reverseLogicalChannelParameters.reverseLogicalChannelNumber** message. It is neither set nor read by the B-LCSE.

The coding of reverseLogicalChannelParameters in the OpenLogicalChannel and OpenLogicalChannelAck messages differ, as shown in the following table. The associated primitive parameters are also shown.

message	field	fields within	associated parameter
OpenLogicalChannel	reverseLogicalChannelParameters	dataType multiplexParameters	REVERSE_PARAM
OpenLogicalChannelAck	reverseLogicalChannelParameters	reverseLogicalChannelNumber portNumber multiplexParameters	REVERSE_DATA

Hence reverseLogicalChannelNumber does not need to be stated explicitly in section 8.5: it is simply a field embedded within REVERSE_DATA.

The following changes are required to section 8.11 to reflect the above.

Note that reverseLogicalChannelNumber serves no purpose in H.310; the return channel is identified by the PID value contained in the mux parameters. reverseLogicalChannelNumber is coded for applications which have no such mux layer identification.

1.2. Section 8.5.2.3 c), REVERSE_DATA

Correct definition of REVERSE_DATA as follows,

c) The REVERSE_DATA parameter specifies parameters associated with the reverse logical 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 OpenLogicalChannelAck message and is carried transparently to the peer B-LCSE user.

1.3. Section 8.5.4.3, REVERSE_DATA

Add entry for ESTABLISH.confirm in Table 33, as shown below.

TABLE 33/H.245

Default primitive parameter values

primitive	parameter	default value [1]
ESTABLISH.indication	FORWARD_PARAM	OpenLogicalChannel.forwardLogicalChannelParameters
	REVERSE_PARAM	OpenLogicalChannel.reverseLogicalChannelParameters
ESTABLISH.confirm	REVERSE_DATA	OpenLogicalChannelAck.reverseLogicalChannelParameters
RELEASE.indication	SOURCE	CloseLogicalChannel.source
	CAUSE	null

1.4. Section 8.5.4.5, Figure 19 (ii)

OpenLogicalChannelConfirm arriving in the AWAITING ESTABLISHMENT state at the incoming B-LCSE is correctly identified as an error condition. However what action should be taken? Currently CloseLogicalChannelAck is sent. This is not in line with correct approach of keeping incoming side passive. The following is recommended,

- remove CloseLogicalChannelAck

Clean up is thus a user issue i.e. use close logical channel signalling entity to request a closure, as occurs for timer expiry in the AWAITING CONFIRMATION state.

2. Section 8.7, Multiplex Table Signalling Entity

The correct field name in which the parameter MUX-DESCRIPTOR is carried in the MultiplexEntrySend message, is **MultiplexEntrySend.multiplexEntryDescriptor.elementList**. The field name currently used varies throughout section 8.7. (Above naming agrees with that adopted for the MultiplexEntrySendReject message cause field in Table 42).

Section 6 says that MultiplexEntryDescriptor is made up of two fields; multiplexTableEntryNumber and elementList. The former is set by the state variable out_ENUM, while the later is set by the MUX-DESCRIPTOR parameter. The procedures need to recognise these two fields.

Hence the following changes;

2.1. Section 8.7.2.3 Parameter definition

2nd sentence a) should read,

" ... This parameter is mapped to the elementList field of the MultiplexEntrySend message ...".

Note missing capital "M" in "MultiplexEntrySend".

2.2. Section 8.7.3.1 Table 42

For the MultiplexEntrySend message in Table 42 change, "multiplexTableEntryNumber" to "multiplexEntryDescriptor.multiplexTableEntryNumber". Change "MultiplexElement" to "multiplexEntryDescriptor.elementList".

2.3. Section 8.7.4.2, Table 43

Change "MultiplexEntrySend.MultiplexEntryDescriptor" to "MultiplexEntrySend.multiplexEntryDescriptor.elementList"

2.4. Section 8.7.4.3, Table 44

For the MultiplexEntrySend message change, "multiplexTableEntryNumber" to "multiplexEntryDescriptor.multiplexTableEntryNumber". Change "MultiplexElement" to "multiplexEntryDescriptor.elementList".

2.5. Section 8.7.4.3, Table 44

The correct reference to the cause parameter of the MultiplexEntrySendReject message is "MultiplexEntrySendReject.rejectionDescriptions.cause". In Table 44 change "cause" to "rejectionDescriptions.cause". Table 42 is correct.

3. Section 8.8, Request Multiplex Entry

3.1. Section 8.8.4.2, Table 48

The correct reference to the cause parameter of the MultiplexEntrySendReject message is "RequestMultiplexEntryReject.rejectionDescriptions.cause". In Table 48 change "cause" to "rejectionDescriptions.cause". Table 46 is correct.

3.2. Section 6, RequestMultiplexEntryReject message

In the syntax for the RequestMultiplexEntryReject message there is an entryNumbers field and multiplexTableEntryNumber; there should be only one occurrence of this field type.

The simplest thing to do is to remove the entryNumbers field (though removal of the RequestMultiplexEntryRejectionDescriptions structure is cleaner).

3.3. Section 8.8.3.1, Table 46

The field "multiplexTableEntryNumber" in Table 46 is inconsistent with the syntax in section 6. Note that the term "MultiplexTableEntryNumber" (capital M) is a field type, and not a field name (I think). The correct name is either of "entryNumbers" or "multiplexTableEntryNumber". There is a slight problem in semantics with "entryNumbers"; it should probably be just "entryNumber".

It is recommended that,

- in section 6, for the request multiplex entry messages, the "entryNumbers" field name be changed to "multiplexTableEntryNumber".

- in section 8.8, all references to this field are corrected. This may effect,
  - Table 46 (currently MultiplexTableEntryNumber)
  - 8.8.3.2 (currently multiplexTableEntryNumber)
  - Table 48 (currently multiplexTableEntryNumber)

In all cases, for the RequestMultiplexEntryReject message the correct reference is **RequestMultiplexEntryReject.rejectionDescriptions.multiplexTableEntryNumber**; but see above point.

3.4. Section 8.8.3.1, Table 46

The RequestMultiplexEntryRelease message also requires the "entryNumber" field in Table 46. Table 48 is correct with respect to this.

3.5. Section 8.8.4.3, SDLs

In Figure 33 (ii), for the REJECT.indication primitive in the RequestMultiplexEntryReject message part, there should be the text caption "CAUSE = RequestMultiplexEntryReject.cause".

While NULL may well be the only current valid entry of the cause field, the SDL should be generalised, so that new additions of the cause field values will not change the SDLs.

4. Section 8.11, Maintenance Loop Signalling Entity

4.1. Section 6, syntax

The syntax for MaintenanceLoopOffCommand turns off all loops simultaneously. It may have been desirable to allow maintenance loops to be turned off individually, though there may have been good reason for not doing this.

With respect to setting of the MaintenanceLoopRequest.type field, the MaintenanceLoopRequest has the following syntax,

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

Note that both,

- the LOOP_TYPE parameter, and
- the logical channel number

are required to set the type field. Section 8.11 should reflect this.

However the MaintenanceLoopAck and the MaintenanceLoopRelease messages use the same syntax for type, when in fact only logical channel number is required (system loop is logical channel number not present): when a logical channel has been looped, it matters not what the Ack or Release messages say about loop type.

Section 8.11 currently has no means to store the LOOP_TYPE value between when the MaintenanceLoopRequest message is received, and when the MaintenanceLoopAck message is sent in response.

The following solutions are possible in order of preference, most preferred first,

a) code MaintenanceLoopAck and MaintenanceLoopRequest messages as (or something like)

```
MaintenanceLoopAck/Request      ::=SEQUENCE
{
    logicalChannelNumber        NULL    (system loop)
    logicalChannelNumber        LogicalChannelNumber
        ...
    },
    ...
}
```

b) do nothing: the LOOP_TYPE information in the type field is anyway not required in the Ack/Release messages, and can be an arbitrary value.

c) include an incoming side state variable that stores the value of LOOP_TYPE when the MaintenanceLoopRequest message is received. This state variable assists in setting the value of the type field in the MaintenanceLoopAck message.

4.2. 8.11.2.3 Parameter definition

Repair as follows,

a) The LOOP_TYPE parameter specifies the type of 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 to the peer MLSE user.

4.3. section 8.11.3.2 Note

Could probably remove note at end of this section, with the clarifications listed here.

4.4. section 8.11.4.3 Note 2

The type field is set by the local state variable and the LOOP_TYPE parameter. In Table 60, modify the first entry and append the note 2 as shown,

TABLE 60/H.245

Default message field values

message	field	default value [1]
MaintenanceLoopRequest	type	LOOP.request(LOOP_TYPE) and out_MLN[2]
MaintenanceLoopAck	type	in_MLN
MaintenanceLoopReject	type cause	in_MLN RELEASE.request(CAUSE)
MaintenanceLoopOffCommand	-	-

Notes:

1. A message field shall not be coded, if the corresponding primitive parameter is null i.e. not present.
2. The value of the type field is determined by both the LOOP_TYPE parameter and the logical channel number. See syntax in 6.

- end -