INTERNATIONAL TELECOMMUNICATION UNION

ITU-T J.181

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(06/2004)

SERIES J: CABLE NETWORKS AND TRANSMISSION OF TELEVISION, SOUND PROGRAMME AND OTHER MULTIMEDIA SIGNALS

Digital transmission of television signals

Digital program insertion cueing message for cable television systems

ITU-T Recommendation J.181

ITU-T Recommendation J.181

Digital program insertion cueing message for cable television systems

Summary

This Recommendation supports the splicing of MPEG-2 transport streams for the purpose of digital program insertion, which includes advertisement insertion and insertion of other content types. An in-stream messaging mechanism is defined to signal splicing and insertion opportunities. A technique for carrying notification of upcoming Splice Points in the transport stream is specified.

Source

ITU-T Recommendation J.181 was approved on 29 June 2004 by ITU-T Study Group 9 (2001-2004) under the ITU-T Recommendation A.8 procedure.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. 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 Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

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

NOTE

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

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

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

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

CONTENTS

ITU-T Recommendation J.181

Digital program insertion cueing message for cable television systems

1 Scope

This Recommendation supports the splicing of MPEG-2 streams for the purpose of digital program insertion, which includes advertisement insertion and insertion of other content types. An in-stream messaging mechanism is defined to signal splicing and insertion opportunities and it is not intended to ensure seamless splicing. As such, this Recommendation does not specify the splicing method used or constraints applied to the streams being spliced, nor does it address constraints placed on splicing devices. This Recommendation also supports accurate signalling of events in the stream.

A fully compliant MPEG-2 transport stream (either Multi-Program Transport Stream or Single Program Transport Stream) is assumed. No further constraints beyond the inclusion of the defined cueing messages are placed upon the stream.

This Recommendation specifies a technique for carrying notification of upcoming Splice Points and other timing information in the transport stream. A splice information table is defined for notifying downstream devices of splice events, such as a network break or return from a network break. The splice information table, which pertains to a given program, is carried in one or more PID(s) referred to by that program's Program Map Table (PMT). In this way, splice event notification can pass through most transport stream remultiplexers without need for special processing.

2 References

2.1 Normative 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; 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. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

– ITU-T Recommendation H.222.0 (2000) | ISO/IEC 13818-1:2000, *Information technology – Generic coding of moving pictures and associated audio information: Systems*.

– ITU-T Recommendation H.262 (2000) | ISO/IEC 13818-2:2000, *Information technology – Generic coding of moving pictures and associated audio information: Video*.

– ISO/IEC 13818-4:1998, *Information technology – Generic coding of moving pictures and associated audio information – Part 4: Conformance testing, plus Corrigendum 2 (1998)*.

– FIPS PUB 46-3-1999, *Data Encryption Standard (DES)*.

– FIPS PUB 81-1980, *DES Modes of Operation*.

2.2 Informative references

– SMPTE 312M, SMPTE Standard for Television – Splice Points for MPEG-2 Transport Streams.

3 Definition of terms

Throughout this Recommendation, the terms below have specific meanings. Because some of the terms that are defined in ISO/IEC 13818 have very specific technical meanings, the reader is referred to the original source for their definition. For terms used in this Recommendation, brief definitions are given below.

3.1 access unit: A coded representation of a presentation unit (see ITU-T Rec. H.262 | ISO/IEC 13818-2).

3.2 analog cue tone: In an analog system, a signal that is usually either a sequence of DTMF tones or a contact closure that denotes to ad insertion equipment that an advertisement avail is about to begin or end.

3.3 avail: Time space provided to cable operators by cable programming services during a program for use by the CATV operator; the time is usually sold to local advertisers or used for channel self promotion.

3.4 break: Avail or an actual insertion in progress.

3.5 component splice mode: A mode of the cueing message whereby the program_splice_flag is set to "0" and indicates that each PID/component that is intended to be spliced will be listed separately by the syntax that follows. Components not listed in the message are not be spliced.

3.6 cueing message: See "message".

3.7 event: A splice event or a viewing event.

3.8 in point: A point in the stream, suitable for entry, that lies on an elementary presentation unit boundary. An In Point is actually between two presentation units rather than being a presentation unit itself.

3.9 in-stream device: A device that receives the transport stream directly and is able to derive timing information directly from the transport stream.

3.10 message: In the context of this Recommendation, a message is the contents of any splice_info_section.

3.11 Multi Program Transport Stream (MPTS): A transport stream with multiple programs.

3.12 out-of-stream device: A device that receives the cue message from an in-stream device over a separate connection from the transport stream. An out-of-stream device does not receive or pass the transport stream directly.

3.13 out point: A point in the stream, suitable for exit, that lies on an elementary presentation unit boundary. An Out Point is actually between two presentation units rather than being a presentation unit itself.

3.14 payload_unit_start_indicator: A bit in the transport packet header that signals, among other things, that a section begins in the payload that follows (see ITU-T Rec. H.222.0 | ISO/IEC 13818-1).

3.15 Packet Identifier (PID): A unique 13-bit value used to identify the type of data stored in the packet payload (see ITU-T Rec. H.222.0 | ISO/IEC 13818-1).

3.16 PID stream: All the packets with the same PID within a transport stream.

3.17 pointer_field: The first byte of a transport packet payload, required when a section begins in that packet (see ITU-T Rec. H.222.0 | ISO/IEC 13818-1).

3.18 presentation time: The time that a presentation unit is presented in the system target decoder (see ITU-T Rec. H.222.0 | ISO/IEC 13818-1).

3.19 presentation unit: A decoded Audio Access Unit or a decoded picture (see ITU-T Rec. H.262 | ISO/IEC 13818-2).

3.20 program: A collection of video, audio, and data PID streams that share a common program number within an MPTS (see ITU-T Rec. H.222.0 | ISO/IEC 13818-1).

3.21 program in point: A group of PID stream In Points that correspond in presentation time.

3.22 program out point: A group of PID stream Out Points that correspond in presentation time.

3.23 program splice mode: A mode of the cueing message whereby the program_splice_flag is set to "1" and indicates that the message refers to a Program Splice Point and that all PIDs/components of the program are to be spliced.

3.24 program splice point: A Program In Point or a Program Out Point.

3.25 receiving device: A device that receives or interprets sections conforming to this Recommendation. Examples of these devices include splicers, ad servers, segmenters and satellite receivers.

3.26 registration descriptor: Carried in the PMT of a program to indicate that, when signalling splice events, splice_info_sections shall be carried in a PID stream within this program. The presence of the Registration Descriptor signifies a program's compliance with this Recommendation.

3.27 reserved: The term "reserved", when used in the clauses defining the coded bit stream, indicates that the value may be used in the future for extensions to this Recommendation. Unless otherwise specified in this Recommendation, all reserved bits shall be set to "1" and this field shall be ignored by receiving equipment.

3.28 Single Program Transport Stream (SPTS): A transport stream containing a single MPEG program.

3.29 splice event: An opportunity to splice one or more PID streams.

3.30 splice immediate mode: A mode of the cueing message whereby the splicing device shall choose the nearest opportunity in the stream, relative to the splice_info_table, to splice. When not in this mode, the message gives a "pts_time" that, when modified by pts_adjustment, gives a presentation time, for the intended splicing moment.

3.31 splice point: A point in a PID stream that is either an Out Point or an In Point.

3.32 viewing event: A television programme or a span of compressed material within a service; as opposed to a splice event, which is a point in time.

4 Abbreviations

This Recommendation uses the following abbreviations:

ATSC Advanced Television Systems Committee

bslbf Bit string, left bit first, where left is the order in which bit strings are written

DVB Digital Video Broadcast

MPTS Multi Program Transport Stream

PMT Program Map Table (see ITU-T Rec. H.222.0 | ISO/IEC 13818-1)

PTS Presentation Time Stamp (see ITU-T Rec. H.222.0 | ISO/IEC 13818-1)

rpchof Remainder polynomial coefficients, highest order first

SPTS Single Program Transport Stream

STC System Time Clock

uimsbf Unsigned integer, most significant bit first

5 Introduction

5.1 Splice points (Informative)

To enable the splicing of compressed bit streams, this Recommendation defines Splice Points. Splice Points in an MPEG-2 transport stream provide opportunities to switch elementary streams from one source to another. They indicate a place to switch or a place in the bit stream where a switch can be made. Splicing at such splice points may or may not result in good visual and audio quality. That is determined by the performance of the splicing device.

Transport streams are created by multiplexing PID streams. In this Recommendation, two types of Splice Points for PID streams are defined: Out Points and In Points. In Points are places in the bit streams where it is acceptable to enter, from a splicing standpoint. Out Points are places where it is acceptable to exit the bit stream. The grouping of In Points of individual PID streams into Program In Points in order to enable the switching of entire programs (video with audio) is defined. Program Out Points for exiting a program are also defined.

Out Points and In Points are imaginary points in the bit stream located between two elementary stream presentation units. Out Points and In Points are not necessarily transport packet aligned and are not necessarily PES packet aligned. An Out Point and an In Point may be co-located; that is, a single presentation unit boundary may serve as both a safe place to leave a bit stream and a safe place to enter it.

The output of a simple switching operation will contain access unit data from one stream up until its Out Point followed by data from another stream starting with the first access unit following an In Point. More complex splicing operations may exist whereby data prior to an Out Point or data after an In Point are modified by a splicing device. Splicing devices may also insert data between one stream's Out Point and the other stream's In Point. The behaviour of splicing devices will not be specified or constrained in any way by this Recommendation.

5.2 Program splice points (Informative)

Program In Points and Program Out Points are sets of PID stream In Points or Out Points that correspond in presentation time.

Although Splice Points in a Program Splice Point correspond in presentation time, they do not usually appear near each other in the transport stream. Because compressed video takes much longer to decode than audio, the audio Splice Points may lag the video Splice Points by as much as hundreds of milliseconds and by an amount that can vary during a program.

This Recommendation defines two ways of signalling which splice points within a program are to be spliced. A program_splice_flag, when true, denotes that the Program Splice Mode is active and that all PIDs of a program may be spliced (the splice information table PID is an exception; splicing or passage of these messages is beyond the scope of this Recommendation). A program_splice_flag, when false, indicates that the Component Splice Mode is active and that the message will specify unambiguously which PIDs are to be spliced and may give a unique splice time for each. This is required to direct the splicing device to splice or not to splice various unspecified data types as well as video and audio.

While this Recommendation allows for a unique splice time to be given for each component of a program, it is expected that most Component Splice Mode messages will utilize one splice time (a default splice time) for all components as described in clause 7. The facility for optionally

specifying a separate splice time for each component is intended to be used when one or more components differ significantly in their start or stop time relative to other components within the same message. An example would be a downloaded applet that must arrive at a set-top box several seconds prior to an advertisement.

5.3 Splice events (Informative)

This Recommendation provides a method for in-band signalling of splice events using splice commands to downstream splicing equipment. Signalling a splice event identifies which Splice Point within a stream to use for a splice. A splicing device may choose to act or not act upon a signalled event (a signalled event should be interpreted as an opportunity to splice; not a command). A splice information table carries the notice of splice event opportunities. Each signalled splice event is analogous to an analog cue tone. The splice information table incorporates the functionality of cue tones and extends it to enable the scheduling of splice events in advance.

This Recommendation establishes that the splice information table is carried on a per-program basis in one or more PID stream(s) with a designated stream_type. The program's splice information PID(s) are designated in the program's program map table (PMT). In this way, the splice information table is switched with the program as it goes through remultiplexing operations. A common stream_type identifies all PID streams that carry splice information tables. Remultiplexers or splicers may use this stream_type field to drop splice information prior to sending the transport stream to the end-user device.

The cue injection equipment may send messages at intervals that do not indicate a splice point to be used as heartbeat messages which help insure the proper operation of the system. This could be performed by periodically issuing splice_null() messages or by sending encrypted splice_insert messages generated with a key that is not distributed. Since cues are currently sent twice per hour on a typical network, an average interval of 5 minutes would be a reasonable interval. If a message was not received in a 10-minute interval, a receiving device could alarm an operator to a possible system malfunction (such behaviour would be implementer dependent).

5.4 PID selection

5.4.1 PID selection (Normative)

Splice information can be carried in multiple PIDs. The maximum number of PIDs that can carry splice information shall not exceed 8. These PIDs can be either in the clear (where the transport scrambling_control bits are set to "00") or scrambled by a CA system. Each cue message PID may include the cue_identifier_descriptor defined in 6.2 to describe the splice commands included in the PID. When multiple PIDs are used to carry splice information, the first cue message PID in the program map table shall only contain the splice command types 0x00 (splice_null), 0x04 (splice_schedule) and 0x05 (splice_insert). In addition, the splice_event_id shall be unique in all splice information PIDs within the program.

5.4.2 PID selection (Informative)

While the use of multiple cue message PIDs is an allowed practice, it should be noted that not all equipment may respond in the same manner to a stream that contains multiple cue message PIDs. Some equipment may limit the number of PIDs that the equipment can pass or receive. If a system utilizes multiple PIDs through various devices with the intention of reaching the set-top, it is suggested that thorough end-end testing be performed.

In many systems, the delivery of PIDs that carry splice information beyond the ad insertion equipment in the head-end is not desired. In these systems, the splicing or multiplexing device will drop any or all of these messages (PIDs) so they will not be delivered to the set-top. In other systems it may selectively pass certain PIDs to the set-top to enable set-top functionality. A third possibility is that the splicing or multiplexing device will aggregate the multiple PIDs that carry

splice information into a single PID to handle downstream, set-top, issues with multiple PIDs. The action of ignoring or passing the message is recommended to be a user provisioned item, with a suitable default behaviour chosen by the implementer.

The default operation, if a splicing or multiplexing device receives a PID based on this Recommendation with the scrambling bits set in the header, should be to drop that PID and not pass it through to the output. This ideally should be a user provisioned operation, as in some instances this PID may be descrambled by a downstream device.

The delivery of messages outside of the receive location to the customer may be based on business agreements. An example would be that one programmer wants the cue messages passed to set-tops to enable a targeted advertising method while a different programmer insists that the messages be dropped to insure that a commercial killer may not utilize the messages.

When multiple splicing PIDs are identified in the PMT, the splicing device should process all of these PIDs. If the cue_identifier_descriptor is utilized, the splicing or multiplexing device may use that information to be more selective of the PIDs on which it will act.

Some possible reasons for utilizing multiple PIDs for this message include selective delivery of cue messages for different tiers of advertising or for separating cue messages from segmentation messages. While one possible method of handling these issues is to use the encryption methods built into this Recommendation, many delivery mechanisms can support conditional delivery by PID in a secure fashion. The delivery equipment (Satellite transmitter/receiver, remultiplexer) may PID filter the stream to only allow one or a small number of the PIDs to be passed in-stream. This method may be used to create multiple programs in the feed based on entitlement. The decision to use one or more PIDs will be based on the security required and the CA hardware available on the system.

5.5 Message flow (Informative)

The messages described in this Recommendation can originate from multiple sources. They are designed to be sent in-stream to downstream devices. The downstream devices may act on the messages or send them to a device that is not in-stream to act upon them. An example would be a splicer communicating via SCTE 30 protocol to an ad server. The in-stream devices could pass the messages to the next device in the transmission chain, or they could, optionally, drop the messages. Implementers are urged to make these decisions user provisioned, rather than arbitrarily hard-coded.

Any device that restamps pcr/pts/dts and that passes these cue messages to a downstream device should modify the pts_time field or the pts_adjustment field in the message in all PIDs conforming to this Recommendation. Modifying the pts_adjustment field is preferred because the restamping device will not have to be knowledgeable of the pts_time field that may occur in multiple commands (and possibly in future commands).

The bandwidth_reservation() message is intended as a message used on a closed path from a satellite origination system (encoder) to a receiver. It is also intended that this message will be dropped (replaced by a NULL packet) by the receiver, but this is not required. Should this message reach an in-stream device (e.g., a splicer) the message should not be forwarded to an out-of-stream device (e.g., Ad Server) and can either be ignored or passed by an in-stream device. The action of ignoring or passing the message is recommended to be a user provisioned item, with a suitable default behaviour chosen by the implementer.

6 PMT descriptors

6.1 Registration descriptor

The registration descriptor (ITU-T Rec. H.222.0 | ISO/IEC 13818-1, Table 2-45 – Registration Descriptor, clause 2.6.8) is defined to identify unambiguously the programs that comply with this

Recommendation. The registration descriptor shall be carried in the program_info loop of the PMT for each program that complies with this Recommendation. It must reside in all PMTs of all complying programs within a multiplex. The presence of the registration descriptor also indicates that, when signalling splice events, splice_info_sections shall be carried in one or more PID stream(s) within this program.

Presence of this registration descriptor in the PMT signals the following:

1) The program elements do not include the splice information table defined by SMPTE 312M.

2) The only descriptors that can be present in the ES_descriptor_loop of the PMT for the PID(s) that carry the splice_information_table are those that are defined in this Recommendation or user private descriptors.

Note that this descriptor applies to the indicated program and not to the entire multiplex. The content of the registration descriptor is specified in Table 6-1 and below:

Table 6-1/J.181 – registration_descriptor()

Syntax	Bits	Mnemonic
registration_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
SCTE_splice_format_identifier	**32**	**uimsbf**
}		

6.1.1 Semantic definition of fields in registration descriptor

descriptor_tag: The descriptor_tag is an 8-bit field that identifies each descriptor. For registration purposes, this field shall be set to 0x05.

descriptor_length: The descriptor_length is an 8-bit field specifying the number of bytes of the descriptor immediately following descriptor_length field. For this registration descriptor, descriptor_length shall be set to 0x04.

SCTE_splice_format_identifier: SCTE has assigned a value of 0x43554549 (ASCII "CUEI") to this 4-byte field to identify the program (within a multiplex) in which it is carried as complying with this Recommendation.

6.2 Cue identifier descriptor

The cue_identifier_descriptor may be used in the PMT to label PIDs that carry splice commands so that they can be differentiated as to the type or level of splice commands they carry. The cue_identifier_descriptor, when present, shall be located in the elementary descriptor loop. If the cue_identifier_descriptor is not utilized, the stream may carry any valid command in this Recommendation. See Table 6-2.

Table 6-2/J.181 – cue_identifier_descriptor()

Syntax	Bits	Mnemonic
cue_identifier_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
cue_stream_type	**8**	**uimsbf**
}		

6.2.1 Semantic definition of fields in Cue Identifier Descriptor

descriptor_tag: The descriptor_tag is an 8-bit field that identifies each descriptor. For cue_identifier_descriptor, this field shall be set to 0x8A.

descriptor_length: The descriptor_length is an 8-bit field specifying the number of bytes of the descriptor immediately following descriptor_length field. For this descriptor, descriptor_length shall be set to 0x01.

cue_stream_type: This 8-bit field is defined in Table 6-3.

Table 6-3/J.181 – cue_stream_type values

cue_stream_type	PID usage
0x00	splice_insert, splice_null, splice_schedule
0x01	All Commands
0x02	Segmentation
0x03	Tiered Splicing
0x04	Tiered Segmentation
0x05-0x7F	Reserved
0x80-0xFF	User Defined

6.2.2 Description of cue_stream_type usage

0x00 – splice_insert, splice_null, splice_schedule: Only these cue messages are allowed in this PID stream. There shall be a maximum of one PID identified with this cue_stream_type. If this PID exists, it shall be the first stream complying with this Recommendation in the PMT elementary stream loop.

0x01 – All Commands: Default if this descriptor is not present. All messages can be used in this PID.

0x02 – Segmentation: This PID carries the time_signal command and the segmentation descriptor. It may also carry all other commands if needed for the application, but the primary purpose is to transmit content segmentation information.

0x03 – Tiered Splicing: Tiered Splicing refers to an insertion system where the operator provides different inserted program possibilities in a given avail for different customers. The physical and logical implementation may be done in several different manners, some of them are outside the scope of this Recommendation.

0x04 – Tiered Segmentation: Tiered Segmentation refers to a system where the operator provides different program segmentation possibilities for different customers. The physical and logical implementation may be done in several different manners, some of them are outside the scope of this Recommendation.

0x05-0x7F: Reserved for future extensions to this Recommendation.

0x80-0xFF: User-defined range.

6.3 Stream identifier descriptor

The stream identifier descriptor may be used in the PMT to label component streams of a service so that they can be differentiated. The stream identifier descriptor shall be located in the elementary descriptor loop following the relevant ES_info_length field. The stream identifier descriptor shall be used if either the program_splice_flag or the program_segmentation_flag is zero. If stream identifier descriptors are used, a stream identifier descriptor shall be present in each occurrence of

the elementary stream loop within the PMT and shall have a unique component tag within the given program. See Table 6-4.

Table 6-4/J.181 – stream_identifier_descriptor()

Syntax	Bits	Mnemonic
stream_identifier_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
component_tag	**8**	**uimsbf**
}		

6.3.1 Semantic definition of fields in stream identifier descriptor

descriptor_tag: The descriptor_tag is an 8-bit field that identifies each descriptor. For stream_identifier_descriptor, this field shall be set to 0x52.

descriptor_length: The descriptor_length is an 8-bit field specifying the number of bytes of the descriptor immediately following descriptor_length field. For this descriptor, descriptor_length shall be set to 0x01.

component_tag: This 8-bit field identifies the component stream for associating it with a description given in a component descriptor. Within a program map section, each stream identifier descriptor shall have a different value for this field.

7 Splice information table

7.1 Overview

The splice information table provides command and control information to the splicer. It notifies the splicer of splice events in advance of those events. It is designed to accommodate ad insertion in network feeds. In this environment, examples of splice events would include:

1) a splice out of a network feed into an ad; or

2) the splice out of an ad to return to the network feed.

The splice information table may be sent multiple times and splice events may be cancelled. Syntax for a splice_info_section is defined to convey the splice information table. The splice_info_section is carried on one or more PID stream(s) with the PID(s) declared in that program's PMT.

A splice event indicates the opportunity to splice one or more elementary streams within a program. Each splice event is uniquely identified with a splice_event_id. Splice events may be communicated in three ways: they may be scheduled ahead of time, a pre-roll warning may be given, or a command may be given to execute the splice event at specified Splice Points. These three types of messages are sent via the splice_info_section. The splice_command_type field specifies the message being sent. Depending on the value of this field, different constraints apply to the remaining syntax.

The following command types are specified: splice_null(), splice_schedule(), splice_insert(), time_signal() and bandwidth_reservation(). If the Receiving Device does not support a command, it can ignore the entire splice_info_section.

The splice_null() command is provided for extensibility. It can be used as a means of providing a heartbeat message to downstream splicing equipment.

The splice_schedule() command is a command that allows a schedule of splice events to be conveyed in advance.

The splice_insert() command shall be sent at least once before each splice point. Packets containing the entirety of the splice_info_table shall always precede the packet that contains the related splice point (i.e., the first packet that contains the first byte of an access unit whose presentation time most closely matches the signalled time in the splice_info_section).

In order to give advance warning of the impending splice (a pre-roll function), the splice_insert() command could be sent multiple times before the splice point. For example, the splice_insert() command could be sent at 8, 5, 4 and 2 seconds prior to the packet containing the related splice point. In order to meet other splicing deadlines in the system, any message received with less than 4 seconds of advance notice may not create the desired result. The splice_insert() message shall be sent at least once a minimum of 4 seconds in advance of the desired splice time for a network Out Point condition.

The time_signal() command is provided for extensibility while preserving the precise timing allowed in the splice_insert() command. This is to allow for new features not directly related to splicing utilizing the timing capabilities of this Recommendation while causing minimal impact to the splicing devices that conform to this Recommendation. This allows the device that will be inserting the time into the cue message to have a defined location.

The bandwidth_reservation() command is provided to allow command insertion devices to utilize a consistent amount of transport stream bandwidth. Descriptors may be used in this command, but they cannot be expected to be processed and sent downstream to provide signalling information.

There are two methods for changing the parameters of a command once it has been issued. One method is to cancel the issued command by sending a splice_info_section with the splice_event_cancel_indicator set and then to send a new splice_info_section with the correct/new parameters. The other method is to simply send a subsequent message with the new data (without cancelling the old message via a cue message that has the splice_event_cancel_indicator bit set).

7.1.1 Time base discontinuities

In the case where a system time base discontinuity is present, packets containing a splice_insert() or time_signal() command with time expressed in the new time base shall not arrive prior to the occurrence of the time base discontinuity. Packets containing a splice_insert() or time_signal() command with time expressed in the previous time base shall not arrive after the occurrence of the time base discontinuity. See ISO/IEC 13818-4.

The complete syntax is presented below, followed by definition of terms, followed by constraints.

7.2 Splice info section

The splice_info_section shall be carried in transport packets whereby only one section or partial section may be in any transport packet. Splice_info_sections must always start at the beginning of a transport packet payload. When a section begins in a transport packet, the pointer_field must be present and equal to 0x00 and the payload_unit_start_indicator bit must be equal to one (per the requirements of section syntax usage per ITU-T Rec. H.222.0 | ISO/IEC 13818-1). See Table 7-1.

Table 7-1/J.181 – splice_info_section()

Syntax	Bits	Mnemonic	Encrypted
splice_info_section() {			
table_id	**8**	**uimsbf**	
section_syntax_indicator	**1**	**bslbf**	
private_indicator	**1**	**bslbf**	
reserved	**2**	**bslbf**	
section_length	**12**	**uimsbf**	
protocol_version	**8**	**uimsbf**	
encrypted_packet	**1**	**bslbf**	
encryption_algorithm	**6**	**uimsbf**	
pts_adjustment	**33**	**uimsbf**	
cw_index	**8**	**uimsbf**	
reserved	**12**	**bslbf**	
splice_command_length	**12**	**uimsbf**	
splice_command_type	**8**	**uimsbf**	**E**
if (splice_command_type == 0x00)			
splice_null()			**E**
if (splice_command_type == 0x04)			
splice_schedule()			**E**
if (splice_command_type == 0x05)			
splice_insert()			**E**
if (splice_command_type == 0x06)			
time_signal()			**E**
if (splice_command_type == 0x07)			
bandwidth_reservation()			**E**
descriptor_loop_length	**16**	**uimsbf**	**E**
for (i = 0; i < N1; i++)			
splice_descriptor()			**E**
for (i = 0; i < N2; i++)			
alignment_stuffing	**8**	**bslbf**	**E**
if (encrypted_packet)			
E_CRC_32	**32**	**rpchof**	**E**
CRC_32	**32**	**rpchof**	
}			

7.2.1 Semantic definition of fields in splice_info_section()

table_id: This is an 8-bit field. Its value shall be 0xFC.

section_syntax_indicator: The section_syntax_indicator is a 1-bit field that should always be set to "0" indicating that MPEG short sections are to be used.

private_indicator: This is a 1-bit flag that shall be set to 0.

section_length: This is a 12-bit field specifying the number of remaining bytes in the splice_info_section immediately following the section_length field up to the end of the splice_info_section. The value in this field shall not exceed 4093.

protocol_version: An 8-bit unsigned integer field whose function is to allow, in the future, this table type to carry parameters that may be structured differently than those defined in the current protocol. At present, the only valid value for **protocol_version** is zero. Non-zero values of **protocol_version** may be used by a future version of this Recommendation to indicate structurally different tables.

encrypted_packet: When this bit is set to "1", it indicates that portions of the splice_info_section, starting with splice_command_type and ending with and including E_CRC_32, are encrypted. When this bit is set to "0", no part of this message is encrypted. The potentially encrypted portions of the splice_info_table are indicated by an E in the Encrypted column of Table 7-1.

encryption_algorithm: This 6-bit unsigned integer specifies which encryption algorithm was used to encrypt the current message. When the encrypted_packet bit is zero, this field is present but undefined. Refer to clause 9, and specifically Table 9-1 for details on the use of this field.

pts_adjustment: A 33-bit unsigned integer that appears in the clear and that shall be used by a splicing device as an offset to be added to the (sometimes) encrypted pts_time field(s) throughout this message to obtain the intended splice time(s). When this field has a zero value, then the pts_time field(s) shall be used without an offset. Normally, the creator of a cueing message will place a zero value into this field. This adjustment value is the means by which an upstream device, which restamps pcr/pts/dts, may convey to the splicing device the means by which to convert the pts_time field of the message to a newly imposed time domain.

It is intended that the first device that restamps pcr/pts/dts and that passes the cueing message will insert a value into the pts_adjustment field, which is the delta time between this device's input time domain and its output time domain. All subsequent devices, which also restamp pcr/pts/dts, may further alter the pts_adjustment field by adding their delta time to the field's existing delta time and placing the result back in the pts_adjustment field. Upon each alteration of the pts_adjustment field, the altering device must recalculate and update the CRC_32 field.

The pts_adjustment shall, at all times, be the proper value to use for conversion of the pts_time field to the current time-base. The conversion is done by adding the two fields. In the presence of a wrap or overflow condition, the carry shall be ignored.

cw_index: An 8-bit unsigned integer that conveys which control word (key) is to be used to decrypt the message. The splicing device may store up to 256 keys previously provided for this purpose. When the encrypted_packet bit is zero, this field is present but undefined.

splice_command_length: A 12-bit length of the splice command. If this field = 0xFFF the length is not defined.

splice_command_type: An 8-bit unsigned integer assigned one of the values shown in Table 7-2.

Table 7-2/J.181 – splice_command_type values

splice_command_type value	Command
0x00	splice_null
0x01	Reserved
0x02	Reserved
0x03	Reserved
0x04	splice_schedule
0x05	splice_insert
0x06	time_signal
0x07	bandwidth_reservation
0x08-0xFF	Reserved

descriptor_loop_length: A 16-bit unsigned integer specifying the number of bytes used in the splice descriptor loop immediately following.

alignment_stuffing: When encryption is used this field is a function of the particular encryption algorithm chosen. Since some encryption algorithms require a specific length for the encrypted data, it is necessary to allow the insertion of stuffing bytes. For example, DES requires a multiple of 8 bytes be present in order to encrypt to the end of the packet. This allows standard DES to be used, as opposed to requiring a special version of the encryption algorithm.

When encryption is not used, this field shall not be used to carry valid data but may be present.

E_CRC_32: This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in ITU-T Rec. H.222.0 | ISO/IEC 13818-1 after processing the entire decrypted portion of the splice_info_section. This field is intended to give an indication that the decryption was performed successfully. Hence, the zero output is obtained following decryption and by processing the fields splice_command_type through E_CRC_32.

CRC_32: This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in ITU-T Rec. H.222.0 | ISO/IEC 13818-1 after processing the entire splice_info_section, which includes the table_id field through the CRC_32 field. The processing of CRC_32 shall occur prior to decryption of the encrypted fields and shall utilize the encrypted fields in their encrypted state.

7.3 Splice commands

7.3.1 splice_null()

The splice_null() command is provided for extensibility of this Recommendation. The splice_null() command allows a splice_info_table to be sent that can carry descriptors without having to send one of the other defined commands. This command may also be used as a "heartbeat message" for monitoring cue injection equipment integrity and link integrity. See Table 7-3.

Table 7-3/J.181 – splice_null()

Syntax	Bits	Mnemonic
splice_null() { }		

7.3.2 splice_schedule()

The splice_schedule() command is provided to allow a schedule of splice events to be conveyed in advance. See Table 7-4.

Table 7-4/J.181 – splice_schedule()

Syntax	Bits	Mnemonic
splice_schedule() {		
splice_count	**8**	**uimsbf**
for (i = 0; i < splice_count; i++) {		
splice_event_id	**32**	**uimsbf**
splice_event_cancel_indicator	**1**	**bslbf**
reserved	**7**	**bslbf**
if (splice_event_cancel_indicator == '0') {		
out_of_network_indicator	**1**	**bslbf**
program_splice_flag	**1**	**bslbf**
duration_flag	**1**	**bslbf**
reserved	**5**	**bslbf**
if (program_splice_flag == '1')		
utc_splice_time	**32**	**uimsbf**
if (program_splice_flag == '0') {		
component_count	**8**	**uimsbf**
for (j = 0; j < component_count; j++) {		
component_tag	**8**	**uimsbf**
utc_splice_time	**32**	**uimsbf**
}		
}		
if (duration_flag)		
break_duration()		
unique_program_id	**16**	**uimsbf**
avail_num	**8**	**uimsbf**
avails_expected	**8**	**uimsbf**
}		
}		
}		

7.3.2.1 Semantic definition of fields in splice_schedule()

splice_count: An 8-bit unsigned integer that indicates the number of splice events specified in the loop that follows.

splice_event_id: A 32-bit unique splice event identifier.

splice_event_cancel_indicator: A 1-bit flag that when set to "1" indicates that a previously sent splice event, identified by splice_event_id, has been cancelled.

out_of_network_indicator: A 1-bit flag. When set to "1", indicates that the splice event is an opportunity to exit from the network feed and that the value of utc_splice_time shall refer to an intended Out Point or Program Out Point. When set to "0", the flag indicates that the splice event is an opportunity to return to the network feed and that the value of utc_splice_time shall refer to an intended In Point or Program In Point.

program_splice_flag: A 1-bit flag that, when set to "1", indicates that the message refers to a Program Splice Point and that the mode is the Program Splice Mode whereby all PIDs/components of the program are to be spliced. When set to "0", this field indicates that the mode is the Component Splice Mode whereby each component that is intended to be spliced will be listed separately by the syntax that follows.

duration_flag: A 1-bit flag that indicates the presence of the break_duration() field.

utc_splice_time: A 32-bit unsigned integer quantity representing the time of the signalled splice event as the number of seconds since 00 hours UTC, 6 January 1980, with the count of intervening leap seconds included. The utc_splice_time may be converted to UTC without the use of the GPS_UTC_offset value provided by the System Time table. The **utc_splice_time** field is used only in the splice_schedule() command.

component_count: An 8-bit unsigned integer that specifies the number of instances of elementary PID stream data in the loop that follows. Components are equivalent to elementary PID streams.

component_tag: An 8-bit value that identifies the elementary PID stream containing the Splice Point specified by the value of splice_time() that follows. The value shall be the same as the value used in the stream_identification_descriptor() to identify that elementary PID stream.

unique_program_id: This value should provide a unique identification for a viewing event within the service. Since this Recommendation is intended for any encompassing standard, a value independent of any standard is required. It provides the same function as the event_id field in ATSC or DVB standards.

avail_num: (previously "avail") This field provides an identification for a specific avail within one unique_program_id. This value is expected to increment with each new avail within a viewing event. This value is expected to reset to one for the first avail in a new viewing event. This field is expected to increment for each new avail. It may optionally carry a zero value to indicate its non-usage.

avails_expected: (previously "avail_count") This field provides a count of the expected number of individual avails within the current viewing event. When this field is zero, it indicates that the avail_num field has no meaning.

7.3.3 splice_insert()

The splice_insert() command shall be sent at least once for every splice event. Please refer to 5.3 for the use of this message. See also Table 7-5.

Table 7-5/J.181 – splice_insert()

Syntax	Bits	Mnemonic
splice_insert() {		
splice_event_id	**32**	**uimsbf**
splice_event_cancel_indicator	**1**	**bslbf**
reserved	**7**	**bslbf**
if (splice_event_cancel_indicator == "0") {		
out_of_network_indicator	**1**	**bslbf**
program_splice_flag	**1**	**bslbf**
duration_flag	**1**	**bslbf**
splice_immediate_flag	**1**	**bslbf**
reserved	**4**	**bslbf**
If ((program_splice_flag == "1") && (splice_immediate_flag == "0"))		
splice_time()		
if (program_splice_flag == "0") {		
component_count	**8**	**uimsbf**
for (i = 0; i < component_count; i++) {		
component_tag	**8**	**uimsbf**
if (splice_immediate_flag == "0")		
splice_time()		
}		
}		
if (duration_flag == "1")		
break_duration()		
unique_program_id	**16**	**uimsbf**
avail_num	**8**	**uimsbf**
avails_expected	**8**	**uimsbf**
}		
}		

7.3.3.1 Semantic definition of fields in splice_insert()

splice_event_id: A 32-bit unique splice event identifier.

splice_event_cancel_indicator: A 1-bit flag that when set to "1" indicates that a previously sent splice event, identified by splice_event_id, has been cancelled.

out_of_network_indicator: A 1-bit flag. When set to "1", indicates that the splice event is an opportunity to exit from the network feed and that the value of splice_time(), as modified by pts_adjustment, shall refer to an intended Out Point or Program Out Point. When set to "0", the flag indicates that the splice event is an opportunity to return to the network feed and that the value of splice_time(), as modified by pts_adjustment, shall refer to an intended In Point or Program In Point.

program_splice_flag: A 1-bit flag that, when set to "1", indicates that the message refers to a Program Splice Point and that the mode is the Program Splice Mode whereby all PIDs/components of the program are to be spliced. When set to "0", this field indicates that the mode is the Component Splice Mode whereby each component that is intended to be spliced will be listed separately by the syntax that follows.

duration_flag: A 1-bit flag that, when set to "1", indicates the presence of the break_duration() field.

splice_immediate_flag: When this flag is "1", it indicates the absence of the splice_time() field and that the splice mode shall be the Splice Immediate Mode, whereby the splicing device shall choose the nearest opportunity in the stream, relative to the splice information packet, to splice. When this

flag is "0", it indicates the presence of the splice_time() field in at least one location within the splice_insert() command.

component_count: An 8-bit unsigned integer that specifies the number of instances of elementary PID stream data in the loop that follows. Components are equivalent to elementary PID streams.

component_tag: An 8-bit value that identifies the elementary PID stream containing the Splice Point specified by the value of splice_time() that follows. The value shall be the same as the value used in the stream_identification_descriptor() to identify that elementary PID stream.

unique_program_id: This value should provide a unique identification for a viewing event within the service. Since this Recommendation is intended for any encompassing standard, a value independent of any standard is required. It provides the same function as the event_id field in ATSC or DVB standards.

avail_num: (previously "avail") This field provides an identification for a specific avail within one unique_program_id. This value is expected to increment with each new avail within a viewing event. This value is expected to reset to one for the first avail in a new viewing event. This field is expected to increment for each new avail. It may optionally carry a zero value to indicate its non-usage.

avails_expected: (previously "avail_count") This field provides a count of the expected number of individual avails within the current viewing event. When this field is zero, it indicates that the avail_num field has no meaning.

7.3.4 time_signal()

The time_signal() provides a time synchronized data delivery mechanism. The syntax of the time_signal() allows for the synchronization of the information carried in this message with the System Time Clock (STC). The unique payload of the message is carried in the descriptor; however, the syntax and transport capabilities afforded to splice_insert() messages are also afforded to the time_signal(). The carriage however can be in a different PID than that carrying the other cue messages used for signalling splice points.

If the time_specified_flag is set to 0, indicating no pts_time in the message, then the command shall be interpreted as an immediate command. It must be understood that using it in this manner will cause an unspecified amount of accuracy error.

Since the time_signal() command utilizes descriptors for most of the specific information, this command could exceed one MPEG transport packet in length. It is strongly recommended to keep this command to one packet if possible. This may not always be possible in situations, for example, where the unique information is long or where another specification is used for the definition of this unique information. See Table 7-6.

Table 7-6/J.181 – time_signal()

Syntax	Bits	Mnemonic

time_signal() {
  splice_time()
}		

7.3.4.1 Semantic definition of time_signal()

This time_signal() provides a uniform method of associating a pts_time sample with an arbitrary descriptor (or descriptors) as provided by the splice_info_section syntax (see Table 7-1). Please refer to clause 8 for splice descriptors.

7.3.5 bandwidth_reservation()

The bandwidth_reservation() command is provided for reserving bandwidth in a multiplex. A typical usage would be in a satellite delivery system that requires packets of a certain PID to always be present at the intended repetition rate to guarantee a certain bandwidth for that PID. This message differs from a splice_null() command so that it can easily be handled in a unique way by receiving equipment (i.e., removed from the multiplex by a satellite receiver). If a descriptor is sent with this command, it cannot be expected that it will be carried through the entire transmission chain and it should be a private descriptor that is utilized only by the bandwidth reservation process. See Table 7-7.

Table 7-7/J.181 – bandwidth_reservation()

Syntax	Bits	Mnemonic
bandwidth_reservation() { }		

7.4 Time

7.4.1 splice_time()

The splice_time() structure, when modified by pts_adjustment, specifies the time of the splice event. See Table 7-8.

Table 7-8/J.181 – splice_time()

Syntax	Bits	Mnemonic

splice_time() {
time_specified_flag
if (time_specified_flag == 1) {
reserved
pts_time
}
else
reserved
}
	1
	
	6
	33
	
	7
		bslbf
		
		bslbf
		uimsbf
		
		bslbf

7.4.1.1 Semantic definition of fields in splice_time()

time_specified_flag: A 1-bit flag that, when set to "1", indicates the presence of the pts_time field and associated reserved bits.

pts_time: A 33-bit field that indicates time in terms of ticks of the program's 90 kHz clock. This field, when modified by pts_adjustment, represents the time of the intended splice point.

7.4.2 break_duration()

The break_duration() structure specifies the duration of the commercial break(s). It may be used to give the splicer an indication of when the break will be over and when the network In Point will occur. See Table 7-9.

Table 7-9/J.181 – break_duration()

Syntax	Bits	Mnemonic
break_duration() {		
auto_return	**1**	**bslbf**
reserved	**6**	**bslbf**
duration	**33**	**uimsbf**
}		

7.4.2.1 Semantic definition of fields in break_duration()

auto_return: A 1-bit flag that, when set to "1", denotes that the duration shall be used by the splicing device to know when the return to the network feed (end of break) is to take place. A splice_insert() command with out_of_network_indicator set to 0 is not intended to be sent to end this break. When this flag is "0", the duration field, if present, is not required to end the break because a new splice_insert() command will be sent to end the break. In this case, the presence of the break_duration field acts as a safety mechanism in the event that a splice_insert() command is lost at the end of a break.

duration: A 33-bit field that indicates elapsed time in terms of ticks of the program's 90 kHz clock.

7.5 Constraints

7.5.1 Constraints on splice_info_section()

The splice_info_section shall be carried in one or more PID stream(s) that are specific to a program and referred to in the PMT. The splice_info_section PID(s) shall be identified in the PMT by stream_type equal to 0x86.

The splice_info_section carried in one or more PID streams referenced in a program's PMT shall contain only information about splice events that occur in that program.

A splice event shall be defined by a single value of splice_event_id.

If the Component Splice Mode will be used, then each elementary PID stream shall be identified by a stream_identifier_descriptor carried in the PMT loop, one for each PID. The stream_identifier_descriptor shall carry a component_tag, which uniquely corresponds to one PID stream among those contained within a program and listed in the PMT for that program.

Any splice_event_id that is sent in a splice_info_section using a splice_schedule() command shall be sent again prior to the event using a splice_insert() command. Hence, there shall be a correspondence between the splice_event_id values chosen for particular events signalled by the splice_schedule() command (distant future) and splice_event_id values utilized in the splice_insert() command (near future) to indicate the same events.

Splice_event_id values do not need to be sent in an incrementing order in subsequent messages nor must they increment chronologically. Splice_event_id values may be chosen at random. When utilizing the splice_schedule() command, splice_event_id values shall be unique over the period of the splice_schedule() command. A splice_event_id value may be reused when its associated splice time has passed.

When the splice_immediate_flag is set to 1, the time to splice shall be interpreted as the current time. This is called the "Splice Immediate Mode". When this form is used with the splice_insert() command, the splice may occur at the nearest (prior or subsequent) opportunity that is detected by the splicer. The "Splice Immediate Mode" may be used for both splicing entry and exit points, i.e., for both states of out_of_network_indicator.

It shall be allowed that any avail may be ended with a Program Splice Mode message, a Component Splice Mode message or no message (whereby the break_duration is reached) regardless of the nature of the message at the beginning of the avail.

7.5.2 Constraints on the interpretation of time

7.5.2.1 Constraints on splice_time() for splice_insert()

For splice_command_type equal to 0x05 (splice_insert()) the following constraints on splice_time() shall apply:

At least one message for a network Out Point must arrive at least 4 seconds in advance of the signalled splice time (pts_time as modified by pts_adjustment) if the time is specified. A Splice Immediate Mode message is allowed for a network Out Point, but the actual splice time is not defined and it is recommended that Splice Immediate Mode messages only be used for the early termination of breaks. When non-Splice Immediate Mode cue messages are used for network In Points, the cue message must arrive at the splicer before the arrival of the signalled In Point picture at the receiver.

An Out Point lies between two presentation units. The intended Out Point of a signalled splice event shall be the Out Point that is immediately prior to the presentation unit whose presentation time most closely matches the signalled pts_time as modified by pts_adjustment.

An In Point lies between two presentation units. The intended In Point of a signalled splice event shall be the In Point that is immediately prior to the presentation unit whose presentation time most closely matches the signalled pts_time as modified by pts_adjustment.

When the Component Splice Mode is in effect and the out_of_network_indicator is "1" (the beginning of a break), each component listed in the splice_insert() component loop shall be switched from the network component to the splicer supplied component at the time indicated. Components not listed in the component loop of the message will remain unchanged: if a splicer output component was the network component then it will remain the network component; if a splicer output component was the splicer supplied component, then it will remain the splicer supplied component.

When the Component Splice Mode is in effect and the out_of_network_indicator is "0" (the end of a break), each component listed in the splice_insert() component loop shall be switched from the splicer supplied component to the network component at the time indicated. Components not listed in the component loop of the message will remain unchanged: if a splicer output component was the network component, then it will remain the network component; if a splicer output component was the splicer supplied component, then it will remain the splicer supplied component.

When the Component Splice Mode is in effect and the Splice Immediate Mode is not in effect, the first component listed in the component loop of the splice_insert() command shall have a valid pts_time in its associated splice_time() and this pts_time is referred to as the default pts_time. Subsequent components listed in the component loop of the same message, which do not have an associated pts_time, shall utilize this default pts_time. It shall be allowed that any and all components following the first listed component of a splice_insert() command may contain a unique pts_time that is different from the default pts_time.

In the Component Splice Mode, all pts_time values given in the splice_insert component loop shall be modified by the pts_adjustment field to obtain each intended value for the signalled Out Point or In Point. The pts_adjustment, provided by any device that generates or modifies a pts_adjustment field value, shall apply to all pts_time fields in the message.

7.5.2.2 Constraints on break_duration() for splice_insert()

For splice_command_type equal to 0x05 (splice_insert), the following constraints on break_duration() shall apply:

The value given in break_duration() is interpreted as the intended duration of the commercial break. It is an optional field to be used when the out_of_network_indicator equals 1. It may be used in the same splice_insert() command that specifies the start time of the break, so that the splicer can calculate the time when the break will be over.

Breaks may be terminated by issuing a splice_insert() command with out_of_network_indicator set to 0. A splice_time() may be given or the Splice Immediate Mode may be used. When a break_duration was given at the start of the break (where the auto_return was set to zero), the break_duration value may be utilized as a backup mechanism for insuring that a return to the network actually happens in the event of a lost cueing packet.

Breaks may also be terminated by giving a break duration at the beginning of a break and relying on the splicing device to return to the network feed at the proper time. The auto_return flag must be 1. This will be referred to as the Auto Return Mode. Auto Return Mode breaks do not require and do not disallow cue messages at the end of the break with out_of_network_indicator set to 0. Hence a receiving device should not expect a cue message at the end of a break in order to function properly. Auto Return Mode breaks may however be terminated early. To end the break prematurely a second splice_insert() command may be given, where the out_of_network_indicator equals 0. The new time of the back to network splice may be given by an updated splice_time(), or the Splice Immediate Mode message may be used. A cue message with out_of_network_indicator set to 0 shall always override the duration field of a previous cue message (with out_of_network_indicator set to 1) if that break's signalled duration is still under way.

8 Splice descriptors

8.1 Overview

The splice_descriptor is a prototype for adding new fields to the splice_info_section. All descriptors included use the same syntax for the first six bytes. In order to allow private information to be added, we have included the "identifier" code. This removes the need for a registration descriptor in the descriptor loop.

Any receiving equipment should skip any descriptors with unknown identifiers or unknown descriptor tags. For descriptors with known identifiers, the receiving equipment should skip descriptors with an unknown splice_descriptor_tag.

Splice descriptors may exist in the splice_info_section for extensions specific to the various commands. See Table 8-1.

Table 8-1/J.181 – Splice descriptor tags

Tag	Descriptors for identifier"CUEI"
0x00	avail_descriptor
0x01	DTMF_descriptor
0x02	segmentation_descriptor
0x03-0xFF	Reserved for future SCTE splice_descriptors

8.2 Splice descriptor

The splice descriptor syntax provided in this clause is to be used as a template for specific implementations of a descriptor intended for the splice_info_section. It should be noted that splice

descriptors are only used within a splice_info_section. They are not to be used within MPEG syntax, such as the PMT, or in the syntax of any other standard. This allows one to draw on the entire range of descriptor tags when defining new descriptors. See Table 8-2.

Table 8-2/J.181 – splice_descriptor()

Syntax	Bits	Mnemonic
splice_descriptor() {		
splice_descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
identifier	**32**	**uimsbf**
for (i = 0; i < N; i++) {		
private_byte	**8**	**uimsbf**
}		
}		

8.2.1 Semantic definition of fields in splice_descriptor()

splice_descriptor_tag: This 8-bit number defines the syntax for the private bytes that make up the body of this descriptor. The descriptor tags are defined by the owner of the descriptor, as registered using the identifier.

descriptor_length: This 8-bit number gives the length, in bytes, of the descriptor following this field. Descriptors are limited to 256 bytes, so this value is limited to 254.

identifier: This is a 32-bit field as defined in ITU-T Rec. H.222.0 | ISO/IEC 13818-1, clauses 2.6.8 and 2.6.9, for the registration_descriptor() format_identifier. Only identifier values registered and recognized by SMPTE Registration Authority, LLC should be used (see http://www.smpte-ra.org/mpegreg.html). Its use in this descriptor shall scope and identify only the private information contained within this descriptor. This 32-bit number is used to identify the owner of the descriptor. The code 0x43554549 (ASCII "CUEI") for descriptors defined in this Recommendation has been registered with SMPTE.

private_byte: The remainder of the descriptor is dedicated to data fields as required by the descriptor being defined.

8.3 Specific splice descriptors

8.3.1 avail_descriptor()

The avail_descriptor is an implementation of a splice_descriptor. It provides an optional extension to the splice_insert() command that allows an authorization identifier to be sent for an avail. Multiple copies of this descriptor may be included by using the loop mechanism provided. This identifier is intended to replicate the functionality of the cue tone system used in analog systems for ad insertion. This descriptor is intended only for use with a splice_insert() command, within a splice_info_section. See Table 8-3.

Table 8-3/J.181 – avail_descriptor()

Syntax	Bits	Mnemonic
avail_descriptor() {		
splice_descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
identifier	**32**	**uimsbf**
provider_avail_id	**32**	**uimsbf**
}		

8.3.1.1 Semantic definition of fields in avail_descriptor()

splice_descriptor_tag: This 8-bit number defines the syntax for the private bytes that make up the body of this descriptor. The splice_descriptor_tag shall have a value of 0x00.

descriptor_length: This 8-bit number gives the length, in bytes, of the descriptor following this field. The descriptor_length field shall have a value of 0x08.

identifier: This 32-bit number is used to identify the owner of the descriptor. The identifier shall have a value of 0x43554549 (ASCII "CUEI").

provider_avail_id: This 32-bit number provides information that a receiving device may utilize to alter its behaviour during or outside of an avail. It may be used in a manner similar to analog cue tones. An example would be a network directing an affiliate or a headend to black out a sporting event.

8.3.2 DTMF_descriptor()

The DTMF_descriptor() is an implementation of a splice_descriptor. It provides an optional extension to the splice_insert() command that allows a receiver device to generate a legacy analog DTMF sequence based on a splice_info_section being received. See Table 8-4.

Table 8-4/J.181 – DTMF_descriptor()

Syntax	Bits	Mnemonic
DTMF_descriptor() {		
splice_descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
Identifier	**32**	**uimsbf**
Preroll	**8**	**uimsbf**
dtmf_count	**3**	**uimsbf**
reserved	**5**	**bslbf**
for (i = 0; i < **dtmf_count**; i++) {		
DTMF_char	**8**	**uimsbf**
}		
}		

8.3.2.1 Semantic definition of fields in DTMF_descriptor()

splice_descriptor_tag: This 8-bit number defines the syntax for the private bytes that make up the body of this descriptor. The splice_descriptor_tag shall have a value of 0x01.

descriptor_length: This 8-bit number gives the length, in bytes, of the descriptor following this field.

identifier: This 32-bit number is used to identify the owner of the descriptor. The identifier shall have a value of 0x43554549 (ASCII "CUEI").

preroll: This 8-bit number is the time the DTMF is presented to the analog output of the device in tenths of seconds. This gives a preroll range of 0 to 25.5 seconds. The splice_info_section shall be sent at least two seconds earlier than this value. The minimum suggested preroll is 4 seconds.

dtmf_count: The value of this flag is the number of DTMF characters the device is to generate.

DTMF_char: This is an ASCII value for the numerals "0" to "9", "*", "#". The device shall use these values to generate a DTMF sequence to be output on an analog output. The sequence shall complete with the last character sent being the timing mark for the preroll.

8.3.3 segmentation_descriptor()

The segmentation_descriptor() is an implementation of a splice_descriptor(). It provides an optional extension to the time_signal() command that allows for program segmentation messages to be sent in a time/video accurate method. This descriptor may be utilized in commands other than the time_signal() command.

Devices that do not recognize a value in any field shall ignore the message and take no action. See Table 8-5.

Table 8-5/J.181 – segmentation_descriptor()

Syntax	Bits	Mnemonic
segmentation_descriptor() {		
splice_descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
identifier	**32**	**uimsbf**
segmentation_event_id	**32**	**uimsbf**
segmentation_event_cancel_indicator	**1**	**bslbf**
reserved	**7**	**bslbf**
if (segmentation_event_cancel_indicator == "0") {		
program_segmentation_flag	**1**	**bslbf**
segmentation_duration_flag	**1**	**bslbf**
reserved	**6**	**bslbf**
if (program_segmentation_flag == "0") {		
component_count	**8**	**uimsbf**
for (i = 0; i < component_count; i++) {		
component_tag	**8**	**uimsbf**
reserved	**7**	**bslbf**
pts_offset	**33**	**uimsbf**
}		
}		
if (**segmentation_duration_flag** == "1")		
segmentation_duration()		
segmentation_upid_type	**8**	**uimsbf**
segmentation_upid_length	**8**	**uimsbf**
segmentation_upid		**uimsbf**
segmentation_type_id	**8**	**uimsbf**
chapter	**8**	**uimsbf**
chapter_count	**8**	**uimsbf**
}		
}		

8.3.3.1 Semantic definition of fields in segmentation_descriptor()

splice_descriptor_tag: This 8-bit number defines the syntax for the private bytes that make up the body of this descriptor. The splice_descriptor_tag shall have a value of 0x02.

descriptor_length: This 8-bit number gives the length, in bytes, of the descriptor following this field.

identifier: This 32-bit number is used to identify the owner of the descriptor. The identifier shall have a value of 0x43554549 (ASCII "CUEI").

segmentation_event_id: A 32-bit unique segmentation event identifier.

segmentation_event_cancel_indicator: A 1-bit flag that when set to "1" indicates that a previously sent segmentation event, identified by segmentation_event_id, has been cancelled.

program_segmentation_flag: A 1-bit flag that should be set to "1" indicating that the message refers to a Program Segmentation Point and that the mode is the Program Segmentation Mode whereby all PIDs/components of the program are to be segmented. When set to "0", this field indicates that the mode is the Component Segmentation Mode whereby each component that is intended to be segmented will be listed separately by the syntax that follows. The program_segmentation_flag can be set to different states during different descriptors messages within a program.

segmentation_duration_flag: A 1-bit flag that should be set to "1" indicating the presence of segmentation_duration() field.

component_count: An 8-bit unsigned integer that specifies the number of instances of elementary PID stream data in the loop that follows. Components are equivalent to elementary PID streams.

component_tag: An 8-bit value that identifies the elementary PID stream containing the Segmentation Point specified by the value of splice_time() that follows. The value shall be the same as the value used in the stream_identification_descriptor() to identify that elementary PID stream. The presence of this field from the component loop denotes the presence of this component of the asset.

pts_offset: A 33-bit unsigned integer that shall be used by a splicing device as an offset to be added to the pts_time in the time_signal() message to obtain the intended splice time(s). When this field has a zero value, then the pts_time field(s) shall be used without an offset. If splice_time() time_specified_flag = 0 or if the command this descriptor is carried with does not have a splice_time() field, this field shall be used to offset the derived immediate splice time.

segmentation_duration(): The segmentation_duration() structure specifies the duration of the program segment. It may be used to give the splicer an indication of when the segment will be over and when the next segmentation message will occur. Must be 0 for end messages. See Table 8-6.

Table 8-6/J.181 – segmentation_duration()

Syntax	Bits	Mnemonic
segmentation_duration() {		
reserved	**7**	**bslbf**
duration	**33**	**uimsbf**
}		

duration: A 33-bit field that indicates elapsed time in terms of ticks of the program's 90 kHz clock.

segmentation_upid_type: A value from the following table. There are multiple types allowed to ensure that programmers will be able to use an id that their systems support. It is expected that the consumers of these ids will have an out-of-band method of collecting other data related to these numbers and therefore they do not need to be of identical types. These ids may be in other descriptors in the program and where the same identifier is used (V-ISAN for example) it shall match between programs. When used with splice insert, the **segmentation_upid** cannot be mandated or assumed to be a match with the **unique_program_id** in the splice_insert message. See Table 8-7.

Table 8-7/J.181 – segmentation_upid_type

Type	Length bytes	Name	Description
0x00	0	Not Used	The **segmentation_upid** is not defined and is not present in the descriptor.
0x01	variable	User Defined	The **segmentation_upid** does not follow a standard naming scheme.
0x02	8	ISCII	The **segmentation_upid** is an 8-character code that is typically used for advertising content.
0x03	12	Ad-ID	Defined by the Advertising Digital Identification, LLC group. This is basically a 12-character identifier. http://www.Ad-id.org/
0x04	24	UMID	SMPTE 330M
0x05	8	ISAN	ISO standard for AV material using 16 Hex digits. http://www.nlc-bnc.ca/iso/tc46sc9/standard/15706e.htm
0x06	12	V-ISAN	Extension of the ISAN standard adding 8 Hex characters for a version identifier.
0x07	12	TID	Tribune Media Systems Program identifier. 12 characters, 2 letters followed by 10 digits.
0x08	8	Origin	Turner Origin ID.
0x09-0x1F	variable	Reserved	Reserved for future standardization.

segmentation_upid_length: Length in bytes of **segmentation_upid**.

segmentation_upid: Length and identification from Table 8-7. This field's contents and length are determined by the **segmentation_upid_type** and **segmentation_upid_length** fields. An example would be a type of 0x06 for V-ISAN and a length of 12 bytes. This field would then contain the V-ISAN identifier for the content that this descriptor refers to.

segmentation_type_id: The 8-bit value shall contain one of the values in Table 8-8 to designate type of segmentation. All unused values are reserved.

Table 8-8/J.181 – segmentation_type_id

Segmentation message	Segmentation_type_id
Program Start	0x10
Program End	0x11
Program Early Termination	0x12
Program Breakaway	0x13
Program Resumption	0x14
Program Runover Planned	0x15
Program Runover Unplanned	0x16
Chapter Start	0x20
Chapter End	0x21
National Advertisement Start	0x30
National Advertisement End	0x31

Table 8-8/J.181 – segmentation_type_id

Segmentation message	Segmentation_type_id
Local Advertisement Start	0x32
Local Advertisement End	0x33
Unscheduled_event_start	0x40
Unscheduled_event_end	0x41

chapter: This field provides identification for a specific chapter within a segmentation_unique_program_id. This value is expected to increment for each new chapter within a segmentation event. This value is expected to reset to one for the first chapter in a new viewing event. This field is expected to increment for each new chapter.

chapter_count: This field provides a count of the expected number of individual chapters within the current segmentation event.

9 Encryption

9.1 Overview

The splice_info_section supports the encryption of a portion of the section in order that one may prevent access to an avail to all except those receivers that are authorized for that avail. This clause describes the various encryption algorithms that may be used. The encryption of the section is optional, as is the implementation of encryption by either the creator of the message, or any receive devices. The use of encryption is deemed optional to allow a manufacturer to ship "in-the-clear" systems without worrying about the export of encryption technology. If encryption is included in the system, any receive device shall implement all of the algorithms listed in this Recommendation, which allows the creator of a splice info table to use any of the algorithms in a transmission. The use of private encryption technology is optional, and out of the scope of this Recommendation.

9.2 Fixed key encryption

The encryption used with this Recommendation assumes a fixed key is to be used. The same key is provided to both the transmitter and the receiver. The method of delivering the key to all parties is unspecified. This Recommendation allows for up to 256 different keys to be available for decryption. The cw_index field is used to determine which key should be used when decrypting a section. The length of the fixed key is dependent on the type of algorithm being used. It is assumed that fixed key delivered to all parties will be the correct length for the algorithm that is intended to be used.

9.3 Encryption algorithms

The encryption_algorithm field of the splice_info_section is a six-bit value, which may contain one of the values shown in Table 9-1. All Data Encryption Standard variants use a 64-bit key (actually 56 bits plus a checksum) to encrypt or decrypt a block of 8 bytes. In the case of triple DES, there will need to be three 64-bit keys, one for each of the three passes of the DES algorithm. The "standard" triple DES actually uses two keys, where the first and third keys are identical. See FIPS PUB 46-3 and FIPS PUB 81.

Table 9-1/J.181 – Encryption algorithm

Value	Encryption algorithm
0	No encryption
1	DES – ECB mode
2	DES – CBC mode
3	Triple DES EDE3 – ECB mode
4-31	Reserved
32-63	User private

9.3.1 DES – ECB mode

This algorithm uses the "Data Encryption Standard" (see FIPS PUB 81) in the electronic codebook mode.

In order to use this type of encryption, the encrypted data must contain a multiple of 8 bytes of data, from splice_command_type through to E_CRC_32 fields. The alignment_stuffing loop may be used to pad any extra bytes that may be required.

9.3.2 DES – CBC mode

This algorithm uses the "Data Encryption Standard" (see FIPS PUB 81) in the cipher block chaining mode. The basic algorithm is identical to DES ECB. Each 64-bit plaintext block is bitwise exclusive-ORed with the previous ciphertext block before being encrypted with the DES key. The first block is exclusive-ORed with an initial vector. For the purposes of this Recommendation, the initial vector shall have a fixed value of zero.

In order to use this type of encryption, the encrypted data must contain a multiple of 8 bytes of data, from splice_command_type through to E_CRC_32 fields. The alignment_stuffing loop may be used to pad any extra bytes that may be required.

9.3.3 Triple DES EDE3 – ECB mode

This algorithm uses three 64-bit keys, each key being used on one pass of the DES – ECB algorithm. See FIPS PUB 46-3. Every block of data at the transmit device is first encrypted with the first key, decrypted with the second key, and finally encrypted with the third key. Every block at the receive site is first decrypted with the third key, encrypted with the second key, and finally decrypted with the first key.

In order to use this type of encryption, the encrypted data must contain a multiple of 8 bytes of data, from splice_command_type through to E_CRC_32 fields. The alignment_stuffing loop may be used to pad any extra bytes that may be required.

9.3.4 User private algorithms

This Recommendation allows for the use of private encryption algorithms. It is not specified how the transmit and receive devices agree on the algorithm to use for any user private code. It is also not specified as to how coordination of private values for the encryption_algorithm field should be registered or administered.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of 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	**Cable networks and transmission of television, sound programme and other multimedia signals**
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure, Internet protocol aspects and Next Generation Networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2004