INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

J.181

(03/2001)

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

Miscellaneous

Digital program insertion cueing message for cable television systems

ITU-T Recommendation J.181

(Formerly CCITT Recommendation)

ITU-T J-SERIES RECOMMENDATIONS

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

General Recommendations	J.1–J.9
General specifications for analogue sound-programme transmission	J.10–J.19
Performance characteristics of analogue sound-programme circuits	J.20–J.29
Equipment and lines used for analogue sound-programme circuits	J.30–J.39
Digital encoders for analogue sound-programme signals	J.40–J.49
Digital transmission of sound-programme signals	J.50–J.59
Circuits for analogue television transmission	J.60–J.69
Analogue television transmission over metallic lines and interconnection with radio-relay links	J.70–J.79
Digital transmission of television signals	J.80–J.89
Ancillary digital services for television transmission	J.90–J.99
Operational requirements and methods for television transmission	J.100–J.109
Interactive systems for digital television distribution	J.110–J.129
Transport of MPEG-2 signals on packetised networks	J.130–J.139
Measurement of the quality of service	J.140–J.149
Digital television distribution through local subscriber networks	J.150–J.159
IPCablecom	J.160–J.179
Miscellaneous	**J.180–J.199**
Application for Interactive Digital Television	J.200–J.209

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

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 prepared by ITU-T Study Group 9 (2001-2004) and approved under the WTSA Resolution 1 procedure on 9 March 2001.

CONTENTS

Page

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.

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. It is expected that transport packet boundary splicing, as intended by ITU-T H.222.0 | ISO/IEC 13818-1 and by SMPTE 312M, will not be sufficient in cable plants due to current and continuing methods such as statistical multiplexing (VBR) and progressive refresh (no I-frames).

This Recommendation specifies a technique for carrying notification of upcoming Splice Points 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 a separate PID 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.

This Recommendation does not address constraints on splicing devices.

2 References

2.1 Normative references

The following ITU-T Recommendation 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.

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

– ITU-T 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).

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**: The coded representation of a video picture or an audio frame (see ITU-T H.262 | ISO/IEC 13818-2).

3.2 **analog cue tone**: In an analog system, a signal which 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 to 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 access unit boundary.

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

3.10 **out point**: A point in the stream, suitable for exit, that lies on an elementary access unit boundary.

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

3.12 **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 H.222.0 | ISO/IEC 13818-1).

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

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

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

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

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

3.18 **program out point**: A group of PID stream Out Points which correspond in presentation time.

3.19 **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.

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

3.21 **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.

3.22 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 the Recommendation. Unless otherwise specified in this Recommendation, all reserved bits shall be set to "1".

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

3.24 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", which is a presentation time, for the intended splicing moment.

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

3.26 viewing event: A television program 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
MPTS	Multi Program Transport Stream
PMT	Program Map Table (see ITU-T H.222.0 | ISO/IEC 13818-1)
PTS	Presentation Time Stamp (see ITU-T H.222.0 | ISO/IEC 13818-1)
rpchof	Remainder polynomial coefficients, highest order first
SPTS	Single Program Transport Stream
uimsbf	Unsigned integer, most significant bit first

5 Introduction

5.1 Splice points

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 from one program 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 access 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 access 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

Program In Points and Program Out Points are sets of PID stream In Points or Out Points which 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 false flag 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

This Recommendation provides a method for in-band signalling of splice_schedule and splice_insert messages 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 a 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 a PID stream with a designated stream_type. The program's splice information PID is designated in the program's program map table. 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 may use this stream_type field to drop splice information prior to sending the transport stream to the end-user device.

6 PMT descriptors

6.1 Registration descriptor

The registration descriptor (see Table 2-45 "Registration descriptor" in 2.6.8/ITU-T H.222.0 | ISO/IEC 13818-1) is defined to identify unambiguously the programs which comply with this Recommendation. The registration descriptor shall be carried in the program_info loop of the PMT for each program that complies wirh 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 a unique PID stream within this program. The PID chosen for this splice information table shall be a unique PID and shall not be in common with the PID for other splice information tables for other programs within the multiplex and shall not be in common with the PID chosen for other splice information tables defined by other standards (i.e. SMPTE 312M).

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 which 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 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 the program_splice_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.

6.2.1 Semantic definition of fields in stream identifier descriptor

See Table 6-2.

Table 6-2/J.181 – Stream_identifier_descriptor

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

descriptor_tag: The descriptor_tag is an 8-bit field which identifies each descriptor. For stream_identifier_descriptor, this field shall be set to 0x52 as specified by DVB.

descriptor_length: The descriptor_length in 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 in its own PID stream with the PID 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 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 allowed: splice_null(), splice_schedule() and splice_insert().

The splice_null() command is provided for extensibility.

The splice_schedule() command is a command which 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 matches that specified 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.

7.1.1 Time base discontinuities

In the case where a system time base discontinuity (TBD) is present, packets containing a splice_insert() command with time expressed in the new time base shall not arrive prior to the occurrence of the TBD. Packets containing a splice_insert() command with time expressed in the previous time base shall not arrive after the occurrence of the TBD. See ISO/IEC 13818-4, Information technology – Generic coding of moving pictures and associated audio information – Part 4: Conformance testing.

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

7.2 splice_info_section

The splice_info_section (see Table 7-1) 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 (according to the requirements of section syntax usage in ITU-T H.222.0 | ISO/IEC 13818-1).

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	**24**	**bslbf**	
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**
descriptor_loop_length	**16**	**uimsbf**	**E**
for (I = 0; I < N; I++)			
splice_descriptor()			**E**
for (I = 0; I < N; 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 which should always be set to "0" indicating that MPEG short sections are to be used.

private_indicator: This is a 1-bit flag which 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 which indicates the version number of the segment of the full table delivered with this section. The value of protocol_version shall be 0x00.

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 which appears in the clear and which 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 which restamps pcr/pts/dts and which 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 which 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_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	Null
0x01	Reserved
0x02	Reserved
0x03	Reserved
0x04	Schedule
0x05	Insert
0x06-0xff	Reserved

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

alignment_stuffing: 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.

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 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 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 the Recommendation. The splice_null() command allows a splice_info_table to be sent which carries new descriptors without having to send one of the existing commands (splice_insert() or splice_schedule()). (See Table 7-3.)

Table 7-3/J.181 – splice_null()

Syntax	Bits	Description

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.

7.3.2.1 Semantic definition of fields in splice_schedule()

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**	**bslbf**
splice_event_cancel_indicator	**1**	**bslbf**
reserved	**7**	**bslbf**

Table 7-4/J.181 – splice_schedule() (*concluded*)

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	**8**	**uimsbf**
avail_count	**8**	**uimsbf**
}		
}		
}		

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 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 splice_time() shall refer to an intended In Point or Program In Point.

program_splice_flag: A 1-bit flag which 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 which 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 which identifies the elementary PID stream containing the Splice Point specified by the value of splice_time() which 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: 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.

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 field has no meaning.

7.3.3 splice_insert()

The splice_insert() command shall be sent at least once for every splice event. (See Table 7-5.)

Table 7-5/J.181 – splice_insert()

Syntax	Bits	Mnemonic
splice_insert() {		
splice_event_id	**32**	**bslbf**
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**

Table 7-5/J.181 – splice_insert() (*concluded*)

avail	**8**	**uimsbf**
avail_count	**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() 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() shall refer to an intended In Point or Program In Point.

program_splice_flag: A 1-bit flag which 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 which, 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 which identifies the elementary PID stream containing the Splice Point specified by the value of splice_time() which 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: 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.

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 field has no meaning.

7.4 Time

7.4.1 splice_time()

The splice_time() structure specifies the time of the splice event. (See Table 7.6.)

Table 7-6/J.181 – splice_time()

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

7.4.1.1 Semantic definition of fields in splice_time()

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

pts_time: A 33-bit field which indicates time in terms of ticks of the program's 90 kHz clock. This field represents the presentation time of the intended splice point, which is equivalent to the presentation time of the access unit following 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 7-7.)

Table 7-7/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) shall 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 which 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 a PID stream which is specific to a program and referred to in the PMT. The splice_info_section PID shall be identified in the PMT by stream_type equal to 0x86.

The splice_info_section carried in a PID stream referenced in a program's PMT shall contain only information about splice events which 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 which 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 (insert) the following constraints on splice_time() shall apply:

For specifying a Program Out Point, i.e. when the program_splice_flag equals 1 and the out_of_network_indicator equals 1, then the value of pts_time shall equal the presentation time of the intended Program Out Point.

For specifying an Out Point in an elementary PID stream, i.e. when the program_splice_flag equals 0 and the out_of_network_indicator equals 1, then the value of pts_time shall equal the presentation time of the intended Out Point of the elementary PID stream which corresponds to the value of component_tag.

An Out Point shall be defined to have the same presentation time as the access unit that immediately follows it in the stream. Thus the intended Out Point of a signalled splice event shall be the Out Point which is immediately prior to the access unit whose presentation time agrees with the signalled splice_time().

For specifying a Program In Point, i.e. when the program_splice_flag equals 1 and the out_of_network_indicator equals 0, then the value of pts_time shall equal the presentation time of the intended Program In Point.

For specifying an In Point in an elementary PID stream, i.e. when the program_splice_flag equals 0 and the out_of_network_indicator equals 0, then the value of pts_time shall equal the presentation time of the intended In Point of the elementary PID stream which corresponds to the value of component_tag.

An In Point shall be defined to have the same presentation time as the access unit that immediately follows it in the stream. Thus the intended In Point of a signalled splice event shall be the In Point which is immediately prior to the access unit whose presentation time agrees with the signalled splice_time().

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 currently the network component then it will remain the network component; if a splicer output component was currently 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 currently the network component, then it will remain the network component; if a splicer output component was currently 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, it shall be a requirement that 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 shall be 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 which is different from the default pts_time.

7.5.2.2 Constraints on break_duration() for splice_insert()

For splice_command_type equal to 0x05 (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 back-up 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 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 may be used.

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
0x01-0xFF	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	Description
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 which 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 32-bit number is used to identify the owner of the descriptor. The SCTE is reserving the code 0x43554549 (ASCII "CUEI").

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	Description
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 which 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.

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 avails 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 which 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.

Table 9-1/J.181 – Encryption algorithm

Value	Encryption Algorithm	Standard
0	No encryption	
1	DES-ECB mode	ANSI X9.32
2	DES-CBC mode	ANSI X9.32
3	Triple DES EDE3-ECB mode	ANSI X9.52
4-31	Reserved	
32-63	User private	

9.3.1 DES-ECB mode

This algorithm uses the "Data Encryption Standard", 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.

9.4 DES-CBC mode

This algorithm uses the "Data Encryption Standard", 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.

9.5 Triple DES EDE3-ECB mode

This algorithm uses three 64-bit keys, each key being used on one pass of the DES-ECB algorithm. 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.

9.6 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.

APPENDIX I

Bibliography

– SMPTE 312M-1999, *Television – Splice Points for MPEG-2 Transport Streams.*

Geneva, 2001