

# ITU-T

TELECOMMUNICATION  
STANDARDIZATION SECTOR  
OF ITU

## J.181

**Amendment 1**  
(09/2014)

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

Digital transmission of television signals – Part 1

---

Digital program insertion cueing message for cable  
television systems

**Amendment 1: New Appendix II: Recommended  
practices for the implementation of ITU-T J.181**

Recommendation ITU-T J.181 (2014) – Amendment 1

ITU-T



# Recommendation ITU-T J.181

## Digital program insertion cueing message for cable television systems

### Amendment 1

#### New Appendix II: Recommended practices for the implementation of ITU-T J.181

#### Summary

The core text of Recommendation ITU-T J.181 has been kept brief in many areas in order to maintain conciseness and accuracy. Amendment 1 to Recommendation ITU-T J.181 provides an informative appendix on recommended practices which enhance Recommendation ITU-T J.181.

#### History

| Edition | Recommendation            | Approval   | Study Group | Unique ID*                                                                |
|---------|---------------------------|------------|-------------|---------------------------------------------------------------------------|
| 1.0     | ITU-T J.181               | 2001-03-09 | 9           | <a href="http://handle.itu.int/11.1002/1000/5386">11.1002/1000/5386</a>   |
| 1.1     | ITU-T J.181 (2001) Amd. 1 | 2003-04-04 | 9           | <a href="http://handle.itu.int/11.1002/1000/6355">11.1002/1000/6355</a>   |
| 2.0     | ITU-T J.181               | 2004-06-29 | 9           | <a href="http://handle.itu.int/11.1002/1000/7379">11.1002/1000/7379</a>   |
| 3.0     | ITU-T J.181               | 2014-01-13 | 9           | <a href="http://handle.itu.int/11.1002/1000/12102">11.1002/1000/12102</a> |
| 3.1     | ITU-T J.181 (2014) Amd. 1 | 2014-09-12 | 9           | <a href="http://handle.itu.int/11.1002/1000/12351">11.1002/1000/12351</a> |

\* To access the Recommendation, type the URL <http://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID. For example, <http://handle.itu.int/11.1002/1000/11830-en>.

## FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). 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, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2015

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

## Introduction

ITU-T J.181 is necessarily brief in many areas in order to maintain conciseness and accuracy. This amendment adds Appendix II to ITU-T J.181 and serves as an informative guideline on practices for the implementation of the Recommendation.



# Recommendation ITU-T J.181

## Digital program insertion cueing message for cable television systems

### Amendment 1

#### New Appendix II: Recommended practices for the implementation of ITU-T J.181

##### 1) Modifications to ITU-T J.181

*Add the following text after Appendix I as a new Appendix II, Recommended practices for the implementation of ITU-T J.181.*

### Appendix II

#### Recommended practices for the implementation of ITU-T J.181

(This appendix does not form an integral part of this Recommendation.)

##### II.1 Scope

This amendment adds Appendix II as an informational companion to Recommendation ITU-T J.181. The appendix itself is not a specification or a standard; the information within is intended as guideline practices for the implementation of the Recommendation. Where this appendix contradicts the main Recommendation, the main Recommendation takes precedence.

##### II.2 Purpose

The purpose of this appendix is to aid splicing and ad insertion equipment designers, as well as the purchasers and users of such equipment, such as the networks that will originate cue messages from their uplink sites. This appendix is also expected to aid in the system integration of advertising related equipment, both at the message origination and at message reception end.

The Recommendation includes content segmentation messages; updated information provided in this appendix is intended to aid the users of these messages. The following new devices that will be interpreting the Recommendation commands include: transcoders, packagers, network personal video recorder (PVR) and other content manipulation, storage and streaming delivery systems.

Crucial information is provided for manufacturers of equipment that pass the cue messages as part of the MPEG-2 transport stream. An example of such equipment is a rate altering remultiplexer, which performs complex processing of the stream. When the stream is demultiplexed and processed and then remultiplexed, it is very important to place the ITU-T J.181 cue message in the proper position relative to the video service and relative to nearby time base discontinuities. Such equipment may also be required to alter the message before retransmission.

##### II.3 Definitions

Throughout this appendix, the terms used have specific meanings. Because some of the terms that are defined in ISO/IEC [ITU-T H.222.0] have very specific technical meanings, the reader is referred to the original source for their definition. For terms used in this appendix, the brief definitions are given

below, and only apply to their use within this appendix. Outside this appendix, the definitions listed in clause 3 of this Recommendation take precedence.

**II.3.1 access unit:** The coded representation of a video picture or an audio frame [ITU-T H.222.0].

**II.3.2 analogue cue tone:** In an analogue 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.

**II.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.

**II.3.4 AVC:** Abbreviation for "Advanced Video Coding" and refers specifically to video compression standardized in [b-ITU-T H.264].

**II.3.5 break:** Avail or an actual insertion in progress.

**II.3.6 C3:** An audience measurement that is specified to include live viewing as well as DVR viewing up to 75 hours from the ad minute original broadcast time.

**II.3.7 cipher block chaining (CBC):** This is a specific method of encryption. It is one of the methods used in DES.

**II.3.8 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.

**II.3.9 cyclic redundancy check (CRC):** A method to verify the integrity of a transmitted Message.

**II.3.10 cueing message:** See Message.

**II.3.11 data encryption standard (DES):** A method for encrypting data with symmetric keys.

**II.3.12 digital video broadcasting (DVB):** An international consortium for the development of digital television systems.

**II.3.13 ITU-T J.181 cue message:** See Message.

**II.3.14 electronic code book (ECB):** This is a specific method of encryption. It is one of the methods used in DES.

**II.3.15 entitlement control message (ECM):** These are private conditional access information messages which specify control words and possibly other, typically stream-specific, scrambling and/or control parameters.

**II.3.16 entitlement management message (EMM):** These are private conditional access information messages which specify the authorization levels or the services of specific decoders. They may be addressed to single decoders or groups of decoders.

**II.3.17 event:** A splice event or a Viewing Event as defined below.

**II.3.18 in point:** A point in the stream, suitable for entry.

**II.3.19 message:** In the context of this appendix, a message is the contents of any splice\_info\_section.

**II.3.20 media presentation description (MPD):** A formalized description for a DASH Media Presentation for the purpose of providing a streaming service,

**II.3.21 out point:** A point in the stream, suitable for exit.

**II.3.22 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 [ITU-T H.222.0].

**II.3.23 packet identifier (PID):** A unique 13-bit value used to identify elementary streams of a program in a single or multi-program Transport Stream [ITU-T H.222.0].

**II.3.24 PID stream:** A stream of packets with the same PID within a transport stream.

**II.3.25 presentation time:** The time that a presentation unit is presented in the system target decoder [ITU-T H.222.0].

**II.3.26 program:** A collection of video, audio, and data PID streams which share a common program number within an MPTS [ITU-T H.222.0].

**II.3.27 program in point:** A group of PID stream In Points that correspond in Presentation Time.

**II.3.28 program out point:** A group of PID stream Out Points that correspond in Presentation Time.

**II.3.29 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.

**II.3.30 program splice point:** A Program In Point or a Program Out Point.

**II.3.31 registration descriptor:** Carried in the PMT of a program to indicate that, when signalling Splice Events, `splice_info_sections` is carried in a PID stream within this program. The presence of the Registration Descriptor signifies a program's compliance with ITU-T J.181.

**II.3.32 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 standard. Unless otherwise specified in the core part of ITU-T J.181, all reserved bits are set to '1'.

**II.3.33 splice event:** An opportunity to splice one or more PID streams.

**II.3.34 splice immediate mode:** A mode of the Cueing Message whereby the splicing device chooses 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.

**II.3.35 splice point:** A point in a PID stream that is either an Out Point or an In Point.

**II.3.36 TVE:** Acronym for "TV Everywhere", after being authenticated as a subscriber to an operator, it is the ability to view TV content on the internet in addition to on one's television.

**II.3.37 uimsbf:** Unsigned integer, most significant bit first.

**II.3.38 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.

## II.4 Abbreviations and acronyms

This appendix uses the following abbreviations and acronyms, in addition to those listed in clause 4 of this Recommendation:

|      |                                      |
|------|--------------------------------------|
| AMF  | Action Message Format                |
| AVC  | Advanced Video Coding                |
| CBR  | Constant Bit Rate                    |
| CDN  | Content Delivery Network             |
| DASH | Dynamic Adaptive Streaming over HTTP |
| ECM  | Entitlement Control Message          |
| EMM  | Entitlement Management Message       |
| ETV  | Enhanced TV                          |

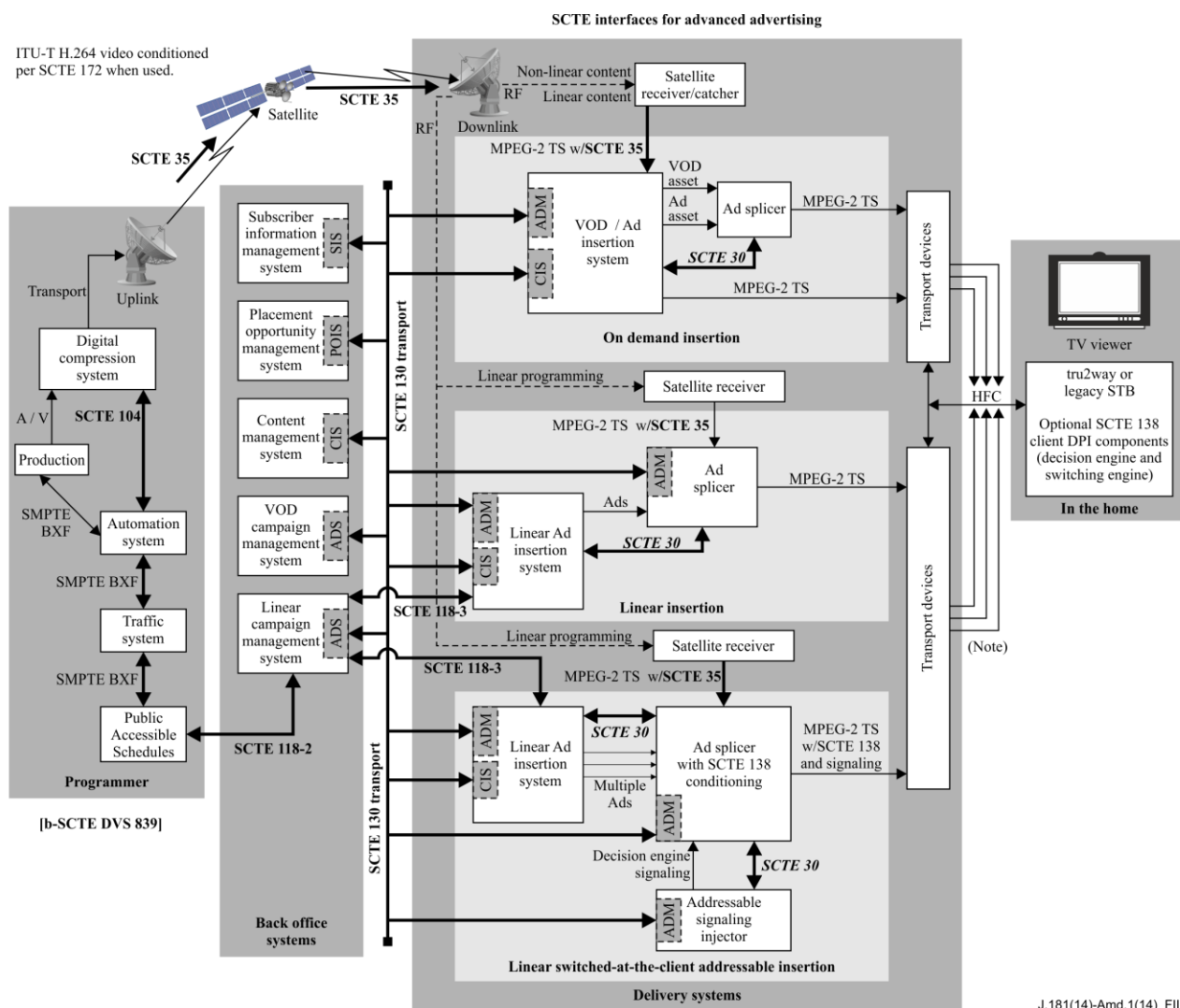
|       |                                          |
|-------|------------------------------------------|
| GOP   | Group of Pictures                        |
| HD    | High Definition                          |
| HLS   | HTTP Live Streaming                      |
| IDR   | Instantaneous Decoding Refresh           |
| MPD   | Media Presentation Description           |
| MVPD  | Multichannel Video Programming           |
| OATC  | Open Authentication Technology Committee |
| OCAP  | OpenCable Application Platform           |
| PVR   | Personal Video Recorder                  |
| RTMP  | Real Time Messaging Protocol             |
| SD    | Standard Definition                      |
| SPTS  | A Single Program Transport Stream        |
| TS    | Transport Stream                         |
| T-STD | Transport Stream System Target Decoder   |
| TVE   | TV Everywhere                            |
| VBR   | Variable Bit Rate                        |
| VBV   | Video Buffer Verifier                    |
| VOD   | Video on Demand                          |

## II.5 Overview

The Recommendation supports the splicing of MPEG-2 transport streams for the purpose of digital program insertion, which includes insertion of advertisement and other content types. An in-stream messaging mechanism is defined in the Recommendation to signal splicing and insertion opportunities. A splicing device is free to ignore splicing events signalled by the cue message because the message is not a command to splice, but is rather an indicator of the presence of an ad avail. Inserting content into the avail is optional.

A cue message can also signal other events such as program start/end, content identification and other program or advertisement related events. These events can be used for applications such as network PVR, TV everywhere, regional blackouts, live video on demand (VOD) capture or other advanced services.

As shown in Figure II.1, cue messages are received and acted upon in the cable system head-ends by splicer and server devices to initiate the insertion of local advertisements by splicing the ad bit stream (typically containing the commercial content) into the bit stream of programming content. Similar to [b-ITU-T J.280], this Recommendation does not differentiate between a splicing device and a server. When the terms "splicer" or "splicing device" are used, the meaning of the sentence may apply to a splicer/server combination as well. In actual practice it is common for ad servers (and not splicers) to parse, interpret and initiate action upon cue messages. Since splicer and server devices can be combined into one, this appendix often uses the term server/splicer to denote a device or set of devices that together perform both functions. Figure II.1 describes the overall functionality and interoperability associated with the systems that accomplish this.



J.181(14)-Amd.1(14)\_Fil.1

NOTE – This diagram provides a high level overview of the major entities, components, and interfaces that play a role in the SCTE Advanced Advertising ecosystem. It is intended for a technical audience familiar with cable technology but not with SCTE standards. As a graphical representation of an idealized, representative architecture, it is intentionally vague about implementation choices and lacks many details of an actual system.

**Figure II.1 – System overview**

In Figure II.1, a compliant MPEG-2 transport stream (either multi program transport stream (MPTS) or single program transport stream (SPTS)) is assumed for the programmer stream. 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 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 packet identifier (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. However, remultiplexers may need to obey certain constraints when they carry the cue message. These constraints are addressed in this Recommendation and are elaborated upon within this appendix.

This Recommendation does not address constraints on splicing devices and splice\_info\_table syntax and is not intended to guarantee seamless splicing.

By using presentation time stamp (PTS) to reference in points and out points in the program or component streams, frame accurate splicing is possible. This appendix offers advice on how to properly encode or condition the video to aid this. While the Recommendation does not require any stream conditioning, it can be advantageous to get the best splice results. [b-SCTE 172] defines constraints on insertion materials at in points and out points and also provides suggestions on splicer behaviour at in points and out points.

### II.5.1 Example decoder

A sample java decoder application written using the Netbeans framework is available. It allows entering a Base64 or hex string of the MPEG section and will decode it with a printout of the data. It will also convert the Base64 or hex to the opposite format. This open source sample code does not have extensive error checking. It may be useful to understand how to properly parse a message. A sample cue message reference decoder is available as a zipped GIT repository on the Society of Cable Telecommunications Engineers (SCTE) Standards site:

<http://www.scte.org/standards>

<http://www.scte.org/documents/standards/scte35-master.zip>

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

<https://netbeans.org/downloads/>

## II.6 Application guidelines

### II.6.1 Practical boundaries for splice\_time() in splice\_insert()

How far in advance of an impending splice must a splice\_insert() message be sent, relative to the picture it refers to, in order to be responded to by an ad insertion system? This Recommendation requires a minimum of four seconds of "pre-roll", which is also termed "arm time".

The arm time denotes the time a cue message precedes that actual insertion. The arm time is typically in the range of five to eight seconds. This is in line with the pre-roll time for analogue cue tones. The arm time duration cannot be so short that the avail passes by before the ad insertion system has time to respond. A minimum of four seconds is required for "start" cue messages (out\_of\_network\_indicator = 1) and a minimum of four seconds is recommended for "stop" cue messages (out\_of\_network\_indicator = 0). It is not necessary to send a "stop" cue message if "duration" is provided. Manufacturers should provide the option to generate a real-time dual tone multi-frequency (DTMF) "stop" cue tone at the actual switch point for legacy analogue insertion systems.

More thought about the possible consequences is required if it is desirable to extend the arm time beyond the recommended eight seconds. It is premature to specify a maximum arm time.

The splice\_info\_section itself does not have a PTS or decoding time-stamp (DTS) value associated with it, as it is not a packetized elementary stream (PES) structure.

### II.6.2 System delays

Most devices that operate on the video and audio components will delay the transport stream from the input to the output of the device. Encoders typically have delays of 0.5 to 2 or more seconds depending on if a low delay mode or multi-pass encoding mode is used. Splicers/statistical multiplexors also have delays in the same range. The satellite transmission of the transport stream incurs some delay as well. Converting a transport stream to an adaptive bit rate format such as [b-ISO/IEC 23009-1] will incur significant delays both on the preparation and playback/buffering side.

While most programmers prefer that the delay be as small as possible, small amounts of additional delay are usually only a concern on live programs that have call-in or simulcast issues.

### II.6.2.1 SCTE 104 insertion delays

The ability to insert messages with at least four seconds of pre-roll to allow the encoders to handle group of picture (GOP) structures properly, especially in advanced video coding (AVC) encoding, is a good practice. There are some events that are manually triggered and can lead to an immediate command being inserted, likely resulting in an inaccurate splice with MPEG-2 video or picture artifacts if AVC video is used.

One possible solution is to add a few seconds of high-definition serial digital interface (HD-SDI) delay before the SCTE 104 vertical ancillary (VANC) inserter and craft the SCTE 104 messages pre-roll value to account for that delay. This would have the effect of adding the additional pre-roll to the message, but the operator would see it as an immediate command.

### II.6.2.2 Encoder delays

Encoder delays should be handled by proper encoding resulting in frame accurate splicing. An encoder can immediately pass the ITU-T J.287 message through as a cue message into the output, essentially adding the encode delay to the pre-roll. In all cases, the encoder needs to consider the encode delay when converting the ITU-T J.287 frame to the presentation time stamp (PTS) time.

### II.6.2.3 Transmission delays

Transmission delays are a factor when generating cue messages; the variability of receipt and processing within receive sites is a major reason why cue messages use the in-stream PTS time.

### II.6.2.4 Splicer/multiplexor delays

Clause II.6.1 describes the suggested minimum pre-roll of four seconds for splice\_insert commands. This allows for one second of message processing before the three seconds minimum to send an ITU-T J.280 splice\_request message. In order to insure proper splice accuracy and quality these minimums need to be adhered to.

Some splicers and servers using MPEG-2 video will attempt to complete insertions when the timing is below the minimums, as this represents lost revenue. Some splicers use their delay and/or have faster processing capabilities to allow for lower minimums and theoretically could have slightly negative splice times due to delays.

## II.6.3 Splice time accuracy

Although precise splice times can be specified using both the program splice mode and component splice mode a server/splicer may choose to perform splices at other than these precise times. Since most splicing devices operate in the compressed domain they are typically constrained to only perform splices at specific locations in the stream if the splice is to meet the semantic and syntactic requirements of the MPEG-2 systems specification [ITU-T H.222.0] and the relevant video coding specification.

See clause 6 for stream requirements for splicing MPEG-2 video. Stream requirements for splicing AVC video are described in [b-SCTE 172]. While the details of out point requirements differ, both video codecs require that in points be intra-coded pictures with no stream dependencies on pictures prior to the in point (a closed GOP in MPEG-2, an instantaneous decoding refresh (IDR) in AVC). In time shifted environments (e.g., VOD), where ads may be inserted into content, the out and in points in the entertainment content may be coincident and, if coincident, they need to meet the requirements of both an in point and an out point.

If the signalled splice times are not aligned with these locations, the server/splicer may choose nearby in or out points, using the splice times as guidelines, if the resulting splice is to be syntactically seamless.

Splicing between AVC streams with differing GOP structures and/or buffer delays may place further requirements on the selection of splice points. Refer to [b-SCTE 172] for additional details.

In the case of audio the video presentation time of the splice will not in general align with the audio presentation time(s) and, therefore, corresponding audio splices may be executed at times that are "close" to the nearest video presentation time. Consequently, it may be necessary for a server/splicer to replace existing audio near the splice point with silence to ensure a clean transition.

Note that specific server/splicer implementations may place further, or fewer, restrictions on in and out points, depending upon their capabilities. Clause 6 of this Recommendation discusses various aspects of server/splicer capabilities and performance.

In order for a splice to be visually seamless, that is occurring at the desired location in the content with no visible transition (for example, black frames) the specified splice points should align with the GOP structure of the content. This can be accomplished by the following means:

- Encoders that insert splice points under control of an automation system can (see [b-ITU-T J.287]) adjust the GOP structure around the splice point so that it meets in or out point requirements without impacting video quality.
- Systems that insert cue messages into existing content may re-encode the GOP surrounding the desired splice point so that it meets in or out point requirements.
- Splice points will often coincide with scene breaks or changes. Encoders that employ scene change detection will often generate streams where GOP structures naturally align with splice points (for example, beginning each new scene with a closed GOP or IDR would enable scene breaks to be used as splice points after encoding).

If the content cannot be encoded such that the desired splice points align with the GOP structure, other techniques can be used such as encoding a number of black frames around the time of the desired splice point to minimize the visual effect if the server/splicer performs the splice at the nearest in or out point.

Some server/splicers may be capable of inserting black or repeat frames, and corresponding audio silence, between the nearest valid in or out point and the specified splice location to minimize the visual effect of adjusting the splice point. If this is used in an environment where an ad is replacing an existing ad in the network feed, the server/splicer is required to ensure that the combined length of the black frames and the inserted content does not exceed the length of the replaced ad. Similarly, there is the possibility that the length or GOP structure of the replacement ad may not precisely align with the original, in which case a splicer may insert black frames to align with the network feed GOP structure when returning from the ad.

Alternatively, some splicers may use their buffers and delay to look for an appropriate in/out point somewhere in their buffer. This technique should be used carefully and every effort to move in opposite directions each time a time-jump is made to even out the offset and not overflow or underflow the buffer.

Thus, behaviour when requested splice points do not align with actual splice points will be a factor that differentiates splicers.

#### **II.6.4 Splice\_event\_id usage and uniqueness**

NOTE – This section also applies to the segmentation\_event\_id carried in the segmentation\_descriptor.

Cue messages can be created by several means: from information embedded in the original audio/video source, by uplink event trigger systems or by head-end event trigger systems. When all of these sources are combined within a service there is potential for collisions in the splice\_event\_id value chosen for the cue message. The 32-bit splice\_event\_id should therefore be partitioned in the following way to allow each source a unique range of splice\_event\_id values as shown in Table II.1:

**Table II.1 – Splice\_event\_id**

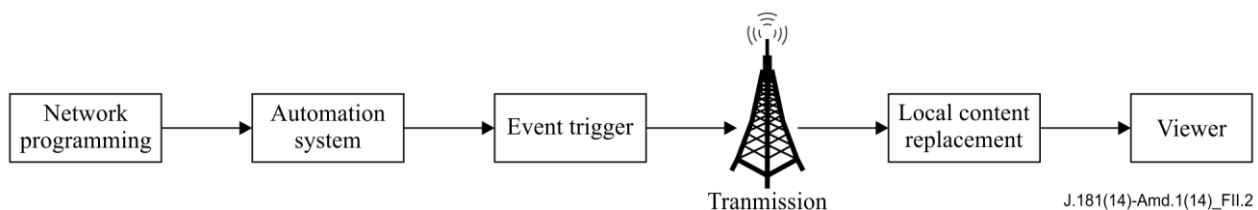
| Syntax                                                                                                                                                                                         | Bits | Type   |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|--------|
| Event_source (Note 1)                                                                                                                                                                          | 4    | uimsbf |
| Event_number (Note 2)                                                                                                                                                                          | 28   | uimsbf |
| NOTE 1 – Event_source is a user assigned number for the source of the cue message.<br>NOTE 2 – Event_number is a number chosen by the event source to identify an instance of the cue message. |      |        |

The splice\_event\_id serves to identify a particular instance of an opportunity to change the multiplex. At least two distinct splice\_events are required to perform one insertion: one to initiate the insertion and one to end it (i.e., unless the duration is included in the message signalling the splice-out).

Each splice event is required to have a unique splice\_event\_id. A splice\_event\_id cannot be reused before the splicing opportunity it describes (fully or in part) has completely passed. A splice\_event\_id is regarded to be in-use from the first cue message where it appears until about one second after the splice time that it is associated with. It may be reused for a new splice event immediately afterwards. It is not possible to use the same splice\_event\_id to signal an avail's start and stop if the stop is signalled before the start has been executed. However, it is possible to reuse the same splice\_event\_id if the first stop message is sent after the start has been executed.

The advertising industry uses additional information, besides the splice\_event\_id, to identify the actual material being played and the time of its insertion. This information includes: service name, time, ad-agency and spot number.

As long as the event\_source is unique for each point at which cue messages can be inserted in the following chain, the event identifiers will not collide. See cue message insertion points in Figure II.2.

**Figure II.2 – Cue message insertion points**

The following values for event\_source, as shown in Table II.2, are suggested:

**Table II.2 – Event\_source values**

| Source of cue message                           | Event_source value | Example splice_event_id ranges         |
|-------------------------------------------------|--------------------|----------------------------------------|
| Cue embedded in original source material        | 0                  | 0x00000000, 0x00000001 .... 0x0fffffff |
| Cue created by automation system switching      | 4                  | 0x40000000, 0x40000001 .... 0x4fffffff |
| Cue created by manual event trigger system      | 6                  | 0x60000000, 0x60000001 .... 0x6fffffff |
| Cue created by local content replacement system | 12                 | 0xC0000000, 0xC0000001 .... 0xCfffffff |

## II.6.5 Use of splice\_schedule() command

The splice\_schedule() command is intended for the announcements of a large schedule of splice events. It is not intended for the announcement of a single event. Current implementations of this Recommendation focus on the support of single events that do not rely on the use of the splice\_schedule() command.

## II.6.6 Component splice mode

While component splice mode is part of this Recommendation, there are currently no known implementations and it should be used with caution.

The primary reason for the component splice mode is to allow the replacement of, or the passage of individual elementary streams of any type in a manner consistent with the program content and the advertiser's intent.

For example, during a local commercial break, it might be useful to allow informational data (e.g., stock market returns) that is being streamed from the network provider to continue, while the viewer is being shown a locally inserted commercial. Conversely, with an advanced set-top box implementation, a viewer may be offered information to be downloaded for later review as part of the commercial. In this case, if the data content exceeds the duration of the commercial, the advertising data might be continued, although the program has resumed.

The methodology defined in this Recommendation allows some data types to be passed while allowing others to be blocked or replaced during a break. The decision of which data types to pass and which to block is made by the message inserter. A splicing device may choose to behave differently if it has more accurate information or is commanded to do so by an entity within the head-end.

It is the intention of the Recommendation, in both the program splice mode and the component splice mode, that the splicer should have a great deal of freedom in choosing the actual access units (both video and audio) on which to splice. Some equipment will provide frame accurate splices and some may not.

The signalled splice\_time may or may not be on an anchor frame or I-frame and the splicer may need to round off to a frame that makes sense for its capabilities.

There appears to be industry consensus that in program splice mode it makes the most sense to give a presentation time of a video frame and then let the splicer choose whichever audio frame is closest or which makes sense for its methods. This was not, however, specified in this Recommendation. It is believed, however, that the industry will develop the best way to use the tools (especially if it differs from what is stated here).

The component splice mode works the same way. A single splice\_time should be used (called the default splice\_time). It is optional to give a splice\_time for each component. There is some concern in the industry that the creator of a message will try to "help" the splicer by giving the exact time of all components and possibly complicate the job of the splicer (or the splicer may need to ignore the splice\_time for some components).

It is understood that [b-SMPTE 312] requires a time per-component to define the exact access unit for each component to be spliced. However, the cable industry is looking at a different usage for the time per-component approach. This Recommendation allows individual components to start or end at very different times from the normal beginning or ending of the break. For instance, a "data" component such as a Java applet may have to be downloaded to the set-top box a number of seconds prior to the ad in order to support the ad. This is the primary usage of the unique splice\_time for each component.

### II.6.6.1 Erroneous component splice commands

When a splice message (in component mode) specifies invalid component tags, the splice should be executed for all correctly identified components. The intent is to let insertion succeed if possible, even if the complete chain of devices results in a violation of the standard. Several potential cases are given below.

In case a device removes one of several audio streams after the splice message has been inserted, but fails to update the splice messages (either because it is unaware of them or for other reasons), the combination of stream and message at the server/splicer will be invalid. It is, however, still possible for the server/splicer to perform a valid insertion.

In case a device adds a component (e.g., a data channel), but fails to update the splice message (same possible reasons as above), it is still possible to perform a valid insertion.

Whenever a discrepancy between the actual stream and splice message exists, a server/splicer should perform the insertion in order to add error resilience to the chain.

### II.6.7 Pre-roll functionality – accomplishing a pre-roll function

A pre-roll function was necessary in the times of analogue ad insertion to allow time for a tape machine to get up to speed by the beginning of the ad avail. Thus, analogue cue tones were often sent about five to eight seconds prior to the avail (each network used its own chosen time and the consistency varied). The early arrival of the cue tone has remained unchanged during the days of hybrid ad insertion where tape machines are replaced by MPEG servers and decoders, but the insertion was still analogue. This early cue tone is useful for digital ad insertion as well. It gives the ad insertion equipment time to access its ad database, to determine which ad to play next, to start accessing its disk drives and to fill the MPEG decoder pipeline. This, however, takes much less time and allows for pre-rolls typically in the four to five second range. However, legacy analogue systems remain in operation and should be considered until the industry "sunsets" the requirement to provide DTMF cue tones and differing "splice return" with pre-roll for digital splices versus actual end-time switch points for analogue splices.

The splice\_insert() command is constructed such that a pre-roll time can be used. It is also possible to repeat a splice\_insert() command to increase the error resilience of the system.

The simplest variant is to send the splice\_insert() command once - about eight seconds prior to an avail. This is similar to the analogue insertion case.

An enhancement is to send the splice\_insert() command three times: at about eight, six and four seconds prior to the avail. This increases the redundancy and prevents lost insertion opportunities. A lost avail nationwide is a very expensive error. In the case of multiple messages denoting a single avail, the splice\_event\_id field in all such messages is required to be identical. It is allowable that the splice time can be different from message to message in the case where the first splice time is approximate and subsequent messages supply a more accurate splice time. Ad servers should only act on the last time received that meets the pre-roll constraints.

While servers should be reading the splice\_event\_id and handling the messages as described in the above paragraph, other implementations are known to exist. Some ad servers, once they receive a valid cue message and have ads to play, will block all subsequent cue messages until playback is finished. If they receive a valid cue message with no ads to play, they may see the subsequent cue messages and incorrectly advance to the next avail within the window. Some ad servers have an adjustable timer to block subsequent cues for a number of seconds to prevent this from happening.

### II.6.8 Conditional access and cue encryption

There are no known implementations of the following sections. The message encryption is typically handled by the proprietary conditional access systems in the satellite feed.

### II.6.8.1 What to encrypt

The format of the splice information section allows for the payload to be optionally encrypted. This includes all data from the "splice\_command\_type" through "E\_CRC\_32". This mechanism allows for any current or future commands to be protected. The reasons for protection can range from anti-piracy (e.g., commercial killers), malicious tampering of the data, and privacy when using cascaded ad insertion devices.

The specification requires that an encrypted CRC be present so that a receive device is able to verify that the encrypted data has not been changed since its origination. This could be due to noise, but normally this type of corruption is detected by the standard CRC. The encrypted CRC has the primary purpose as a signature to detect that the receiver is authorized to receive the message (i.e., that they have the correct control word). It is also used to detect wilful modification of the encrypted data. This CRC is not present when the section is sent in-the-clear.

### II.6.8.2 Operation in a cue insertion device

A cue insertion device is used to create splice\_info\_sections and insert them into a transport stream. When encryption is desired, the cue inserter uses a fixed key entered by an operator, plus the algorithm selected, to encrypt the section before being transmitted.

The cue insertion device should be able to maintain multiple simultaneous keys for the same program. The reason is that each ad inserter in a cascade could require one or more different keys to decrypt the splice\_info\_section.

A cue insertion device is only required to implement one of the standard encryption methods. In recommended practice, however, the cue insertion device should implement all standard algorithms.

### II.6.8.3 Operation in an ad insertion device

An ad insertion device consumes splice\_info\_sections. When a splice\_info\_section is encrypted, the ad inserter will only act upon the section if the decryption is successful. This is determined by checking the encrypted CRC before interpreting the data. A failure will usually mean that the device is simply not authorized to interpret the information that the section contains.

The ad inserter is expected to allow the encrypted splice\_info\_section to pass through it to the next device. This is true whether the decryption was successful or not. For security reasons the ad inserter never releases the contents of a decrypted section.

Once the section has been decrypted, the device may act on the information it contains, or discard it. This is the same operation that would be performed if the section had been sent in-the-clear.

The ad inserter device should be able to hold 256 different decryption keys. The cw\_index field in the section indicates which of these decryption keys should be used. The method of key interchange is out of scope for this appendix.

For normal operation, it is expected that one or two keys will be used for any one pair of send/receive devices. There is provision for a large number of keys to allow an ad inserter to connect to multiple programs in a transport stream, or to allow cascaded ad inserters to use different ranges of cw\_indexes when connecting to the same program.

An ad insertion device should implement all defined decryption algorithms in this Recommendation. This allows it to receive encrypted sections from any cue insertion device. Any cue insertion device may choose any standard algorithm to protect the section, and the ad inserter is required to be able to decrypt any message it is authorized to receive.

## II.6.8.4 Theory of operation

### II.6.8.4.1 Encryption versus scrambling

Scrambling refers to the use of the "standard" data encryption standard (DES) or digital video broadcasting (DVB) common scrambling algorithm that is used for video and audio services.

When deciding how best to encrypt the `splice_info_section`, consider that the ad insertion functionality is typically not located in a set-top box. There has been a provision made for this model, but it is not the primary model. Most systems will employ an ad insertion device in a digital head-end (or some other distribution point). These standalone devices are typically computers. Using a descrambler would require a custom circuit designed to descramble the stream. Also, since the entire transport packet is scrambled, the ad insertion device has no access to the header data as it does with the current model. Some of the information, such as `pts_adjustment`, may need to be modified even when the section is not descrambled. There are similar considerations when creating the stream.

The model used for the `splice_info_section` is closer to the entitlement control message (ECM) model, than the elementary stream model. An ECM section is independently encrypted (and decrypted), and authorized by some external mechanism. In the case of the ECM, it is usually the entitlement management message (EMM) that controls authorization. In the case of the `splice_info_section`, it is a manual authorization.

The fixed key model was chosen for simplicity. The key distribution was left undefined and may use any mechanism that gets the key to the decryptor in a secure and timely manner. It is conceivable that a committee may standardize on some type of ECM and/or EMM to distribute the keys in the transport stream.

### II.6.8.4.2 Standard encryption methods

This Recommendation provides for three types of encryption algorithms. All three algorithms use the DES decryptor as the basic building block. For triple DES, the DES encryptor is also required. Software implementations of the DES algorithm are readily available.

#### II.6.8.4.2.1 Section stuffing

All DES type algorithms (block encryptors rather than stream encryptors) require the length of the data to be an exact multiple of eight bytes. To achieve this, stuffing is often required. The `alignment_stuffing` field may be present whether the section is encrypted or not. The number of bytes of stuffing can be determined during the interpretation of the data. The stuffing bytes are never used, so they may take on any value.

To determine the length of the stuffing consider the fact that the total length is known from the `section_length` field. Knowing the size of the header, it is possible to determine the start of a `splice_command`. Every command has a size completely determined by the syntax within the command itself. The length is:

$$\langle \text{section\_length} \rangle + 3 - \langle \text{end\_of\_command} \rangle - \langle \text{length\_trailing} \rangle$$

where `<length_trailing>` is 4 for non-encrypted sections and 8 for encrypted sections.

Algorithmically, it is easier to simply ignore the stuffing. Work forward to decrypt and interpret the command, and work backward from the end to find the CRCs.

#### II.6.8.4.2.2 DES ECB algorithm

DES electronic codebook (ECB) requires a 56-bit key plus 8 parity bits to make up a complete 64-bit key, which is then used by the algorithm to encrypt or decrypt a stream. The DES algorithm is symmetric. The same key is used in both the encryptor and the decryptor. The actual encryption and decryption algorithms are slightly different to allow this symmetry to work.

It is suggested that the full 64-bit key be distributed between the two devices. The key will be entered as a 16-digit hexadecimal number. For example, 0x123456789ABCDEF0 would be distributed. In engineering notation, the leftmost digit is the most significant digit, and the most significant bit (MSB) of this nibble is the MSB of the key (i.e., bit 63). Cipher algorithms generally use FIPS notation to represent keys. In this case, the leftmost bit of the key is numbered bit 1, through bit 64 on the right. Therefore, bit 63 of the distributed key value will be loaded into bit 1 of the initial key register. Similarly, bit 0 of the key value is loaded into bit 64 of the key register.

The ECB method of encryption uses the *same* key for every 8-byte block of the original message. Figure II.3 gives an example of a simple 3-block message being encrypted. The arrows represent operations. Across the top, the key is being loaded into the key register. Side to side, the data is being shifted into the encryption register, and across the bottom, the DES algorithm is being applied.

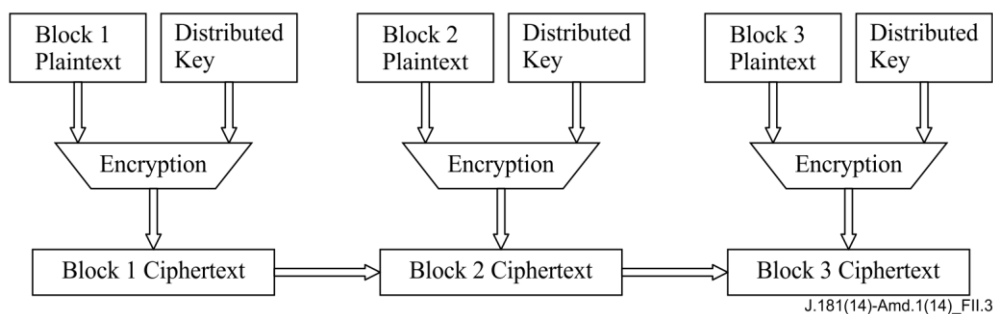


**Figure II.3 – DES ECB example**

#### II.6.8.4.2.3 DES cipherblock chaining algorithm

DES cipherblock chaining (CBC) requires a 56-bit key plus 8 parity bits to make up a complete 64-bit key that is used by the algorithm to encrypt or decrypt a stream. The same bit ordering rules and key distribution methods can be used for CBC that were used for the DES ECB algorithm described in clause II.6.8.4.2.2.

The CBC method of encryption uses a different key for every 8-byte block of the original message as shown in Figure II.4.

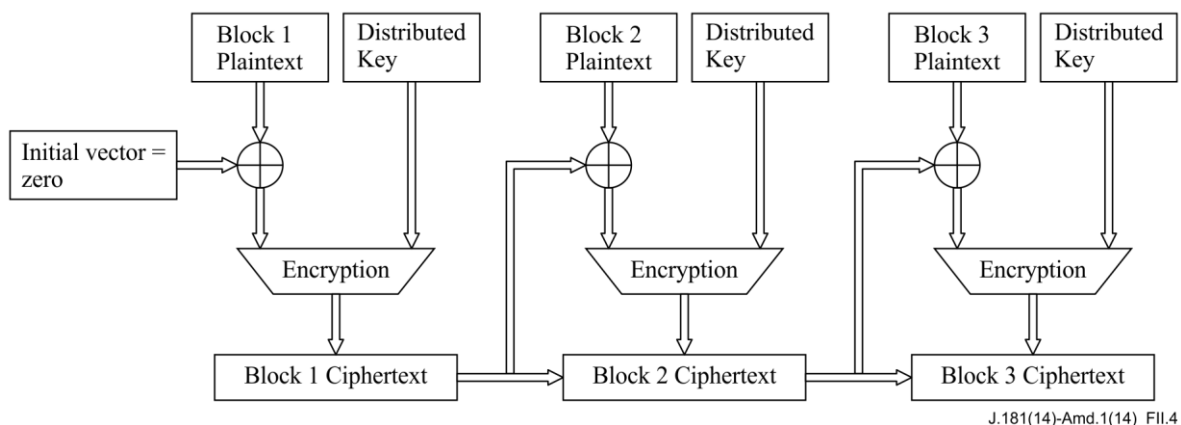


**Figure II.4 – DES CBC encryption example**

Figure II.4 shows that the plaintext data is modified by the result of the previous encryption block. For the first block, an "initial vector" is needed to apply to the exclusive-or block. It turns out that the initial vector provides no extra security, whether it is known or not. Thus, for this system, the initial vector can be zero, which removes the need for it to be distributed with the key.

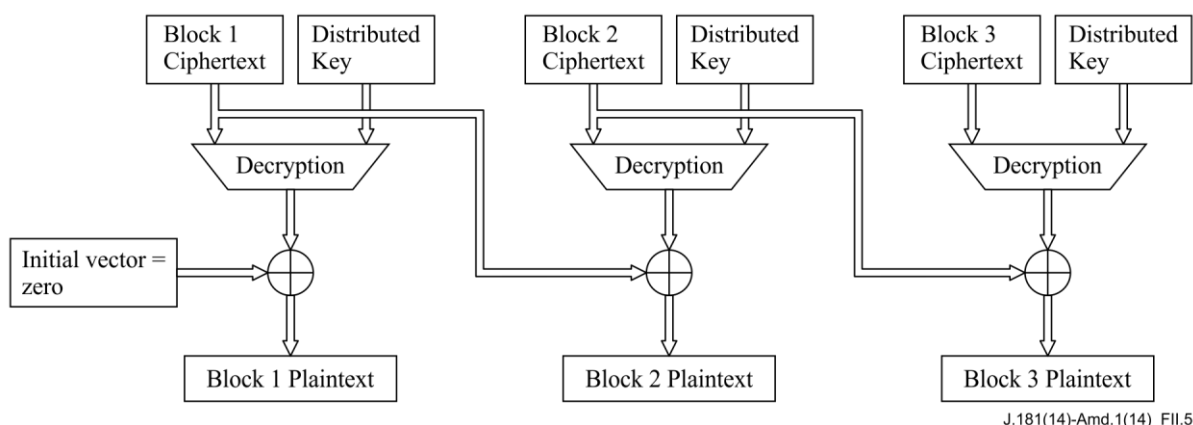


**Figure II.5 – DES CBC decryption example**

Figure II.5 above shows that the DES CBC decryption is slightly different from encryption. The exclusive-or block requires the previous block of encrypted data to arrive at the proper cipher key for the DES algorithm. In the decryptor, this encrypted block is derived from the transmitted data.

#### II.6.8.4.2.4 Triple DES ECB mode (EDE method) algorithm

Triple DES uses the standard DES algorithm, but applies the algorithm three times for each block of input data. This provides a significant security enhancement. The disadvantage is that distributing a 192-bit key (i.e., three 64-bit keys) is needed to obtain the security enhancement.

Using the combination of two algorithms and three passes (encryption and decryption are different), it is possible to generate eight different triple DES variants<sup>1</sup>. The variants are designated by a three-letter code. Of these variants, the most common is selected for use in a splicing system. This variant is designated the encrypt-decrypt-encrypt (EDE) method, referring to the fact that pass one uses encryption, pass two uses decryption, and pass three uses the encryption algorithm again. For simplicity, the DES ECB mode of operation is used for the triple DES algorithm<sup>2</sup>. See Figure II.6.

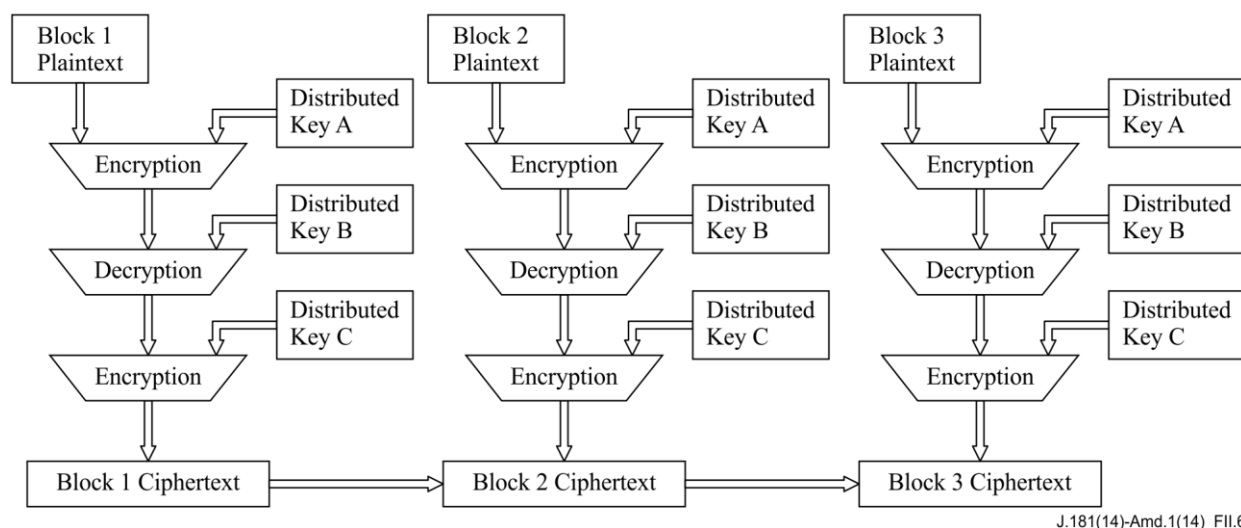


**Figure II.6 – Triple DES ECB encryption example**

<sup>1</sup> Triple DES also has two key variants. This can be simulated by making KEY A == KEY C.

<sup>2</sup> Triple DES can also use the cipherblock chaining mode, but this not one of the standard methods.