

SCTE[®] | STANDARDS

Digital Video Subcommittee

AMERICAN NATIONAL STANDARD

ANSI/SCTE 30 2021

Digital Program Insertion Splicing API

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards and Operational Practices (hereafter called “documents”) are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interoperability, interchangeability, best practices, and the long term reliability of broadband communications facilities. These documents shall not in any way preclude any member or non-member of SCTE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE members.

SCTE assumes no obligations or liability whatsoever to any party who may adopt the documents. Such adopting party assumes all risks associated with adoption of these documents and accepts full responsibility for any damage and/or claims arising from the adoption of such documents.

NOTE: The user’s attention is called to the possibility that compliance with this document may require the use of an invention covered by patent rights. By publication of this document, no position is taken with respect to the validity of any such claim(s) or of any patent rights in connection therewith. If a patent holder has filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license, then details may be obtained from the standards developer. SCTE shall not be responsible for identifying patents for which a license may be required or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Patent holders who believe that they hold patents which are essential to the implementation of this document have been requested to provide information about those patents and any related licensing terms and conditions. Any such declarations made before or after publication of this document are available on the SCTE web site at <https://scte.org>.

All Rights Reserved
©2021 Society of Cable Telecommunications Engineers, Inc.
140 Philips Road
Exton, PA 19341

Document Types and Tags

Document Type: Specification

Document Tags:

- | | | |
|--|------------------------------------|--|
| ☐ Test or Measurement | ☐ Checklist | ☒ Facility |
| ☐ Architecture or Framework | ☐ Metric | ☐ Access Network |
| ☒ Procedure, Process or Method | ☐ Cloud | ☐ Customer Premises |

Document Release History

Release	Date
SCTE 30 2001	11/6/2001
SCTE 30 2006	2/13/2006
SCTE 30 2009	4/10/2009
SCTE 30 2015	4/6/2015
SCTE 30 2017	6/13/2017

Note: Standards that are released multiple times in the same year use: a, b, c, etc. to indicate normative balloted updates and/or r1, r2, r3, etc. to indicate editorial changes to a released document after the year.

Table of Contents

Title	Page Number
NOTICE.....	2
Document Types and Tags	3
Document Release History	3
Table of Contents	4
1. Introduction.....	7
1.1. Executive Summary	7
1.2. Scope	7
1.3. Benefits	7
1.3.1. Improvements in ad timing synchronization.....	7
2. Normative References	8
2.1. SCTE References	8
2.2. Standards from Other Organizations	8
2.3. Other Published Materials	8
3. Informative References	9
3.1. SCTE References	9
3.2. Standards from Other Organizations	9
3.3. Other Published Materials	9
4. Compliance Notation	9
5. Abbreviations and Definitions.....	9
5.1. Abbreviations.....	9
5.2. Definitions.....	10
6. Introduction.....	11
6.1. System Block Diagram.....	12
6.2. Arbitration Priorities	14
6.3. Abnormal Terminations	16
6.4. Splicing Requirements	16
6.5. Communication	16
7. API Syntax.....	17
7.1. Splicing_API_Message Syntax	17
7.2. Conventions and Requirements	19
7.3. Initialization.....	19
7.3.1. Init_Request Message	19
7.3.2. Init_Response Message	20
7.4. Embedded Cueing Messages	20
7.4.1. Cue_Request Message	21
7.5. Splice Messages	21
7.5.1. Splice_Request Message	22
7.5.2. Splice_Response Message	24
7.5.3. SpliceComplete_Response Message	25
7.6. Alive Messages	26
7.6.1. Alive_Request Message	26
7.6.2. Alive_Response Message	26
7.7. Extended Data Messages	27
7.7.1. ExtendedData_Request Message	27
7.7.2. ExtendedData_Response Message	27
7.8. Abort Messages	28
7.9. Abort_Request Message.....	28
7.10. Abort_Response Message.....	28
7.11. TearDownFeed_Request Message	28

7.12.	TearDownFeed_Response Message	29
7.13.	Requesting Configuration Settings	29
7.13.1.	GetConfig_Request Message	29
7.13.2.	GetConfig_Response Message	29
7.14.	General_Response Message	29
8.	Additional Structures	29
8.1.	Version	29
8.2.	Hardware_Config	30
8.3.	splice_elementary_stream()	33
8.4.	time() Field Definition	34
8.5.	splice_API_descriptor() Field Definition	35
8.5.1.	playback_descriptor() Field Definitions	35
8.5.2.	muxpriority_descriptor() Field Definitions	36
8.5.3.	missing_Primary_Channel_action_descriptor() Field Definitions	37
8.5.4.	port_selection_descriptor() Field Definitions	37
8.5.5.	asset_id_descriptor() Field Definitions	39
8.5.6.	create_feed_descriptor() Field Definitions	40
8.5.7.	source_info_descriptor() Field Definitions	41
9.	Time Synchronization	42
9.1.	Introduction [Informative]	42
9.2.	Splicing Compressed Streams	43
9.3.	NTP Time Synchronization	44
9.4.	PTP Time synchronization	48
10.	System Timing	50
10.1.	DPI Splice Signal Flow	50
10.2.	DPI Splice Initiation Timeline	51
Appendix A	Result Codes	53
Appendix B	Example use of Logical Multiplex Type 0x0006 and the port_selection_descriptor()	55

List of Figures

Title	Page Number
Figure 1 - Single Server / Single Splicer	12
Figure 2 - Multiple Servers / Multiple Splicers	13
Figure 3 - OverridePlaying Flag Operation	15
Figure 4 - Normative NTP Configuration	45
Figure 5 - Informative NTP Configuration 1	46
Figure 6 - Informative NTP Configuration 2	47
Figure 7 - Normative PTP Configuration	49
Figure 8 - Single Event Splice	50
Figure 9 - Multiple Event Splice	51
Figure 10 - DPI Splice Initiation Timeline	52

List of Tables

Title	Page Number
Table 1 - Splicing_API_Message	17
Table 2 - MessageID Values	18
Table 3 - Init_Request_Data	20

Table 4 - Init_Response_Data	20
Table 5 - Cue_Request_Data	21
Table 6 - Splice_Request_Data	23
Table 7 - Splice_Response_Data	24
Table 8 - SpliceComplete_Response_Data	25
Table 9 - Alive_Request_Data	26
Table 10 - Alive_Response_Data	26
Table 11 - Alive_Response Message States	26
Table 12 - ExtendedData_Request_Data	27
Table 13 - ExtendedData_Response_Data	27
Table 14 - Abort_Request Data	28
Table 15 - Abort_Request Data	28
Table 16 - GetConfig_Response Data	29
Table 17 - Version()	30
Table 18 - Hardware_Config()	30
Table 19 - Logical Multiplex Type	30
Table 20 - Type 0x0006 Structure	31
Table 21 - Type 0x0007 Structure	32
Table 22 - Splice_Elementary_Stream()	34
Table 23 - Time()	35
Table 24 - Splice_Api_Descriptor()	35
Table 25 - Playback_Descriptor()	36
Table 26 - BitrateRule Values	36
Table 27 - Muxpriority_Descriptor()	36
Table 28 - Missing_Primary_Channel_Action_Descriptor ()	37
Table 29 - Pv4 Port_Selection_Descriptor ()	38
Table 30 - IPv6 Port_Selection_Descriptor ()	39
Table 31 - Asset_Id_Descriptor ()	40
Table 32 - Create_Feed_Descriptor ()	41
Table 33 - Source_Info_Descriptor ()	42
Table 34 - Frame Rate Codes (Informative)	42

1. Introduction

1.1. Executive Summary

This Application Program Interface (API) creates a standardized method of communication between Servers and Splicers for the insertion of content into any MPEG-2 Output Multiplex.

1.2. Scope

This Application Program Interface (API) creates a standardized method of communication between Servers and Splicers for the insertion of content into any MPEG-2 Output Multiplex in the Splicer. This API is flexible enough to support one or more Servers attached to one or more Splicers. Digital Program Insertion includes content such as spot advertisements of various lengths, program substitution, public service announcements or program material created by splicing portions of the program from a Server.

1.3. Benefits

1.3.1. *Improvements in ad timing synchronization*

This section discusses some benefits of proper ad insertion time synchronization and why it is important for a digital ad insertion system to maintain proper time synchronization.

If non cue based ad insertion (i.e. wall clock triggered) is being done, then the implementer should use whatever methods meet the application requirements.

In cue based insertion systems many content providers and distributors have issues associated with implementing correct ad insertion splice timing in the field. Although [SCTE 35] messages are capable of containing frame-accurate splicing information, that information is not always present. Additionally, splice times are usually not well monitored and must be periodically readjusted. In fact, the method selected for time synchronization can further exacerbate insertion splice timing challenges.

Currently operators must periodically adjust local ad insertion timing for one of the following reasons:

- Network programming is clipped because local ads are switched in too early.
- Frames of underlying national ads are visible because local ads switch in too late.
- Frames of underlying national ads are visible because local ads switch back to the network too early.
- Network programming is clipped because local ads switch back to the network too late.
- The underlying content used by the content provider is not the exact size of the signaled break duration.

Periodic timing readjustment is not ideal as most Splicers insert into a compressed video stream which only allows splices on certain key frames. Any adjustment made by the operator will typically not be frame-accurate (usually ± 2 frames), and depending on how close the splice is to a key frame, adjustments may cause large, indeterminate jumps.

This standard requires a certain amount of synchronization between the Splicer and Server time references where cues are described in PTS time. In doing so, the time synchronization system must be able to keep the Splicer and Server within ± 15 ms of each other.

The bit stream representing the primary channel is subject to various delays, which may include upstream splicing, satellite links, and other transmission and conditioning processes. These delays may range

milliseconds to seconds. However, these delays do not affect the accuracy of a cue message embedded in the primary channel. The cue message uses the PCR to indicate the correct time of insertion, so it retains its original accuracy relative to the content.

The Server providing the insertion-channel content is aware of clock time (UTC) only, and the insertion windows with which it has been programmed are relative to clock time. However, the Server depends on the Splicer to indicate the exact moment to begin streaming the content. When the Splicer receives the program bit stream, all delays to that stream have occurred.

2. Normative References

The following documents contain provisions which, through reference in this text, constitute provisions of this document. The editions indicated were valid at the time of subcommittee approval. All documents are subject to revision and, while parties to any agreement based on this document are encouraged to investigate the possibility of applying the most recent editions of the documents listed below, they are reminded that newer editions of those documents might not be compatible with the referenced version.

2.1. SCTE References

[SCTE 35] ANSI/SCTE 35 2020, Digital Program Insertion Cueing Message.

[SCTE 128-1] SCTE 128-1 2020 AVC Video Systems and Transport Constraints for Cable Television

[SCTE 54] SCTE 54 2020 Digital Video Service Multiplex and Transport System Standard for Cable Television.

2.2. Standards from Other Organizations

[13818-1] ITU-T Rec. H.222.0 / ISO/IEC 13818-1 (2021), Information Technology ---- Generic Coding of Moving Pictures and Associated Audio Information: Systems

[13818-2] ITU-T Rec. H.262 / ISO/IEC 13818-2 (2012), Information Technology ---- Generic Coding of Moving Pictures and Associated Audio Information: Video

[IEEE 1588] 2008 - Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

[SMPTE ST2059-1] ST 2059-1 2021 - Profile for Use of IEEE-1588 Precision Time Protocol in Professional Broadcast Applications

[SMPTE ST2059-2] ST 2059-2 2021 - Profile for Use of IEEE-1588 Precision Time Protocol in Professional Broadcast Applications

[AES67] AES67-2018: AES standard for audio applications of networks - High-performance streaming audio-over-IP interoperability

[EG 40] EG 40:2016 Conversion of Time Values Between SMPTE ST 12-1 Time Code, MPEG-2 PCR Time Base and Absolute Time

2.3. Other Published Materials

No normative references are applicable.

3. Informative References

The following documents might provide valuable information to the reader but are not required when complying with this document.

3.1. SCTE References

[SCTE 67] ANSI/SCTE 67, Digital Program Insertion Cueing Message for Cable – Interpretation for SCTE 35.

3.2. Standards from Other Organizations

[RFC 3810] IETF RFC3810 --- Multicast Listener Discovery Version 2 (MLDv2) for IPv6

[RFC 5905] IETF RFC5905 --- Network Time Protocol Version 4: Protocol and Algorithms Specification

3.3. Other Published Materials

M. Kar, S. Narasimhan, and R. Prodan, “Local Commercial Insertion in the Digital Headend”, Proceedings of NCTA 2000 Conference, New Orleans, USA.

“Cable Advertising”, white paper, March 1997, Cable Television Laboratories, Louisville, CO.

4. Compliance Notation

<i>shall</i>	This word or the adjective “ <i>required</i> ” means that the item is an absolute requirement of this document.
<i>shall not</i>	This phrase means that the item is an absolute prohibition of this document.
<i>forbidden</i>	This word means the value specified <i>shall</i> never be used.
<i>should</i>	This word or the adjective “ <i>recommended</i> ” means that there <i>may</i> exist valid reasons in particular circumstances to ignore this item, but the full implications <i>should</i> be understood and the case carefully weighed before choosing a different course.
<i>should not</i>	This phrase means that there <i>may</i> exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications <i>should</i> be understood and the case carefully weighed before implementing any behavior described with this label.
<i>may</i>	This word or the adjective “ <i>optional</i> ” indicate a course of action permissible within the limits of the document.
deprecated	Use is permissible for legacy purposes only. Deprecated features <i>may</i> be removed from future versions of this document. Implementations <i>should</i> avoid use of deprecated features.

5. Abbreviations and Definitions

5.1. Abbreviations

AAL	ATM adaptation layer
ADI	Asset Distribution Interface

API	Application Program Interface (also Application Programming Interface)
ASCII	American Standard Code for Information Interchange
ATM	asynchronous transfer mode
CNN	Cable News Network
CRC	cyclic redundancy check
DVS	Digital Video Subcommittee
DVB-ASI	Digital Video Broadcast – Asynchronous Serial Interface
GPS	Global Positioning System
ID	Identifier
IEC	International Electrotechnical Commission
IGMP	Internet group management protocol
IP	Internet protocol
IPv4	Internet protocol version 4
IPv6	Internet protocol version 6
ISO	International Organization for Standardization
ITU	International Telecommunication Union
kHz	kilohertz
MAC	media access control
MLD	Multicast Listener Discovery
MPEG	Moving Picture Experts Group
MPTS	Multi-Program Transport Stream
NCTA	National Cable & Telecommunications Association
NTP	Network Time Protocol
PAT	Program Association Table
PCR	Program Clock Reference
PID	Packet Identifier
PMT	Program Map Table
PSI	Program specific information
SAPI	Source Access Point Identifier
SCTE	Society of Cable Telecommunications Engineers
SPTS	Single Program Transport Stream
tcimbsf	twos complement integer, most significant bit first
TCP/IP	Transmission Control Protocol/ Internet Protocol
UDP	user datagram protocol
uimbsf	unsigned integer, most significant bit first
UTC	Coordinated Universal Time
VCI	virtual channel identifier
VOD	video on demand
VPI	virtual path identifier

5.2. Definitions

Definitions of terms used in this document are provided in this section. Defined terms that have specific meanings are capitalized. When the capitalized term is used in this document, the term has the specific meaning as defined in this section.

API Connection	A TCP/IP socket connection between a Server and a Splicer for transferring API messages.
----------------	--

Back-To-Back Insertion	Two or more temporally contiguous Sessions without return to the Primary Channel between Sessions.
Channel	A Channel is a synonym for a “Service” in DVB terminology, or a “Program” in MPEG terminology.
Deprecated	Use is permissible for legacy purposes only. Deprecated features may be removed from future versions of the standard. Implementations should avoid use of deprecated features.
Insertion Channel	The Insertion Multiplex Channel(s) that replace the Primary Channel in whole or in part of the duration for a splice event.
Insertion Multiplex	This is the source of the Insertion Channel. A Multiplex produced by a Server may under some circumstances exclude PSI information, thus it is understood that this Multiplex may be a non-compliant MPEG-2 transport stream.
Multiplex	A Multiplex is a collection of one or more channel(s) that may include the associated service information. A Multiplex is an MPEG-2 Transport Stream with the possible exception of an Insertion Multiplex.
Output Channel	The Channel that is produced at the output of the Splicer.
Output Multiplex	The MPEG-2 Transport Stream produced by multiplexing one or more Output channels.
Primary Channel	The Primary Multiplex Channel that is replaced in whole or in part. A single Primary Channel may result in multiple Output Channels.
Primary Multiplex	This is the source of the Primary Channel(s).
Server	The device that originates the Insertion Channel(s) to be spliced into the Primary Channel(s). This device communicates with the Splicer about when and what to splice.
Session	A Session is the insertion of content (such as spot advertisements of various lengths, program substitution, public service announcements, or program material created by splicing portions of the program from a Server). Each Session is identified by a unique SessionID .
Splice-in	The splice at the start of the insertion. This happens at the time specified in the Splice_Request message.
Splice-out	The splice at the end of the insertion. The expected insertion end time is calculated by adding the start time and the duration specified in the Splice_Request message, however this may occur earlier due to error conditions.
Splicer	The device that splices the Insertion Channel(s) into the Primary Channel(s). It may receive [SCTE 35] cue messages. This device also communicates with the Server about when and what to splice.

6. Introduction

Note: This version of the standard has a **Revision_Num** of 2.

6.1. System Block Diagram

This API may be used with many different configurations of Server(s) and Splicer(s). This API focuses on the single Server, single Splicer configuration shown in Figure 1. However, this can be expanded to the multiple Servers, multiple Splicers configuration as shown in Figure 2.

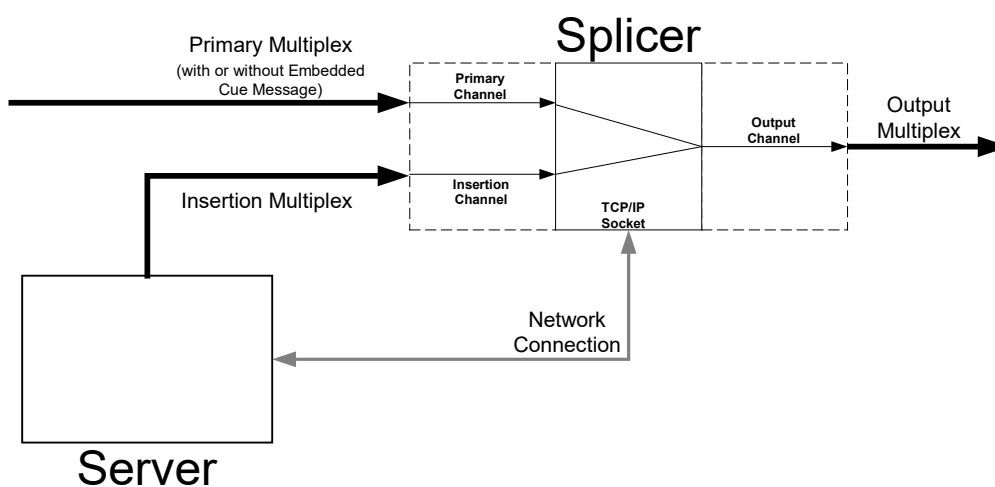


Figure 1 - Single Server / Single Splicer

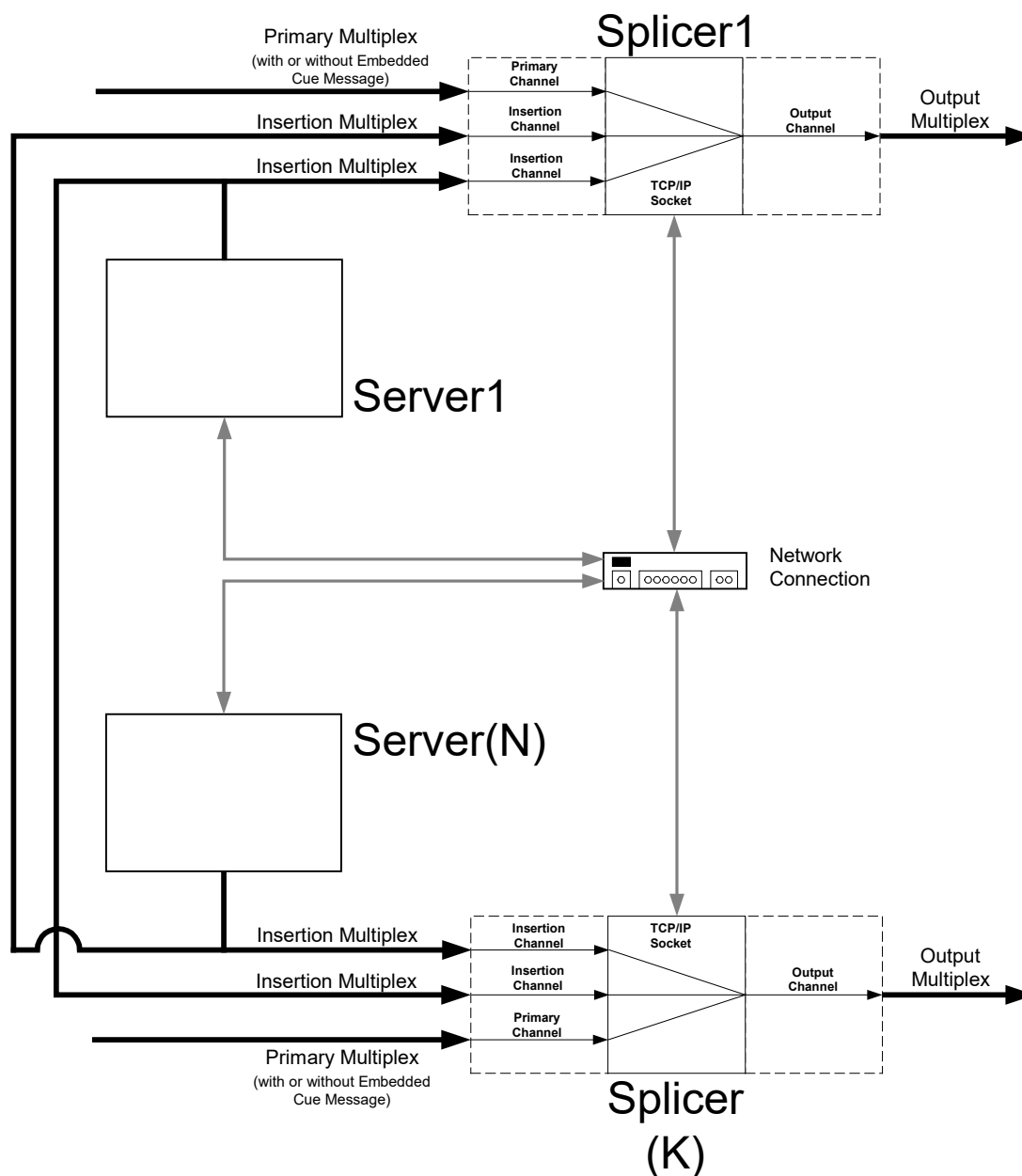


Figure 2 - Multiple Servers / Multiple Splicers

The model referenced in this API has one or more Splicers with one or more Multiplex inputs. The Splicer logically separates the Channel(s) in the Multiplex(s) and presents these Channel(s) to a switch. This switch is capable of mapping any input to any Output Channel. The initial configuration maps Primary Channel(s) to the Output Channel(s). The Server may then direct the Splicer to switch from a Primary Channel to an Insertion Channel for a specified duration. It may then direct the Splicer to switch to another Insertion Channel following the initial switch. The Splicer may then create an MPEG-2 Transport Stream produced by multiplexing one or more Output channels that shall be compliant with [SCTE 54].

Logically, a splice involves two input Channels and one output Channel. The Splicer is responsible for joining the various elementary streams (audio, video and data) together. The optimal splice point may occur at slightly different times for each elementary stream, so the Splicer should perform the splice that will supply the best quality output. Splicing may not always be performed from the Primary Channel, i.e. programming network, to the Insertion Channel, i.e. spot advertisement, and back to the Primary Channel. The Splicer may splice content that is stored solely on the server and arrives over a single input Multiplex. It is possible to use this API in a situation where a server has one Multi-Program Transport Stream (MPTS) output that contains program and interstitial material and uses the splicer to create proper splices between the content.

This API supports all combinations of single and multiple Servers communicating with single and multiple Splicers. A separate API Connection is associated with each Output Channel.

In some configurations, there can be either multiple Servers or multiple channels within the Insertion Multiplex connected to a Splicer. In these cases, the Splicer will have multiple API Connections associated with an Output Channel. When an [SCTE 35] Cueing Message is received in a Primary Channel, the **Cue_Request** message must be sent to Servers over all of the API Connections that were made for the associated Output Channel(s). It is also possible that more than one API Connection will transport a **Splice_Request** message for the same insertion at the same time for an Output Channel.

6.2. Arbitration Priorities

Deprecated: There are no known implementations of arbitration.

Different levels of access are used to ensure that the correct Insertion Channel is utilized. There are ten different levels of access, 0 through 9, with 9 being the highest priority which may override any lower priority connection. The **OverridePlaying** flag in the **Splice_Request** message specifies whether an insertion request is honored when the Splicer is currently queuing or performing an insertion. If the flag is set to 1, then the higher priority insertion can interrupt the same or lower priority currently playing insertion. If the flag is set to 0, the Splicer will not replace the insertion currently playing, even if the new request is of a higher priority.

The **Splice_Request** message should be sent at least three seconds before the splice time() in order to be valid. If the three second minimum is not met, the outcome of the **Splice_Request** message is not determined by this API. If multiple Servers initiate splice requests for the same time with the same priority, the Splicer will prioritize the requests on a first come-first served basis. All other requests will be denied and a collision error will be sent in the **Splice_Response** message (unless the **OverridePlaying** flag is set).

For example, during the period of time immediately preceding the initiation of an insertion, the following is true: if a priority 5 **Splice_Request** is received for the same splice time as a priority 3 **Splice_Request**, a collision error is returned for the priority 3 request. If a priority 7 **Splice_Request** is later received for the same time, a collision error is returned for the priority 5 request and the priority 7 request is queued. If a second priority 7 request is received with the **OverridePlaying** flag set to 0, then the second priority 7 request would receive a collision error. However, if the **OverridePlaying** flag is set to 1 on the second priority 7 request, the original priority 7 request would receive a collision error and be overridden.

In Figure 6-3, three Splicer inputs are shown. The shaded areas indicate which input source will be directed to the Output Channel at any given moment.

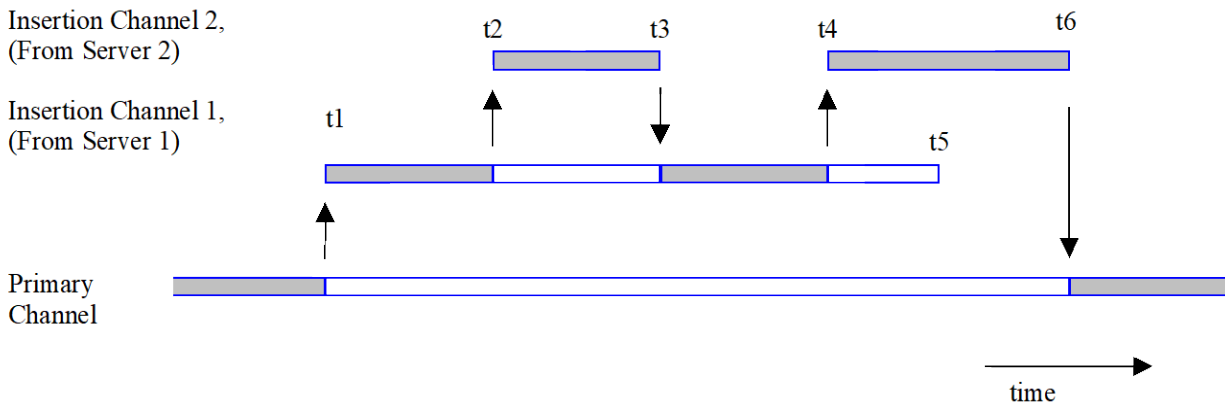


Figure 3 - OverridePlaying Flag Operation

t1 - Server 1 issues a **Splice_Request** Message and begins its stream to the Splicer. Splicer switches this Insertion Channel stream to the Output Channel. The **Splice_Request** has requested an insertion duration from time t1 thru time t5. The Splicer shall send Server 1 a **SpliceComplete_Response** message with the **SpliceType** flag set to Splice_in and a Result Code set to 100, “Successful Response”.

t2 - Server 2 issues a **Splice_Request**, with the **OverridePlaying** flag set to 1 and has an equal or higher priority. At the time specified by the **Splice_Request**, Insertion Channel 2’s stream is switched to the Output Channel (replacing the ongoing stream from Server 1). Server 2’s **Splice_Request** requests a duration from time t2 thru time t3. The Splicer shall send Server 1 a **SpliceComplete_Response** message with the **SpliceType** flag set to Splice_out and a Result Code set to 125, Channel Override. The Splicer shall send Server 2 a **SpliceComplete_Response** message with the **SpliceType** flag set to Splice_in and a Result Code set to 100, “Successful Response”. If Server 1 determines that the Channel Override is an error, it may send an **Abort_Request** and terminate its stream at this time. This behavior is not shown in Figure 6-3.

t3 - The insertion duration is completed and the Splicer returns to the material from Server 1 to direct to the Output Channel. Note that the Splicer did not return to the Primary Channel for direction to the Output Channel. The Splicer shall send Server 1 a **SpliceComplete_Response** message with the **SpliceType** flag set to Splice_in and a Result Code set to 125, Channel Override. The Splicer shall send Server 2 a **SpliceComplete_Response** message with the **SpliceType** flag set to Splice_out and a Result Code set to 100, “Successful Response”.

t4 - Server 2 issues another **Splice_Request**, with the **OverridePlaying** flag set to 1. At the time specified by the **Splice_Request**, Insertion Channel 2’s stream is switched to the Output Channel (replacing the still ongoing stream from Server 1). Server 2’s **Splice_Request** requests a duration from time t4 thru time t6. The Splicer shall send Server 1 a **SpliceComplete_Response** message with the **SpliceType** flag set to Splice_out and a Result Code set to 125, Channel Override. The Splicer shall send Server 2 a **SpliceComplete_Response** message with the **SpliceType** flag set to Splice_in and a Result Code set to 100, “Successful Response”.

t5 - Server 1’s insertion stream ends with 2 portions of its duration having been played and 2 portions having been overridden by Server 2’s stream.

t6 - The final insertion duration is completed and the Splicer returns to the material from Primary Channel for direction to the Output Channel. The Splicer shall send Server 2 a **SpliceComplete_Response** message with the **SpliceType** flag set to Splice_out and a Result Code set to 100, “Successful Response”.

It is also possible that multiple Servers will need to split a **Cue_Request** message; a 60-second duration splicing opportunity where one Server will use the first 30 seconds and the second Server will use the last 30 seconds, for example. Depending on the priorities and when the **Splice_Request** messages are received, the Splicer shall indicate a Result Code 109 (Splice Collision) if one exists. This API does not coordinate the ability of the two Servers to be able to perform this functionality. This can be done by mutual agreement between the Servers or by a Server-to-Server API.

6.3. Abnormal Terminations

It is possible that an insertion will be overridden at some time during playback by a higher priority insertion. In this case the Splicer shall return to the overridden insertion at the end of the higher priority insertion. If the higher priority insertion is aborted by an **Abort_Request** message, the Splicer shall return to the overridden insertion. If the initial Insertion Channel is no longer available, then the Splicer shall return to the Primary Channel if possible.

If the Server requests a splice on a Primary Channel that currently has no valid input, the Splicer shall perform the splice and report a Result Code 111 (No Primary Channel Found) in the **SpliceComplete_Response** to the Server. Likewise, a splice from an Insertion Channel back to a Primary Channel that has no valid input shall complete with Result Code 111 (No Primary Channel Found).

The Splicer vendor may consider adding software to ensure that the Splicer always returns to the Primary Channel. It is highly desirable to have the Splicer fail-safe to the Primary Channel on any error condition that would cause the Output Channel to stop transmitting

6.4. Splicing Requirements

The Splicer requires information about the Insertion Channel before it can be spliced into the Primary Channel. Some of this information shall be sent in the API Connection (such as **Init_Request** and **Splice_Request**) and some of it may be sent in the MPEG Multiplex (such as PAT and PMT). All of the information is required before the splice.

ChannelName is used for Output Channel identification. This is a unique name assigned to each Output Channel (e.g. CNN) in the Splicer setup and is needed by the Server to determine which Primary Channel shall be replaced by each Insertion Channel.

The Splicer needs to know which Insertion Channel to splice into the Primary Channel. This includes the Insertion Multiplex location (**Init_Request**) and which Channel in the Insertion Multiplex (**Splice_Request**) to use.

6.5. Communication

The communication between the Server and the Splicer is conducted over one TCP/IP socket connection per Output Channel. Once this API Connection is established it remains established until one of the devices terminates it, at which time re-initialization is required to splice again.

All messages exchanged between the Splicer and Server share a common general format detailed in Section 7.1. Only messages adhering to this format shall be used for communication between the Splicer

and Server. The format allows for a class of “User Defined” type messages that can be used as a template for private data messages between the Server and Splicer, but is beyond the scope of this document.

All request messages require a response from either the Splicer or the Server, depending on which device is making the request. Most of the response messages only indicate a result and do not contain any other data, but are needed to ensure the requestor that the message was received and interpreted correctly. If there are errors, the message might be resent. An example would be an error return with a 111 No Primary Channel Found message and there is still time to execute the command, the ad server may resend it.

7. API Syntax

7.1. Splicing_API_Message Syntax

Messages in this API all contain a general message structure that wraps the data for the specific message being sent. This is done so that when the message is received a common parsing routine can store the message, determine what the structure of the data is and ensure that the message is received correctly.

Table 1 - Splicing_API_Message

Syntax	Bytes	Type
Splicing_API_Message {		
MessageID	2	uimbsf
MessageSize	2	uimbsf
Result	2	uimbsf
Result_Extension	2	uimbsf
data()		
}		

MessageID – An integer value that indicates what message is being sent. See Table 2

MessageSize – The size of the data() field being sent in bytes.

Result – The results to the requested message. See *Appendix A (53) – Result Codes* for details on the result codes. On request messages, this is set to 0xFFFF.

Result_Extension – This shall be set to 0xFFFF unless used to send additional result information in a response message.

data() – Specific data structure for the message being sent. Details on each of the messages containing data are described below. The size of this field is equal to the **MessageSize** and is determined by the size of the data being added to the message. Not all messages utilize the data() field.