

International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

J.280

(03/2013)

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

Digital transmission of television signals

Digital program insertion: Splicing application program interface

Recommendation ITU-T J.280

Downloaded from ITU-T J.280 2013

Recommendation ITU-T J.280

Digital program insertion: Splicing application program interface

Summary

The splicing application program interface (API) focuses on the single server, single splicer configuration. This version of the Recommendation increases the value of Revision_Num from 1 to 2.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T J.280	2004-03-15	9	11.1002/1000/7204-en
2.0	ITU-T J.280	2005-12-14	9	11.1002/1000/8668-en
3.0	ITU-T J.280	2013-03-01	9	11.1002/1000/11829-en

* 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 2014

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

Table of Contents

	Page
1 Scope	1
2 References.....	1
3 Definitions	1
3.1 Terms defined elsewhere	1
3.2 Terms defined in this Recommendation.....	1
4 Abbreviations and acronyms	2
5 Conventions	3
6 Requirements	3
6.1 System block diagram	3
6.2 Arbitration priorities	5
6.3 Abnormal terminations	6
6.4 Splicing requirements	7
6.5 Communication	7
7 API syntax	7
7.1 Splicing_API_Message syntax	7
7.2 Conventions and requirements	9
7.3 Initialization.....	10
7.4 Embedded cueing messages	11
7.5 Splice messages	12
7.6 Alive messages	16
7.7 Extended data messages	17
7.8 Abort messages.....	18
7.9 Abort_Request message	18
7.10 Abort_Response message	18
7.11 TearDownFeed_Request message.....	18
7.12 TearDownFeed_Response message	19
7.13 Requesting configuration settings	19
7.14 General_Response message.....	19
8 Additional structures.....	19
8.1 Version	19
8.2 Hardware_Config	20
8.3 splice_elementary_stream().....	23
8.4 time() field definition	24
8.5 splice_API_descriptor() field definition.....	25
9 Time synchronization	32
10 System timing	33
10.1 DPI splice signal flow	33

	Page
10.2 DPI splice initiation timeline	34
Appendix I – Result codes	36
Appendix II – Example use of Logical_Multiplex type 0x0006 and the port_selection_descriptor()	39
II.1 Informative example 1	39
II.2 Informative example 2	39
Appendix III – Frame rate codes	40
Bibliography	41

iv

Recommendation ITU-T J.280

Digital program insertion: Splicing application program interface

1 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.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- [ITU-T H.222.0] Recommendation ITU-T H.222.0 (2012) | ISO/IEC 13818-1:2012, *Information technology – Generic coding of moving pictures and associated audio information: Systems*.
- [ITU-T H.262] Recommendation ITU-T H.262 (2012) | ISO/IEC 13818-2:2012, *Information technology – Generic coding of moving pictures and associated audio information: Video*.
- [ANSI/SCTE 35] ANSI/SCTE 35 2012 (2012), *Digital Program Insertion Cueing Message for Cable*.
- [ANSI/SCTE 54] ANSI/SCTE 54 2009 (2009), *Digital Video Service Multiplex and Transport System Standard for Cable Television*.
- [ANSI/SCTE 128] ANSI/SCTE 128 2010a (2010), *AVC Video Systems and Transport Constraints for Cable Television*.

3 Definitions

3.1 Terms defined elsewhere

None.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 API connection: A TCP/IP socket connection between a server and a splicer for transferring API messages.

3.2.2 back-to-back insertion: Insertion of two or more temporally contiguous sessions without returning to the primary channel between sessions.

3.2.3 channel: A channel is a synonym for a "Service" in DVB terminology, or a "Program" in MPEG terminology.

- 3.2.4 insertion channel:** The insertion multiplex channel(s) that replaces the primary channel in whole or in part of the duration for a splice event.
- 3.2.5 insertion multiplex:** This is the source of the insertion channel. A multiplex produced by a server may under some circumstances exclude program specific information (PSI), thus it is understood that this multiplex may be a non-compliant MPEG-2 transport stream.
- 3.2.6 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.
- 3.2.7 output channel:** The channel that is produced at the output of the splicer.
- 3.2.8 output multiplex:** The MPEG-2 transport stream produced by multiplexing one or more output channels.
- 3.2.9 primary channel:** The primary multiplex channel that is replaced in whole or in part. A single primary channel may result in multiple output channels.
- 3.2.10 primary multiplex:** This is the source of the primary channel(s).
- 3.2.11 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.
- 3.2.12 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.
- 3.2.13 splice-in:** The splice at the start of the insertion. This happens at the time specified in the Splice_Request message.
- 3.2.14 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.
- 3.2.15 splicer:** The device that splices the insertion channel(s) into the primary channel(s). It may receive ANSI/SCTE 35 cue messages. This device also communicates with the server about when and what to splice.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

API	Application Program Interface
DVB-ASI	Digital Video Broadcast – Asynchronous Serial Interface
ID	Identifier
MLD	Multicast Listener Discovery
MPEG	Moving Picture Experts Group
MPTS	Multi-Program Transport Stream
NTP	Network Time Protocol
PAT	Program Association Table
PCR	Program Clock Reference
PID	Packet Identifier
PMT	Program Map Table
PSI	Program Specific Information

SPTS	Single Program Transport Stream
tcimsbf	twos complement integer, most significant bit first
TCP/IP	Transport Control Protocol/Internet Protocol
uimsbf	unsigned integer, most significant bit first
UTC	Coordinated Universal Time

5 Conventions

None.

6 Requirements

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 6-1. However, this can be expanded to the multiple servers, multiple splicers configuration as shown in Figure 6-2.

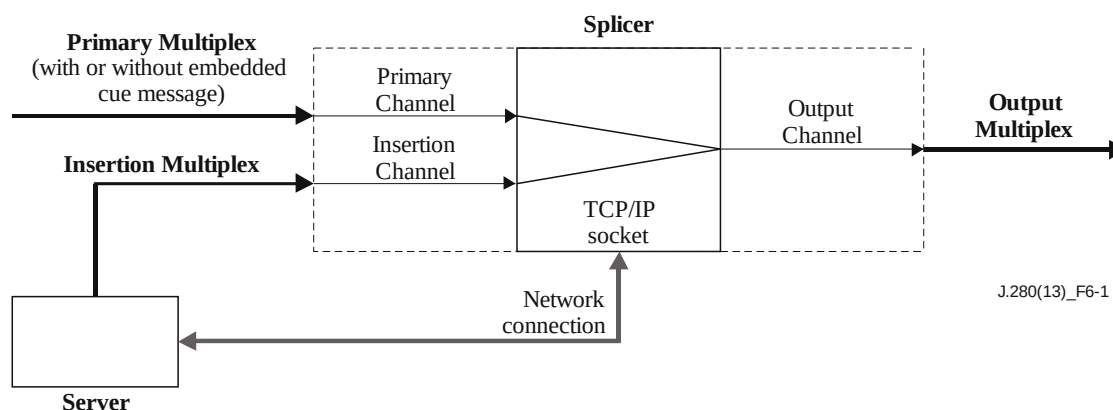


Figure 6-1 – Single server/Single splicer

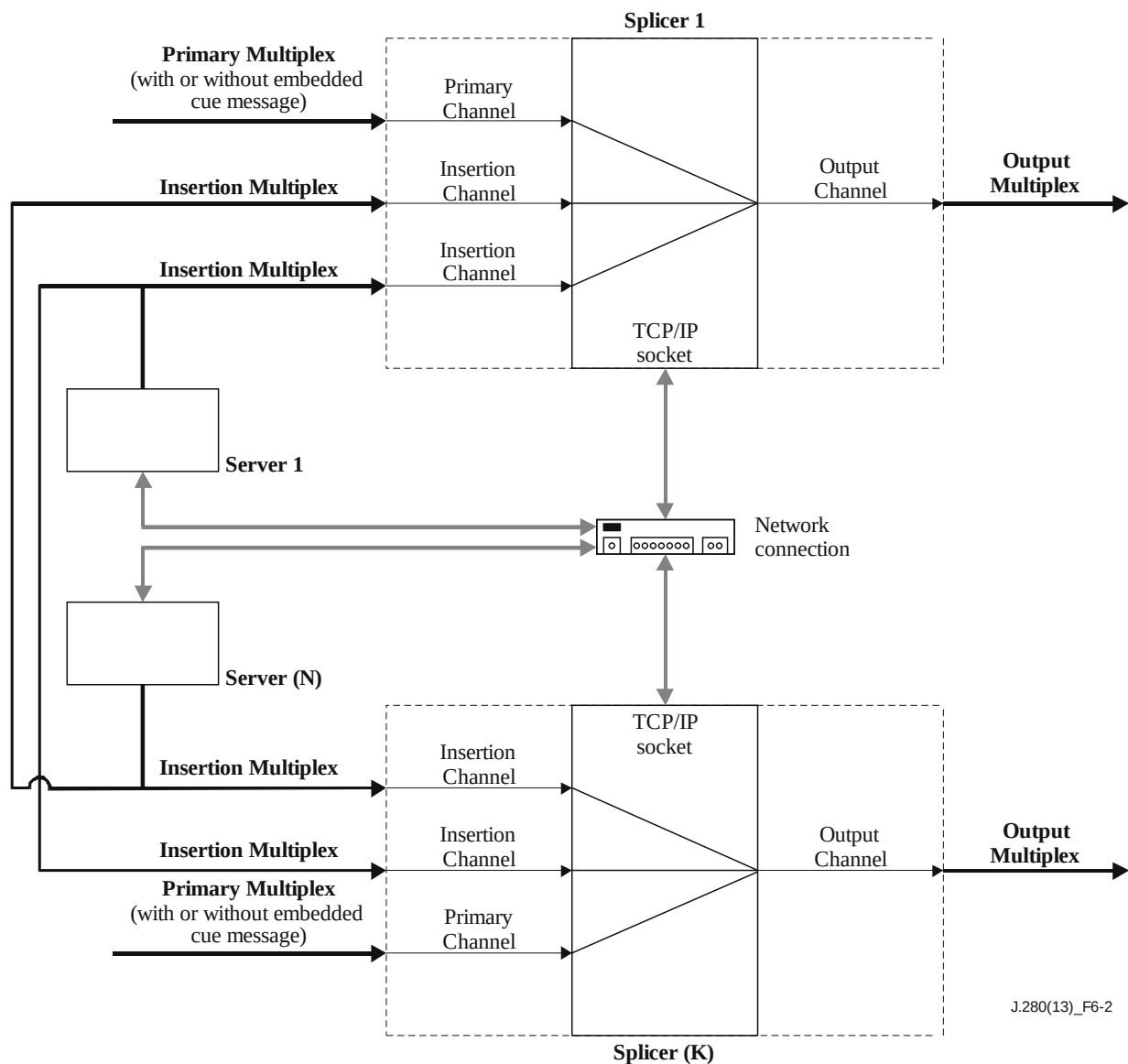


Figure 6-2 – Multiple servers/Multiple splicers

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(es) 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 [ANSI/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 ANSI/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

Different levels of access are used to ensure that the correct insertion channel is utilized. There are ten different levels of access, 0 to 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 honoured when the splicer is currently queueing 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.

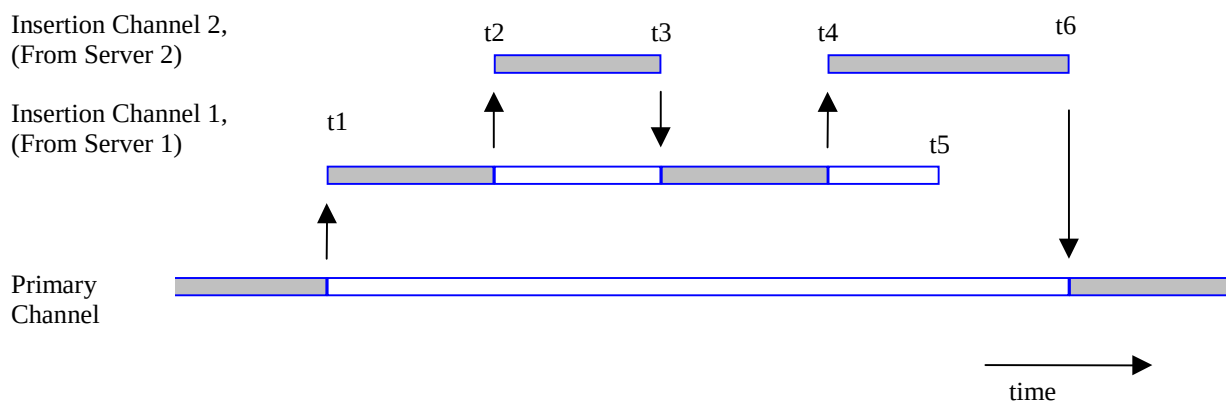


Figure 6-3 – OverridePlaying flag operation

t1 – Server 1 issues a **Splice_Request** message and begins its stream to the splicer. The splicer switches this insertion channel stream to the output channel. The **Splice_Request** has requested an insertion duration from time t1 to 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 to 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 behaviour 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 to 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 two portions of its duration having been played and two portions having been overridden by Server 2's stream.

t6 – The final insertion duration is completed and the splicer returns to the material from the 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 the 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 and some of it may be sent in the MPEG multiplex. 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, Cable News Network) in the splicer set-up 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 and which channel in the insertion multiplex to use. This information is available in the **Splice_Request** message.

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 clause 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 Recommendation.

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 that the message was received and interpreted correctly by the requester. If there are errors, the message can be resent.

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 7-1 – Splicing_API_Message

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

MessageID – An integer value that indicates what message is being sent. See Table 7-2.

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

Result – The results to the requested message. See Appendix I 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.

Table 7-2 – MessageID values

MessageID	Message name	Sent by	Description
0x0000	General_Response	Splicer or Server	Used to convey asynchronous information between the devices. There is no data() associated with this message.
0x0001	Init_Request	Server	Initial message to the splicer on port 5168.
0x0002	Init_Response	Splicer	Initial response to the server on the established connection.
0x0003	ExtendedData_Request	Server	Request for detailed playback information from the splicer.
0x0004	ExtendedData_Response	Splicer	Vendor unique response of extended playback data from the requested playback event.
0x0005	Alive_Request	Server	Sends an alive message to acquire current status.
0x0006	Alive_Response	Splicer	Response to the alive message indicating current status.
0x0007	Splice_Request	Server	Request to splice at a specific time.
0x0008	Splice_Response	Splicer	Response to indicate that the Splice_Request was received and that the splicer is preparing to splice.
0x0009	SpliceComplete_Response	Splicer	Response at the splice in and splice out.
0x000A	GetConfig_Request	Server	Request to get the current splice configuration for this API connection.

Table 7-2 – MessageID values

MessageID	Message name	Sent by	Description
0x000B	GetConfig_Response	Splicer	Contains all of the splice information for the API connection.
0x000C	Cue_Request	Splicer	Splicer sending the cue info section to the server.
0x000D	Cue_Response	Server	Acknowledgment that the cue info section was received.
0x000E	Abort_Request	Server	Request to immediately return to the primary channel or overridden insertion channel.
0x000F	Abort_Response	Splicer	Acknowledgment that the Abort_Request message was received. A SpliceComplete_Response shall also be generated if necessary.
0x0010	TearDownFeed_Request	Server	Request to delete an output channel created using the create_feed_descriptor().
0x0011	TearDownFeed_Response	Splicer	Response to indicate that the output channel has been deleted.
0x0012-0x7FFF 0xFFFF	Reserved		Range reserved for future standardization.
0x8000-0xFFFE	User defined		Range available for user-defined functions.

7.2 Conventions and requirements

- Each message that contains data is outlined with its data fields and types below. Additional structures are indicated as functions and are described in clause 8.
- All string lengths have space reserved for a null terminator character and must use null terminated strings. For example, a string that is defined as 16 characters can be at most 15 characters of data followed by a null (0x00) character immediately after the last data character. Once a null is encountered in scanning a string, the rest of the characters in the string are undefined. The size defined for the string is constant and will not vary depending on the length of the string. This specification uses 8-bit ASCII characters for strings.
- All time values are UTC.
- This specification uses all 1s for a DON'T CARE condition. For a 4-byte field, this value would be 0xFFFFFFFF.
- Response messages shall be sent out without unnecessary delay. The device expecting a response should consider no response within 5 seconds to indicate a timeout. When a server suspects a timeout, it shall send an Alive_Request message. If the splicer does not answer as specified in this Recommendation, the connection for this channel shall be dropped and re-established.
- A server receiving a response message indicating a failure to parse a message (error code 123) shall send an Alive_Request message. If it does not receive the appropriate Alive_Response message, the connection for this channel shall be dropped and re-established.
- The Result field in the Splicing_API_Message is used to return a result code. Multiple response codes can be returned by sending multiple **General_Response** messages at any time.