



***Society of Cable
Telecommunications
Engineers***

**ENGINEERING COMMITTEE
Digital Video Subcommittee**

AMERICAN NATIONAL STANDARD

ANSI/SCTE 104 2004

**Automation System to Compression System
Communications Applications Program Interface (API)**

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability and ultimately 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, whether used domestically or internationally.

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

Attention is called to the possibility that implementation of this standard may require the use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. 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 standard 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 <http://www.scte.org>.

All Rights Reserved

© Society of Cable Telecommunications Engineers, Inc.
140 Philips Road
Exton, PA 19341

TABLE OF CONTENTS

AUTOMATION SYSTEM – COMPRESSION SYSTEM COMMUNICATIONS API	1
1.0 SCOPE	1
2.0 DEFINITIONS AND ACRONYMS.....	1
3.0 NORMATIVE REFERENCES	7
3.1 SCTE REFERENCES.....	7
3.2 STANDARDS FROM OTHER ORGANIZATIONS	7
4.0 INFORMATIVE REFERENCES	8
4.1 SCTE REFERENCES.....	8
4.2 STANDARDS FROM OTHER ORGANIZATIONS	8
4.3 PUBLISHED MATERIALS	9
5.0 OVERVIEW	9
6.0 DATA COMMUNICATIONS	13
6.1 CONCERNING DATA COMMUNICATIONS (INFORMATIVE)	13
6.2 DATA COMMUNICATIONS REQUIREMENTS FOR THIS API (NORMATIVE)	13
6.3 CONVEYANCE QUALITY-OF-SERVICE CONSIDERATIONS (INFORMATIVE).....	14
6.4 UNI-DIRECTIONAL SYSTEM CONSIDERATIONS (INFORMATIVE).....	14
7.0 MESSAGE FORMATS.....	15
7.1 TERMINOLOGY (INFORMATIVE).....	15
7.2 MESSAGE STRUCTURES (NORMATIVE).....	15
7.3 OPERATION TYPES (NORMATIVE)	23
7.4 CONVENTIONS AND REQUIREMENTS	29
8.0 AUTOMATION SYSTEM TO INJECTOR COMMUNICATION	30
8.1 INITIALIZATION	30
8.2 ALIVE (“HEARTBEAT”) COMMUNICATIONS.....	32
8.3 SPLICE REQUESTS	34
8.4 ENCRYPTION SUPPORT (NORMATIVE)	41
8.5 COMPONENT MODE SUPPORT	44
8.6 RESPONSE MESSAGES	45
8.7 SCTE 35 SPLICE_SCHEDULE() SUPPORT REQUESTS	49
8.8 MISCELLANEOUS REQUESTS	55
9.0 PAMS TO THE AUTOMATION SYSTEM COMMUNICATIONS.....	63
9.1 SYSTEM DESIGN PHILOSOPHY	63
9.2 PAMS FUNCTIONALITY	65
9.3 SERVICE CONTINUITY	66
9.4 SYSTEM INITIALIZATION MESSAGES.....	66

9.5	INJECTOR SERVICE NOTIFICATION.....	69
9.6	FAILURE NOTIFICATION MESSAGES (DEVICE OR COMMUNICATIONS)	72
9.7	PAMS TO AS PERMANENT “LINK ALIVE” MESSAGES.....	74
9.8	PAMS TO AS COMMON ELEMENTS	75
10.0	PAMS TO INJECTOR COMMUNICATIONS (INFORMATIVE).....	76
10.1	THE PAMS IMPLEMENTATION	77
10.2	INJECTOR PROVISIONING	77
10.3	PAMS STRUCTURE	77
10.4	SUPPORT OF MULTIPLE DPI PIDs	77
11.0	COMMON ELEMENTS.....	78
11.1	VALUES OF SPLICE_EVENT_ID USED IN THIS INTERFACE	78
11.2	VALUES OF UNIQUE_PROGRAM_ID USED IN THIS INTERFACE	78
11.3	MINIMUM PRE-ROLL TIME SUPPORTED BY THIS INTERFACE	78
11.4	TIME() DEFINITION	79
11.5	TIMESTAMP() DEFINITION.....	80
12.0	SYSTEM ARCHITECTURE AND PROVISIONING (INFORMATIVE).....	82
12.1	ONE WAY PROTOCOL – AUTOMATION SYSTEM TO INJECTOR	82
12.2	TWO WAY PROTOCOL – AUTOMATION SYSTEM TO INJECTOR ONLY	93
12.3	TWO WAY PROTOCOL – AUTOMATION SYSTEM TO INJECTOR WITH PAMS	113
13.0	RESULT CODES (NORMATIVE).....	125
APPENDIX A: TCP/IP CONVEYANCE		128
APPENDIX B: ANSI/TIA/EIA-232-F CONVEYANCE		128
B.1	THE BASIC LINK LAYER SYNTAX.....	129
B.2	THE ESCAPE SEQUENCE	130
APPENDIX C: ANALOG VIDEO SYSTEM CONVEYANCE		130

LIST OF FIGURES

FIGURE 5-1 – SCTE 35 OVERALL SYSTEM BLOCK DIAGRAM WITH BI-DIRECTIONAL DATA COMMUNICATIONS	11
FIGURE 5-2 – SCTE 35 OVERALL SYSTEM BLOCK DIAGRAM WITH UNI-DIRECTIONAL DATA COMMUNICATIONS	12
FIGURE 8-1 - MULTIPLE_OPERATION_MESSAGE() TO SCTE 35 SECTION FIELD MAPPING (INFORMATIVE).....	40
FIGURE 12-1 - ONE-WAY PROTOCOL EMBEDDED IN VIDEO WITH INTEGRATED INJECTOR.....	83
FIGURE 12-2 - ONE-WAY PROTOCOL WITH MULTIPLE AS TO EXTERNAL INJECTOR	84
FIGURE 12-3 - ONE-WAY FLOW DIAGRAM WITH DELAYED PROCESSING.....	91
FIGURE 12-4 - ONE-WAY FLOW DIAGRAM FOR EARLY RETURN	92
FIGURE 12-5 - TWO-WAY BLOCK DIAGRAM WITH INTERNAL INJECTOR	93
FIGURE 12-6 - TWO-WAY BLOCK DIAGRAM WITH EXTERNAL INJECTOR.....	94
FIGURE 12-7 - TWO-WAY FLOW DIAGRAM FOR INITIALIZATION	106
FIGURE 12-8 - TWO-WAY FLOW DIAGRAM WITH DELAYED PROCESSING.....	107
FIGURE 12-9 - TWO-WAY FLOW DIAGRAM WITH IMMEDIATE PROCESSING	108
FIGURE 12-10 – TWO-WAY FLOW DIAGRAM FOR EARLY RETURN.....	109
FIGURE 12-11 - TWO-WAY CANCELLATION BEFORE BEING PROCESSED.....	110
FIGURE 12-12 - TWO-WAY CANCELLATION AFTER BEING PROCESSED.....	111
FIGURE 12-13 - TWO-WAY FLOW DIAGRAM CANCEL AFTER SPLICE POINT	112
FIGURE 12-14 - TWO-WAY BLOCK DIAGRAM WITH INTERNAL INJECTOR	114
FIGURE 12-15 - TWO-WAY BLOCK DIAGRAM WITH EXTERNAL INJECTOR.....	115
FIGURE 12-16 – AS/PAMS FLOW DIAGRAM FOR INITIALIZATION	120
FIGURE 12-17 - PAMS TWO-WAY INITIALIZATION OF A PERMANENT CONNECTION	121
FIGURE 12-18 - PAMS DETECTS AN INJECTOR FAILURE.....	122
FIGURE 12-19 - AS DETECTS AN INJECTOR FAILURE.....	123
FIGURE 12-20 - INJECTOR SOCKET FAILED AND RECOVERED.....	124

LIST OF TABLES

TABLE 2-1 - TERMS AND ACRONYMS	1
TABLE 2-2 - WIDELY USED TERMS AND ACRONYMS (INFORMATIVE)	6
TABLE 7-1 - SINGLE OPERATION MESSAGE	18
TABLE 7-2 - MULTIPLE OPERATION MESSAGE	21
TABLE 7-3 - OPID ASSIGNED VALUES AND MEANINGS FOR SINGLE_OPERATION_MESSAGES	24

TABLE 7-4 - OPID ASSIGNED VALUES AND MEANINGS FOR MULTIPLE_OPERATION_MESSAGES.....	27
TABLE 8-1 - INIT_REQUEST_DATA	31
TABLE 8-2 - INIT_RESPONSE_DATA	31
TABLE 8-3 - ALIVE_REQUEST_DATA	33
TABLE 8-4 - ALIVE_RESPONSE_DATA	33
TABLE 8-5 - SPLICE_REQUEST_DATA.....	34
TABLE 8-6 - SPLICE_INSERT_TYPE ASSIGNED VALUES.....	36
TABLE 8-7 - SPLICE_INSERT_TYPE CORRESPONDING SPLICE_INSERT() FIELD SETTINGS (INFORMATIVE).....	39
TABLE 8-8 - ENCRYPTED_DPI_REQUEST_DATA	42
TABLE 8-9 - UPDATE_CONTROLWORD_DATA	43
TABLE 8-10 - DELETE_CONTROLWORD_DATA	44
TABLE 8-11 - COMPONENT_MODE_DPI_REQUEST_DATA	45
TABLE 8-12 - GENERAL_RESPONSE_DATA	46
TABLE 8-14 - INJECT_RESPONSE DATA.....	46
TABLE 8-15 – INJECT_RESPONSES.....	47
TABLE 8-16 - INJECT_COMPLETE RESPONSE DATA.....	48
TABLE 8-17 – INJECT_COMPLETE_RESPONSES.....	49
TABLE 8-18 - START_SCHEDULE_DOWNLOAD_REQUEST_DATA	50
TABLE 8-19 - SCHEDULE_DEFINITION_DATA.....	52
TABLE 8-20 - SPLICE_SCHEDULE COMMAND TYPE ASSIGNED VALUES	52
TABLE 8-21 - SCHEDULE_COMPONENT_REQUEST_MODE.....	53
TABLE 8-22 - TRANSMIT_SCHEDULE_REQUEST_DATA.....	54
TABLE 8-23 - TIME_SIGNAL_REQUEST_DATA	55
TABLE 8-24 - SPLICE_NULL_REQUEST_DATA	56
TABLE 8-25 - INJECT_SECTION_DATA_REQUEST	57
TABLE 8-26 - INSERT_AVAIL_DESCRIPTOR_REQUEST_DATA	58
TABLE 8-27 - INSERT_DESCRIPTOR_REQUEST_DATA.....	59
TABLE 8-28 - INSERT_DTMF_DESCRIPTOR_REQUEST_DATA.....	60
TABLE 8-29 – INSERT_SEGMENTATION_DESCRIPTOR_REQUEST_DATA.....	61
TABLE 8-30 - PROPRIETARY_COMMAND_REQUEST_DATA	62
TABLE 9-1 - CONFIG_REQUEST_DATA	67
TABLE 9-2 - CONFIG_RESPONSE_DATA	68
TABLE 9-3 - PROVISIONING_REQUEST_DATA.....	70
TABLE 9-4 – PROVISIONING_RESPONSE_DATA	72
TABLE 9-5 - FAULT_REQUEST_DATA	73
TABLE 9-6 - FAULT_RESPONSE_DATA	74
TABLE 9-7 - AS_ALIVE_REQUEST_DATA	75
TABLE 9-8 - AS_ALIVE_RESPONSE_DATA	75
TABLE 9-9 - INJECTOR_COMPONENT_LIST()	76
TABLE 11-1 - TIME().....	79
TABLE 11-2 - TIMESTAMP()	80
TABLE 12-1 – SUPPORTED PROTOCOL MESSAGES.....	87
TABLE 12-2 – UNSUPPORTED PROTOCOL MESSAGES.....	88

TABLE 12-3 – OPTIONAL PROTOCOL MESSAGES.....	89
TABLE 12-4 – UNUSED PAMS PROTOCOL MESSAGES	90
TABLE 12-5 – SUPPORTED PROTOCOL MESSAGES.....	100
TABLE 12-6 – SUPPORTED PROTOCOL MESSAGES (CON'T).....	102
TABLE 12-7 – OPTIONAL PROTOCOL MESSAGES.....	104
TABLE 12-8 – UNUSED PAMS PROTOCOL MESSAGES	105
TABLE 12-9 –PAMS PROTOCOL MESSAGES.....	119
TABLE 13-1 - RESULT CODES	125
TABLE B-1 - SERIAL_LINKLAYER STRUCTURE.....	129

AUTOMATION SYSTEM – COMPRESSION SYSTEM COMMUNICATIONS API

1.0 SCOPE

This Standard defines the Communications API between an Automation System and the associated Compression System that will insert SCTE 35 2004 private sections into the outgoing Transport Stream. This document serves as a companion to both SCTE 35 2004 and SCTE 30 2001.

2.0 DEFINITIONS AND ACRONYMS

Throughout this document, the terms used have specific meanings. Because some of the terms that are defined in ISO/IEC 13818-1 have very specific technical meanings, the reader is referred to the original source for their definition. For terms used in this document, brief definitions are given below.

Table 2-1 - Terms and Acronyms

TERM	DESCRIPTION
API	Application Program Interface. A mechanism whereby one software system asks another software system to perform a service.
API Connection	A communications connection between an Automation System and an Injector for transferring API messages.
AS	Automation System
ATSC	Advanced Television Systems Committee
Automation System	A control system for a program origination facility which controls operation of the production facilities and devices.
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.
Basic	A category of Request or Response operation supported by this API. See Section 7.3.
backoff	A mechanism, commonly used in data communications, to randomize the interval between retries.
BER	Abbreviation for bit-error rate.

TERM	DESCRIPTION
bslbf	Bit string, left bit first, where “left” is the order in which bit strings are written in the Standard. Bit strings are written as a string of 1s and 0s within single quote marks, e.g. ‘1000 0001’. Blanks within a bit string are for ease of reading and have no significance. (See ISO/IEC 13818-1).
Component Splice Mode	A mode of the splice_info_section 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 splice_info_section are not to be spliced.
Control	A category of Request operation supported by this API. See Section 7.3.
Control Word	A multiple key value used by the encryption mechanisms specified in SCTE 35.
CRC	Cyclic Redundancy Check. A method to verify the integrity of a transmitted message.
CW	Abbreviation for Control Word.
dB	Abbreviation for decibel.
DCS	Digital Compression System
DES	Data Encryption Standard. A method for encrypting data with symmetric keys.
DPI	Digital Program Insertion
DPI PID	A single PID carrying SCTE 35 splice_info_sections.
DVB	Digital Video Broadcasting. An international consortium for the development of digital television systems.
Event	A splice event or a viewing event as defined below.
GPI	General Purpose Interface, commonly used to source or sink contact closures in video facilities.
HANC	<u>H</u> orizontal <u>A</u> NCillary data space in digital video streams
IJ	Abbreviation for Injector.

TERM	DESCRIPTION
Injector	A device or combination of devices within the DCS capable of converting DVS/525 message data into a SCTE 35 splice_info_section(), including a program-specific PCR splice time value, if necessary, and multiplexing the resulting section data along with the other program components into the eventual MPEG SPTS or MPTS.
Injector Instance	A specific instance of an Injector, constrained to place a single DPI PID into a single MPEG program in a single Transport Stream.
In Point	A point in the stream, suitable for entry, that lies on an access unit boundary.
ISO	Abbreviation for International Organization for Standardization.
ITU	Abbreviation for International Telecommunications Union.
Message	In the context of this document a message is a single communication between the Automation System and the Compression System or between the Automation System and the PAMS. A message may contain one or more operations.
MPTS	A Multi Program Transport Stream.
MSB	Abbreviation for Most Significant Bit.
Normal	A category of Request operation supported by this API. See Section 7.3.
Out Point	A point in the stream, suitable for exit, that lies on an access unit boundary.
OSI	Abbreviation for Open Systems Interconnection.
PAMS	Abbreviation for Provisioning and Alarm Management System (see Section 5.0).
PID	Packet identifier; a unique 13-bit value used to identify elementary streams of a program in a single or multi-program Transport Stream. (See ISO/IEC 13818-1).
PID stream	A stream of packets with the same PID within a transport stream.
PMT	Program Map Table. (See ISO/IEC 13818-1).
port	See “socket.” Refers to a bit-field defined in a TCP header. May also refer to a specific physical connector mounted on a device.
PPP	Point-to-Point Protocol. Defined in RFC 1661.

TERM	DESCRIPTION
Presentation Time	The time that a presentation unit is presented in the system target decoder.
Program	A collection of video, audio, and data PID streams which share a common program number within a SPTS or MPTS.
PTS	Presentation Time Stamp. (See ISO/IEC 13818-1).
Registration Descriptor	An MPEG-2 (ISO/IEC 13818-1) construct to uniquely and unambiguously identify formats of private data. As used in this context, it is carried in the PMT of a program to indicate the program's compliance with SCTE 35 2001. (See ISO/IEC 13818-1 Section 2.6.8).
Request	A single directive, from either the Automation System, the Injector, or the PAMS, to another portion of the overall system. "Request" and "Command" are used interchangeably. A request is always carried within a message. A request is normally answered by a response message.
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 this standard, all reserved bits shall be set to '1'.
Response	A reply message to a request directive from the other portion of the system. Responses are made by the Automation System, the Compression System, and the PAMS in reply to requests. A response is always carried within a single_operation message.
rpchof	Remainder polynomial coefficients, highest order first. (See ISO/IEC 13818-1).
SDI	Abbreviation for Serial Digital Interface (see SMPTE 259M)
Section	A private_section structure as defined by ISO/IEC 13818-1 and (in this case) SCTE 35. As used here, the term is usually "splice_info_section". See SCTE 35 Section 6.2 and ISO/IEC 13818-1, Section 2.4.4.10.
Simple Profile	A defined subset of the Automation to Injector messages in this API which supports all basic splicing functionality while excluding schedules, encryption, and component mode. An implementer may choose to support only the Simple Profile or features beyond it. The implementer can then describe their implementation in common terms (for example "Simple Profile plus encryption").
SNR	Abbreviation for signal to noise ratio.

TERM	DESCRIPTION
socket	A TCP/IP mechanism used for connection-oriented communications. Sometimes also called “port” in an interchangeable manner.
Splice Event	An opportunity to splice one or more PID streams.
Splice_info_section	Basic SCTE 35 structure for carrying DPI commands in a TS to downstream equipment. See SCTE 35 Section 6.2.
Splice Immediate Mode	A mode of the splice_info_section whereby the splicing device shall choose the nearest opportunity in the stream, relative to the splice_info_table, to splice. When not in this mode, the splice_info_section gives a “PTS_time”, which is a presentation time, for the intended splicing moment.
Splice Point	A point in a PID stream that is either an Out Point or an In Point.
SPTS	A Single Program Transport Stream.
Supplemental	A category of request operation supported by this API. See Section 7.3.
TS	Abbreviation for Transport Stream.
uimsbf	Unsigned integer, most significant bit first. (See ISO/IEC 13818-1).
UTC	“ <u>U</u> niversel <u>T</u> emps <u>C</u> oordonné” in French. Coordinated Universal Time in English.
VANC	<u>V</u> ertical <u>A</u> NCillary data space in digital video streams
VITC	<u>V</u> ertical <u>I</u> nterval <u>T</u> ime <u>C</u> ode
WST	World System Teletext. (See ITU-R BT.653-3)

In addition to the precisely defined terms and acronyms in Table 2-1, there are many widely used, but less precisely defined terms related to Digital Program Insertion. A table of these appears below.

Table 2-2 - Widely Used Terms and Acronyms (Informative)

TERM	DESCRIPTION
Analog Cue Tone	In an analog system, a signal which is usually either a sequence of DTMF tones or a contact closure that denotes to ad insertion equipment that an advertisement avail is about to begin or end.
Break	Avail or an actual insertion in progress.
Command	A single directive from the Automation System to the Compression System. A command is always carried within a multiple_operation message. This term is also used to specify specific SCTE 35 commands.
Cueing Message	See splice_info_section. A term used in SCTE 35; a “Cueing Message” is a Cueing Section in this document.
DPI Cue Message	See splice_info_section. A term used in SCTE 35; a “DPI Cue Message” is a splice_info_section in this document.
Digital Cue Tone	Widely used term to refer to an SCTE 35 splice_info_section().
Long Form Insertion	Refers to insertions of material with a duration generally greater than 10 minutes, i.e. program length material
Short Form Insertion	Refers to insertions of material with a duration generally less than 10 minutes, i.e. advertising or promotional material. As of this writing, the primary use of DPI technology.
Spot	Term for the contents of an advertisement, sometimes also used to refer to an avail.
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.

3.0 NORMATIVE REFERENCES

The following documents contain provisions, which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreement based on this standard are encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

3.1 SCTE References

1. SCTE 35 2004, Digital Program Insertion Cueing Message for Cable, Society of Cable Telecommunications Engineers (SCTE), 2004. (Also standardized as ITU-T Recommendation J.181).
2. SCTE 30 2001, Digital Program Insertion Splicing API, Society of Cable Telecommunications Engineers (SCTE), 2001.

3.2 Standards from other Organizations

3. ISO/IEC 13818-1; Information Technology ---- Generic Coding of Moving Pictures and Associated Audio Information: Systems, International Organization for Standardization/International Electrotechnical Commission, 2000. (Also standardized as ITU-T Recommendation H.222.0).
4. ITU-R BT.653-3, Teletext Systems, International Telecommunications Union (ITU), Radiocommunication Assembly, 1998.
5. ANSI/EIA-516, North American Basic Teletext Specification (NABTS), Electronic Industries Association (EIA), 1988. (Defined in BT.653-3 as “System C”). (For the purposes of this document, only Chapters 1, 2, 3, and 4 are normative. Chapters 5 through 8 are informative).
6. ETSI ETS 300 706, Enhanced Teletext specification, European Telecommunications Standards Institute (ETSI), 1997. (Defined in BT.653-3 as “System B”).
7. ETSI ETS 300 708, Data transmission within Teletext, European Telecommunications Standards Institute (ETSI), 1997.
8. SMPTE 334M, Vertical Ancillary Data Mapping for Bit-Serial Interface, Society of Motion Picture and Television Engineers, 2000.
9. SMPTE 291M, Ancillary Data Packet and Space Formatting, Society of Motion Picture and Television Engineers, 1998.

4.0 INFORMATIVE REFERENCES

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

4.1 SCTE References

1. SCTE 67 2002, Digital Program Insertion Cueing Message for Cable -- Interpretation for SCTE 35 2001, Society of Cable Telecommunications Engineers (SCTE), 2002.

4.2 Standards from other Organizations

2. SMPTE 259M, 10-Bit 4:2:2 Component and 4fsc Composite Digital Signals --- Serial Digital Interface, Society of Motion Picture and Television Engineers, 1997.
3. SMPTE 312M, Splice Points for MPEG-2 Transport Streams, Society of Motion Picture and Television Engineers, 2001.
4. SMPTE 12M, Time and Control Code, Society of Motion Picture and Television Engineers, 1999
5. SMPTE EG 40, Conversion of Time Values Between SMPTE 12M Time Code, MPEG-2 PCR Time Base and Absolute Time, Society of Motion Picture and Television Engineers, 2002.
6. ISO/IEC 11172-3, Information Technology ---- Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s, Part 3: Audio, International Organization for Standardization/International Electrotechnical Commission, 1993
7. ATSC Doc. A/52, Digital Audio Compression Standard (AC-3), Advanced Television Systems Committee, 1995
8. ATSC Doc. A/52A, Digital Audio Compression Standard (AC-3), Revision A, Advanced Television Systems Committee, 2001
9. ETSI TR 101 233, Code of practice for allocation of services in the Vertical Blanking Interval (VBI), European Telecommunications Standards Institute (ETSI), 1998.
10. IETF RFC 793, Transmission Control Protocol, The Internet Society, 1981.
11. IETF RFC 2728, The Transmission of IP Over the Vertical Blanking Interval of a Television Signal, The Internet Society, 1999.
12. ITU-T X.200, Open Systems Interconnection -- Basic Reference Model, International Telecommunications Union (ITU), Telecommunication Standardization Sector, 1994.

13. SMPTE 298M, Universal Labels for Unique Identification of Digital Data, Society of Motion Picture and Television Engineers, 1997
14. SMPTE 330M, Unique Material Identifier (UMID), Society of Motion Picture and Television Engineers, 2000
15. ATSC A/57, Program/Episode/Version Identification, Advanced Television Systems Committee, 1996
16. SMPTE RP168, Definition of Vertical Interval Switching Point for Synchronous Video Switching, Society of Motion Picture and Television Engineers, 2002
17. ANSI/EIA/TIA-250-C, Electrical Performance for Television Transmission Systems, Electronic Industries Association (EIA), 1989.
18. ANSI/TIA/EIA-232-F, Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange, Electronic Industries Association (EIA), 1997
19. IETF RFC 1305, Network Time Protocol (Version 3), Specification, Implementation and Analysis, The Internet Society, 1992.
20. IETF RFC 1661, The Point-to-Point Protocol (PPP), The Internet Society, 1994.

4.3 Published Materials

(None)

5.0 OVERVIEW

The block diagrams below (Figure 5-1 and Figure 5-2) are based on Figure 6-1 from SCTE 30. They show a single Automation System, a single Injector and a single splicer. In reality, the Injector is part of an overall Digital Compression System (DCS), which includes several other Injectors (for other channels), multiplexers, and usually conditional access. Typically this system will include redundancy, to prevent a single device failure from forcing the system offline. Splice_info_section injection may actually be done by encoders, multiplexers, or other devices. As a result, the injecting device will be referred to in the rest of this document simply as an “Injector.”

All of these components are under the watch of a master Provisioning and Alarm Management System, or “PAMS”. The primary task of the PAMS is to monitor device health within the DCS, to notify human operators of any failures, and to switch redundant units into service as directed by the operator. The secondary task of the PAMS is to provide provisioning for the equipment contained within the DCS. Provisioning is usually defined as setting service parameters for each device. The Automation System, the Digital Compression System, and the PAMS are frequently located at the program origination facility, sometimes referred to as the “uplink” facility.

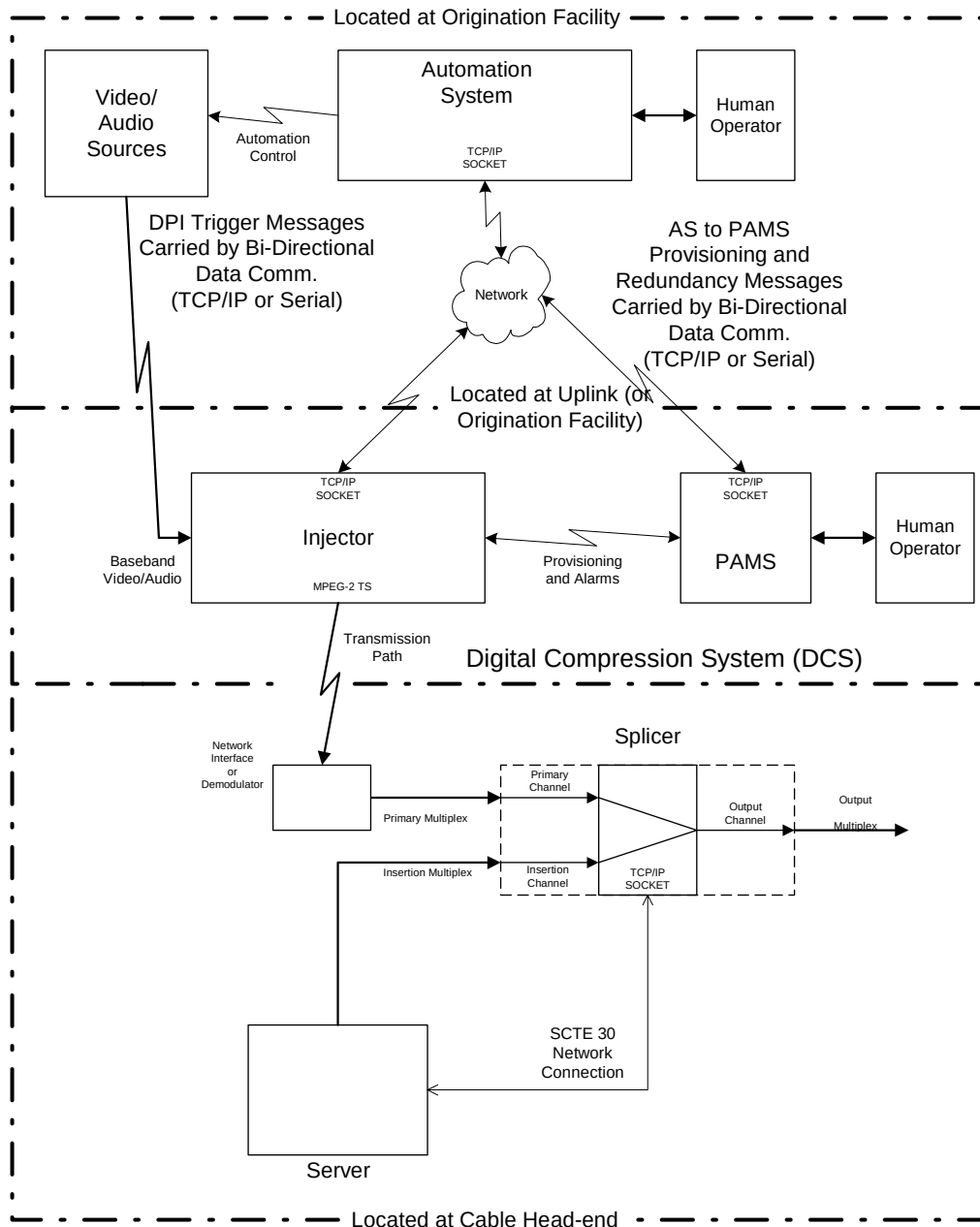
Note that TCP/IP networks for use with this standard are intended as strictly private, closed networks for the use of the Automation, Compression, and Splicing systems. As a result, latency is not expected to be a major factor in system design. None of these should be connected to either the commercial Internet or any other LAN or WAN without appropriate routing and firewall systems in place to ensure exclusion of intrusive traffic, planned (malicious) or unplanned (accidental).

Latency in a moderately trafficked TCP/IP network should be much less than 1 video frame time (33.37 ms for NTSC and 40 ms for PAL). As a result, the use of time-stamping is not mandatory, and thus is an optional portion of this API.

The following two end-to-end system block diagrams are intended as informative high-level overviews of the components of systems compliant with this standard. They illustrate both bi-directional (TCP/IP or serial) data communications, as well as uni-directional (video conveyed or serial) data communications.

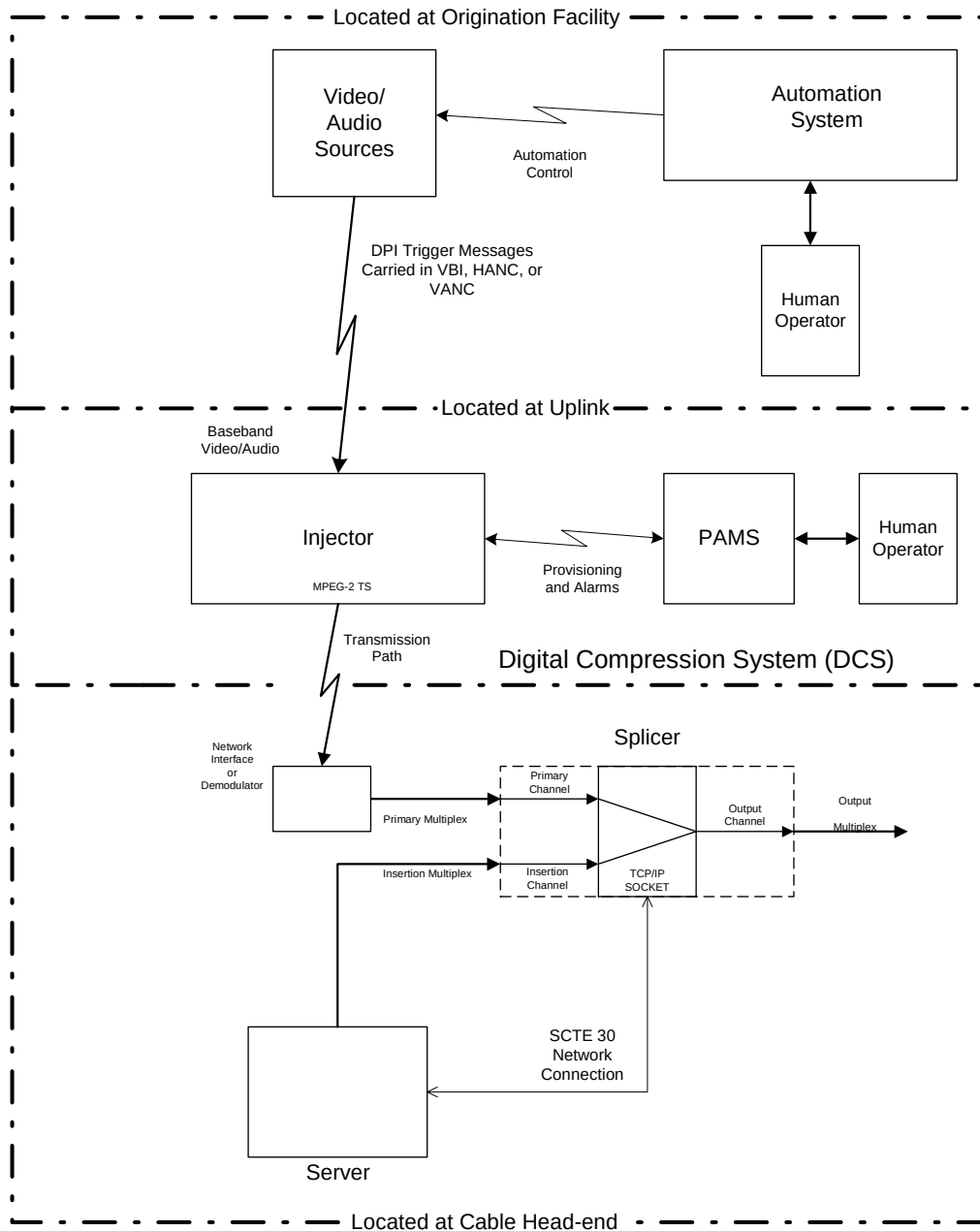
Actual system architectures are discussed in Section 12.0.

In addition to TCP/IP, this API also supports data communications through other physical layers, such as uni-directional or bi-directional serial (ANSI/TIA/EIA-232-F) or uni-directional serial digital video (VANC or HANC). In these systems the exact physical carriage is implementer defined, and the API messages are formatted the same regardless of the underlying implementation. For these system designs, the use of time-stamping is highly recommended.



Overall System Block Diagram (Informative)
(end-to-end)
Bi-directional Data Communications of API
Messages

**Figure 5-1 – SCTE 35 Overall System Block Diagram
with Bi-directional Data Communications**



Overall System Block Diagram (Informative)
(end-to-end)
Uni-directional Data Communications of API
Messages

**Figure 5-2 – SCTE 35 Overall System Block Diagram
with Uni-directional Data Communications**

6.0 DATA COMMUNICATIONS

The data communications system for this Standard can be described according to the Open Systems Interconnection (OSI) Basic Reference Model specified in ITU-T X.200. According to this functional model, information and services may be delivered from device to device by arranging the information into logical groupings or messages, delivering them to lower functional layers for transmission and, after reception, reconstituting the information into the proper form for use by the recipient.

6.1 Concerning Data Communications (Informative)

In what follows, the names of the layers are those adopted by the ISO and the ITU in ITU-T X.200 (1994), Open Systems Interconnection (OSI) -- Basic Reference Model [Informative Reference 12].

Some of these names are also commonly used in broadcasting technology to express different concepts. This particularly applies to the terms “network” and “link” and care must be taken to avoid confusion. This is especially important to readers of this Standard, since concise usage of terminology may confuse others who are less familiar with this Standard and the OSI Reference Model.

Readers unfamiliar with the OSI Reference Model are referred to the many tutorial web sites available which explain these concepts in detail. The layers of the Reference Model are: Layer 1: Physical; Layer 2: Link; Layer 3: Network; Layer 4: Transport; Layer 5: Session; Layer 6: Presentation; and Layer 7: Application.

6.2 Data Communications Requirements for this API (Normative)

This Standard defines the Application, Presentation, and Session Layers of the OSI Basic Reference Model and relies upon other well-defined Standards to provide the lower-level Layers necessary to function.

The data communication requirements for this Standard are based on those of SCTE 30 2001, which expects a high quality-of-service, bi-directional, connection-oriented, end-to-end reliable communications system using TCP. These expectations should be understood as the norm for this API. In this case, TCP over IP will provide the bottom 4 Layers of the OSI Basic Reference Model.

In addition to the above normative data communications system requirements, Automation System to Injector messages defined in this Standard may also be carried over a low noise, high quality-of-service, bi-directional point-to-point communications systems consisting of encapsulation within serial digital (SDI) video signal(s) in one or both directions (or a return path of suitable bandwidth capable of carrying TCP segments in compliance with RFC 793

[Informative Reference 10]). Within SDI video, the segments shall be carried in VANC or HANC.

A subset of the Automation System to Injector messages defined in this Standard may also be carried over a low noise, high quality-of-service, uni-directional, point-to-point communications system using encapsulation within a video signal. The physical conveyance may be either NABTS [Normative Reference 5] or WST [Normative References 4, 6, and 7] (for analog composite systems) or VANC/HANC [Normative References 8 and 9] for serial digital component systems. In this case, those communications methods will provide the bottom 4 Layers of the OSI Basic Reference Model. A full set of these messages may be carried in VANC/HANC for a serial digital component system, however a subset of them will not actually be used (see Section 12.1).

It is also possible to construct a high quality-of-service, connection-oriented, bi-directional communications system with one direction conveyed within the video signal and responses via a different path. Such a system will take careful engineering and may have some additional risks. Such a system may be able to gracefully degrade to a standard uni-directional, connection-oriented communications system upon loss of the return path

6.3 Conveyance Quality-of-Service Considerations (Informative)

The fundamental requirement for all modes of operation under this Standard is to provide high quality of service to the API messages. For TCP/IP communications, this may seem obvious. For video conveyance, the requirements are less obvious.

Serial digital component video must have a signal loss in a link of less than 30 dB (see SMPTE 259M, [Informative Reference 2]), which translates into a maximum bit-error rate of 2×10^{-7} or a signal to noise ratio of better than 17.1 dB. This is actually marginal performance, since it translates into an error per frame of video (most viewers will judge it noisy). Realistic performance of a link should be a SNR of better than 20 dB (a BER of 8×10^{-14}) which in viewer's terms is one error a day.

In analog video terms, this level of performance requires ANSI/EIA/TIA-250-C [Informative Reference 9] "short haul" link performance, with a minimum analog signal-to-noise ratio of 57 dB.

It is recognized that all "real world" communications systems may be subject to periodic degradation from external sources ("rain fade") that temporarily add considerable noise to the link. As a result, the conveyance requirements outlined in this document will endeavor to add extra link margin to their message designs.

6.4 Uni-directional System Considerations (Informative)

The requirements for uni-directional conveyance in the video signal are reasonably straightforward. The messages will be inserted via an insertion unit designed for inserting signals in video.

This Standard assumes correct delivery of each message. It must be understood that in a uni-directional, video conveyed system, the Automation System may choose to operate in a “best effort is OK” manner, and retransmit messages at least twice to ensure they have been completely received. Such a system architecture may be desirable where the Digital Compression System is located some distance away from the origination facilities (and hence, the Automation System) and no return path can be provided.

7.0 MESSAGE FORMATS

Messages in this API all possess a general message structure that wraps the data for the specific requests or responses being sent. This is done so that when the message is received, a common parsing routine can store it, determine what the structure of the data is, and ensure that the request and/or response and associated data is processed correctly. The end result of operations carried by this API are the placement of SCTE 35 Transport Stream (TS) private sections in the outgoing TS and transmission to downstream splicing equipment. Within this document, these private sections will be referred to as “splice_info_sections,” using the specific terminology of SCTE 35.

7.1 Terminology (Informative)

The following terms will be used to indicate which level of the communications structure is being discussed. A splice_info_section will indicate information in the resultant TS, on one or more PIDs designated for this purpose, which communicate with downstream splicing devices. A “message” will indicate information communicated between the Automation System and the Compression System via this API. A given “operation” may be termed a “request” or a “response,” and will indicate an individual specific action to be taken by either the Compression System or the Automation System. Such action may result in a splice_info_section being generated.

There are 4 different categories of operations (requests and responses) provided by this API. These are “Basic,” “Normal,” Control, and “Supplemental.” Basic operations supply the base communications required to support the system. Normal operations supply the base DPI-related functions (splicing, schedules, etc.) Supplemental operations are modifiers of Normal operations. Control operations manage the Control Word database required for encryption support. Detail is provided in Section 7.3.1.

7.2 Message Structures (Normative)

Messages in this API are defined assuming they will be carried via TCP/IP and all delivered as part of a single datagram.

Messages in this API carried via TIA/EIA-232-F (or TIA/EIA-422-B) shall utilize the Basic Link Layer Syntax specified in Appendix B.

Messages in this API carried in video (analog or digital) shall also utilize the Basic Link Layer Syntax specified in Appendix B. The implementation details are left to the system manufacturer. A discussion of the link layer requirements is found in Appendix C.

Where field lengths in the resulting SCTE 35 splice_info_section are less than a byte, they are padded on the MSB side to fill an even byte count for ease of debugging. The high-order byte in multiple byte fields is transmitted first, the lower order byte last. The Injector can pull the required number of bits from the message in forming the resulting actual splice_info_section TS packet. Each message begins with an operation identifier field, followed by a length field.

7.2.1 Addressing of Particular Items within a System

Two variables are provided in each of the messages to ensure the ability to uniquely identify the origination and the destination of messages. For a request for section insertion into an output TS, **AS_index** identifies the Automation System generating the request and the specific program component (**DPI_PID_index**) for which the resulting SCTE 35 splice_info_section is intended. For responses, this indicates the specific Automation System (**AS_index**) for which the response is intended.

The presence of these variables within this API is not intended to require support of the generation of multiple DPI PIDs by a single Injector, since the support of multiple DPI PIDs is optional (see Section 10.4).

7.2.1.1 AS_index

AS_index uniquely identifies the source of the message (since it is possible to have several automation systems active at once). The number ranges from 0 to 255 and shall be zero if this index is not required. This variable takes the value returned by the “AS_index” field of the config_response message (see Section 9.3.2). A redundant AS shall be assigned one single value of **AS_index** which applies to both primary and backup. Either the primary or the backup is active at a given time, but not both. An Injector Instance shall be connected to only one AS at a given time. If non-zero, **AS_index** shall be unique within a single DCS.

In systems where the PAMS to AS communications are not utilized, it is the operational responsibility of the Digital Compression System operator and the Automation System operator to each assign values such that they are unique for each automation system communicating with a given Injector Instance through this API and that only one automation system at a time will communicate with a given Injector Instance (a single value of **DPI_PID_index** for that Injector). The Injector will insure that messages received via the automation interface will only be used if authorized.

7.2.1.2 DPI_PID_index

DPI_PID_index specifies the index to the DPI PID which will carry the resulting splice_info_sections. The number ranges from 0 to 65535. **DPI_PID_index** shall be zero if not required by the system architecture.

The **DPI_PID_index** allows a given Automation System to direct messages to a specific DPI PID within a specific MPEG program in a specific Transport Stream (TS) within the purview

of the operational system (DCS). This is especially important when there are multiple DPI PIDs referenced by the PMT of a single MPEG program.

DPI_PID_index is required only if multiple Injector Instances (logical injectors) are present for any physical connection or if one or more Injector Instances are generating more than one DPI PID. Examples of situations requiring non-zero values of **DPI_PID_index** are multiple injectors listening to the same physical connection, such as multiple injectors receiving the same video stream, or multiple Injector Instances located behind a single IP address and port number.

Ordinarily, there shall be one value of **DPI_PID_index** for each DPI PID referenced by a program's PMT for each program within the purview of the DCS. The exception to the rule is the case where a single DPI PID is shared by more than one program within a single TS. In this case, more than one PMT may make reference to the same shared DPI PID via a common value for **DPI_PID_index**.

Multiple language versions of the same movie are an example where this facility may be utilized. The AS is expected to know what these programs are and that the same value of **DPI_PID_index** may be assigned for each. In this example, the different programs share a video PID but have different audio PIDs for each language. The associated DPI PID for the video could be the same or different in this case.

The AS may validate for shared PIDs before sending a provisioning_response message (see Section 9.5.1.2).

In all other circumstances, each value of **DPI_PID_index** shall be unique.

This value is normally furnished to the AS by the PAMS during system initialization as part of the Injector Service Notification (via the provisioning_request message, see Section 9.4). In systems without PAMS to AS service, this value must be manually provided to the automation system.

It is recommended that even trivial system architectures utilize non-zero values of **DPI_PID_index**.

7.2.2 Single Operation Message

This variable length structure carries a single instance of an operation (request or response as it will be normally termed) listed in Table 7-3 and whose structural details are provided in Sections 8.0 and 9.0 of this document.

Operations listed in Table 7-3 shall use the single_operation_message() and shall not use multiple_operation_message().

Table 7-1 - single operation message

Syntax	Bytes	Type
single_operation_message() {		
opID	2	uimsbf
messageSize	2	uimsbf
result	2	uimsbf
result_extension	2	uimsbf
protocol_version	1	uimsbf
AS_index	1	uimsbf
message_number	1	uimsbf
DPI_PID_index	2	uimsbf
data()	*	Varies
}		

7.2.2.1 Semantics of fields in single_operation_message()

opID – An integer value that indicates what message is being sent. See Table 7-3. It shall only take values whose “Usage” column entries are listed as “Basic Request” or “Basic Response.”

messageSize – The size of the entire single_operation_message() structure in bytes.

result – The results to the requested message. See Section 13.0 (Result Codes) for details on the result codes. For message Usage types (as shown in the Usage column of Table 7-3) other than Basic Response messages, this shall be set to 0xFFFF.

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

protocol_version – An 8-bit unsigned integer field whose function is to allow, in the future, this message type to carry parameters that may be structured differently than those defined in the current protocol. It shall be zero (0x00). Non-zero values of

protocol_version may be used by a future version of this standard to indicate structurally different messages.

AS_index – Defined in Section 7.2.1 above. Response messages return the value from the corresponding Request message.

message_number – The **message_number** can be any number in the range 0 to 255. The **message_number** must be unique for the life of a message within a given Injector Instance. The **message_number** is used to identify an individual request.

When multiple copies of the same message are sent, they can be identified because they have the same **message_number**. If not in current use, the **message_number** may freely vary over the range of 0 to 255.

In a uni-directional system, the message number can be assumed to be available for reuse after the last copy of the message is sent and after the associated processing timestamp() time has passed.

DPI_PID_index – Defined in Section 7.2.1 above. Response messages return the value from the corresponding Request 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** minus the size of the fixed portion of **single_operation_message()** and is determined by the size of the data being added to the message. Not all messages utilize the **data()** field.

7.2.3 Multiple Operation Message

This variable length structure carries one or more of the operations (or requests) listed in Table 7-4, which must be either “Normal”, “Control”, or “Supplemental” in Usage category and whose structural details are provided in Section 8.0 of this document. Each request in the **data()** structure includes a **opID** value (2 bytes) and a length (2 bytes). Thus the first 4 bytes of every request within the repeating structure is identical to easily permit a receive device to skip a request if the **opID** is unknown. This allows for extensions to the protocol in the future.

Use of the **multiple_operation_message()** will normally result in the insertion of at least one SCTE 35 **splice_info_section** into the resultant TS, unless the Injector (IJ) detects fatal errors in the message. In multiple byte fields the first byte received is the most significant byte.

Note that the use of the **multiple_operation_message()** will result in a **single_operation_message** in response, since response messages are defined as Basic Usage responses (which, by definition, use the **single_operation_message**).

7.2.3.1 Order of Request Execution

This structure permits multiple requests to be grouped together to permit transmission in one message (and execution as appropriate). Its use is permitted in both bi-directional (serial or TCP/IP-based) and uni-directional systems. The data() structure is populated with one or more of the request structures defined in Section 8.0 (within the constraints identified elsewhere in this document). The time of processing may be instantaneous or delayed, as required.

All requests are executed in the order that they exist within the data() structure. If requests are time based, then the time is referenced to the start of the video frame that the last byte is received, not the frame in which it was actually processed.

Requests listed in Table 7-3 shall not use the multiple_operation_message().

Some requests are order dependant, such as the various Supplemental requests. The Supplemental request modifies the characteristics of a Normal request, so they must be carried following the associated Normal request. In this way, multiple Normal requests with Supplemental requests can be carried without confusing which Supplemental request is associated with which Normal request.

Each instance of data() shall begin with a Normal or a Control request. A Normal request may be followed by zero or more Supplemental requests which modify or augment it. Unless otherwise specified, Supplemental request operations may occur in any order, except that they must follow the Normal operation to which they apply. It may then be followed by additional Normal requests for which the AS requests time deferral. The placement of a new Normal request shall indicate that the definition of the preceding Normal request is complete and that the resulting SCTE 35 splice_info_section can be formatted and output at the time indicated by timestamp().

As used here, the term “processed” refers to whatever operations the Injector must accomplish to emit an SCTE 35 section or sections or change a CW database. Processing begins when the timestamp() time has expired and ends when the section or sections are placed in the TS or the database is updated.

7.2.3.2 Format of the multiple_operation_message() structure

Table 7-2 - multiple operation message

Syntax	Bytes	Type
multiple_operation_message() {		
Reserved	2	uimsbf
messageSize	2	uimsbf
protocol_version	1	uimsbf
AS_index	1	uimsbf
message_number	1	uimsbf
DPI_PID_index	2	uimsbf
SCTE35_protocol_version	1	uimsbf
timestamp()	*	Varies
num_ops	1	uimsbf
for (i=0; i < num_ops; i++) {		
opID	2	
data_length	2	
data()	*	Varies
}		
}		

7.2.3.3 Semantics of fields in multiple_operation_message()

Reserved – This field shall be set to all ones (0xFFFF).

messageSize – The size of the entire multiple_operation_message() structure in bytes.

protocol_version – An 8-bit unsigned integer field whose function is to allow, in the future, this message type to carry parameters that may be structured differently than those defined in the current protocol. It shall be zero (0x00). Non-zero values of **protocol_version** may be used by a future version of this standard to indicate structurally different messages.

AS_index – Defined in Section 7.2.1 above.

message_number – An integer value that is used to identify an individual message. The **message_number** variable must be unique for the life of a message. When multiple copies of the same message are sent, they can be identified because they have the same **message_number**. This means that for messages that are to be processed in the future, the **message_number** may not be reused until the message has been processed. If not in current use, the **message_number** may freely vary over the range of 0 to 255.

In a uni-directional system, the message number can be assumed to be available for reuse after the associated processing timestamp() time has passed.

DPI_PID_index – Defined in Section 7.2.1 above.

SCTE35_protocol_version – This 8-bit unsigned integer field indicates the version of SCTE 35 protocol that the section which results from this message conforms to. Its function is to allow, in the future, this section type to carry parameters that may be structured differently than those defined in the current protocol. At present, the only valid value defined by SCTE 35 is zero (0x00). Non-zero values of **SCTE35_protocol_version** may be used by a future version of this standard to indicate structurally different sections.

timestamp() – This field delivers the exact time to process all of the requests in this message (see Section 11.5). The **time_type** field of timestamp() may be zero, indicating the messages are processed immediately. The timestamp() may contain either the UTC time or the VITC time specifying when to process the requests. The timestamp() may alternatively contain the number of the GPI to use for triggering the messages to be processed. Once the GPI is triggered, all requests associated with that edge of the GPI will be processed.

num_ops – An integer value that indicates the number of requests contained within the data() loop.

opID – An integer value that indicates what request is being sent. See Table 7-4.

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

data() – Specific data structure for the request being sent. Details on each of the requests containing data are described in Sections 8.3.1, 8.4, 8.5, 8.7, and 8.8 of this document. The size of this field is equal to **data_length** and is determined by the size of the data being added to the multiple_operation_message() structure.

7.2.3.4 Detailed Discussion of Message Syntax and Semantics

Note that each **opID** in Table 7-4 has an associated “Usage” column, which indicates the class of each request. Normal requests have no associations with other requests and (once the time value specified in the timestamp() structure is reached) are immediately

formatted into the appropriate SCTE 35 message and dispatched. Each Normal request may be followed by zero or more “Supplemental” requests. The Supplemental requests must follow immediately after the Normal request that they are modifying. Some Supplemental requests are specific to a certain type of Normal request. Others are a general Supplemental request that can be associated with any Normal request, when appropriate. The Injector must ensure in processing any Normal requests that it checks for the existence of associated Supplemental requests before inserting the transport packet into the multiplex.

For the Control requests, only one request per Control Word index is permitted within a single `multiple_operation_message()`. It is permitted to send several requests in the same message, each operating on different Control Words. For example, update `CW_index 1` and delete `CW_index 2` in the same message is permitted. It would not be permitted to update `CW_index 1` and then delete `CW_index 1` within the same message.

DPI Schedules are potentially very large. The system is downloading a playlist of future ad avail periods, one splice point at a time. There is a single start message, and a single stop message, to frame the downloading of the data. Like other messages in this API, the schedules have Normal and Supplemental features. If Supplemental features are required, they must be included in the same message as the basic schedule request, and immediately following the associated basic request.

If multiple Normal requests are present in a message, then the requests are processed in the same order that they appear in the message. If the **time_type** field of `timestamp()` is zero, all Normal request timing is relative to the arrival time of the last byte of the message. Please see Section 7.2.3.1 for additional information.

7.3 Operation Types (Normative)

Table 7-3 and Table 7-4 contain the assigned values for each type of operation (request or response) supported by this API. Other columns in the tables list information identifying the normal originator and recipient, and other useful information.

Those operations required for the Simple Profile appear in the column labeled “In Simple Profile,” with an indication of “Y.” An “N” indicates that support of the Request is not required for compliance. “n/a” indicates “not applicable.” With the sole exception of the “`general_response_data()`” message, compliant implementations may also omit support for those messages in Table 7-3 which show PAMS as either the “Sent By” or “Sent To” when the PAMS is not a constituent portion of the overall system. Systems should with PAMS as constituent portion of the overall system should indicate this as “Simple Profile with PAMS,” or, if applicable (and as an example), “Simple Profile plus encryption with PAMS.”

Table 7-3 - opID Assigned Values and Meanings for single_operation_messages

opID assigned value	Operation Name	Sent By	Sent To	In Simple Profile	Description	Usage
0x0000	general_response_data()	PAMS, Automation or Injector	PAMS, Automation or Injector	Y	Used to convey asynchronous information between the devices. There is no data() associated with this message.	basic response
0x0001	init_request_data()	Automation	Injector	Y	Initial Message to Injector on predefined port	basic request
0x0002	init_response_data()	Injector	Automation	Y	Initial Response to Automation on the established connection	basic response
0x0003	alive_request_data()	Automation	Injector	Y	Sends an alive message to acquire current status.	basic request
0x0004	alive_response_data()	Injector	Automation	Y	Response to the alive message indicating current status.	basic response
0x0005 - 0x0006	User Defined			n/a	Receiving devices shall ignore these values. Used in legacy systems.	
0x0007	inject_response_data()	Injector	Automation	Y	Response to indicate that the request was received and that Injector is preparing to send SCTE 35 message or messages.	basic response

opID assigned value	Operation Name	Sent By	Sent To	In Simple Profile	Description	Usage
0x0008	inject_complete_response_data()	Injector	Automation	Y	Response from Injector when all resultant SCTE 35 splice messages are sent.	basic response
0x0009	config_request_data()	Automation	PAMS	n/a	Automation sends PAMS its IP configuration	basic request
0x000A	config_response_data()	PAMS	Automation	n/a	Responds to Config_Request	basic response
0x000B	provisioning_request_data()	PAMS	Automation	n/a	PAMS notification of the Injectors provisioned for DPI service	basic request
0x000C	provisioning_response_data()	Automation	PAMS	n/a	Response from Automation that the message is received and DPI is starting	basic response
0x000D -0x000E	Reserved			n/a	Range Reserved Used in legacy systems.	
0x000F	fault_request_data()	Automation	PAMS	n/a	Automation discovered communication problem with an Injector	basic request
0x0010	fault_response_data()	PAMS	Automation	n/a	Response from PAMS	basic response
0x0011	AS_alive_request_data()	PAMS	Automation	n/a	Maintain PAMS to AS communications	basic response
0x0012	AS_alive_response_data()	Automation	PAMS	n/a	Maintain AS to PAMS communications	basic response
0x0013 -0x00FF	Reserved for future basic requests			n/a	Range Reserved for future	

opID assigned value	Operation Name	Sent By	Sent To	In Simple Profile	Description	Usage
	or responses				standardization.	
0x0100 -0x7FFF	Reserved			n/a	Range Reserved for Table 7-4 uses	
0x8000 -0xBFFF	User Defined	Automation or PAMS	Injector or PAMS	n/a	Range available for user defined functions.	
0xC000 - 0xFFFE	Reserved				Range Reserved for user defined Table 7-4 uses.	
0xFFFF	Reserved				Reserved value	

Table 7-4 - opID Assigned Values and Meanings for multiple_operation_messages

opID assigned value	Operation Name	Sent By	Sent To	In Simple Profile	Description	Usage
0x0000 - 0x00FF	Reserved			n/a	Range Reserved (see Table 7-3).	
0x0100	inject_section_data_request()	Automation	Injector	Y	Generates an SCTE 35 section directly	Normal
0x0101	splice_request_data()	Automation	Injector	Y	Normally used request to send SCTE 35 message or messages.	Normal
0x0102	splice_null_request_data()	Automation	Injector	Y	Generates an SCTE 35 splice_null operation	Normal
0x0103	start_schedule_download_request_data()	Automation	Injector	N	Initiates schedule download	Normal
0x0104	time_signal_request_data()	Automation	Injector	Y	Generates an SCTE 35 time_signal operation	Normal
0x0105	transmit_schedule_request_data()	Automation	Injector	N	Initiates schedule transmission	Normal
0x0106	component_mode_DPI_request_data()	Automation	Injector	N	Adds component mode to a DPI request	Supplemental
0x0107	encrypted_DPI_request_data()	Automation	Injector	N	Adds encryption to a DPI request	Supplemental
0x0108	insert_descriptor_request_data()	Automation	Injector	Y	Adds a descriptor to another operation	Supplemental
0x0109	insert_DTMF_descriptor_request_data()	Automation	Injector	Y	Adds a DTMF descriptor to another operation	Supplemental
0x010A	insert_avail_descriptor_request_data()	Automation	Injector	Y	Adds an avail_descriptor to the SCTE 35	Supplemental

opID assigned value	Operation Name	Sent By	Sent To	In Simple Profile	Description	Usage
					section	
0x010B	insert_segmentation_descriptor_request_data()	Automation	Injector	Y	Adds a segmentation descriptor to another operation	Supplemental
0x010C	proprietary_command_request_data()	Automation	Injector	Y	Adds a proprietary descriptor to another operation	Supplemental
0x010D	schedule_component_mode_request_data()	Automation	Injector	N	Adds component mode to an avail definition	Supplemental
0x010E	schedule_definition_data() request	Automation	Injector	N	Single avail definition	Supplemental
0x010F - 0x02FF	Reserved			n/a	Range Reserved for future standardization (additional Normal or Supplemental operations).	
0x0300	delete_ControlWord_data()request	Automation	Injector	N	Maintains CW database	Control
0x0301	update_ControlWord_data() request	Automation	Injector	N	Maintains CW database	Control
0x0302 - 0x7FFF	Reserved			n/a	Range Reserved for future standardization (additional Control operations).	
0x8000 - 0xBFFF	Reserved			n/a	Range Reserved (see Table 7-3).	
0xC000 - 0xFFFFE	User Defined	Automation or PAMS	Injector or PAMS	n/a	Range available for user defined functions for multiple operation messages.	
0xFFFF	Reserved				Reserved value.	

7.3.1 Meaning of the Usage Field in Table 7-3 and Table 7-4

The Usage field indicates the class of each request or response and the messages with which they may be used:

- Basic requests or responses shall always use the `single_operation_message()` structure (see Section 7.2.2).
- Normal requests shall have no linkage with other Normal requests and are normally formatted into the appropriate SCTE 35 `splice_info_section` and dispatched. Normal requests shall use the `multiple_operation_message()` structure (see Section 7.2.3.2). While multiple Normal requests may be grouped together into a single instance of `multiple_operation_message()`, they may not have any dependencies beyond execution order (see Section 7.2.3.1).
- Supplemental requests are also carried only by the `multiple_operation_message()` structure (See Section 7.2.3). Each Supplemental request follows immediately after the Normal request that they are modifying. Some Supplemental requests are specific to a certain request. Others are a general request that can be associated with any Normal request, when appropriate.
- Control requests are also carried only by the `multiple_operation_message()` structure (See Section 7.2.3). Each Control request shall be independent of any other contained within the same `data()` structure and shall be executed at the time specified in the `timestamp()`. Multiple Control requests may be present within the `data()` structure. Supplemental requests do not modify Control requests.

7.4 Conventions and Requirements

1. Each message that contains data is outlined with its data fields and types below. Additional structures are indicated as functions and are described in Section 11.0 of this document.
2. The Injector shall retain the following data values while messages are being processed:
 - `message_number`
 - `splice_event_id`

These are retained until the `inject_complete_response` message is sent to the AS. In addition, each Injector which supports splice schedule messages must retain any descriptors defined via this API during the output of the individual SCTE 35 `splice_schedule()` sections which result from a single `schedule_definition` request (see Section 8.7).

3. All string lengths have space reserved for a null terminator character (0x00) and shall use null terminated strings. The size defined for the string is constant and will not vary depending on the actual length of the string. As an example a string that is defined as 16 characters can have at most 15 characters of data followed by a null character. Once a null is encountered in scanning a string, the rest of the characters in the string are undefined and ignored. This specification uses 8 bit ASCII characters for strings.
4. 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 the Automation System suspects a timeout, it shall send an `alive_request` message. If the Injector does not answer as specified in this document, the connection for this channel shall be dropped and re-established.
5. Initialization (or re-initialization) of the communications between the AS and the Injector shall not cause interruption of any of the audio, video, or DPI message insertions currently being processed by either the AS or the Injector. Initialization can be safely conducted at any point in time. This includes changes to Injector services or Injectors themselves. These events may be expected to occur at random intervals.
6. When a device is polling to start or restart communications, a suitable interval (30 to 60 seconds) may be left between attempts. Such an interval might be randomly determined, with exponential backoff, as is commonly used in Ethernet-based protocols.

8.0 AUTOMATION SYSTEM TO INJECTOR COMMUNICATION

8.1 Initialization

The methods of initializing the TCP/IP communications parameters are discussed in Section 9.4

For TCP/IP, the initial communication begins with Injector listening on predefined port 5167 and the Automation System opening an API Connection to the Injector via that socket. If another socket number has been furnished in the `provisioning_request` message (via the `injector_socket_number` field), that socket should be used instead of the default socket 5167. The Automation System sends an **init_request** message to the Injector. The Automation System then listens for the response from the Injector on the established API Connection. All further communication is done on this API Connection. Either the Automation System or Injector may terminate communications by closing this API Connection. Each device is responsible for detecting and properly handling a closed API Connection.

The Injector should support multiple Automation System connections simultaneously if provisioned to do so. When the Injector initializes the TCP listener on port 5167 it should allow for the number of API Connections it is provisioned for (see Section 10.4). No two Automation Systems may have an active connection to any given Injector Instance at any one time. The

Injector Instance shall return a response of “Injector already in use” (see Table 13-1) if this occurs.

The **protocol_version** fields in `single_operation_message()` and `multiple_operation_message()` permit the Automation System and the Digital Compression System to “negotiate” at which level of the protocol the system will function. The lesser value shall be taken as the operating point for the system as initialized. Please note that this value may have implications upon the possible values for the **SCTE35_protocol_version** field (see Sections 7.2.2 and 7.2.3).

In a uni-directional system, the AS and Injector must both be configured to operate at a compatible protocol version.

8.1.1 init_request AS ==> IJ

This basic usage request is sent by the Automation System to the Injector to initialize a TCP/IP connection. The appropriate value for desired **protocol_version** shall be furnished to the Injector in this message.

Table 8-1 - init_request_data

Syntax	Bytes	Type
<code>init_request_data(){ }</code>		

8.1.2 init_response IJ ==> AS

This basic usage response is sent by the Injector to the Automation System to indicate the receipt of the `init_request`. The appropriate value for desired **protocol_version** shall be furnished to the AS in this message. All devices supporting this API shall operate from this point forward at the lesser of the furnished **protocol_version** values.

Table 8-2 - init_response_data

Syntax	Bytes	Type
<code>init_response_data(){ }</code>		

8.2 Alive (“Heartbeat”) Communications

For bi-directional communications, once initialization is complete, then the Automation System shall send **alive_request** messages to ensure that the Injector and the communications path remain up and running. Each **alive_response** message (wrapped in the `single_operation_message()`) contains a **result** field that may be used to signal if DPI support has been stopped on the recipient’s end. If there has been no activity on the connection in the preceding 60 seconds, then an **alive_request** message shall be sent.

If TCP/IP is being used and the user de-provisions DPI support in the Injector, the Injector will close the socket connection to the Automation System without waiting for the next **alive_request**.

For uni-directional communications this message also serves to provide a mechanism that the receiving device shall use to verify a working connection to the automation computer. This message shall be sent at least once every 60 seconds. If the messages fail to arrive, then the receiving Injector shall notify its PAMS or a human operator that communications may be lost.

The second function is to provide clock synchronization for UTC or VITC time-stamped splice messages. The time () structure provides the time for the start of the associated video frame. This requires the sender and the receiver to both be examining synchronous video of the same frame rate. In multi-standard systems, this requirement is very important.

The receiving device can synchronize to the vertical interval of its incoming video and the received time () value and thus maintain a local UTC or VITC time base to use with time-stamped messages.

For TCP/IP-based systems, implementers may choose to use an external time standard to keep the internal clocks of the Automation System and the Injector in sync. This is not strictly necessary for the simplest implementation that meets the requirements of SCTE 35.

If the Automation System has access to a facility master clock, and it makes sense to both parties, then the current value of facility time-of-day timecode can be transmitted in the “**alive_request**” messages from the Automation System to the Injector and conversely in the Injector to the Automation System “**alive_response**” responses. Alternatively, facility time-of-day time samples may be conveyed to the Injector in the video signal proper as VITC.

8.2.1 alive_request AS ==> IJ

This basic request serves to ensure that the AS to Injector communications path remains open and reliable. In addition it may be used to ensure the internal time within each is synchronized. If deferred requests are to be used with a time-value trigger, then it is vital that synchronization be maintained.

Table 8-3 - alive_request_data

Syntax	Bytes	Type
<pre>alive_request_data(){ time() }</pre>		

8.2.1.1 Semantics of fields in alive_request_data ()

time() – This is an optional structure, unless the **time_type** field of the timestamp() structure carried in multiple operation messages is non-zero. The current UTC time clock of the sending device checked as close as possible to the sending of the message. This is designed to be used by the Injector and the Automation System to check on how well the two systems are time synchronized. See Section 11.4 for a definition of **time()**. If this time synchronization is not being used in a given system, the value of time() may be set to zero.

8.2.2 alive_response IJ ==> AS

This basic response serves to ensure that the AS to Injector communications path remains open and reliable. In addition it may be used to ensure the internal time within each is synchronized. If deferred requests are to be used with a time-value trigger, then it is vital that synchronization be maintained.

Table 8-4 - alive_response_data

Syntax	Bytes	Type
<pre>alive_response_data(){ time() }</pre>		

8.2.2.1 Semantics of fields in alive_response_data ()

time() – This is an optional structure, unless the **time_type** field of the timestamp() structure carried in multiple operation messages is non-zero. The current UTC time clock of the sending device checked as close as possible to the sending of the message. This is designed to be used by the Injector and the Automation System to check on how well the

two systems are time synchronized. See Section 11.4 for a definition of **time()**. If this time synchronization is not being used in a given system, the value of time() may be set to zero.

8.3 Splice Requests

After initializing communications with the Injector, the Automation System can issue (via a multiple operation message), one of the Normal requests listed in the Usage column of Table 7-4. Issuing typically a splice_request to initiate placement of one or more SCTE 35 splice_info_sections into the outgoing TS. The Automation System may choose to send any of the messages multiple times before the designated in-point (especially if return path communications is unavailable). The Injector can detect that these are duplicates of one another by comparison of the **message_number** fields.

The two messages that are returned (in a bi-directional system) from the splice request messages are the inject_response message and the inject_complete_response message. A inject_response message is returned upon receipt of the splice request. A inject_complete_response message is returned once the SCTE 35 section has been generated.

8.3.1 splice request AS ==> IJ

This Normal request is the usual carrier of splicing requests. It may be further elaborated upon by various Supplemental type requests which may follow it within the data() structure of a multiple_operation_message.

Table 8-5 - splice_request_data

Syntax	Bytes	Type
splice_request_data() {		
splice_insert_type	1	uimsbf
splice_event_id	4	uimsbf
unique_program_id	2	uimsbf
pre_roll_time	2	uimsbf
break_duration	2	uimsbf
avail_num	1	uimsbf
avails_expected	1	uimsbf
auto_return_flag	1	uimsbf
}		

8.3.1.1 Semantics of fields in splice_request_data()

splice_insert_type – An 8-bit unsigned integer defining the type of insertion operation desired. These will result in the generation of one or more SCTE 35 splice_info() sections with a **splice_command_type** field value of splice_insert with other inferred field values also being set within the resulting splice_info() section. The other inferred field values are noted with the discussion of each assigned value. Please refer to Section 8.3.2 below for additional clarification of the inferred values.