ISO/IEC 14165-321

Edition 1.0 2009-11

INTERNATIONAL STANDARD

Information technology – Fibre channel –
Part 321: Audio video (FC-AV)

ISO/IEC 14165-321:2009(E)

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

ISO/IEC 14165-321

Edition 1.0 2009-11

INTERNATIONAL STANDARD

Information technology – Fibre channel –
Part 321: Audio video (FC-AV)

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE X

ICS 35.200

ISBN 2-8318-9016-0

PATENT STATEMENT

CAUTION: The developers of this standard have requested that holder's of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Table of Contents

List of Tables

List of Figures

INFORMATION TECHNOLOGY –
FIBRE CHANNEL –

Part 321: Audio video (FC-AV)

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.

4) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IECpublication and the corresponding national or regional publication should be clearly indicated in the latter.

6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC publication.

7) All users should ensure that they have the latest edition of this publication.

8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.

9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

10) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14165-321 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

The list of all currently available parts of the ISO/IEC 14165 series, under the general title *Information technology - Fibre channel,* can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies and the voting results may be obtained from the address given on the second title page.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

INTRODUCTION

This International Standard defines a protocol for transmitting AV streams using Fibre Channel Sequences and Exchanges. Fibre Channel is a high speed serial interface using either optical or electrical connections (i.e., the physical layer) at data rates currently up to 2 Gbit/s with a growth path to 10 Gbit/s. The topologies supported by Fibre Channel include point-to-point, switched fabric, and arbitrated loop. Fibre Channel connections used for transmitting AV streams utilize standard FC frame format and sequence/exchange hierarchy.

The *Fibre Channel Audio-Video* (FC-AV) standard is divided into 9 clauses and 7 annexes as follows:

Clause 1 - Scope

Clause 2 - Normative references

Clause 3 - Definitions, abbreviations, and conventions

Clause 4 - Overview of the protocol for transmitting FC-AV containers or AV frames over Fibre Channel

Clause 5 - FC-AV Container system

Clause 6 - Compressed FC-AV Stream transmission

Clause 7 - Frame Header Control Protocol

Clause 8 - Simple Streaming protocol for Simple Content Movement Architecture

Clause 9 - SCSI-3 FCP mapping of the Simple Streaming protocol

Annex A (normative) - Simple Parametric Digital Video (SPDV) profile that defines a mapping based on the FC-AV Container system.

Annex B (normative) - Object Type data.

Annex C (normative) - Television video primer.

Annex D (informative) - Audio and video information sender to receiver synchronization issues

Annex E (informative) - Three techniques that are in common use to make TCP/IP go fast on fast networks

Annex F (informative) - FC-AV container Header for allowed Video Frame Rates

Annex G (informative) - Data packing guidelines.

INFORMATION TECHNOLOGY – FIBRE CHANNEL – Part 321: Audio video (FC-AV)

1 Scope

This part of ISO/IEC 14165-321 specifies the transport of digital Audio and Video formats over Fibre Channel.

Specifications are included for:

– a coherent framework (i.e., an FC-AV Container and Objects) for mapping current and future digital Audio and Video formats to Fibre Channel;

– mapping the formats defined by the ITU-R BT-601 and SMPTE family of standards to Fibre Channel;

– mapping the formats defined by the ISO/IEC 3818 family of standards (which include MPEG and related compression systems) to Fibre Channel;

– a profile (i.e., Simple Parametric Digital Video) that parametrically defines the characteristics of Audio and Video information for specific applications; and,

– data packing guidelines recommended for AV data within the Fibre Channel transmission words.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Availability and contact information is provided as needed.

IEC 61179, *Helical-scan digital composite video cassette recording system using 19 mm magnetic tape, format D2 (NTSC, PAL, PAL-M)*

IEC 61834 (all parts), *Recording – Helical-scan digital video cassette recording system using 6.35 mm magnetic tape for consumer use (525-60, 625-50, 1125-60 and 1250-50 systems)*

IISO/IEC 14165-251, *Information technology – Fibre Channel – Part 251: Framing Signaling (FC-FS)*

ISO/IEC 14776-222, *Information technology – Small Computer System Interface (SCSI) – Part 222: Fibre Channel Protocol for SCSI, Version 2 (FCP-2)*

ISO/IEC 14165-331, *Information technology - Fibre Channel - Part 331: Virtual Interface (FC-VI)*

AES3-1992(r1997)(ANSI S4.401992), *AES Recommended Practice for Digital Audio Engineering - Serial transmission format for two-channel linearly represented digital audio data*

ANSI X3.230-1994, *Fibre Channel – Physical and Signaling Interface (FC-PH)*

ANSI/SMPTE 125M-1995, *Television – Component Video Signal 4:2:2 Bit-Parallel Interface*

ANSI/SMPTE 170M-1994, *Television – Composite Analog Video Signal – NTSC for Studio Applications*

ANSI/SMPTE 253M-1998, *Television – Three-Channel RGB Analog Video Interface*

ANSI/SMPTE 259M-1997, *Television – 10-Bit 4:2:2 Component and 4fsc Composite Digital Signals – Serial Digital Interface*

ANSI/SMPTE 274M-1995, *Television - 1920 x 1080 Scanning and Interface*

ANSI/SMPTE 291M-1996, *Television - Ancillary Data Packet and Space Formatting*

ANSI/SMPTE 292M-1996, *Television - Bit-Serial Digital Interface for High-Definition Television Systems*

ANSI/SMPTE 293M-1996, *Television - 720 x 483 Active Line at 59.94-Hz Progressive Scan Production - Digital Representation*

ANSI/SMPTE 294M-1997, *Television - 720 x 483 Active Line at 59.94-Hz Progressive Scan Production - Bit-Serial Interfaces*

ANSI/SMPTE 296M-1997, *Television - 1270 x 720 Scanning, Analog and Digital Representation and Analog Interface*

ANSI/SMTPE 298M-1997, *Universal Labels for Unique Identification of Digital Data*

ANSI/SMPTE 305M-2000, *For Television - Serial Data Transport Interface*

ANSI/SMPTE 314M-1999, *Data Structure for DV-Based Audio, Data and Compressed Video - 25 and 50 Mbits/s*

ANSI/SMPTE RP-168-1993, *Definition of Vertical Interval Switching Point for Synchronous Video Switching*

ANSI/SMPTE RP 177-1993, *Derivation of Basic Television Color Equations (R1997)*

ANSI/SMPTE RP 211-2000, *Implementation of 24P, 25P, and 30P Segmented Frames for 1920x1080 Production Format*

ATSC A/53-1995, *Digital Television Standard*

CIE 15.2-1986, *Colorimetry, 2nd Edition*

International Telecommunication Union Recommendation ITU-R BT.601-5, *Encoding Parameters of Digital Television for Studios*

International Telecommunication Union Recommendation ITU-R BT.709-3, *Parameter values for the HDTV standards for production and international programme exchange*

3 Terms, definitions, abbreviations, keywords and conventions

3.1 Terms and definitions

The following definitions apply to this standard; words used that are defined in referenced standards shall use that definition; and, words not defined here or in the referenced standards shall have the standard technical English meaning.

3.1.1 active video: the portion of a complete video scan that is visible on the display screen.

3.1.2 Audio frame: an audio Object.

3.1.3 AV frame: an audio and/or video Object.

3.1.4 chrominance: the color portion of a video signal. Chrominance includes hue and saturation information but not brightness.

3.1.5 client: a requestor for action on a Stream (e.g., Record or Play).

3.1.6 clip: a set of Containers.

3.1.7 component video: the unencoded output of a camera or other device consisting of three primary color signals: red, green, and blue (RGB) that together convey all necessary picture information.

3.1.8 composite video: an encoded video signal, such as NTSC or PAL video, that includes horizontal and vertical synchronizing information.

3.1.9 Compressed A/V stream: see Video Program.

3.1.10 Content Movement: a full-duplex Streaming protocol for the inter-action and communication between the client and the server without respect to the underlying transport.

3.1.11 consumer: see Video Sink.

3.1.12 Container: see FC-AV Container.

3.1.13 Content: a set of audio frames, video frames, AV frames or Containers.

3.1.14 FC-AV Container: an entity that consists of a Container Header and Objects (see clause 5).

3.1.15 FC frame: the Fibre Channel frame as defined in FC-FS.

3.1.16 field: half of the interlaced horizontal lines needed to create a complete picture. Two interlaced fields create a complete frame or picture.

3.1.17 Full Stream Structure: content is defined in SMPTE 259M. It consists of an active video data (payload data), ancillary space data and synchronizing data.

3.1.18 Interlaced: a scanning system for video whereby the even lines of a picture are scanned first and transmitted as field 1 and then the odd lines are scanned and transmitted as field 2. The two fields make up a frame and the field and scanning rate are therefore twice the frame rate. (See SMPTE 170 M.)

3.1.19 luminance: the portion of a video signal which carries the brightness and contrast information.

3.1.20 Object: a collection of data of like kind (e.g., uncompressed video, audio, ancillary data, or compressed video).

3.1.21 producer: see Video Source.

3.1.22 Progressive: a system for video whereby all the lines, odd and even, of a picture are scanned and transmitted as one field with the odd and even lines in vertical progression. The single field makes up a frame and the scanning rate is therefore equal to the frame rate. (See SMPTE 293 M.)

3.1.23 Real Time: information delivery at a data rate sufficient to match the human perception of time.

3.1.24 server: a grantor of action on a Stream (e.g., Record or Play).

3.1.25 Segmented: a scanning system for video whereby all the lines, odd and even, of a picture are scanned progressively, and then the even lines are transmitted as field 1, followed by the transmission of the odd lines as field 2, permitting a progressive scanned picture to be displayed by an interlaced monitor. The two fields make up a frame, and the frame rate is equal to the scanning rate. (See SMPTE RP 211.)

3.1.26 Stream: a mechanism used to control the transfer of Content.

3.1.27 Stream mode: from the point of view of the client, a state where Content is to flow from the client to the server (Record mode) or from the server to the client (Play mode), but not in both directions in the same stream.

3.1.28 Stream state: the current operational state (e.g., Play, Pause, Record) of a single Stream. Two Streams may be in different states.

3.1.29 Video frame: a video Object or Objects. A video frame is a complete scan which may be progressive, interlaced, or segmented.

3.1.30 Video Program: Audio/Video Objects produced from systems employing audio/video compression.

3.1.31 Video Sink: the receiver of video Content in a Stream.

3.1.32 Video Source: the sender of video Content in a Stream.

3.1.33 YC_BC_R: a signal designation developed as part of Recommendation ITU-R BT.601-5. It is a scaled and offset version of the YUV color space.

3.1.34 YIQ: The YIQ system is the color primary system adopted by National Television System Committee (NTSC) for color TV broadcasting. The YIQ color solid is made by a linear transformation of the RGB cube. Its purpose is to exploit certain characteristics of the human eye to maximize the utilization of a fixed bandwidth. The human visual system is more sensitive to changes in luminance than to changes in hue or saturation, and thus a wider bandwidth should be dedicated to luminance than to color information. Y is similar to perceived luminance, I and Q carry color information and some luminance information. The Y signal usually has 4,2 MHz bandwidth in a 525 line system. Originally, the I and Q had different bandwidths (1,5 MHz and 0,6 MHz), but now they commonly have the same bandwidth of 1 MHz.

3.1.35 YUV: A color encoding scheme in which luminance and chrominance information are separate. In the YUV color space, black-and-white information (Y) is combined with color information (U and V) in such a way that a black-and-white receiver can display normal black-and-white images. Color receivers combine the additional color information with the black-and-white to produce color images. The Y, U, and V components are derived from gamma corrected R, G and B signals by multiplication with an encoding matrix.

3.2 Abbreviations and acronyms

Abbreviations and acronyms applicable to this standard are listed. Definitions of several of these items are included in 3.1.

DIF	Digital Interface
DV	Digital Video
DVB	Digital Video Broadcast
DVCPRO	DV-based (Panasonic trademark for a 1/4 inch tape recording format)
FC-FS	Fibre Channel Framing and Signaling
FCP	Fibre Channel Protocol
FC-PH	Fibre Channel Physical and Signaling Interface
FEC	Forward Error Correction
FHCP	Frame Header Control Protocol
HIPPI	High Performance Parallel Interface
JPEG	Joint Photographic Experts Group
lsb	Least significant bit
LSB	Least significant byte
MPEG	Moving Picture Experts Group
MPEG-TS	Moving Picture Experts Group - Transport Stream
msb	Most significant bit
MSB	Most significant byte
NTSC	National Television Systems Committee
OUI	Organizational Unit Identifier
PAL	Phase Alternate Line
QoS	Quality of Service
RT	Real Time
SCMA	Simple Content Movement Architecture
SCSI	Small Computer System Interface
SPDV	Simple Parametric Digital Video
SSRB	Simple Streaming Request/Response Block
ULP	Upper Level Protocol

3.3 Keywords

3.3.1 Keywords overview

Several keywords are used to differentiate between different levels of requirements and optionality.

3.3.2 expected: used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

3.3.3 mandatory: indicates items required to be implemented as defined by this standard.

3.3.4 may: a keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

3.3.5 may not: a keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

3.3.6 obsolete: indicates items that were defined in prior FCI standards, but have been removed from this standard.

3.3.7 optional: describes features that are not required to be implemented by this standard. However, if any optional feature defined by the standard is implemented, it shall be implemented as defined by this standard.

3.3.8 reserved: a keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as error.

3.3.9 shall: indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other standard conformant products.

3.3.10 should: indicates flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “it is recommended.”

3.3.11 vendor-specific: items (e.g., a bit, field, code value, etc.) that are not defined by this standard and may be vendor defined.

3.4 Conventions

Lower case is used for words having the normal English meaning. Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 3 or in the text where they first appear.

An alphanumeric list (e.g., a, b, c or A, B, C) of items indicate the items in the list are unordered.

A numeric list (e.g., 1, 2, 3) of items indicate the items in the list are ordered (i.e., item 1 shall occur or complete before item 2).

In the event of conflicting information the precedence for requirements defined in this standard is:

1. text,
2. tables, then
3. figures.

Not all tables or figures are fully described in text. Tables show data format and values.

The ISO convention of numbering is used (i.e., the thousands and higher multiples are separated by a space and a comma is used as the decimal point as in 65 536 or 0,5).

The additional conventions are:

a) The names of abbreviations, commands, and acronyms used as signal names are in all uppercase (e.g., IDENTIFY DEVICE);

b) Fields containing only one bit are referred to as the "NAME" bit instead of the "NAME" field;

c) Field names are in SMALL CAPS to distinguish them from normal English;

d) Numbers that are not immediately followed by lower-case b or h are decimal values;

e) Numbers “0-1” followed by lower-case b (‘xx’b and ‘xxxx xxxx’b) are binary values;

f) Numbers and letters “A-F” immediately followed by lower-case h (xxh) are hexadecimal values;

g) The most significant bit of a binary quantity is shown on the left side and represents the highest algebraic value position in the quantity;

h) If a field is specified as not meaningful or it is to be ignored, the entity that receives the field shall not check that field.

4 Structure and concepts

4.1 Relationship with FC-FS

FC-FS describes a high-performance transport vehicle for Upper Level Protocols (ULPs). This document, FC-AV, describes standard methods, applications, and services that enable audio, video, ancillary data and control streams to use FC-FS as a transport service.

Fibre Channel is organized in a layered structure as illustrated in figure 1. The highlighted portion of figure 1 is the subject of this document.

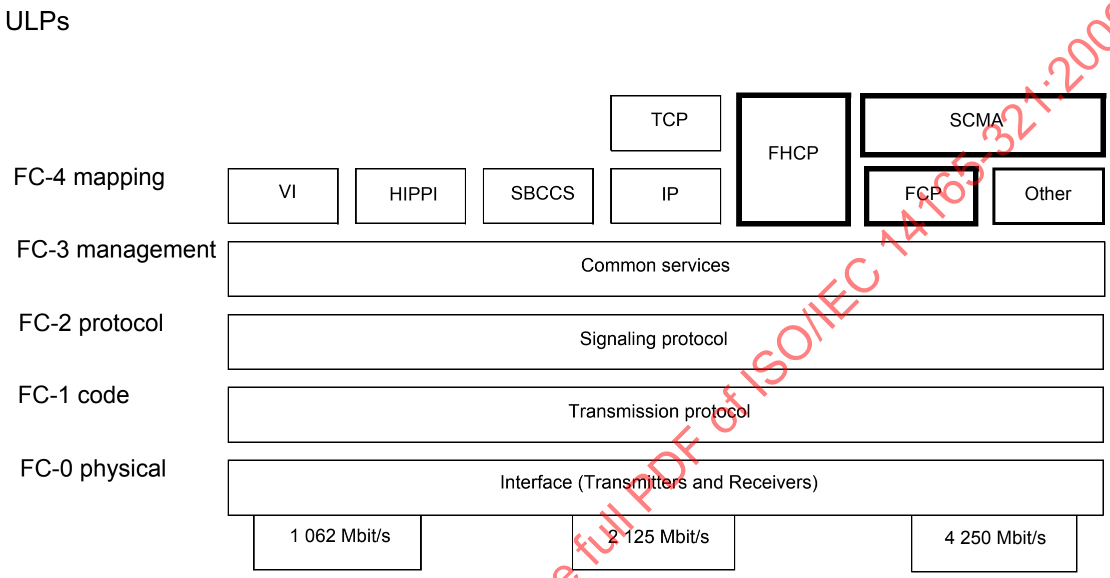


Figure 1 – FC-AV model

4.2 FC-AV Container system

The FC-AV Container system (see clause 5) is designed to optimize the transport of Video data over Fibre Channel. An essential element of the optimization is the grouping of video, audio and ancillary data into relatively large data sets to be transported as a unit. Using 4:2:2 uncompressed video as an example, the data set is on the order of one megabyte of data per Video frame. Fibre Channel is well suited to block transmissions of such a large data set, and is much more efficient when data is grouped into large blocks rather than small Packets.

4.3 Simple Content Movement

4.3.1 Overview

4.3.1.1 General

The Simple Content Movement Architecture (SCMA) provides the basic function to support most post-production and general editing applications, but it is not intended to support general broadcast environments.

The Simple Content Movement Architecture consists of two layers: the Content Movement layer and the Content Transport layer (see figure 2).

NOTE – In this document, Content is defined as Audio frames, Video frames, AV frames, or Containers.

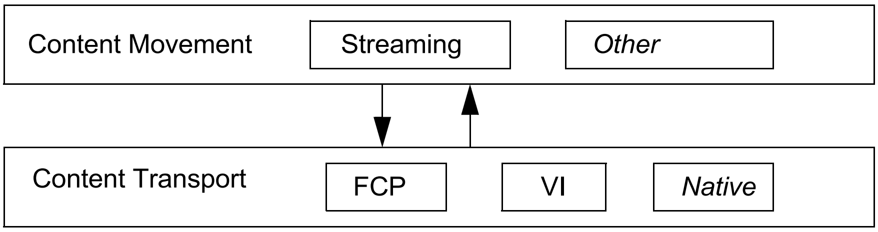


Figure 2 – Simple Content Movement Architecture

4.3.1.2 Content Movement layer

The first layer, known as the Content Movement layer, defines a simple full duplex Streaming protocol for the interaction and communication between the client and the server without respect to the underlying transport.

The intent of SCMA is to provide a simple mechanism to transfer Content between two nodes.

Only a Simple Streaming protocol is presently defined for SCMA (see clause 8).

4.3.1.3 Content Transport layer

The second layer, known as the Content Transport layer, allows the Content Movement layer to be mapped to one or more transport protocols and physical systems.

Any defined protocol used to support the Content Movement layer shall be capable of supporting full duplex activity.

Most of the Content Movement protocol is in the form of request-reply actions. Only the Content transfer activity has any asynchronous elements. Any defined protocol used to support the Content Movement layer shall be capable of supporting this asynchronous activity.

The SCSI-3 FCP protocol (see clause 9) is the only defined transfer protocol at this time. However, SCMA allows for other protocol mappings.

4.4 Frame Header Control Protocol

Frame Header Control Protocol (FHCP) is a low-overhead mechanism for the transport of FC-AV Containers (see clause 7). FHCP utilizes FC-2 level properties and as a result, is optimized toward low latency, efficient data transport. While Simple Streaming Protocol provides operator control such as *Start*, *Stop, Play, Pause, Resume*, nodes utilizing FHCP are considered to be constantly in the *Play* mode.

5 FC-AV Containers

5.1 Overview

The FC-AV Container is designed to optimize the transport of Audio/Video data over Fibre Channel. An essential element of this optimization is the grouping of video, audio and ancillary data into relatively large data sets to be transported as a unit. Using 4:2:2 uncompressed video as an example, the data set is on the order of one megabyte per Video frame. Fibre Channel is well suited to block transmission of such a large data set, and is much more efficient when data is grouped in large blocks rather than small Packets. Data transported in a Container is segregated into Objects, each representing a data type.

Fibre Channel implementations use sophisticated chip sets to “Exchange” data. At the lowest level, the Exchange is composed of FC frames limited to 2112 bytes of payload. In practice, these FC frames are not visible to any part of the system above the interface chip(s). The lowest level that is visible is referred to as a Sequence. A Sequence is an element of an Exchange based on the particular protocol used. The Container proposed in this clause maps to the Fibre Channel Sequence.

Clause 5 does not address higher level protocols required to set up Exchanges of Sequences, but only the packaging of the data into a single Sequence. Mechanisms are provided to insure the Containers can be sent in the correct order, and provide enough information to disseminate the data by type (Video, Audio and Ancillary data) at the lowest possible level.

An important aspect of the Container definition is the ability to utilize the Container Header as a mechanism for directing Objects composed of elemental data (such as video samples or audio samples) directly to decoders with a minimum of software overhead. This direction is designed to happen at the lowest possible level for high efficiency.

More sophisticated applications transporting streams with multiple elements, and metadata may require the intervention of a higher protocol layer.

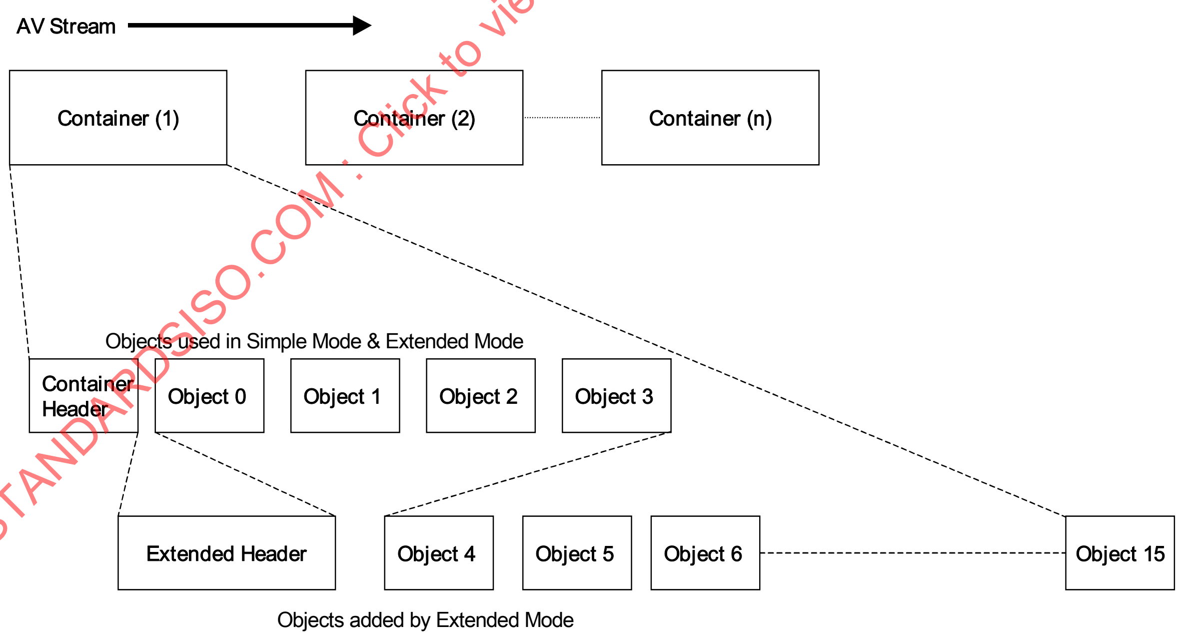


Figure 3 – The FC-AV Container system

5.2 The FC-AV Container system

5.2.1 Overview

A Container system (see figure 3) is defined for transporting video or graphics with associated audio and ancillary data. Each Container is intended to hold a frame (optionally two fields) of Video, Audio and Ancillary data. Each Container consists of a Container Header and Objects. Objects are classified by Type and further defined by an Index. Each data type is held within an object. Simple mode supports four objects of somewhat restricted types. Extended Header mode supports up to sixteen objects. A single Container maps exactly into one single Fibre Channel Sequence.

5.2.2 The FC-AV Container

Within each Container a number of objects exist containing video, audio and metadata elements. A fixed length Container Header precedes data Objects. The Container Header is designed to have enough information to allow direct processing of the data in the objects without having to “crack” the individual objects for additional information. This is intended to allow hardware processing at the Container level if desired. An optional Extended Header directly follows the fixed length header for the Extended Header mode. The first data object directly follows the end of the Container Header or Extended Header (if present). Specific Container types are defined in the Object classification system (see 5.6).

Two basic modes of operation are defined: Simple mode and Extended Header mode.

Simple mode is targeted at applications desiring to send a stream of video and/or audio with minimum overhead. It has attributes designed to be easy to implement in hardware. The Container Header addresses exactly four Objects. The Container Header is of fixed length, with Object descriptions in sequence and restricted to certain data types.

Extended Header mode provides support for more complex streams composed of a number of elements requiring additional features not included in the Simple mode. Specifically, the Extended Header mode supports the use of a variable number of Objects from 5 to 16 (Object 0 to Object 15). Physically, the Extended Header follows the Container Header, and shares the same structure. This allows the combination of the Container Header and the Extended Header to be viewed as a single header of variable length.

5.2.3 FC-AV concepts for Containers in Simple mode

Following is a list of basic concepts describing Containers in Simple mode:

1) The Container encapsulates video, audio and graphic data. There are exactly four Objects in Simple mode.
2) The Simple mode Container has restricted data types for each of the four Objects. Valid Objects and Types are:

Object 0 - Restricted to Ancillary data types	(Object Type number 5xh)
Object 1 - Restricted to Audio data types	(Object Type number 4xh)
Object 2 - Restricted to Video data types	(Object Type number 1xh | 2xh)
Object 3 - Restricted to Video data types	(Object Type number 1xh | 2xh)

NOTE – If any Object is not used in a Container, the Object size is set to zero.

3) The Container Header is in a fixed format in Simple mode, 22 words in length.

5.3 Container Header structure

The organization of the header is fixed to encourage hardware decoding of header information. Each word maps to a Fibre Channel transmission word (32 bits). To facilitate use in high performance computers, the header is organized to parse well into 64-bit words. All other references to "words" in this clause refer to 32-bit FC transmission words.

The Container Header (in Simple mode) is shown in table 1. The Container Header is a fixed length of 22 four-byte words. The first 6 words of the header apply to the entire Container. The remaining words are divided into identically structured sections, one for each data object. The Simple mode header uses only Objects 0-3. Extended Header mode may add up to 12 more identically structured sections to describe Objects 4-15 (see table 4).

Table 1 – Container Header (Simple Mode)

Word	Identifier	Byte 0	Byte 1	Byte 2	Byte 3
0	Container count	MSB	-container count-		LSB
1	Clip ID	MSB	-clip ID-		LSB
2-3	Container Time Stamp	MSB	-Time Stamp-		
					LSB
4	Transmission Type	Video Fr. Rate	Trans. Rate	Reserved	Reserved
5	Container Type	Mode	# of Objects	Reserved	Sz of Ext Hdr
6	Object 0 Class	Type 5xh	Link pointer	Index (xxxxh)	
7	Object 0 Size	MSB	-Size (bytes)-		LSB
8	Object 0 Offset	MSB	-Offset (bytes)-		LSB
9	Object 0 O.T. Defined	O.T. Defined			
10	Object 1 Class	Type 4xh	Link pointer	Index (xxxxh)	
11	Object 1 Size	MSB	-Size (bytes)-		LSB
12	Object 1 Offset	MSB	-Offset (bytes)-		LSB
13	Object 1 O.T. Defined	O.T. Defined			
14	Object 2 Class	Type 1xh | 2xh	Link pointer	Index (xxxxh)	
15	Object 2 Size	MSB	-Size (bytes)-		LSB
16	Object 2 Offset	MSB	-Offset (bytes)-		LSB
17	Object 2 O.T. Defined	O.T. Defined			
18	Object 3 Class	Type 1xh | 2xh	Link pointer	Index (xxxxh)	
19	Object 3 Size	MSB	-Size (bytes)-		LSB
20	Object 3 Offset	MSB	-Offset (bytes)-		LSB
21	Object 3 O.T. Defined	O.T. Defined			

5.4 Description of Container Header contents

5.4.1 Container Information Block

The Container Information Block consists of words 0 to 5 and is common to all Containers.

Word 0 - Container Count: A 32-bit unsigned integer used to identify the position of a Container in a series of Containers of the same Clip ID. The Container Count shall be incremented by one for each Container in the series.

Word 1 - Clip ID: A 32-bit unsigned integer used for identification of a group of Containers, e.g., a video clip.

Word 2-3 - Container Time Stamp: A 64-bit word used to store an FC-AV time stamp for transmission purposes. This time stamp represents the Container time. There may be other time stamps and time codes associated with each Object.

The time stamp format is described as the FC-PH Expiration Time which is a 64-bit fixed-point number valued in seconds. The integer part of the number is the most significant 32 bits and the fractional part is the least significant 32 bits. The maximum integer is therefore 2E32-1 and there are 32 bits for the fraction of a second. The timer start time is 1 January 1900, 0 Hours UTC.

Word 4, Byte 0 - Video Fr. Rate: The encoding of the Video frame rate field is shown in table 2.

Table 2 – Video frame rate – encoding

Frame rate per second	Code
Null	00h
15	01h
20	02h
24	03h
24*1000/1001	83h
24 (Segmented frames)	23h
24*1000/1001(Segmented frames)	A3h
25 (PAL)	44h
30	45h
30*1000/1001 (29.97 NTSC)	C5h
50	06h
60	07h
60*1000/1001 (59.94 NTSC)	87h

Word 4, Byte 1 - Trans. Rate: An 8-bit signed integer Transmission Rate. Positive values represent n * Video frame rate. Negative values represent Video frame rate / n.

This allows the protocol processing to operate the same for Real-Time and faster than Real-Time transmission of Video data not compressed temporally (e.g., JPEG and DV).

Word 4, Byte 2-3: reserved

Word 5, Byte 0 - Mode: Simple or Extended mode:

– Bit 7 = 0, Simple mode
– Bit 7 = 1, Extended Header mode
– Bits 6:0 = reserved

Word 5, Byte 1 - # of Objects: The integer number of Objects in the Container. In Simple mode the # of Objects is 04h.

Word 5, Byte 2: reserved

Word 5, Byte 3 - Sz of Ext Hdr: The Size of Extended Header field shall be zero in Simple mode and non-zero in Extended Header mode. The value in bytes is equal to the total number of Objects minus 4, multiplied by 16.

5.4.2 Object Information Block

Each Object Information Block in the Container Header consists of a 4-word field (see table 1) that describes relevant information about that Object. The Object Information Block is shown in table 3.

Table 3 – Object Information Block

Item	Size (Bytes)
Object n Type	1
Object n Link pointer	1
Object n Index	2
Object n Size	4
Object n Offset	4
Object n Object Class defined	4

Object n Type: (see 5.6)

Object n Link pointer: An unsigned byte used to link an object with another object within the same Container. The value is the Object number of the associated object.

Object n Index : (see 5.6)

Object n Size: Object size in bytes.

Object n Offset: Object offset in bytes from the beginning of the Container.

Object n Object Class defined (O.T. defined): (see 5.6)

5.5 Extended Header description

The FC-AV Container system defines an Extended Header to address requirements of applications not directly supported by the simple mode. Provisions include the support of a larger and variable number of Objects. The Extended Header is shown in table 4.

The Extended Header is a logical extension of the Container Header. It is physically located directly after the Container Header and before Object 0. Object descriptions follow the same format as the Object descriptions within the Container Header. There are no restrictions as to the Object Class for any of the Objects addressed by the Extended Header. The length of the Extended Header is dependent on the number of Objects in the Container. The Extended Header length equals 4 words for each Object beyond Object 3. The maximum number of Objects is 16, thus the maximum length is 4 words * (16-4) = 48 words.

Table 4 – Extended Container Header

Word	Identifier	Byte 0	Byte 1	Byte 2	Byte 3
22	Object 4 Class	Type xxh	Link pointer	Index (xxxxh)	
23	Object 4 Size	MSB	-Size (bytes)-		LSB
24	Object 4 Offset	MSB	-Offset (bytes)-		LSB
25	Object 4 O.T. defined	O.T. defined			
26	Object 5 Class	Type xxh	Link pointer	Index (xxxxh)	
27	Object 5 Size	MSB	-Size (bytes)-		LSB
28	Object 5 Offset	MSB	-Offset (bytes)-		LSB
29	Object 5 O.T. defined	O.T. defined			
	...				
n*4+6	Object n Class	Type xxh	Link pointer	Index (xxxxh)	
n*4+7	Object n Size	MSB	-Size (bytes)-		LSB
n*4+8	Object n Offset	MSB	-Offset (bytes)-		LSB
n*4+9	Object n O.T. defined	O.T. defined			

5.6 Object classification system

5.6.1 Object classification overview

The Container system provides a mechanism for defining Object content using an Object Class and Index classification system (see table 5). The Object classification is represented in the Container Header by the Type and Index fields of the Object Class word for each Object. The Type byte is a coarse classification of the Object. The Index bytes provide a refined classification of the Object. This classification system, using the Type and Index fields, provides a quick, shorthand method for FC-AV implementations to provide or determine Object content and format without requiring transmission of complex descriptions or negotiation.

The Object Classification system includes five major classifications of Types:

1) Null - unused Object entry, size is always zero;
2) Assigned - defined in this standard;
3) Reserved - for future standard assignment;
4) Negotiated - for use by a higher level protocol to dynamically assign data types; and,
5) Vendor specific - for open use and guaranteed not to be assigned in the future.

Each Type byte represents a table of data formats. The Index bytes provide a key into that table. These tables then provide information to define the Object's data format.

The Type tables contain condensed information that can be used to quickly determine the definition or defining documents of an object. The Type tables contain an Index or Index ranges which follow the same five major classifications as are defined for Type (Null, Assigned, Reserved, Negotiated, and Vendor specific). The Assigned classification Indices in each table provide references to "standard" formats for Audio/Video Objects such as those documented by SMPTE or other organizations. Indices that are Assigned or Reserved are listed in annex B. For example, the Video Uncompressed Type (Type 10h) Assigned Indices are provided in B.1.

Table 5 – Object Class Hierarchy

Object Class Type		Object Type Index	
00h	Null	Not Applicable	
10h	Video - uncompressed	0000h	Null
		0001-DFFFh	Assigned or Reserved (see B.1)
		E000-EFFFh	Negotiated
		F000-FFFFh	Vendor Specific
11h	Video - compressed	0000h	Null
		0001-DFFFh	Assigned or Reserved (see B.2)
		E000-EFFFh	Negotiated
		F000-FFFFh	Vendor Specific
12-1Fh	Video - reserved	Not Applicable	
20h	Video Program	0000h	Null
		0001-DFFFh	Assigned or Reserved (see B.3)
		E000-EFFFh	Negotiated
		F000-FFFFh	Vendor Specific
21-2Fh	Video Program - reserved	Not Applicable	
30h	Graphics	0000h	Null
		0001-DFFFh	Assigned or Reserved (see B.4)
		E000-EFFFh	Negotiated
		F000-FFFFh	Vendor Specific
31-3Fh	Graphics - reserved	Not Applicable	
40h	Audio - uncompressed	0000h	Null
		0001-DFFFh	Assigned or Reserved (see B.5)
		E000-EFFFh	Negotiated
		F000-FFFFh	Vendor Specific
41h	Audio - compressed	0000h	Null
		0001-DFFFh	Assigned or Reserved (see B.6)
		E000-EFFFh	Negotiated
		F000-FFFFh	Vendor Specific
42-4Fh	Audio - reserved	Not Applicable	
50-5Fh	Ancillary Data	Not Applicable	
60h	Full Stream - structures	0000h	Null
		0001-DFFFh	Assigned or Reserved (see B.7)
		E000-EFFFh	Negotiated
		F000-FFFFh	Vendor Specific
61-6Fh	Full Stream - reserved	Not Applicable	
70-DFh	Reserved	Not Applicable	
E0-EFh	Negotiated	Not Applicable	
F0-FFh	Vendor Specific	Not Applicable	

5.6.2 Object Type byte

5.6.2.1 General

The Object Type byte (Object n Class, byte 0) is an unsigned value. The values *assigned* are shown in table 5 and described in the following clauses. Values listed as *reserved* are for future assignment by this standard. Values labeled *vendor specific* are intended for proprietary use by a specific vendor.

Requirements for assignment of Object Type bytes are as follows:

1) The scope of the Object Type byte is chosen to cover data types likely to be decoded by a dedicated device.
2) A definition for the function (if any) of Object Type Defined Word of the 4 word Object Information Block (see 5.4.2).
3) Negotiated Object Types are for use by a higher level protocol.

5.6.2.2 Null Object

An encoded value of 00h in the Object Type field designates no valid Object Type present.

5.6.2.3 Video, uncompressed

An Object Type of 10h designates Video objects produced from systems not employing video compression techniques. The assigned indices are provided in B.1.

5.6.2.4 Video, compressed

An Object Type of 11h designates Video objects produced from systems employing video compression techniques. The assigned indices are provided in B.2.

5.6.2.5 Reserved for future Video

Reserved for assignment to future Video Object Types. The encoded value range is shown in table 5.

5.6.2.6 Compressed FC-AV stream (Video Program)

An Object Type of 20h designates Audio/Video objects produced from systems employing audio/video compression techniques.The assigned indices are provided in B.3.

5.6.2.7 Reserved for Video Program (multiplexed stream)

Reserved for assignment to future Video Program Object Types. The encoded value range is shown in table 5.

5.6.2.8 Graphics

An Object Type of 30h designates graphic/static image objects. The assigned indices are provided in B.4.

5.6.2.9 Reserved for future Graphics

Reserved for assignment to future Graphics Object Types. The encoded value range is shown in table 5.

5.6.2.10 Audio, uncompressed

An Object Type of 40h designates Audio objects produced from systems not employing audio compression techniques. The assigned indices are provided in B.5.

5.6.2.11 Audio, compressed

An Object Type of 41h designates Audio objects produced from systems employing audio compression techniques. The assigned indices are provided in B.6.

5.6.2.12 Reserved for future Audio

Reserved for assignment to future Audio Object Types. The encoded value range is shown in table 5.

5.6.2.13 Ancillary data

An Object Type of 50-5Fh designates ancillary data objects (e.g., parametric headers, metadata, and inband control data). Indices are not applicable to ancillary data objects.

5.6.2.14 Full Stream - structures

An Object Type of 60h designates Audio/Video objects including synchronization signals and formats. The assigned indices are provided in B.7.

5.6.2.15 Reserved for future Full Stream - structures

Reserved for assignment to future Full Stream - structures Object Types. The encoded value range is shown in table 5.

5.6.2.16 Reserved

The range of encoded values reserved for assignment to future Object Types. The encoded value range is shown in table 5.

5.6.2.17 Negotiated Object Types

Range of encoded values reserved for Object Types that can be established between vendors for specific implementations. These may or may not be adopted as future standard Object Types. The encoded value range is shown in table 5.

5.6.2.18 Vendor Specific Object Types

The Vendor Specific Types are intended to be used at the implementer's discretion. These types do not necessarily have a public Object description and may not even have a unique Object description. A vendor may use them to differentiate private or proprietary Objects within a vendor's own product line or with cooperating partners. The Vendor Specific code values shall never become an Assigned Object classification. An Object, which has been in use as a Vendor Specific type, could later be given a different code value within the Assigned Type code range. Vendor Specific Types are not registered to a specific vendor. The only mechanism to uniquely identify that two vendors have chosen to use identical Vendor Specific code values is through use of the Object Type Defined Word requirement for Vendor Specific classifications (see 5.6.4.2). The encoded value range is shown in table 5.

5.6.3 Object Index bytes

5.6.3.1 General

The Object Index bytes (Object n Class, bytes 2 and 3) are defined as a 2-byte unsigned value representing an index to a table for each Object Type byte. For the case of a null Object, the bytes are set to 0000h. Tables of assigned indices are provided in annex B.

Requirements for assignment of Object Index values are as follows:

1) A clear description of the data format standard to be represented by this index. Normally this would be a recognized standard number. When available this may be described by SMPTE 298M.
2) Sufficient information to describe a single data type. For example, SMPTE 259M would not be sufficient as many different types are described. A SMPTE 298M proposed type identifier for a specific SMPTE 259M type would be sufficient providing the definition of the type is readily available. An example is SMPTE 259M type A: 270 Mbit/s, 720x486 pixels, 4:3 aspect, 59.97 field rate, 2:1 interlace.
3) The Object size range is optional, but when it is present, it shall be in bytes.
4) Object index values are a 16-bit unsigned integer. The values Fxxxh are vendor specific values and shall not be reassigned. All other values are either null (i.e., 00h), assigned, negotiated (i.e., E000h to EFFFh), or reserved for future standards.

5.6.3.2 Vendor Specific Object Indices

Within Vendor Specific Object Types there can be Vendor Specific Object Indices. If Vendor Specific Object Indices are used, the values are to comply with the rules for Object Type Defined word for Vendor Specific Types (see 5.6.4.2).

5.6.4 Object Type Defined word

5.6.4.1 General

The Object Type Defined word (Object n O.T. Defined) is a 4-byte value which contains optional user-defined parameters associated with the Object Type.

5.6.4.2 Object Type Defined word definition for Vendor Specific classifications

If an Object is using a Vendor Specific Object Type or a Vendor Specific Object Index the Object Type Defined word shall be defined as one of the two following formats:

1) Organization Unique Identifier (OUI) Object Type Defined Word format: the Organizational Unique Identifier Object Type Defined Word format provides a method within the object header to uniquely identify an Object within the Vendor Specific classifications.
   - The MSB, bit 31 is set to a one.
   - 7 Bits, 30 through 24 may be defined by the vendor
   - 24 bits, 23 through 0 are the IEEE OUI Company ID

Table 6 – Organization Unique Identifier Object Type Defined Word format

<table>
<tr><th colspan="2">31 24</th><th>23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>1</td><td>7 bit Vendor defined value</td><td colspan="3">IEEE OUI Company ID</td></tr>
</table>

2) Non Organization Unique Identifier Object Type Defined Word format: the Non Organizational Unique Identifier Object Type Defined Word format provides a format that gives the vendor value control of 31 of the 32 bits of the Object Type Defined Word.

– The MSB, bit 31 is set to a zero.

– 7 Bits, 30 through 24 may be defined by the vendor.

Table 7 – Organization Unique Identifier Object Type Defined Word format

31 24	23 16	15 8	7 0
0	31 bit Vendor defined value		

5.6.5 Object Data Packing

Data Packing is the placement of the data (bytes and bits) within the transmission words. Information on Data Packing is contained in annex G.

6 Compressed FC-AV stream transmission

6.1 Overview

6.1.1 General

The compressed FC-AV stream is multiplexed in a Video Program Object. Object Class 20h of the Container and its format is based on the sub-Container format as shown in figure 4. The sub-Container is composed of a Stream Header and single or multiple Compressed Data Stream (CDS) Packets.

Stream Header	CDS Packet	CDS Packet	CDS Packet	...	CDS Packet

Figure 4 – Component AV Stream Object format

6.1.2 Stream Header

The Stream Header is composed of a 32-bit Stream descriptor word, a 32-bit Stream Time Stamp and a 32-bit CDS Packet length word as shown in figure 5.

	Bits: 31 … 0
Word 0	**Stream descriptor**
Word 1	**Stream Time Stamp**
Word 2	**CDS Packet length**

Figure 5 – Stream Header format

6.1.3 Stream descriptor

6.1.3.1 Overview

The Stream descriptor is composed of a 16-bit Packet type and a 16-bit Stream type as shown in figure 6.

31 … 16	15 … 0
Packet type	**Stream type**

Figure 6 – Stream descriptor

6.1.3.2 Packet type

The Packet type is a 16-bit half-word and indicates the CDS Packet structure. The Packet type is defined below.

- Bit 31: The existence indication bit of the Packet Time Stamp word (described in 6.1.7).

  0 - indicates that the Packet Time Stamp word exists.

  1 - indicates that the Packet Time Stamp word does not exist.

- Bit 30: The existence indication bit of the byte count word (described in 6.1.8).

0 - indicates that the byte count word exists.

1 - indicates that the byte count word does not exist.

- Bit 29: The existence indication bit of the FEC (Forward Error Correction) word (described in 6.1.10)

  0 - indicates that the FEC word exists.

  1 - indicates that the FEC word does not exist.

- Bit 28: The indication bit that indicates multiple Packets or single Packet.

  0 - indicates that the sub-Container is composed of a single CDS Packet.

  1 - indicates that the sub-Container is composed of multiple CDS Packets.

- Bit 27: The indication bit that the Stream Time Stamp word is valid or invalid.

  0 - indicates that the Stream Time Stamp word is valid.

  1 - indicates that the Stream Time Stamp word is invalid.

- Bit 26: The indication bit that indicates fixed Packet size or variable Packet size.

  0 - indicates that the CDS Packet size is fixed.

  1 - indicates that the CDS Packet size is variable and CDS Packet length word is invalid.

- Bit 25, Bit 24: reserved for future purposes (default value is '00'b).

- Bit 23 through Bit 16: indicates the byte length of the FEC.

6.1.3.3 Stream type

The Stream type is a 16-bit half-word. The definition of the stream type depends on the compressed data stream format. Instances of two compressed stream types are described in 6.4.

6.1.4 Stream Time Stamp

The Stream Time Stamp is composed of a 32-bit word that indicates the point of time at which the compressed AV stream Object is multiplexed into the Container. The Stream Time Stamp is used to recover the relative stream Object timing from end to end and is valid when Bit 27 of the Packet type is set to 0. The synchronization scheme using this word is described in 6.2.

6.1.5 CDS Packet length

The CDS Packet length is composed of a 32-bit word that indicates the byte length of the CDS Packet multiplexed immediately after the Stream Header. The CDS Packet length word is valid when Bit 26 of the Packet type is set to 0. The CDS Packet length is valid only when the CDS Packet size is fixed.

6.1.6 CDS Packet format

The CDS Packet format is composed of a 32-bit Packet Time Stamp word, a 32-bit byte count word, an *n* byte CDS Packet payload and an *m* byte FEC (see figure 7).

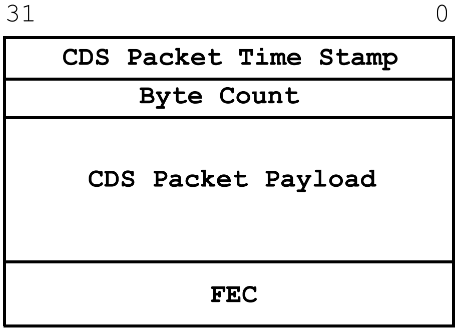


Figure 7 – CDS Packet Format

6.1.7 CDS Packet Time Stamp

The CDS Packet Time Stamp is a 32-bit word and exists when Bit 31 of the Packet type is set to 0. The CDS Packet Time Stamp is used to recover the relative CDS Packet timing from end to end. The reference clock for the Packet Time Stamp is the same frequency as for the Object Time Stamp in the Container Header. The synchronization scheme using this word is described in 6.2.

6.1.8 Byte count

The byte count is a 32-bit word that exists when Bit 30 of the Packet type is set to 0. The byte count indicates the byte length of the variable CDS Packet payload size or the valid byte length in the fixed CDS Packet payload. If the Packet Time Stamp word does not exist, the byte count word is located at the head word of the CDS Packet.

6.1.9 CDS Packet payload

The CDS Packet payload format depends on the compressed data stream and is described in 6.4.

6.1.10 Forward Error Correction

The Forward Error Correction (FEC) is an *m* byte code and exists when Bit 29 of the Packet type is set to 0. The FEC applies to the first byte through the last byte of the CDS Packet. The error management scheme is described in 6.3.

6.2 Synchronization scheme

6.2.1 Overview

The synchronization scheme is classified according to the compression scheme and divided into two groups, MPEG group and frame/field compression group.

6.2.2 MPEG

In the MPEG system, the relative timing of MPEG Stream Packets (MPEG-TS Packets or MPEG-PS Packets) should be maintained from end to end, because time stamp information (PTS/DTS) is embedded in the MPEG Stream Packets.

In the sender that has a master clock (e.g., 27 MHz), each CDS Packet Time Stamp (CTS) and each Stream Time Stamp (STS) is counted up by the master clock. Each CTS is multiplexed in each CDS Packet respectively at the input timing of CDS Packet formulator. On the other hand, each STS is multiplexed in each Stream Header at the input timing of the Object formulator. In the receiver, each STS is decoded in the receiver Object deformatter and relative Object timing should be recovered using the received STSs at the output point of the Object deformatter. Furthermore, the relative Packet timing should be recovered using the received CTSs at the output point of CDS Packet deformatter.

If the relative timing of Objects within the Container does not need to be recovered, the Stream Time Stamp is not necessary.

6.2.3 Frame/Field compression

In the frame/field compression system such as DVCPRO or JPEG, if only the frame/field frequency is recovered accurately, it is not necessary to recover the relative CDS Packet timing accurately except to recover the CDS Packets within the frame/field in which they were generated in the sender. Therefore, STS or CTS are not always necessary.

6.3 Error management scheme

6.3.1 Overview

The error management scheme should be applied flexibly according to the application of the stream transport. But, in the stream transport application, it is necessary for the error management scheme not to disturb the real-time transport.

The error management scheme is classified into the following three modes:

1) Process policy without FEC
2) Process policy with FEC
3) Discard policy with immediate retransmission

6.3.2 Process policy without FEC

Process policy without FEC mode is applicable for the video delivery system that does not require high reliability.

6.3.3 Process policy with FEC

Process policy with FEC mode is applicable for the studio system that requires high reliability and is useful when the bandwidth of the Fibre Channel link is not enough to retransmit the stream.

NOTE – Fibre Channel links have a BER of $\leq$ 10E-12. If the bit rate of a compressed stream is 27 Mbit/s, the mean failure time interval is about 10 hours.

6.3.4 Discard policy with immediate retransmission

Discard policy with immediate retransmission mode is applicable for the studio system that requires high reliability and is useful when the bandwidth of the Fibre Channel link is enough to retransmit the stream.

6.4 Instances of compressed AV streams over Fibre Channel

6.4.1 DV based compression stream

6.4.1.1 Overview

An instance of a DV based compression stream over Fibre Channel is described below. The data structure of DV based compression streams is classified into two groups, 25 Mbit/s structure and 50 Mbit/s structure. These are specified in SMPTE 314M (see reference []).

6.4.1.2 Stream Header

6.4.1.2.1 Stream descriptor

6.4.1.2.1.1 Packet type

- Bit 31 = 1: The Packet Time Stamp word does not exist.
- Bit 30 = 1: The byte count word does not exist.
- Bit 29 = 0 or 1: According to the error scheme which application of DV based stream transport adopts.
- Bit 28 = 1: The sub-Container is composed of multiple CDS Packets.
- Bit 27 = 0 or 1: According to the relation of other Objects in the Container.
- Bit 26 = 0: The CDS Packet size is fixed.
- Bit 25, Bit 24: Default value ’00’b
- Bit 23 through Bit 16: When the FEC word exists, the length of FEC shall have the value 04h (4 bytes). When the FEC word does not exist, the length of FEC shall have the value 00h (0 bytes).

6.4.1.2.1.2 Stream type

Bits 7-0 of the stream type is defined as the signal type. The signal type format is shown in figure 8.

7	6	5	4	3	2	1	0
Rsvd	DIF Structure			Reserved			

Figure 8 – Signal type format

- Bit 15 through Bit 8: reserved for future purposes. (Default value is ‘0000 0000’b)
- Bit 7: reserved for future purposes. (Default value is ‘0’b)
- Bit 6 through Bit 4: The indication bits of the DIF structure of the DV based compression stream.

  ’011’b - indicates that the DV based compression stream is a 25 Mbit/s structure.

  ’101’b - indicates that the DV based compression stream is a 50 Mbit/s structure.

Other values are reserved.

- Bit 3 through Bit 0: reserved for future purposes. (Default value is ‘0000’b)

6.4.1.2.2 CDS Packet length

When the FEC word exists, the Packet length shall have the value 000000A4h (164 bytes).

When the FEC word does not exist, the Packet length shall have the value 000000A0h (160 bytes).

6.4.1.3 CDS Packet format

6.4.1.3.1 Overview

According to the Packet type, the CDS Packet format of the DV based compression stream is shown in figure 9.

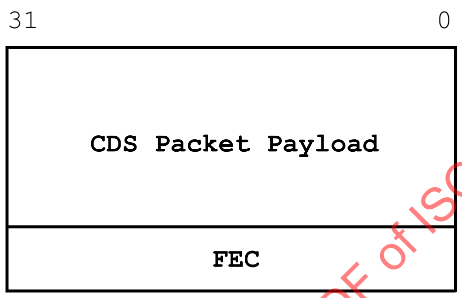


Figure 9 – CDS Packet Format

6.4.1.3.2 CDS Packet payload (SMPTE)

The CDS Packet payload format is shown figure 10. The DIF block ID (ID0-2) and DIF block data are specified in SMPTE 314M (see reference []). A CDS Packet includes two DIF block data. In 525/60 system, one compressed AV stream Object is composed of 750 CDS Packets (1500 DIF blocks) for 25 Mbit/s structure or 1500 CDS Packets (3000 DIF blocks) for 50 Mbit/s structure. In 625/50 system, one compressed AV stream Object is composed of 900 CDS Packets (1800 DIF blocks) for 25 Mbit/s structure or 1800 CDS Packets (3600 DIF blocks) for 50 Mbit/s structure.

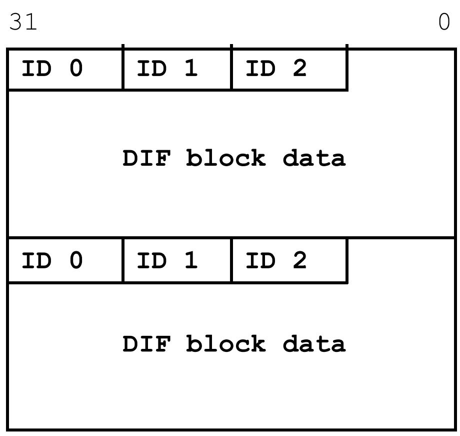


Figure 10 – CDS Packet Payload format (SMPTE)

6.4.1.3.3 FEC

The CDS Packet data is optionally protected by the Forward Error Correction (FEC). The FEC is a 4-byte code and is located immediately after the CDS Packet payload when bit 29 of the Packet type is set to 0. The FEC applies to the first byte through the last byte of the CDS Packet. A 4-byte Reed-Solomon error correction shall be used as the method of FEC. The error management scheme is described in 6.3.

6.4.2 MPEG-TS

6.4.2.1 Overview

An instance of MPEG-TS over Fibre Channel is described below. The specification of MPEG-TS is defined in reference [3] The MPEG-TS Packet is not based on the frame/field structure, but can be transported over the Container as a Video Object.

6.4.2.2 Stream Header

6.4.2.2.1 Stream descriptor

6.4.2.2.1.1 Packet type

- Bit 31 = 0: The Packet Time Stamp word exists.
- Bit 30 = 1: The byte count word does not exist.
- Bit 29 = 0 or 1: According to the error scheme which application of MPEG-TS transport adopts.
- Bit 28 = 1: The sub-Container is composed of multiple CDS Packets.
- Bit 27 = 0 or 1: According to the relation of other Objects in the Container.
- Bit 26 = 0: The CDS Packet size is fixed.
- Bit 25, Bit 24: Default value ’00’b
- Bit 23 through Bit 16: When the FEC word exists, the length of FEC shall have the value 04h (4 bytes). When the FEC word does not exist, the length of FEC shall have the value 00h (0 bytes).

6.4.2.2.1.2 Stream type

Bits 15-8 of the stream type is defined as the signal type. The signal type format is shown in figure 11.

15	14	13	12	11	10	9	8
TS Packet Structure			Reserved				

Figure 11 – Signal type format

- Bit 15 through Bit 13: the indication bits of MPEG-TS Packet structure as follows:
  – ’000’b - indicates that the payload of the CDS Packet is a 188 byte MPEG-TS Packet.
  – ’100’b - indicates that the payload of the CDS Packet is a 204 byte DVB Packet format containing a 188 byte MPEG-TS Packet and a 16 byte Reed-Solomon (RS) error correction code. The DVB format is specified in ETS 300 429.
  – ’101’b - indicates that the payload of the CDS Packet is a 204 byte Packet containing a 188 byte MPEG-TS Packet and a 16 byte zero code.

– Other values are reserved.

- Bit 12 through Bit 8: reserved for future purposes (default value is ’0000 0’b).
- Bit 7 through Bit 0: reserved for future purposes. (Default value is ’0000 0000’b).

6.4.2.2.2 CDS Packet length

When the FEC word exists, the Packet length shall have the value 000000C4h (196 Bytes) or the value 000000D4h (212 bytes).

When the FEC word does not exist, the Packet length shall have the value 000000C0h (192 Bytes) or the value 000000D0h (208 bytes).

6.4.2.3 CDS Packet format (MPEG-TS)

6.4.2.3.1 Overview

The CDS Packet format according to the Packet type of the MPEG-TS is shown in figure 12.

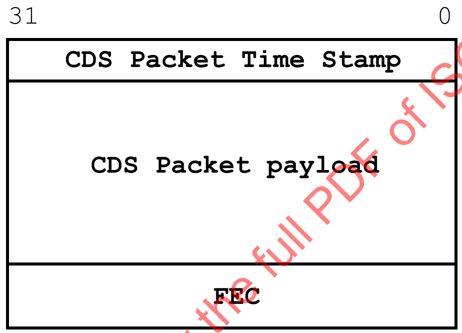


Figure 12 – CDS Packet format (MPEG-TS)

6.4.2.3.2 CDS Packet Time Stamp

CDS Packet Time Stamp is a 32-bit word. The reference clock frequency for the Packet Time Stamp is synchronized with house sync frequency.

6.4.2.3.3 CDS Packet payload (TS Packet)

The CDS Packet payload format is shown figure 13. According to the TS Packet structure, the CDS Packet payload is either a 188 byte MPEG-TS Packet or a 204 byte Packet containing a 16-byte RS error correction code or a 16-byte zero code.

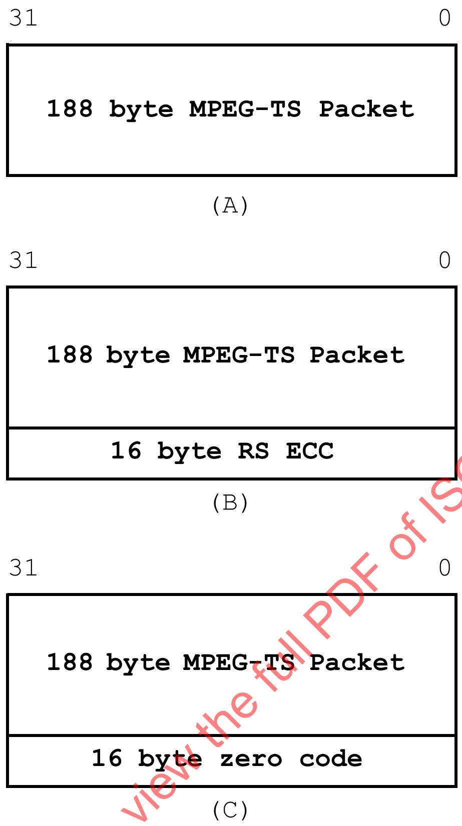


Figure 13 – CDS Packet format (TS Packet)

6.4.2.3.4 FEC

The CDS Packet data is optionally protected by the FEC which is a 4-byte code and is located immediately after the CDS Packet payload when bit 29 of the Packet type is set to 0. The FEC applies to the first byte through the last byte of the CDS Packet. A 4-byte Reed-Solomon error correction shall be used as the method of FEC. The error management scheme is described in 6.3.

7 Frame Header Control Protocol

7.1 Overview

The Frame Header Control Protocol (FHCP) is a low-overhead mechanism for the transport of Containers in Fibre Channel networks. FHCP utilizes FC-2 level properties and as a result, is optimized toward efficient, low-overhead data transport. Alternatively, Simple Streaming Protocol described in clause 8 provides operator control such as *Start*, *Stop, Play, Pause, Resume*. Nodes utilizing FHCP are considered to be constantly in the *Play* mode.

7.2 Network profile

7.2.1 Topology

FHCP shall be applicable to all FC topologies (e.g., point-to-point, arbitrated loop, and switched fabric).

7.2.2 Classes of service

FHCP shall be applicable to all FC classes of service.

7.2.3 Error handling

Error handling functions are not addressed by FCHP and shall be handled by other means (e.g., out of network control or human interface.)

7.2.4 Source and Sink rules

The Video source and sink nodes on the network may support other protocols (e.g., FC-VI, FCP, etc.), but shall be capable of supporting transmission of Containers via FHCP.

7.3 Frame header control fields

7.3.1 Overview

FHCP transport control of Containers shall be characterized by the states of the native control fields of the FC frame header as depicted in figure 14.

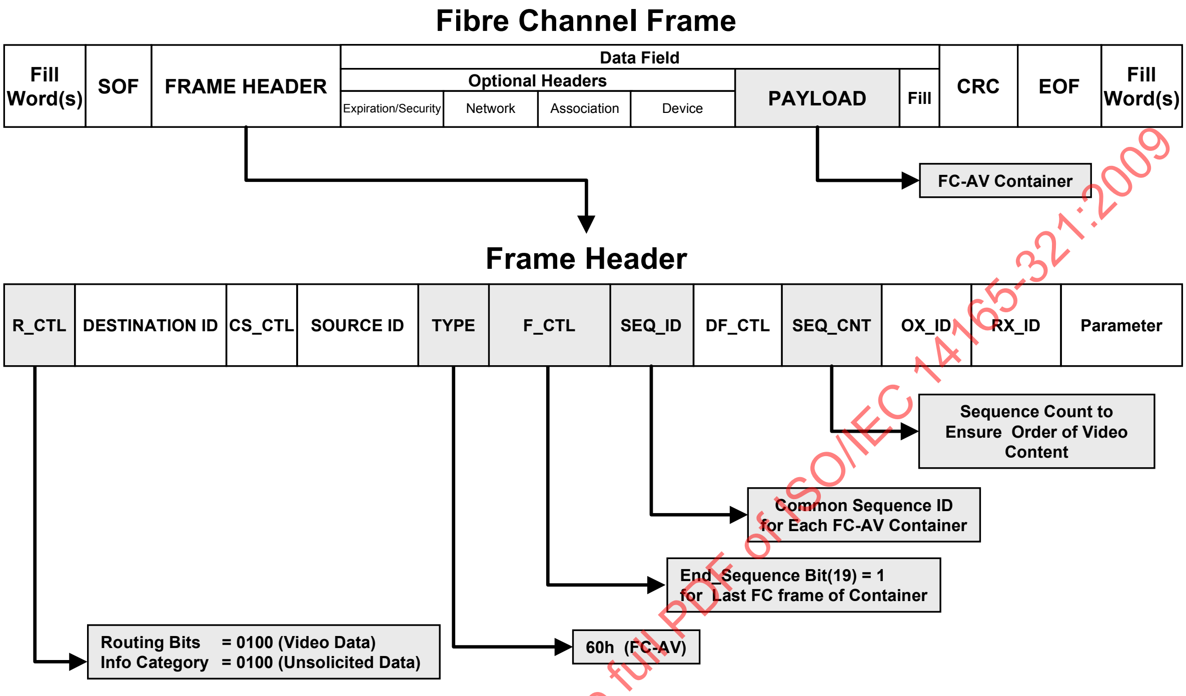


Figure 14 – FHCP Frame Header

7.3.2 Routing Control (R_CTL)

The R_CTL byte shall have its routing bits set to ‘0100’b to designate *Video Data*. The information category bits shall be set to ‘0100’b denoting *Unsolicited Data*.

7.3.3 Type

The Type code field shall be set to 60h as is allocated to Container transfer.

7.3.4 Frame Control (F_CTL)

Bit 19 of the Frame Control field shall be set for the last FC frame of a container. The state of this bit may be used by the recipient node to denote the completion of a Container.

7.3.5 Sequence Identifier (SEQ_ID)

The Sequence Identifier field is set to a single value that identifies all FC frames of a container. This feature may be used by the sink node for data management of the Container. No order information is implied by this field.

7.3.6 Sequence Count (SEQ_CNT)

The Sequence Count field is a value that identifies the FC frame within a container. This feature may be used by the sink node for data management of the Container.

7.3.7 Payload

The FC frame payload carries all the components of the Container (Container Header, Ancillary data, Object data).

7.4 Example FHCP network

An example FHCP network is the avionics system comprised of the following: on-board sensors such as radar and other electro-optic devices (Video sources); one or more display heads (Video sinks); and, a fabric switch. A simplified block diagram of such a network is given in figure 15. Functionally, the sink nodes receive the source Video data in a streaming fashion and display the video in real time. Sensors are assumed to be controlled by external means.

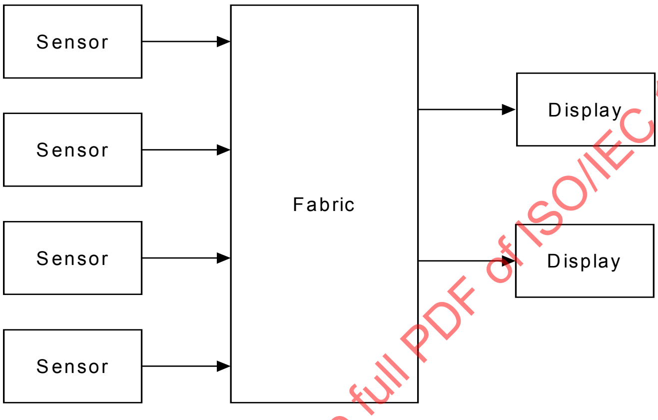


Figure 15 – Example FHCP network

8 Simple Streaming protocol for SCMA

8.1 Overview

The Simple Streaming protocol provides a basic mechanism to transfer Content (AV frames or Containers) between a client and a server.

A server shall be capable of retrieving Content by name and storing new Content by name. Servers do not initiate any Stream establishment operations.

A client usually does not have server capabilities, but that is not precluded in this architecture.

8.2 Stream establishment

8.2.1 Overview

A Stream is a control mechanism used to control the transfer of Content.

A Stream shall be created between a client and a server to agree upon basic operational parameters associated with subsequent Stream operations on Content and to exchange Content. Following the establishment of a Stream, one or more Stream operations may occur between the client and the server.

A Stream operates in either Play mode or Record mode. A single Stream may operate in only one mode. There is no provision in the architecture to change the Stream mode. A Stream shall be torn down and re-established in the opposite mode.

The server shall not permit two Streams on the same Content in opposite modes. The server shall not permit two Streams in Record mode on the same Content.

8.2.2 Stream states

The following states are defined for a Play mode Stream:

- Play Idle
- Play
- Play Pause

The following states are defined for a Record mode Stream:

- Record Idle
- Record
- Record Pause

8.2.3 Establish Stream request

A client shall initiate Stream establishment when the client desires any exchange of Content by sending an Establish Stream request to the server. The Establish Stream request contains operational parameters that shall be valid and agreed upon between the client and server prior to Stream establishment. Following successful Stream establishment, the new Stream enters either the **Play Idle** state or the **Record Idle** state, depending on Stream mode requested.

8.2.4 Establish Stream response

If the Establish Stream request contains operational parameters that are valid and acceptable to the server, then a positive Establish Stream response is sent to the client. This signals the client that Stream operations may begin. The Stream remains in **Play Idle** state or **Record Idle** state at this point. The Stream may be torn down after this point while in any state.

If the parameters are not valid or not agreed to by the server, the server shall send a negative Establish Stream response to the client. The client may choose to send a modified request to the server and attempt to establish a Stream using different operational parameters.

8.2.5 Content Attributes for Establish Stream operations

8.2.5.1 Overview

Content Attributes are transported between the client and the server as part of the Establish Stream operation.

When specifying Play mode for an Establish Stream request, the Content Attributes shall be transported from the server to the client in the associated response.

When specifying Record mode for an Establish Stream requests, Content Attributes shall be transported from the client to the server in the request.

8.2.5.2 Content Attributes for Play mode

Content Type: Specifies the Content type (audio frame, video frame, AV frame, or Container).

Total Number of Frames: A count of the total number AV frames or Containers in the Content.

Total Time: Specifies the total running time at the normal presentation rate.

Frame Size: Specifies the maximum AV frame or Container size in bytes.

Frame Rate: Specifies the normal presentation rate.

Access Control Field: reserved.

8.2.5.3 Content Attributes for Record mode

Content Type: Specifies the Content type (audio frame, video frame, AV frame, or Container).

Total Number of Frames: A count of the total number AV frames or Containers in the Content.

Total Time: Specifies the total running time at the normal presentation rate.

Frame Size: Specifies the maximum AV frame or Container size in bytes.

Frame Rate: Specifies the normal presentation rate.

Access Control Field: Specifies write protected or not write protected.

8.3 Stream Tear-down

8.3.1 Overview

The Stream Tear-down operation may be initiated by the client in any Stream state. As a result of this capability, Stream Tear-down may be an asynchronous operation.

Normally, when the client has completed Content Movement between the client and the server, the Stream Tear-down request is sent by the client to the server. Following a successful Stream tear-down process, resources belonging to the client and server that were associated with the Stream shall be released.

8.3.2 Stream Tear-down request

Following the exchange of Content between the client and the server, the client requests that the Stream be torn down. The client requests this by sending a Stream Tear-down request to the server.

8.3.3 Stream Tear-down response

In response to the Stream Tear-down request, the server indicates that the Stream is torn down by sending a positive Stream Tear-down response to the client.

If the Stream Identifier in the Stream Tear-down request does not match an existing active Stream identifier in the server associated with the client, the server shall respond with a negative Stream Tear-down response.

Successful completion of the Stream Tear-down request shall free any client and server resources associated with the Stream.

8.4 Operations on established Streams

8.4.1 Overview

Following the establishment of a Stream, the client may request Stream operations with the server to begin and manage Content Movement transactions.

A client may use the following Stream operations. Some of these operations shall only be used with Streams in a particular Stream mode (e.g., Play for Streams in Play mode).

- Play
- Record
- Stop
- Pause
- Resume

NOTE – To simplify the architecture, the Simple Streaming protocol does not define fast forward or rewind operations.

8.4.2 Play operation

To initiate a Play operation, a Play operation request shall be sent by a client to a server. The Play operation request is valid only for Play mode Streams in the **Play Idle** state.

If the parameters in the Play operation request are acceptable to the server and the Stream exists in the correct state, the server shall send a positive Play operation response to the client. The Stream makes the transition from to **Play Idle State** to the **Play** state.

The server may then initiate Content Movement transactions. No additional transactions between the client and server are required before Content flows to the client.

A Play operation completes when:

- All the requested Content has been delivered to the client;
- When the client issues a Stop operation request to the server; or,
- When an error occurs during a Content Movement transaction.

When the Play operation completes normally, the server shall send an Operation Complete notification to the client, and the stream makes the transition to the **Play Idle** state.

NOTE – A Pause operation does not complete a Play operation; it merely halts Content Movement transactions.

If the parameters in the Play operation request are unacceptable to the server (i.e., the Stream does not exist in the server), or the Stream, as viewed by the server, is not in the **Play Idle** state, the server shall send a negative Play operation response to the client. If a Stream exists, it shall remain in its current state in the server, however, the server shall not initiate Content Movement transactions.

If an error is detected during the Play operation, the Play operation is complete. The Stream enters the **Play Idle** state and the server shall terminate Content Movement transactions.

If the client detects an error, the client shall send an Operation Error notification to the server.

If the server detects an error, the server shall send an Operation Error notification to the client.

8.4.3 Record operation

To initiate a Record operation, the client sends a Record operation request to the server to indicate to the server that the client intends to start Content Movement transactions.

The Record operation request is valid only for Record mode Streams in the **Record Idle** state.

If the parameters in the Record operation request are acceptable to the server and the Stream exists in the correct state, the Stream makes the transition to the **Record** state. The server shall send a positive Record operation response to the client, and the client may then initiate Content Movement transactions to the server.

A Record operation completes when:

- All the requested Content has been delivered to the server;
- When the client issues a Stop operation request to the server; or,
- When an error occurs during a Content Movement transaction.

When a Record operation completes normally, the client shall send an Operation Complete notification to the server and the stream shall make the transition to the **Record Idle** state.

NOTE – A Pause operation does not complete a Record operation; it merely halts Content Movement transactions.

If the parameters in the Record operation request are unacceptable to the server, the Stream does not exist, or the Stream in the server is not in the **Record Idle** state, the server shall send a negative Record operation response to the client. The Stream, if it exists, shall remain in its current state.

If an error is detected during the Record operation, the steam shall enter the **Record Idle** state, and Content Movement transactions are halted.

If the client detects an error, the client shall send an Operation Error notification to the server.

If the server detects an error, the server shall send an Operation Error notification to the client.

8.4.4 Stop operation

8.4.4.1 Overview

The Stop operation consists of a Stop operation request sent by a client and a Stop operation response sent by a server. The Stop operation request is valid for Streams in Play mode or Record mode, in the **Play** or **Record** states, respectively.

The client issues a Stop operation request to the server for a Stream in the Play mode to halt all Content Movement transactions from the server with an active Play operation request. The Stop operation request may be an asynchronous operation for a Stream in Play mode.

For a Stream in Record mode, the client shall terminate all Content Movement transactions before the Stop operation request is sent. The Stop operation request is not an asynchronous operation for a Stream in Record mode.

8.4.4.2 Stop operation in Play mode

For a Stream in Play mode, a valid Stop operation request indicates to the server that the Play operation is to be terminated. In this case, a positive Stop operation response indicates that the server has halted Content Movement transactions, and has caused the Stream to enter the **Play Idle** state. While in the **Play Idle** state, Play operations may be initiated by the client.

If the parameters in the Stop operation request are unacceptable to the server, the Stream does not exist, or the Stream in the server is not in the **Play** state, the server sends a negative Stop operation response to the client. The Stream, if it exists, shall remain in its current state.

NOTE – After a client sends a Stop operation request for a Stream in Play mode, the client may still receive Content transactions from the server until the Stop operation response is received.

8.4.4.3 Stop operation in Record mode

For an existing Stream in Record mode, a valid Stop operation request indicates to the server that the client has halted Content Movement transactions for the Stream and that the Record operation is to be terminated in the server.

For a valid Stop operation request, the server shall send a positive Stop operation response to indicate that the server shall accept no more Content Movement transactions from the client for the Stream, and causes the Stream to enter the **Record Idle** state. While the Stream is in the **Record Idle** state, the client may initiate subsequent Record operations.

If the parameters in the Stop operation request are unacceptable to the server (i.e., the Stream does not exist in the server) or the Stream in the server is not in the **Record** state, the server shall send a negative Stop operation response to the client. The Stream, if it exists, shall remain in its current state.

8.4.4.4 Stop operation response

Following sending a Stop operation response, the server shall not release any resources associated with the Stream, but may discard any state information about the prior Play or Record operation, except the Stream state.

The Stream associated with a valid Stop operation request shall make the transition to the **Play Idle** state or **Record Idle** state, as appropriate for the Stream mode.

8.4.5 Pause operation

8.4.5.1 Overview

The pause operation is valid for a Stream in Play mode or Record mode in the **Play** or **Record** states, respectively. The pause operation is an asynchronous operation for the Stream in either state.

8.4.5.2 Pause operation in Play mode

For Streams in Play mode, a valid Pause operation request causes the server to halt Content Movement transactions, and causes the Stream to enter the **Play Pause** state. The server shall maintain the appropriate state of the current Play operation so that Playback may resume following a subsequent valid Resume operation request from the client. To indicate that a valid pause operation is complete, the server shall send a positive Pause operation response to the client indicating the proper status.

NOTE – After a client sends a Pause operation request for a Stream in Play mode, the client may still receive Content transactions from the server until the Pause operation response is received.

If the parameters in the Pause operation request are unacceptable to the server, the Stream does not exist, or the Stream in the server is not in the **Play** state, the server shall send a negative Pause operation response to the client. The Stream, if it exists, shall remain in its current state.

8.4.5.3 Pause operation request in Record mode

For Streams in Record mode, a valid Pause operation request shall indicate to the server that the Content Movement transactions have been halted by the client, and causes the Stream to make the transition to the **Record Pause** state. To indicate that the pause operation is complete, the server shall send a positive Pause operation response to the client. The server shall maintain the appropriate state of the current Record operation so that recording may resume following a subsequent valid Resume operation request.

If the parameters in the Pause operation request are unacceptable to the server, the Stream does not exist, or the Stream in the server is not in the **Record** state, the server shall send a negative Pause operation response to the client. The Stream, if it exists, shall remain in its current state.

8.4.5.4 Pause operation response

Following sending a positive Pause operation response, the server shall not release any resources associated with the Stream.

The Stream associated with a valid Pause operation request shall make the transition to the **Play Pause** state or **Record Pause** state, as appropriate for the Stream mode.

8.4.6 Resume operation

8.4.6.1 Overview

The Resume operation is valid for a Play mode Stream in the **Play Pause** state or for a Record mode Stream in the **Record Pause** state, and causes Content Movement transactions between the client and the server to resume.

8.4.6.2 Resume operation request for Play mode

For Streams in Play mode and in the **Play Pause** state, a valid Resume operation request causes the Stream to enter the **Play** state. When the server receives a valid Resume operation request, the server shall send a positive Resume operation response to the client, and Content Movement transactions may resume.

Content Movement transactions shall resume at the exact point after which they were halted by the previous Pause operation request.

If the parameters in the Resume operation request are unacceptable to the server or the Stream in the server is not in the **Play Pause** state, the server shall send a negative Resume operation response to the client. The Stream shall remain in its current state.

8.4.6.3 Resume operation request in Record mode

For Streams in Record mode and in the **Record Pause** state, the Resume operation request indicates to the server that Content Movement transactions are to begin and that recording is to resume. When the server receives the Resume operation request, the Stream enters the **Record** state, a positive Resume operation response shall be sent to the client; Content Movement transactions may resume.

Content Movement transactions shall resume after the exact point where the Pause operation request was halted.

If the parameters in the Resume operation request are unacceptable to the server, the Stream does not exist, or the Stream in the server is not in the **Record Pause** state, the server shall send a negative Resume operation response to the client. The Stream, if it exists, shall remain in its current state.

8.4.6.4 Resume operation response

Following sending a positive Resume operation response, the server shall not release any resources associated with the Stream.

The Stream associated with a valid Resume operation request shall make the transition to the **Play** state or to the **Record** state, as appropriate for the Stream mode.

8.5 Operation notifications

8.5.1 Overview

In addition to operation requests, two asynchronous Operation notifications are defined by the Simple Streaming protocol. An operation notification may be an asynchronous operation.

The operation notifications are:

- Operation Complete notification
- Operation Error notification.

8.5.2 Operation Complete notification

8.5.2.1 Overview

The Operation Complete notification is sent by either the client or server, depending on the Stream mode (Play or Record).

8.5.2.2 Streams in Play mode

For Streams in Play mode, the Operation Complete notification is sent from the server to the client to indicate that the Play operation is complete.

Following the last Content Movement transaction associated with a Play operation, the server shall send an Operation Complete notification to the client. When the client receives the Operation Complete notification, the Stream makes the transition to the **Play Idle** state.

The client may initiate subsequent Play requests while the Stream is in the **Play Idle** state.

There shall be no response to an Operation Complete notification for Streams in Play mode.

8.5.2.3 Streams in Record mode

For Streams in Record mode, the Operation Complete notification is sent from the client to the server when the Record operation completes.

When the last Content Movement transaction has occurred between the client and the server, the client shall send an Operation Complete notification to the server. Upon receipt of the Operation Complete notification, the server performs any necessary processing and the Stream makes the transition to the **Record Idle** state.

The client may initiate subsequent Record requests while in the **Record Idle** state.

There shall be no response to an Operation Complete notification for Streams in Record mode.

8.5.3 Operation Error notification

When an error is detected during Play or Record operations in a Stream, the Operation Error notification shall be used to indicate the error condition to the non-detecting entity.

If the client detects an error, the client shall send an Operation Error notification to the server.

If the server detects an error, the server shall send an Operation Error notification to the client.

The Stream makes the transition to the **Play Idle** or **Record Idle** state as appropriate to the Stream mode.

While in the **Play Idle** or **Record Idle** state, the client may initiate subsequent operation requests appropriate to the Stream mode.

There shall be no response to an Operation Error notification.

8.6 Content Movement transactions

8.6.1 Overview

When a Stream is in the **Play** state or in the **Record** state, Content Movement transactions may occur until:

- the requested operation completes,
- a Stop operation request is processed,
- a Pause operation request is processed,
- a Stream Tear-down request is processed, or
- an error is detected.

Depending on the Stream mode, the client or the server assumes the role of the **producer** and the other the **consumer**.

For the Play operation, the client assumes the role of the consumer and the server assumes the role of the producer.

For the Record operation, the client assumes the role of the producer and the server assumes the consumer the other role.

During each Content Movement transaction, one Audio frame, one Video frame, one AV frame, or one Container compatible with the Stream Content Attributes shall be transferred from the producer to the consumer.

NOTE – Audio frames and Video frames may be quite large (on the order of several megabytes).

A Content Movement transaction consists of the following steps:

1. The producer initiates the Content Movement transaction. An AV frame or Container is transported in the Content Movement transaction from the producer to the consumer.
2. After the last Content Movement transaction, an Operation Complete notification shall be sent from the producer to the consumer to indicate completion of all Content Movement transactions.
3. If an error is detected, the detecting entity (client or server) shall send an Operation Error notification to the other entity.

8.6.2 Content Movement request

A specific Content Movement request is not used in SCMA. Successful completion of a Play operation request or a Record operation request is sufficient to start Content Movement (i.e., implicit permission to begin Content Movement transactions).

For a Play operation, receipt of the Play operation request followed by sending a positive Play operation response grants permission to the producer (server) to start Content transfer. No Content Movement shall occur if the server sends a negative Play operation response.

For a Record operation, sending a Record operation request followed by receipt of a positive Record operation response grants permission to the producer (client) to start Content transfer. No Content Movement shall occur if the server sends a negative Record operation response.

8.6.3 Content Movement response

A specific Content Movement response is not used in SCMA.

When all Content Movement transaction(s) for an operation are complete, the producer shall send an Operation Complete notification to the client indicating completion of all Content Movement transactions for the operation. The Operation Complete notification indicates the successful or unsuccessful completion of the Content Movement transactions. If an error is detected, and Operation Error notification shall be sent by either the producer or the consumer.

8.7 Simple Streaming data format

8.7.1 Overview

The data formats that are associated with the Simple Streaming protocol are defined. Simple Streaming requests, responses, and notifications are transported using the Simple Streaming Request/Response block (SSRB). The SSRB is independent from the underlying transport and is transferred between the FC-AV client and the Video server.

8.7.2 SSRB format

8.7.2.1 Overview

The general format of the SSRB is shown in table 8.

Table 8 – SSRB Format

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Revision	Reserved	Request Identifier	
1	Streaming Operation codes		Reserved	
2	Reserved	Reason Code	Reason Code Explanation	Vendor Unique
3-n	Operation specific data (payload)			

8.7.2.2 Revision field

This field denotes the revision of the SSRB. This field shall be set to a value of 01h.

8.7.2.3 Request Identifier

This field is a 16 bit unsigned integer that provides an identifier for a given request or notification. For requests and notifications, it is the responsibility of the originator of the request or notification to set this value. For responses the value shall be identical to the value specified in the associated request.

8.7.2.4 Streaming Operation code

This field contains the Streaming Operation code which specifies whether the SSRB is a request, response, or notification. Table 9 specifies the Streaming Operation codes.

Table 9 – Streaming Operation codes

Encoded Value	Description
0001h to 7FFFh	Streaming Operation request codes (see table 10 for specific codes)
8001h	Streaming Operation response - Negative
8002h	Streaming Operation response - Positive
FFFEh	Streaming Notification - Operation Error
FFFFh	Streaming Notification - Operation Complete
other values	Reserved

8.7.2.5 Reason Code field

The Reason Code field shall designate the Reason Code associated with a negative Streaming Operation response.

8.7.2.6 Reason Code explanation

The Reason Code explanation field shall designate a Reason Code explanation associated with a negative Streaming Operation response.

8.7.2.7 Vendor Unique field

The Vendor Unique field shall designate a vendor unique explanation associated with a negative Streaming Operation response.

8.7.2.8 Operation specific data

See 8.7.5 for operation of specific data (payloads).

8.7.3 Simple Streaming operation requests

Table 10 indicates the valid Simple Streaming operation requests and their associated operation codes and payload contents.

Table 10 – Simple Streaming operation requests

Code	Mnemonic	Description	Request Attributes	Positive Response Attributes
0101h	ESST	Establish Stream	Content Name, Stream Parameters	Stream ID, Stream Parameters
0201h	STP	Stop	Stream ID	Null
0202h	PAS	Pause	Stream ID	Null
0203h	RSM	Resume	Stream ID	Null
0204h	PLY	Play	Stream ID, Play Parameters	Null
0205h	RCD	Record	Stream ID	Null
0300h	TDST	Tear-Down Stream	Stream ID	Null

8.7.4 Simple Streaming responses

8.7.4.1 Overview

Two types of responses are returned by the server to the client as follows:

1. Negative Operation response
2. Positive Operation response.

8.7.4.2 Negative operation response

When the server is unable to perform a requested operation, a negative operation response is sent from the server to the client. Negative operation responses specify a Reason Code of 09h (Unable to perform operation request).

Table 11 contains the Reason Code explanations for Simple Streaming operations.

Table 11 – Reason Code explanation

Encoded Value	Description
00h	No Additional Explanation
01h	Invalid Stream Parameters
02h	Content with Specified ID Not Found
03h	Server Resources Not Available to Create Stream
04h	Invalid SSRB
05h	Operation Not Initiated - Stream Not Available
10h	Stream Not Resumed - Not in Proper State
11h	Stream Not Paused - Not in Proper State
12h	Stream Not Stopped - Not in Proper State
20h	Invalid Play Parameters
30h	Server Resources Not Available to Perform Operation
others	Reserved

8.7.4.3 Positive Operation response

When the server has successfully initiated a Play or Record operation, a Positive Operation response is sent to the client indicating that the requested operation has been initiated. When the server has successfully completed a Stop, Pause, or Resume operation, a Positive Operation response is sent to the client indicating that the requested operation has been completed.

8.7.5 Simple Streaming operation data

8.7.5.1 Overview

Simple Streaming operation requests and responses may contain data as a payload in the SSRB. The Simple Streaming operations performed by the server and their associated payloads are described below.

8.7.5.2 Establish Stream (ESST) operation

8.7.5.2.1 Overview

The server shall, when it receives an Establish Stream operation request, create the specified Stream as shown in table 12.

Table 12 – ESST Request payload

Item	Size (Bytes)
Stream Mode	1
Reserved	3
Content Name	256
Content Attributes (Record Mode Only)	64

The format of the ESST Response Payload is shown in table 13.

Table 13 – ESST Response payload

Item	Size (Bytes)
Stream Identifier	4
Content Attributes (Play Mode Only)	64

8.7.5.2.2 Stream Mode

The Stream Mode field specifies whether the Stream is a Play mode or Record mode Stream. The values are provided below:

01h - Play Mode Stream
02h - Record Mode Stream
other values - reserved

8.7.5.2.3 Content name

The format of the Content name is depicted in table 14.

Table 14 – Content Name

Item	Size (Bytes)
Content Name Length (m)	1
Content Name	m
Reserved	255-m

8.7.5.2.4 Content Attributes

The format of the Content Attributes structure is provide in table 15. Frames generically refer to either AV frames, or Containers.

Table 15 – Content Attributes

Item	Size (Bytes)
Reserved	1
Content Type	1
Access Control Field	1
Frame Rate	1
Total Frames	4
Maximum Frame Size	4
Total Time	8
Reserved	44

8.7.5.2.4.1 Content Type

This field specifies the type of Content. Valid values are:

01h - Audio frame
02h - Video frame
03h - AV frame
04h - FC-AV Container
other values - reserved.

8.7.5.2.4.2 Access Control Field

This field specifies Access Control attributes associated with the Content. This field is only valid for Record mode Stream Requests. The valid values are:

01h - not write protected
02h - write protected
other values - reserved.

If the client designates the Content as write protected, the server shall not allow the Content to be over-written by a subsequent Record operation.

8.7.5.2.4.3 Frame rate

This field specifies the rate at which the AV frames or Containers shall be processed. The encoded values are the same as the Video Frame Rate field defined in 5.4.1 table 2.

8.7.5.2.4.4 Total frames

This field specifies the total number of AV frames or Containers for a given piece of Content.

8.7.5.2.4.5 Maximum frame size

This field specifies the maximum AV frame or Container size in bytes for a given piece of Content.

8.7.5.2.4.6 Total Time

This field specifies the total running time for a given piece of Content in seconds. The format of the timer is described in the FC-PH Expiration Timer which is a 64-bit fixed point number valued in seconds.

8.7.5.2.5 Stream Identifier

The Stream Identifier uniquely identifies the Stream with respect to the server. The value is provided by the server in response to an Establish Stream Request operation.

8.7.5.3 Tear-Down Stream (TDST) operation

The server shall, when it receives a Tear-Down Stream operation request, tear down the specified Stream. The format of the TDST request payload is shown in table 16.

Table 16 – TDST Request payload

Item	Size (Bytes)
Stream Identifier	4

The TDST response payload shall be null.

8.7.5.4 Stop (STP) operation

The server shall, when it receives a Stop operation request, perform the stop operation on the specified Stream. The format of the STP request payload is shown in table 17.

Table 17 – STP Request payload

Item	Size (Bytes)
Stream ID	4

The Stop response payload shall be null.

8.7.5.5 Pause (PAS) operation

The server shall, when it receives a Pause operation request perform the pause operation on the specified Stream.The format of the PAS request payload is shown in table 18.

Table 18 – PAS Request payload

Item	Size (Bytes)
Stream ID	4

The Pause response payload shall be null.

8.7.5.6 Resume (RSM) operation

The FC-AV server shall, when it receives a Resume operation request perform the resume operation on the specified Stream.The format of the RSM request payload is shown in table 19.

Table 19 – RSM Request payload

Item	Size (Bytes)
Stream ID	4

The Resume response payload shall be null.

8.7.5.7 Play (PLY) operation

8.7.5.7.1 Overview

The server shall, when it receives a Play operation request initiate the Play operation on the specified Stream. The format of the PLY request payload is shown in table 20 and in table 21.

Table 20 – PLY Request payload

Item	Size (Bytes)
Stream ID	4
Play Parameters	64

Table 21 – Play parameters

Item	Size (Bytes)
Format	1
Reserved	3
Begin Relative Frame	4
End Relative Frame	4
Begin Relative Time	8
End Relative Time	8
Reserved	36

8.7.5.7.2 Format field

The format field specifies how the server interprets the remaining Play parameters:

01h - the entire piece of Content is played back. Remaining Play parameters shall be ignored.
02h - the Begin Relative Frame and End Relative Frame fields specify the range of frames to be played back. The Content is played back from the frame specified in the Begin Relative Frame field to the frame specified in the End Relative Frame field.
03h - the Begin Relative Time and End Relative Time fields specify the range of frames to be played back. The Content is played back from the relative time specified in the Begin Relative Time field to the relative time specified in the End Relative Time field.
other values - Reserved.

8.7.5.7.3 Begin Relative Range field

The Begin Relative Range field indicates to the server the beginning frame number for playback.

8.7.5.7.4 End Relative Range field

The End Relative Range field indicates to the server the ending frame number for playback. If the end relative frame number is smaller than the begin relative frame number, a negative response is sent to the client indicating invalid Play parameters.

8.7.5.7.5 Begin Relative Time field

The Begin Relative Time field indicates to the server the beginning relative time for playback.

8.7.5.7.6 End Relative Time field

The End Relative Time field indicates to the server the ending relative time for playback. If the end relative time is smaller than the begin relative time, a negative response is sent to the client indicating invalid Play parameters.

The Play response payload shall be null.

8.7.5.8 Record (RCD) operation

The server shall, when it receives a Record operation request initiate the Record operation on the specified Stream. The format of the RCD request payload is shown in table 22.

Table 22 – PLY Request payload

Item	Size (Bytes)
Stream ID	4

The Record response payload shall be null.

8.7.6 Simple Streaming Notifications

Table 23 indicates the valid Simple Streaming Notifications and their associated codes and payload Contents.

Table 23 – Simple Streaming Notifications

Code	Mnemonic	Description	Notification Attributes
FFFE	NOE	Operation Error	Operation Error Reason Code
FFFF	NOC	Operations Complete	Null

The format of the Operation Error reason and Operation Error codes is shown in table 24 and table 25.

Table 24 – Operation Error Reason Code

Item	Size (Bytes)
Error Code	1
Reserved	3

Table 25 – Operation Error codes

Encoded Values	Description
00h	No Additional explanation
01h	Client Detected Resource error
02h	Client Detected Transport error
03h	Client Detected Processing error
11h	Server Detected Resource error
12h	Server Detected Transport error
13h	Server Detected Processing error
other values	*Reserved*

9 Simple Streaming protocol - SCSI-3 FCP mapping

9.1 Overview

The Simple Streaming protocol is mapped to the SCSI-3 FCP transport. Both Simple Streaming operations and Content Movement transactions are mapped to the FCP transport—-referred to as the Simple Content Movement Architecture (SCMA). The client and server shall each implement the role of initiator and target.

As a target, the client and server shall both implement at least one logical unit that supports the processor device class. The device server for the logical unit shall support the following commands.

The operational processor type command for SCMA operations is the Send command. Other commands required by the various SCSI-3 standards that are applicable to the processor peripheral device class shall be implemented.

There are no optional SCSI-3 commands, command functions, mode pages, log pages or inquiry data required by SCMA that are not required by any applicable technical report.

According to SCSI-3 FCP, if both the initiator and target during process login agree that they can suppress the FCP_XFER_RDY IU for the first burst of a Send command, the client and server shall operate in this mode.

According to SCSI-3 FCP, if either the client or the server cannot suppress the FCP_XFER_RDY IU for the first burst of a Send command, the operations shall follow the protocol as described in 9.3 through 9.12.

9.2 Client and Server requirements

9.2.1 Classes of service

SCMA is restricted only to a class of service common to the client, the server and any intervening switched fabric.

9.2.2 Topology

There are no topology restrictions for SCMA, except as may be imposed by the class of service used.

9.2.3 Quality of Service for Classes 1, 2, and 3

Quality of Service guarantees shall not be made for Classes 2 and 3. For Class 1, each producer shall meter transmission of Content Movement transactions to maintain the agreed upon arrival rate in the Content Attributes for a Stream.

9.2.4 Quality of Service for Class 4

The QoS server in the fabric (well-known address FFFFF9h) shall be responsible to meter credit for the virtual circuit for a Stream. It is the responsibility of the producer to respond to the VC_RDYs for the virtual circuit associated with a Stream to maintain a proper arrival rate at the consumer.

9.2.5 Login

9.2.5.1 Fabric Login

When a fabric is present and the client and server are connected to the fabric, they shall use appropriate explicit login protocols to establish access to the fabric. These transactions are outside the scope of this standard.

9.2.5.2 Port Login

The client shall complete explicit Port Login with any server it intends to generate SCMA operations with. The server is not required to attempt port login to any client. These transactions are outside the scope of this standard.

NOTE – Explicit Port Login is a requirement of SCSI-3 FCP even though it is not required for Class 3 operation.

9.2.5.3 Process Login

Process Login is required by SCSI-3 FCP using the parameters specified in the SCSI-3 FCP standard. For this mapping, the client and the server shall both declare that they are initiator and target mode capable. For other parameters of the PRLI ELS, see SCSI-3 FCP and any appropriate technical report (e.g., FC-PLDA or FC-FLA).

9.3 Establish Stream operation using SCSI-3 FCP

9.3.1 Overview

An Establish Stream operation is mapped to FCP as a Record operation request and a Record operation response as shown in table 26.

Table 26 – Establish Stream operation

Client	IU	Server
Command request	T1, FCP_CMND -->	
	<-- I1, FCP_XFER_RDY	(Prepare for data transfer)
Establish Stream request	T6, FCP_DATA -->	(Receive request)
(Indicate command completion)	<-- I4, FCP_RSP	Send status
	<-- T1, FCP_CMND	Command request
(Prepare for data transfer)	I1, FCP_XFER_RDY -->	
(Receive response)	<-- T6, FCP_DATA	Establish Stream response
Send status	I4, FCP_RSP -->	(Indicate command completion)

9.3.2 Establish Stream request

An FCP_CMND T1 IU is sent from the client initiator to the server processor logical unit containing a send command to the server.

The server transfers an FCP_XFER_RDY I1 IU to the client requesting data for the send command.

The client sends the Establish Stream request to the server in the FCP_Data T6 IU.

The server sends an FCP_RSP I4 IU to the client indicating good status. Good status does not mean that the Establish Stream request is accepted. Good status only means that the request was received correctly.

Establish Stream request parameters:

- **Request Identifier:** uniquely identifies an Establish Stream request from the point of view of the Client. The identifier is not guaranteed to be unique at the server across multiple clients. The client shall use a Request identifier that is different than that associated with any active Stream. A Request Identifier may be reused after a Stream is successfully torn down.

- **Content Name:** identifies the Content requested from a server or to be sent to a server. For a Play mode Stream, the Content Name shall specify an existing piece of Content located on the server. (See Stream mode.) If Content with the specified name does not exist on the server or it exists but is not available to this client, the request shall be rejected with a negative Establish Stream response. For a Record mode Stream the Content Name specifies the name of the Content to be created. If Content with the specified name already exists on the server and the Content is not write-protected on the server, then the Content shall be replaced. If Content with the specified name already exists on the server and the Content is not write-protected on the server, the request shall be rejected with a negative Establish Stream response.
- **Stream mode:** specifies the mode of operation for the specified Stream requested by the client. The two valid Stream modes are Play mode and Record mode.
- **Content Attributes for Record mode:** specifies the basic attributes associated with a piece of Content (see 8.2.5.3). These parameters shall not be specified for Play mode Stream requests. If these parameters are present in the Establish Stream request for a Stream mode of Play, the server shall ignore them.

9.3.3 Establish Stream response

An FCP_CMND T1 IU is sent from the server initiator to the client processor logical unit containing a send command to the server.

The client transfers an FCP_XFER_RDY I1 IU to the server requesting data for the send command.

The server sends the Establish Stream response to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the Establish Stream response is accepted. Good status only means that the response was received correctly.

Establish Stream response parameters:

- **Request Identifier:** the value sent by the client in the Establish Stream request in the Request Identifier.
- **Response type:** shall be set to positive or negative.
- **Stream Identifier:** contains the Stream identifier assigned by the server when the Response type is positive. When the Response type is positive, the server shall assign a value to the Stream Identifier that is different from the value assigned to any active Stream with the same client. The server is not required to assign Stream Identifiers that are unique between two or more clients. If the Response type is negative, the client shall treat the Stream Identifier as not meaningful.
- **Stream mode:** contains the Stream mode of operation requested by the client in the Establish Stream request whether the Response type is positive or negative. The two valid Stream modes are Play mode and Record mode. If the Stream mode returned by the server is different than that requested by the client in the Establish Stream request, the client shall issue a Stream Tear-down request.
- **Content Attributes for Play mode:** specifies the basic attributes associated with a piece of Content (see 8.2.5.2). These parameters shall not be specified for Record mode Stream requests. If these parameters are present in the Establish Stream response for a Stream mode of Record, the client shall ignore them.
- **Reason Code:** if the Response type is negative, the Reason Code contains additional information to help the client manage the negative response. Values: see table 11.

9.4 Stream Tear-down operation using SCSI-3 FCP

9.4.1 Overview

A Stream Tear-down operation is mapped to FCP as a Stream Tear-down operation request and a Stream Tear-down response as shown in table 27.

Table 27 – Stream Tear-down operation

Client	IU	Server
Command request	T1, FCP_CMND -->	
	<-- I1, FCP_XFER_RDY	(Prepare for data transfer)
Stream Tear-down request	T6, FCP_DATA -->	(Receive request)
(Indicate command completion)	<-- I4, FCP_RSP	Send status
	<-- T1, FCP_CMND	Command request
(Prepare for data transfer)	I1, FCP_XFER_RDY -->	
(Receive response)	<-- T6, FCP_DATA	Stream Tear-down response
Send status	I4, FCP_RSP -->	(Indicate command completion)

9.4.2 Stream Tear-down request

An FCP_CMND T1 IU is sent from the client initiator to the server processor logical unit containing a send command to the server.

The server transfers an FCP_XFER_RDY IU to the client requesting data for the send command.

The client sends the Stream Tear-down request to the server in the FCP_Data T6 IU.

The server sends an FCP_RSP I4 IU to the client indicating good status. Good status does not mean that the Stream Tear-down request is accepted. Good status only means that the request was received correctly.

Stream Tear-down request parameter:

- **Stream Identifier:** contains the Stream identifier assigned by the server in the Establish Stream response.

9.4.3 Stream Tear-down response

An FCP_CMND T1 IU is sent from the server initiator to the client processor logical unit containing a send command to the server.

The client transfers an FCP_XFER_RDY IU to the server requesting data for the send command.

The server sends the Stream Tear-down response to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the Stream Tear-down response is accepted. Good status only means that the response was received correctly.

Stream Tear-down response parameters:

- **Stream Identifier:** shall be the same value as that used in the Stream Tear-down request associated with this response.
- **Response type:** shall be set to positive or negative.
- **Reason Code:** if the Response type is negative, the Reason Code contains additional information to help the client manage the negative response. Values: see table 11.

9.5 Play operation using SCSI-3 FCP

9.5.1 Overview

A Play operation is mapped to FCP as a Play operation request and a Play operation response as shown in table 28.

Table 28 – Play operation

Client	IU	Server
Command request	T1, FCP_CMND -->	
	<-- I1, FCP_XFER_RDY	(Prepare for data transfer)
Play request	T6, FCP_DATA -->	(Receive request)
(Indicate command completion)	<-- I4, FCP_RSP	Send status
	<-- T1, FCP_CMND	Command request
(Prepare for data transfer)	I1, FCP_XFER_RDY -->	
(Receive response)	<-- T6, FCP_DATA	Play response
Send status	I4, FCP_RSP -->	(Indicate command completion)

9.5.2 Play operation request

An FCP_CMND T1 IU is sent from the client initiator to the server processor logical unit containing a send command to the server.

The server transfers an FCP_XFER_RDY I1 IU to the client requesting data for the send command.

The client sends the Play operation request to the server in the FCP_Data T6 IU.

The server sends an FCP_RSP I4 IU to the client indicating good status. Good status does not mean that the Play operation request is accepted. Good status only means that the request was received correctly.

Play operation request parameters:

- **Stream Identifier:** specifies the Stream Identifier assigned by the server to an active Stream in play mode.
- **Play Range type:** the Play Range type specified the unit of measure for the Play Range field. The valid units of measure are Frame Count or Relative Time Units.
- **Play range:** the starting Frame Count or Relative Time for the Play operation.

9.5.3 Play operation response

An FCP_CMND T1 IU is sent from the server initiator to the client processor logical unit containing a send command to the server.

The client transfers an FCP_XFER_RDY I1 IU to the server requesting data for the send command.

The server sends the Play operation response to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the Play operation response is accepted. Good status only means that the response was received correctly.

Play operation response parameters:

- **Stream Identifier:** shall be the same value as that used in Play operation request associated with this response.
- **Response type:** shall be set to positive or negative.
- **Reason Code:** if the Response type is negative, the Reason Code contains additional information to help the client manage the negative response. Values: see table 11.

9.6 Record operation using SCSI-3 FCP

9.6.1 Overview

A Record operation is mapped to FCP as a Record operation request and a Record operation response as shown in table 29.

Table 29 – Record operation

Client	IU	Server
Command request	T1, FCP_CMND -->	
	<-- I1, FCP_XFER_RDY	(Prepare for data transfer)
Record request	T6, FCP_DATA -->	(Receive request)
(Indicate command completion)	<-- I4, FCP_RSP	Send status
	<-- T1, FCP_CMND	Command request
(Prepare for data transfer)	I1, FCP_XFER_RDY -->	
(Receive response)	<-- T6, FCP_DATA	Record response
Send status	I4, FCP_RSP -->	(Indicate command completion)

9.6.2 Record operation request

An FCP_CMND T1 IU is sent from the client initiator to the server processor logical unit containing a send command to the server.

The server transfers an FCP_XFER_RDY I1 IU to the client requesting data for the send command.

The client sends the Record operation request to the server in the FCP_Data T6 IU.

The server sends an FCP_RSP I4 IU to the client indicating good status. Good status does not mean that the Record operation request is accepted. Good status only means that the request was received correctly.

Record operation request parameter:

- **Stream Identifier:** assigned by the server to an active Stream in Play mode or Record mode.

9.6.3 Record operation response

An FCP_CMND T1 IU is sent from the server initiator to the client processor logical unit containing a send command to the server.

The client transfers an FCP_XFER_RDY I1 IU to the server requesting data for the send command.

The server sends the Record operation response to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the Record operation response is accepted. Good status only means that the response was received correctly.

Record operation response parameters:

- **Stream Identifier:** shall be the same value as that used in Record operation request associated with this response.

- **Response type:** shall be set to positive or negative.
- **Reason Code:** if the Response type is negative, the Reason Code contains additional information to help the client manage the negative response. Values: see table 11.

9.7 Stop operation using SCSI-3 FCP

9.7.1 General

A Stop operation is mapped to FCP as a Stop operation request and a Stop operation response as shown in table 30.

Table 30 – Stop operation

Client	IU	Server
Command request	T1, FCP_CMND -->	
	<-- I1, FCP_XFER_RDY	(Prepare for data transfer)
Stop request	T6, FCP_DATA -->	(Receive request)
(Indicate command completion)	<-- I4, FCP_RSP	Send status
	<-- T1, FCP_CMND	Command request
(Prepare for data transfer)	I1, FCP_XFER_RDY -->	
(Receive response)	<-- T6, FCP_DATA	Stop response
Send status	I4, FCP_RSP -->	(Indicate command completion)

9.7.2 Stop operation Request

An FCP_CMND T1 IU is sent from the client initiator to the server processor logical unit containing a send command to the server.

The server transfers an FCP_XFER_RDY I1 IU to the client requesting data for the send command.

The client sends the Stop operation request to the server in the FCP_Data T6 IU.

The server sends an FCP_RSP I4 IU to the client indicating good status. Good status does not mean that the Stop operation request is accepted. Good status only means that the request was received correctly.

Stop operation request parameter:

- **Stream Identifier:** specifies the Stream identifier assigned by the server to an active Stream in play mode.

9.7.3 Stop operation response

An FCP_CMND T1 IU is sent from the server initiator to the client processor logical unit containing a send command to the server.

The client transfers an FCP_XFER_RDY I1 IU to the server requesting data for the send command.

The server sends the Stop operation response to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the Stop operation response is accepted. Good status only means that the response was received correctly.

Stop operation response parameters:

- **Stream Identifier:** shall be the same value as that used in Stop operation request associated with this response.
- **Response type:** shall be set to positive or negative.
- **Reason Code:** if the Response type is negative, the Reason Code contains additional information to help the client manage the negative response. Values: see table 11.

9.8 Pause operation using SCSI-3 FCP

9.8.1 Overview

A Pause operation is mapped to FCP as a Pause operation request and a Pause operation response as shown in table 31.

Table 31 – Pause operation

Client	IU	Server
Command request	T1, FCP_CMND -->	
	<-- I1, FCP_XFER_RDY	(Prepare for data transfer)
Pause request	T6, FCP_DATA -->	(Receive request)
(Indicate command completion)	<-- I4, FCP_RSP	Send status
	<-- T1, FCP_CMND	Command request
(Prepare for data transfer)	I1, FCP_XFER_RDY -->	
(Receive response)	<-- T6, FCP_DATA	Pause response
Send status	I4, FCP_RSP -->	(Indicate command completion)

9.8.2 Pause operation request

An FCP_CMND T1 IU is sent from the client initiator to the server processor logical unit containing a send command to the server.

The server transfers an FCP_XFER_RDY I1 IU to the client requesting data for the send command.

The client sends the Pause operation request to the server in the FCP_Data T6 IU.

The server sends an FCP_RSP I4 IU to the client indicating good status. Good status does not mean that the Pause operation request is accepted. Good status only means that the request was received correctly.

Pause operation request parameter:

- **Stream Identifier**: assigned by the server to an active Stream in play mode or Record mode.

9.8.3 Pause operation response

An FCP_CMND T1 IU is sent from the server initiator to the client processor logical unit containing a send command to the server.

The client transfers an FCP_XFER_RDY I1 IU to the server requesting data for the send command.

The server sends the Pause operation response to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the Pause operation response is accepted. Good status only means that the response was received correctly.

Pause operation response parameters:

- **Stream Identifier:** shall be the same value as that used in Pause operation request associated with this response.
- **Response type:** shall be set to positive or negative.
- **Reason Code:** if the Response type is negative, the Reason Code contains additional information to help the client manage the negative response. Values: see table 11

9.9 Resume operation using SCSI-3 FCP

9.9.1 Overview

A Resume operation is mapped to FCP as a Resume operation request and a Resume operation response as shown in table 32.

Table 32 – Resume operation

Client	IU	Server
Command request	T1, FCP_CMND -->	
	<-- I1, FCP_XFER_RDY	(Prepare for data transfer)
Resume request	T6, FCP_DATA -->	(Receive request)
(Indicate command completion)	<-- I4, FCP_RSP	Send status
	<-- T1, FCP_CMND	Command request
(Prepare for data transfer)	I1, FCP_XFER_RDY -->	
(Receive response)	<-- T6, FCP_DATA	Resume response
Send status	I4, FCP_RSP -->	(Indicate command completion)

9.9.2 Resume operation request

An FCP_CMND T1 IU is sent from the client initiator to the server processor logical unit containing a send command to the server.

The server transfers an FCP_XFER_RDY I1 IU to the client requesting data for the send command.

The client sends the Resume operation request to the server in the FCP_Data T6 IU.

The server sends an FCP_RSP I4 IU to the client indicating good status. Good status does not mean that the operation notification complete is accepted. Good status only means that the request was received correctly.

Resume operation request parameter:

- **Stream Identifier:** assigned by the server to an active Streaming Protocol in Play mode or Record mode.

9.9.3 Resume operation response

An FCP_CMND T1 IU is sent from the server initiator to the client processor logical unit containing a send command to the server.

The client transfers an FCP_XFER_RDY I1 IU to the server requesting data for the send command.

The server sends the Resume operation response to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the Resume operation response is accepted. Good status only means that the response was received correctly.

Resume operation response parameters:

- **Stream Identifier:** shall be the same value as that used in Resume operation request associated with this response.

- **Response type:** shall be set to positive or negative.
- **Reason Code:** if the Response type is negative, the Reason Code contains additional information to help the client manage the negative response. Values: see table 11.

9.10 Operation Complete notification using SCSI-3 FCP

An Operation Complete notification is mapped to FCP as a Resume operation request and a Resume operation response as shown in table 33.

Table 33 – Operation Complete notification

Client	IU	Server
	<-- T1, FCP_CMND	Command request
(Prepare for data transfer)	I1, FCP_XFER_RDY -->	
(Receive notification)	<-- T6, FCP_DATA	Operation Complete notification
Send status	I4, FCP_RSP -->	(Indicate command completion)

An FCP_CMND T1 IU is sent from the server initiator to the client processor logical unit containing a send command to the server.

The client transfers an FCP_XFER_RDY I1 IU to the server requesting data for the send command.

The server sends the Operation Complete notification to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the Operation Complete notification is accepted. Good status only means that the notification was received correctly.

There is no response to an Operation Complete notification.

Operation Complete notification parameters:

- **Stream Identifier:** value shall be the associated with the operation that has completed.
- **Response type:** shall be set to positive.

9.11 Operation Error notification using SCSI-3 FCP

An Operation Error notification is mapped to FCP as a Resume operation request and a Resume operation response as shown in table 34.

Table 34 – Operation Complete notification

Client	IU	Server
	<-- T1, FCP_CMND	Command request
(Prepare for data transfer)	I1, FCP_XFER_RDY -->	
(Receive notification)	<-- T6, FCP_DATA	Operation Error notification
Send status	I4, FCP_RSP -->	(Indicate command completion)

An FCP_CMND T1 IU is sent from the server initiator to the client processor logical unit containing a send command to the server.

The client transfers an FCP_XFER_RDY I1 IU to the server requesting data for the send command.

The server sends the Operation Error notification to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the Operation Error notification is accepted. Good status only means that the notification was received correctly.

There is no response to an Operation Error notification.

Operation Error notification parameters:

- **Stream Identifier:** shall be the same value as that used in Resume operation request associated with this response.
- **Response type:** shall be set to positive or negative.
- **Reason Code:** if the Response type is negative, the Reason Code contains additional information to help the client manage the negative response. Values are:
  - Transport error;
  - Processing error; and,
  - Local Resource error.

9.12 SCMA Content Movement transaction

9.12.1 Overview

There are two Content Movement directions in SCMA as shown in table 35.

Table 35 – SCMA Content Movement operation

Client	IU	Server
Command request	T1, FCP_CMND -->	
	<-- I1, FCP_XFER_RDY	(Prepare for data transfer)
Send AV frame	T6, FCP_DATA -->	(Receive AV frame)
(Indicate command completion)	<-- I4, FCP_RSP	Send status
Command request	T1, FCP_CMND -->	
(Receive AV frame)	<-- I3, FCP_DATA	Send AV frame
(Indicate command completion)	<-- I4, FCP_RSP	Send status

9.12.2 SCMA Content Movement from the server to the client (Play operations)

An FCP_CMND T1 IU is sent from the producer initiator to the consumer processor logical unit containing a send command to the server. The send command contains the length of the Container to be transferred.

The client transfers an FCP_XFER_RDY I1 IU to the server requesting the total amount of data for the send command.

The server sends the AV frame or Container to the client in the FCP_Data T6 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the frame or container is accepted. Good status only means that the content was received correctly.

9.12.3 SCMA Content Movement from the client to the server (Record operations).

An FCP_CMND T1 IU is sent from the producer initiator to the consumer processor logical unit containing a send command to the client. The send command contains the length of the Container to be transferred.

The server sends the AV frame or Container to the client in the FCP_Data I3 IU.

The client sends an FCP_RSP I4 IU to the server indicating good status. Good status does not mean that the frame or container is accepted. Good status only means that the content was received correctly.

Annex A (normative)

Simple Parametric Digital Video profile

A.1 Overview

The Simple Parametric Digital Video (SPDV) profile defines a mapping based on the Container system as defined in Clause 5. The target applications of this profile includes real-time distribution of digital video for sensors and displays in avionics systems. To minimize redundancy within this specification, all definitions profiled within this annex are not restated herein. Reference to the appropriate clause, table, and figure numbers of the basic specification are provided as required. All content of this annex shall be considered normative unless otherwise noted.

A.2 Container Header format

A.2.1 Overview

SPDV shall utilize the Simple Header Mode of operation and as such the Container Header shall be as defined in 5.4 and table 1.

A.2.2 Container Count

SPDV shall support the use of Container Count to identify a particular Container in a video clip.

A.2.3 Clip ID

SPDV shall support the use of Clip ID to identify the Container as an element of a particular video clip.

The Clip ID identifies a particular video stream in cases where more than one video stream is to be transferred over a single Fibre Channel link.

NOTE – If a clip is defined as exactly one Container, the Container Count becomes a Video frame counter.

A.2.4 Container Time Stamp

SPDV may support the optional use of Container Time Stamp.

A.2.5 Transmission type

A.2.5.1 Video frame rate

SPDV shall support the use of Video frame rate field as shown in table 2.

NOTE – The Null frame rate entry (code 00h) may be used to denote image data which may be aperiodic and/or is updated at frame rates less than 15 Hz.

A.2.5.2 Transmission rate

SPDV shall support the use of Transmission rate.

NOTE – The majority of applications have this field set to 01h.

A.2.6 Container type

A.2.6.1 Mode

SPDV shall, by definition, operate in the Simple mode as described in 5.2.3. The mode field (Word 5-Byte 0) shall be set to 00h.

A.2.6.2 Number of Objects

SPDV Containers shall, by definition of Simple mode, have four objects. Word 5, Byte 1 shall be set to 04h.

NOTE – Some implementations may not use all objects (e.g., no Audio data present or progressive scan video.) Unused objects are denoted with Object size = 0 (see A.2.7.4).

A.2.6.3 Size of Extended Header

SPDV does not support the Extended Header mode. This field shall be set to 00h.

A.2.7 Object Information Block

A.2.7.1 Object n Type

SPDV Containers are comprised of the valid object types supported in the Simple mode as defined in 5.2.3. Specific values for the Object n Type for SPDV are defined in A.3.

A.2.7.2 Object n Link Pointer

SPDV may support the use of Object Link Pointer.

A.2.7.3 Object n Index

A description of Object Index values is given is 5.6.3. The Object Index value assigned to all SPDV object types shall be D000h.

A.2.7.4 Object n Size

SPDV shall utilize the Object Size value in bytes. For unused objects (e.g., Object 1-Audio Data and/or Object 3-Even Video Field Data) Object Size shall be set to 00000000h.

A.2.7.5 Object n Offset

SPDV shall utilize the Object Offset value in bytes from the beginning of the Container.

NOTE – Since the size of an unused object is 0, that object does not contribute to the offset and thus will be equal to the offset for the next object.

A.2.7.6 Object n Type defined

SPDV shall not utilize Object Type defined and shall set this field to 00000000h.

A.3 Object description

A.3.1 General

Valid objects in simple mode for SPDV are:

Object 0 – Restricted to Ancillary data types	(Object Type number 50h)
Object 1 – Restricted to Audio data object types	(Object Type number 40h)
Object 2 – Restricted to Video data object types	(Object Type number 10h)
Object 3 – Restricted to Video data object types	(Object Type number 10h)

Each Container shall have a container Header; however, each Container can optionally have data in Objects 0, 1, 2, 3. This annex presently addresses Objects 0, 2 and 3.

A.3.2 Object 0– Ancillary Data – Uncompressed Video

A.3.2.1 General

Ancillary data, in the context of this annex, defines the characteristics of the data being transferred from the sender to the recipient. The following paragraphs in conjunction with figure A.1 define the format of Object 0 Ancillary data for uncompressed SPDV applications.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Number of Rows														Number of Columns														Frame/Field			

Word 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CI				PA				PAO				PTN				Bits/Subpixel A				Bits/Subpixel B				Bits/Subpixel C				Bits/Subpixel D			

CI: Color Information Type
PA: Pixel Aspect Ratio (Width:Height)
PAO: Pixel Array Order
PTN: Packing Table Number

Word 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Defined																															

Word 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Defined																															

Words 4-259*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Palette Number								Red Value								Green Value								Blue Value							

Note: Words 4-259 are only used when Color Information Type is Palletized Color - Ancillary Object Table (8h)

Figure A.1 – Object 0 – Ancillary data definition

A.3.2.2 Object 0 Word 0

A.3.2.2.1 Number of Rows

This field of 14-bits shall represent the total Number of Rows within the entire Video frame. 000h shall represent a null. The maximum number of rows is 16383 represented by 3FFFh.

A.3.2.2.2 Number of Columns

This field of 14-bits shall represent the total Number of Columns within the entire Video frame. 000h shall represent a null. The maximum number of columns is 16383 represented by 3FFFh.

A.3.2.2.3 Frame/Field Based Video

This 4-bit field shall represent the format of the transmitted video. Table A.1 defines frame based and field based representations. The most significant bit (bit 3) is intended as a flag to indicate to the recipient that the Object is field based video.

Table A.1 – Video Format codes

Video Format	Code
Full Video Frame	0h
Spare	1h to 7h
Field based; both fields within this Container	8h
Field based; Field 1 only	9h
Field based; Field 2 only	Ah
Spare	Bh to Fh

A.3.2.3 Object 0 Word 1

A.3.2.3.1 Color Information

This 4-bit field shall represent the format of the pixel data; either monochrome or color. Table A.2 defines the Color Information coding. Setting the Color Information (CI) field to 8h (Color Palette-Ancillary Data table) shall indicate to the recipient that Object 0 Words 4-259 are required to be read in order to interpret the Video Object(s) (Objects 2 and/or 3). Setting the CI field to 9h (Color Palette-External table) shall indicate to the recipient that a static look-up table has been transmitted and this table shall be read in order to interpret the Video Object(s) (Objects 2 and/or 3).

Table A.2 – Color Information codes

Color Information	Code
Monochrome	0h
RGB (Red, Green, Blue)	1h
RGGB (Red, Green, Green, Blue)	2h
RGBA (Red, Green, Blue, Alpha)	3h
YIQ	4h
YCbCr (4:2:2)	5h
Spare	6h to 7h
Color Palette; Ancillary Object table	8h
Color Palette; External table	9h
Spare	Ah to Fh

A.3.2.3.2 Pixel Aspect ratio

This 4-bit field shall represent the Pixel Aspect ratio of width to height of an entire pixel. Table A.3 defines the Pixel Aspect ratio coding.

Table A.3 – Pixel Aspect ratio codes

Pixel Aspect Ratio	Code
1:1	0h
1:1,2	1h
1,2:1	2h
Spare	3h to Fh

A.3.2.3.3 Pixel Array order

This 4-bit field shall represent the Pixel Array order in which Video data is packed in the Video Object(s) with respect to their position in the Video frame (or field). Table A.4 defines the Pixel Array order coding.

Table A.4 – Pixel Array order codes

Video Data Order	Codes
Left to Right, Top to Bottom	0h
Right to Left, Top to Bottom	1h
Left to Right, Bottom to Top	2h
Right to Left, Bottom to Top	3h
Top to Bottom, Left to Right	4h
Top to Bottom, Right to Left	5h
Bottom to Top, Left to Right	6h
Bottom to Top, Right to Left	7h
Spare	8h to Fh

A.3.2.3.4 Packing Table number

This 4-bit field shall contain a hexadecimal number representing Packing Table formats as listed in table A.5. The bit and byte placements of the samples associated with these formats are illustrated in figure A.2. These formats are a subset of those described in annex G.

Table A.5 – Packing Table number

Description	PTN
8-bit samples, 4 samples per transmission word, Msb first	0h
10-bit samples, 3 samples per transmission word, Msb first	1h
12-bit samples, 2 2/3 samples per transmission word, Msb first	2h
16-bit samples, 2 samples per transmission word, Msb first	3h
20-bit samples, 1 12/20 samples per transmission word, Msb first	4h
24-bit samples, 1 1/3 samples per transmission word, Msb first	5h
32-bit samples, 1 sample per transmission word, Msb first	6h
Spare	7h to Fh

For n-bit samples not shown (e.g., 14-bit or 22-bit), the next higher sample size shown shall be used as the packing guideline and the unused bits shall be padded with zeroes in the Lsb position.

<table>
<tr><td colspan="8">Byte 0</td><td colspan="8">Byte 1</td><td colspan="8">Byte 2</td><td colspan="8">Byte 3</td><td rowspan="2">**PTN**</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Msb Sample 0 Lsb</td><td colspan="8">Msb Sample 1 Lsb</td><td colspan="8">Msb Sample 2 Lsb</td><td colspan="8">Msb Sample 3 Lsb</td><td>0h</td></tr>
<tr><td>0</td><td>0</td><td colspan="10">Msb Sample 0 Lsb</td><td colspan="10">Msb Sample 1 Lsb</td><td colspan="10">Msb Sample 2 Lsb</td><td>1h</td></tr>
<tr><td colspan="12">Msb Sample 0 Lsb</td><td colspan="12">Msb Sample 1 Lsb</td><td colspan="8">Msb Sample 2</td><td rowspan="3">2h</td></tr>
<tr><td colspan="4">S2 Lsb</td><td colspan="12">Msb Sample 3 Lsb</td><td colspan="12">Msb Sample 4 Lsb</td><td colspan="4">Msb S5</td></tr>
<tr><td colspan="8">Sample 5 Lsb</td><td colspan="12">Msb Sample 6 Lsb</td><td colspan="12">Msb Sample 7 Lsb</td></tr>
<tr><td colspan="16">Msb Sample 0 Lsb</td><td colspan="16">Msb Sample 1 Lsb</td><td>3h</td></tr>
<tr><td colspan="20">Msb Sample 0 Lsb</td><td colspan="12">Msb Sample 1</td><td rowspan="5">4h</td></tr>
<tr><td colspan="8">Sample 1 Lsb</td><td colspan="20">Msb Sample 2 Lsb</td><td colspan="4">Msb S3</td></tr>
<tr><td colspan="16">Sample 3 Lsb</td><td colspan="16">Msb Sample 4</td></tr>
<tr><td colspan="4">S4 Lsb</td><td colspan="20">Msb Sample 5 Lsb</td><td colspan="8">Msb Sample 6</td></tr>
<tr><td colspan="12">Sample 6 Lsb</td><td colspan="20">Msb Sample 7 Lsb</td></tr>
<tr><td colspan="24">Msb Sample 0 Lsb</td><td colspan="8">Msb Sample 1</td><td rowspan="3">5h</td></tr>
<tr><td colspan="16">Sample 1 Lsb</td><td colspan="16">Msb Sample 2</td></tr>
<tr><td colspan="8">Sample 2 Lsb</td><td colspan="24">Msb Sample 3 Lsb</td></tr>
<tr><td colspan="32">Msb Sample 0 Lsb</td><td>6h</td></tr>
</table>

Figure A.2 – Bit and Byte Packing

A.3.2.3.5 Bits per subpixel (n)

These four fields, each 4-bits long, shall independently specify the number of bits that represent each subpixel within the Video object(s). The number of applicable fields will be governed by the subpixels required to support the specified Color Information format (see A.3.2.3.1 and table A.6). The minimum number of bits for a subpixel is 1 represented by 0h. The maximum number of bits for a subpixel is 16 represented by Fh.

Table A.6 – Color Information format per subpixel mapping

Color Information	Bits/Subpixel A	Bits/Subpixel B	Bits/Subpixel C	Bits/Subpixel D
Monochrome	M	X	X	X
RGB	R	G	B	X
RGGB	R	G	G	B
RGBA	R	G	B	A
YIQ	Y	I	Q	X
YCbCr (4:2:2)	Y	Cb	Cr	X

X=Don't Care

A.3.2.4 Object 0, Words 2 and 3 – User Defined words

Words 2 and 3 of Object 0 shall be designated as User Defined words. When not used, the words shall be set to the default value of 00000000h.

A.3.2.5 Object 0 Words 4-259 Ancillary Object Color palette

These words shall only be used when the Color Information field, Object 0 Word 1 Bits 28-31, is set to 8h (the value for Color Palette-Ancillary Object table as defined in A.3.2.3.1). These words shall represent the mapping of a color palette to red, green and blue intensity bits and shall be used by the recipient for displaying/processing Video data contained in Objects 2 and 3. Table A.7 defines the Color Palette word format.

Table A.7 – Word 4-259 Color Palette word format

Palette Number	Palette Color	Red Intensity Value	Red Intensity Code	Green Intensity Value	Green Intensity Code	Blue Intensity Value	Blue Intensity Code
0-255	00h-FFh	0-255	00h-FFh	0-255	00h-FFh	0-255	00h-FFh

A.3.3 Object 1 - Audio data

SPDV does not currently address the use of Audio data.

A.3.4 Objects 2 and 3 - Video data

Objects 2 and 3 shall be comprised of Video data. The order of this data shall be governed by the Pixel Array Order field defined in Object 0, Word 1 in A.3.2.3.3. If the Video format is progressive (i.e., frame based) only Object 2 shall contain data. If the Video format is interlaced (field based), Object 2 shall contain Field 1 (odd field) and Object 3 shall contain Field 2 (even field). A.3.2.2.3 defines the possible formats of frame and field mappings within Containers.

A.4 Profile examples (Informative)

A.4.1 Monochrome Video example

By utilizing the information contained in the Container Header (as depicted in figure A.3) in conjunction with parameter definitions provided by Object 0 (as depicted in figure A.4), the format of Object 2 is established. For this example, a 480 x 480 monochrome source is progressively scanned at a 30 Hz frame rate with 8-bits per sample. Pixels are transmitted in sequence from left to right and top to bottom. The mapping of the pixel data to the FC frame payloads is depicted in figure A.5.

	Byte 0	Byte 1	Byte 2	Byte 3
Container Count	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Clip ID	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Container Time Stamp	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Container Time Stamp	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Transmission Type	0 1 0 0 0 1 0 1	0 0 0 0 0 0 0 1	x x x x x x x x	x x x x x x x x
Container Type	0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0	x x x x x x x x	0 0 0 0 0 0 0 0
Object 0 Class	0 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0	1 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0
Object 0 Size	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 0
Object 0 Offset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 1 0 1 1 0 0 0
Object 0 Type Defined	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Object 1 Class	0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0	1 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0
Object 1 Size	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Object 1 Offset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 1 1 0 1 0 0 0
Object 1 Type Defined	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Object 2 Class	0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0	1 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0
Object 2 Size	0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 1	1 0 0 0 0 1 0 0	0 0 0 0 0 0 0 0
Object 2 Offset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 1 1 0 1 0 0 0
Object 2 Type Defined	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Object 3 Class	0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0	1 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0
Object 3 Size	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Object 3 Offset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 1	1 0 0 0 0 1 0 0	0 1 1 0 1 0 0 0
Object 3 Type Defined	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x

Note: Entries of **x** denote "don't cares"

Figure A.3 – Monochrome example – Container Header

NOTE – The Byte numbers shown in figure A.5 do not include the 88 bytes of Container Header or the 16 bytes of Object 0. Implementors may choose to transmit these bytes in a separate FC frame.

Object 0 Word 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
480 Rows														480 Columns														Frame Based			

Object 0 Word 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X
Monochrome				1:1				Left to Right/ Top to Bottom				8 Bit Samples PTN=0				8 Bits per Pixel				Don't Care				Don't Care				Don't Care			

Object 0 Words 2&3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Defaulted to 0 (No User Defined Data)

Figure A.4 – Monochrome example - Object 0 (Ancillary data)

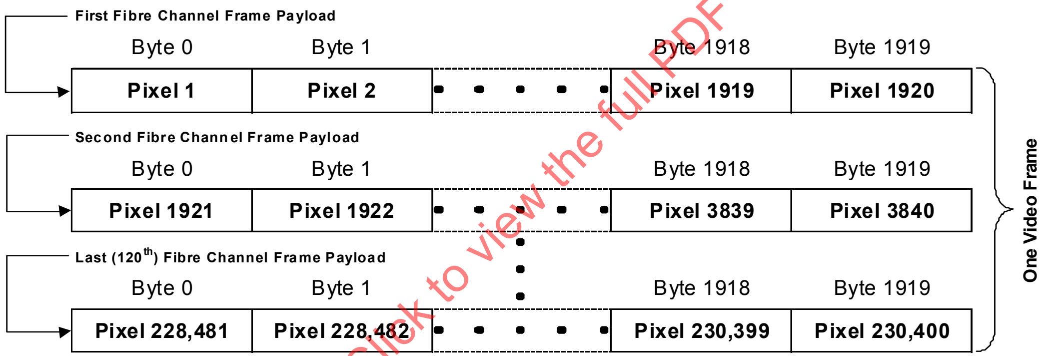
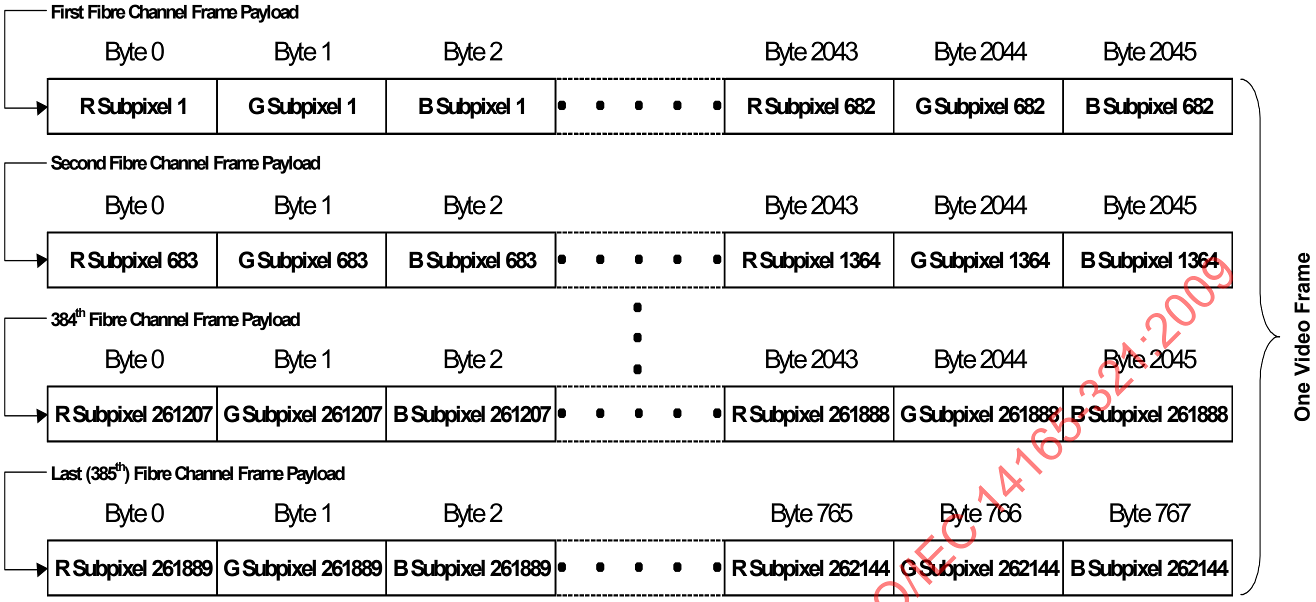


Figure A.5 – Monochrome example - Object 2 (Video data)

A.4.2 RGB Video example



By utilizing the information contained in the Container Header (as depicted in figure A.6) in conjunction with parameter definitions provided by Object 0 (as depicted in figure A.7), the format of Object 2 is established. For this example, each pixel is comprised of 24 bits, representing three subpixels (i.e., 8-bit samples of R, G, and B). The format is 512 x 512 progressively scanned at a 60 Hz frame rate. Pixels are transmitted in sequence from left to right and top to bottom. The mapping of the pixel data to the FC frame payloads is depicted in figure A.8.

	Byte 0	Byte 1	Byte 2	Byte 3
Container Count	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Clip ID	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Container Time Stamp	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Container Time Stamp	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Transmission Type	0 0 0 0 0 1 1 1	0 0 0 0 0 0 0 1	x x x x x x x x	x x x x x x x x
Container Type	0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0	x x x x x x x x	x x x x x x x x
Object 0 Class	0 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0	1 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0
Object 0 Size	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 0
Object 0 Offset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 1 0 1 1 0 0 0
Object 0 Type Defined	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Object 1 Class	0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0	1 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0
Object 1 Size	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Object 1 Offset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 1 1 0 1 0 0 0
Object 1 Type Defined	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Object 2 Class	0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0	1 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0
Object 2 Size	0 0 0 0 0 0 0 0	0 0 0 0 1 1 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Object 2 Offset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 1 1 0 1 0 0 0
Object 2 Type Defined	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x
Object 3 Class	0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0	1 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0
Object 3 Size	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Object 3 Offset	0 0 0 0 0 0 0 0	0 0 0 0 1 1 0 0	0 0 0 0 0 0 0 0	0 1 1 0 1 0 0 0
Object 3 Type Defined	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x

Note: Entries of **x** denote "don't cares"

Figure A.6 – RGB example - Container Header

Object 0 Word 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
512 Rows														512 Columns														Frame Based			

Object 0 Word 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	1	1	1	X	X	X	X
RGB				1:1				Left to Right/ Top to Bottom				8 Bit Samples PTN = 0				8 Bits per Red Subpixel				8 Bits per Green Subpixel				8 Bits per Blue Subpixel				Don't Care			

Object 0 Words 2&3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Defaulted to 0 (No User Defined Data)																															

Figure A.7 – RGB example - Object 0 (Ancillary Data)

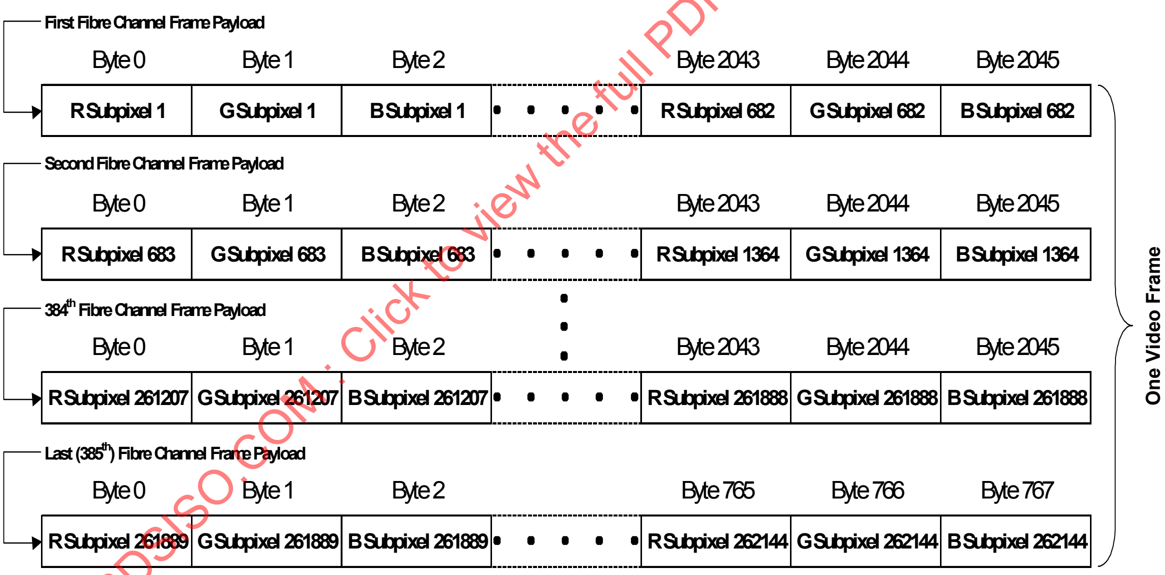


Figure A.8 – RGB example - Object 2 (Video Data)

NOTE – The Byte numbers shown in figure A.8 do not include the 88 bytes of Container Header or the 16 bytes of Object 0. Implementors may choose to transmit these bytes in a separate FC frame.