SCTE · ISBE

STANDARDS

Digital Video Subcommittee

AMERICAN NATIONAL STANDARD

ANSI/SCTE 223 2018

Adaptive Transport Stream

NOTICE

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

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

Attention is called to the possibility that implementation of this document may require the use of subject matter covered by patent rights. By publication of this document, no position is taken with respect to the existence or validity of any patent rights in connection therewith. SCTE•ISBE shall not be responsible for identifying patents for which a license may be required or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

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

140 Philips Road
Exton, PA 19341

Table of Contents

Figures

TABLES

1 INTRODUCTION

1.1 Overview

There are a variety of Adaptive Streaming wire formats. Some are based on an MPEG-2 Transport Stream container such as HLS (HTTP Live Streaming: Apple) and others on a fragmented MP4 container such as HSS (HTTP Smooth Streaming: Microsoft) and HDS (HTTP Dynamic Streaming: Adobe); whereas DASH (Dynamic Adaptive Streaming over HTTP: MPEG) supports both containers. While different, they utilize common video and audio compression formats; namely: ISO/IEC 14496-10 (AVC) and ISO/IEC 14496-3 (AAC). Additional audio formats, such as Dolby Digital Plus and DTS-HD, *may* also be supported by these or a subset of these Adaptive Bit Rate (ABR) formats.

In a unified ABR encoding and packaging system, video and audio data are encoded and conditioned for adaptive streaming purposes and the resultant elementary compressed access units are fed to one or more ABR packagers or encapsulators to be formatted into ABR-specific wire formats.

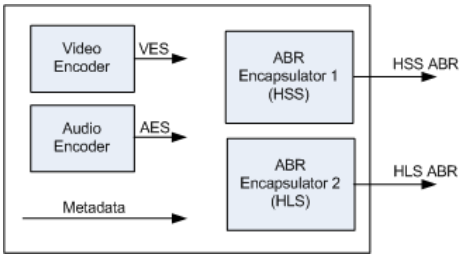


Figure 1 - Unified Transcoder/Packager

The Adaptive Transport Stream (ATS) format described in this specification allows for streaming/storage of adaptive streaming content originating as Transport Streams in a generic manner without restricting this to a particular adaptive streaming delivery technology (HSS/HLS/HDS). As Figure 2 illustrates, this allows for a separation of the transcoding process from the encapsulation process that produces ABR-specific formats.

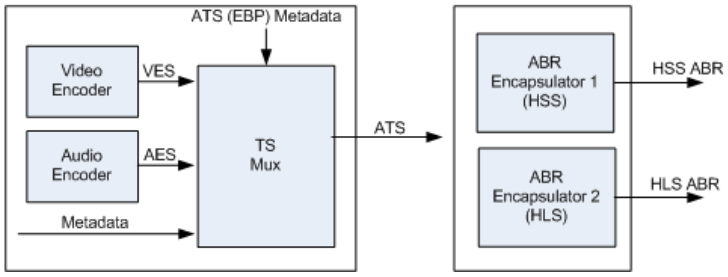


Figure 2 - Separate Transcoder/Packager with ATS between

This specification defines a fully **compliant** continuous single program MPEG-2 Transport Stream which follows the HRD model and provides markers in the stream through the use of PES frame encapsulated af_descriptors (adaptation field descriptors) [11]. These markers identify conditioned points in the stream that are virtual segments that can be partitioned into segments used for ABR applications [11]. Downstream encapsulation is expected to re-encapsulate an ATS, which *may* involve partial or complete de-encapsulation (demux) prior to encapsulating into a target ABR format. Since this downstream encapsulation does no re-encoding of the media data, the video and audio access units in the ATS need to be pre-conditioned for adaptive streaming purposes [in accordance with section 8 of this specification]. Additionally this specification defines af_descriptor metadata, Boundary Descriptor /Labeling Descriptor/Timeline Descriptor and collectively called Encoder Boundary Points (EBP) data [as defined in section 7.5], that are injected and carried in the transport stream layer to provide adaptive boundary information to downstream processing tasks. The EBP data provides a hinting mechanism for taking continuous streams conditioned for adaptive streaming and creating discrete chunks of decodable content with boundaries in one

component stream in the multiplex (Fragment) or across the multiplex (Segment). This specification also references additional PMT descriptors as specified in 13818-1 [11] that may be carried in the ATS for informative purposes to describe the various conditioning and boundary points used in the stream(s).Lastly, structural information on a related set of ATS streams *may* be carried through an ATS Source Description.

The EBP data contained within an ATS stream is carried as data descriptors of the public adaptation field (af_descriptors) of an MPEG-2 TS packet for video or audio and can be applied to each video packetized elementary stream (PES) and audio PES packet, resolved down in many cases to a single Video access unit (AU) or a group of audio AUs [11]. It contains a set of af_descriptors to indicate Boundary, Labeling, and Timeline information.

1.2 Purpose of Document

The purpose of this document is to define an ATS stream, the boundary points within it, both explicit and implicit, how boundary points map to various ABR formats such as HSS Fragments and HLS Segments (both in the video and audio domain), and the time stamps, durations and byte ranges of these chunks.

To that end, a significant portion of this document describes the basic requirements for adaptive video and audio conditioning. These sections detail conditioning considerations such as varying frame rates, advertising splice points and input loss handling.

1.3 Scope

This standard describes the requirements and constraints on a single program transport stream (SPTS) that allow it to be used as an Adaptive Transport Stream, including stream conditioning and signaling of segment boundary points. Typically, multiple ATSs will be generated from a single input and sent to a packager, recorder or other device. The EBP structure can be inserted at the time of encoding or added during the transcoding process. This specification does not describe how an ATS is stored or how it *may* be converted to target delivery formats.

This document describes the wrapping, chunking, and conditioning of packetized elementary streams carried over MPEG-2 TS. These elementary streams are codec independent and could carry AVC, HEVC, or even MPEG-2 video. Reference is made to SCTE 128 [3][33] to be compliant with MPEG-2 Systems layer constraints on the use of adaptation field public data.

These created ATS streams are then sent to a packager (also called a fragmentor or encapsulator) directly or stored to be sent to an packager upon request at a later time. Upon receiving such streams, an packager then processes these streams with EBP data and produces chunks according to the one or more adaptive streaming encapsulating technologies.

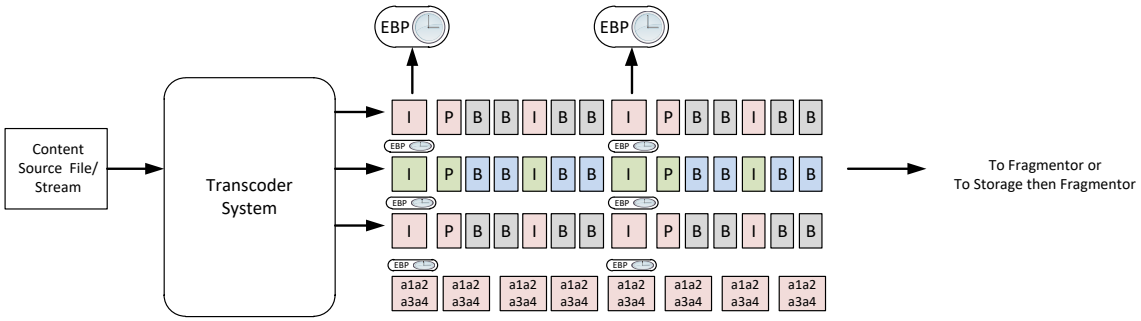


Figure 3 – ATS Streams with EBP data inserted at transcoder

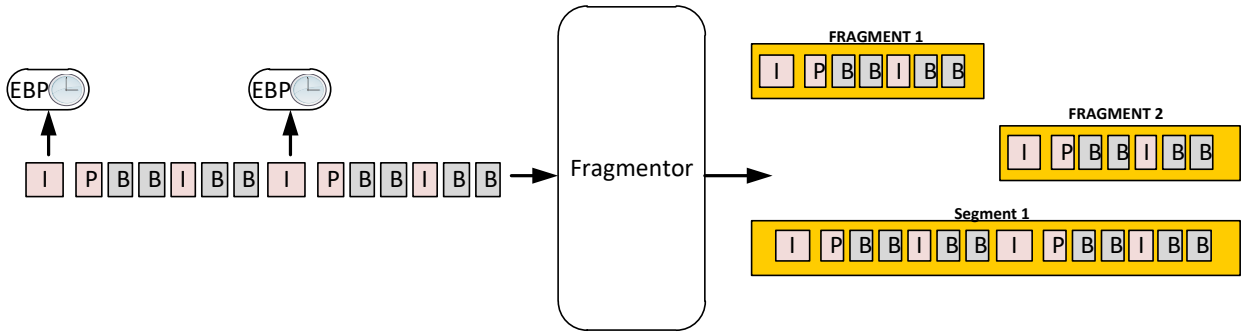


Figure 4 – Fragmentation of an ATS generated stream

In addition, this standard describes both the static and time-sensitive metadata that describes the ATS stream and where this information is located through EBP, PMT, SCTE 35, private metadata, and configuration information. The collection and aggregation of static metadata is defined in a metadata format, the ATS Source Description that describes the attributes of a set of ATSs. The purpose of the ATS Source Description is to provide information needed to configure a downstream device to package or record the set of ATSs and only carries information related to that task, for example, bitrates, codec parameters, etc. The ATS Source Description is based on the DASH MPD schema, but does not carry any time variant data such as Segment names or Periods. This specification does not describe how a device obtains an ATS Source Description.

1.4 Requirements

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

shall	This word or the adjective "***required***" means that the item is an absolute requirement of this specification.
shall not	This phrase means that the item is an absolute prohibition of this specification.
should	This word or the adjective "*recommended*" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighted before choosing a different course.
should not	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
may	This word or the adjective "*optional*" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

2 REFERENCES

2.1 Normative References

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

[1] IETF RFC 5905, Network Time Protocol Version 4: Protocol and Algorithm Specification, June 2010.

[2] ANSI/SCTE 54 2015, Digital Video Service Multiplex and Transport System Standard for Cable Television.

[3] ANSI/SCTE 128-2 2013, AVC Video Constraints for Cable Television Part 2 – Transport.

[4] ISO/IEC 14496-12:2015, Information technology - Coding of audio-visual objects - Part 12: ISO base media file format.

[5] ANSI/SCTE 193-1 2014, MPEG-4 AAC Family Audio System – Part 1 Coding Constraints for Cable Television.

[6] ANSI/ 35 2017, Digital Program Insertion Cueing Message for Cable.

[7] IETF STD 66 Uniform Resource Identifier (URI): Generic Syntax, January 2005.

[8] ANSI/SCTE 214-1 2016, MPEG DASH for IP-Based Cable Services Part1: MPD Constraints and Extensions.

[9] ANSI/SCTE 214-2 2016, MPEG DASH for IP-Based Cable Services Part 2: DASH/TS Profile.

[10] ANSI/SCTE 215-2 2015, HEVC Video Constraints for Cable Television Part 2- Transport.

[11] ISO/IEC 13818-1:2018, Information Technology- Generic coding of Moving picture and associated audio information—Part 1: Systems.

2.2 Informative References

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

[12] ITU-T Recommendation H.264 | ISO/IEC 14496-10, Information Technology - Coding of audio visual objects - Part 10: Advanced Video Coding.

[13] ISO/IEC 23091-2, Information Technology— Coding-independent code points – Part 2: Video.

[14] ISO/IEC 23009-1, Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats.

[15] DASH-IF Implementation Guidelines: Interoperability Points; Version 4.2, https://dashif.org/wp-content/uploads/2018/04/DASH-IF-IOP-v4.2-clean.pdf.

[16] Real-time Event Signaling and Management (ESAM) API, OC-SP-ESAM-API-I03-131025, October 25, 2013, Cable Television Laboratories, Inc.

[17] Adobe HTTP Dynamic Streaming, http://www.adobe.com/products/hds-dynamic-streaming.html.

[18] HTTP Live Streaming, Apple Inc., http://tools.ietf.org/html/draft-pantos-http-live-streaming-19 , April 4, 2016.

[19] Timed Metadata for HTTP Live Streaming https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/HTTP_Live_Streaming_Metadata_Spec/HTTP_Live_Streaming_Metadata_Spec.pdf.

[20] Smooth Streaming Transport Protocol, http://learn.iis.net/page.aspx/684/smooth-streaming-transport-protocol.

[21] [MS-SSTR]: Smooth Streaming Protocol, http://msdn.microsoft.com/en-us/library/ff469518.aspx.

[22] IETF RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.

[23] IETF RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.

[24] IETF RFC 7232: Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests.

[25] IETF RFC 7233: Hypertext Transfer Protocol (HTTP/1.1): Range Requests.

[26] IETF RFC 7234: Hypertext Transfer Protocol (HTTP/1.1): Caching.

[27] IETF RFC 7235: Hypertext Transfer Protocol (HTTP/1.1): Authentication.

[28] IETF BCP 47, Tags for Identifying Languages.

[29] IETF RFC 6381, The 'Codecs' and 'Profiles' Parameters for "Bucket" Media Types.

[30] ANSI/SCTE 104, Automation System to Compression System Communications Applications Program Interface (API).

[31] ETSI TR 101 154, Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream.

[32] ISO/IEC 13818-2, Information technology – Generic coding of moving pictures and associated audio information – Part 2: Video.

[33] ANSI/SCTE 128-1, AVC Video Constraints for Cable Television Part 1- Coding.

[34] ANSI/SCTE 193-2, MPEG-4 AAC Family Audio System - Part 2 Constraints for Carriage over MPEG-2 Transport.

[35] ANSI/SCTE 194-2, DTS-HD AUDIO SYSTEM- Part 2: Constraints for carriage over MPEG-2 Transport.

[36] Encoder Boundary Point Specification, http://www.cablelabs.com/wp-content/uploads/specdocs/OC-SP-EBP-I01-130118.pdf

[37] Adaptive Transport Stream Specification, http://www.cablelabs.com/wp-content/uploads/specdocs/OC-SP-ATS-I01-140214.pdf

[38] OpenCable Enhanced TV Application messaging Protocol Specification, OC-SP-ETV-AM1.0.1, http://cablelabs.com/specification/opencable-enhanced-tv-application-messaging-protocol-specification/

[39] OpenCable Enhanced TV Binary Interchange Format 1.0 Specification, OC-SP-ETV-BIF1.0.1, http://cablelabs.com/specification/opencable-enhanced-tv-binary-interchange-format-1-0-specification/

2.3 Reference Acquisition

- Internet Engineering Task Force (IETF) Secretariat, 48377 Fremont Blvd., Suite 117, Fremont, California 94538, USA, Phone: +1-510-492-4080, Fax: +1-510-492-4001, http://www.ietf.org
- ISO Central Secretariat: International Organization for Standardization (ISO), 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland; Internet: http://www.iso.ch/
- SCTE - Society of Cable Telecommunications Engineers Inc., 140 Philips Road, Exton, PA 19341 Phone: +1-610-363-6888 / +1-800-542-5040; Fax: +1-610-363-5898; http://www.scte.org/
- ETSI - http://www.etsi.org

3 TERMS AND DEFINITIONS

This standard uses the following terms:

Acquisition Time	An NTP-derived time which *may* be carried within an EBP data field to indicate the time at which the Transcoder/Encoder acquired this Access Unit (AU). Acquisition Time *may* be carried in EBP structures of linear media streams to identify time-based recording boundaries, and provide a record of actual content acquisition time. Acquisition Time has no normative use in packaged or VOD content.
Adaptation Streams	Set of interchangeable encoded versions of one or several media content components.
Adaptive Sync	ATS streams that are both Time Synced and Chunk Synced.
Adaptive Transport Stream	MPEG-2 continuous (virtually segmented) compliant single-program Transport Stream conditioned as defined in DVS 1196
AF Descriptor	One of a set of descriptor in the adaptation field as defined in MPEG-2 Systems specification e.g. TEMI [11]
ATS File	ATS stream stored as file. (AN ATS stream physically written into separate segments is a DASH/TS)
ATS Set	ATS Set is a group of ATS streams in different bit rate and resolution, but with same content and well aligned with EBP data.
ATS Source Description	XML document providing descriptive and retrieval information for one or more ATS streams. NOTE: ATS SD closely resembles a DASH MPD.
Boundary Descriptor **Boundary Structure** **Boundary Data**	af_descriptor to indicate accurate boundary points of PES frames. These often indicate first frame of a fragment and can be used by packagers to signal chunking of fixed or variable duration chunks. Structure of the Boundary af_descriptor Data in the boundary af_descriptor
Chunk	A discrete section of content that can be independently decoded, possibly given additional initialization information.
Chunk Boundary Point	A specialized Encoder Boundary Point that indicates the beginning of a chunk, and is a stream access point.
Chunk Sync	Chunk sync implies the identical AUs across representations at boundaries indicated by EBP.
Chunk Type	Refers to a specific partition that is contained in the ATS stream. Segment and fragment are examples of different types of chunks.
Conditioned Stream	A transcoded stream that contains independently decodable sections of content (e.g., Fragments and Segments). This stream can be sent to an packager to create fragments and segments in specific ABR formats.
DASH/TS	A profile of MPEG DASH which uses MPEG-TS segments.
Encoder	A subsystem that compresses digital media. The input can be uncompressed digital media or a mezzanine level packetized elementary stream, and output is a compressed stream for delivery to consumers in real-time or via storage media.

Encoder Boundary Point	An indicated point in the stream that is assigned to a PES packet containing one or more access units. Often indicates the frame of a chunk or segment.
EBP Structures	Collective af_descriptor data structures (Boundary Descriptor, Labeling Descriptor, Timeline Descriptor) in the public adaptation field of MPEG packet with the PES Header Information which boundary markers to create different size chunks through partitions, labeling of those chunks, and time information indicators. When referring to the collective structure, it is usually referring to data positioning within the stream.
EBP Data	Referring to the collective data contained in the EBP structure (boundary markers, labeling, timing) without the need to referring to each data descriptor within the stream.
Explicit EBP Stream	The elementary stream carries the EBP structure in the adaptation public data field associated with boundary point AU. Chunks containing an EBP structure in these types of streams can be called "explicit chunks".
Explicit Partition	A partition which references EBP structures carried on its own stream PID.
Fragment	A chunk that is aligned with boundaries in one component stream in the source multiplex. Fragment boundaries are typically explicit for each component. Smooth Streaming [20][21] is an example ABR format that uses fragments.
Identical AUs	Access Units in multiple encoded bitstreams that represent the same content period. These access units represent the same content period. The access units may represent different encodings of the same source content.
Implicit EBP Stream	The elementary stream does not carry the EBP structure in the adaptation public data field associated with a boundary point AU. It *may* depend on another elementary stream to reference this information. The EBP PMT descriptor *may* be used to indicate if the elementary stream is an implicit EBP stream. Chunks that do not contain an EBP structure in these types of streams can be called "implicit chunks".
Implicit Partition	A partition that references the EBP structure of an explicit partition carried on a different stream PID (and, possibly, in a different multiplex).
Labeling Descriptor	af_descriptor to indicate labeling of PES frames. These can contain UPID labels and identifiers.
Labeling Structure	Structure of the Labeling af_descriptor
Labeling Data	Data in the Labeling af_descriptor
Media Content Component	A single type of media such as audio, video, or text as defined in section 3.1.16 of DASH Standard [14]

Packager	Processes a conditioned continuous group of elementary streams to create specific ABR-format chunks of mixed or separated elementary streams that are stored in a file or transmitted. Each file is wrapped to be in one or more adaptive streaming formats. A packager does not normally perform any transcoding functions but depends on the conditioned stream to create those independently decodable sections. A packager can also be known as a fragmenter, encapsulator, or segmentor.
Partition	A set of continuous chunks within a media stream. A stream can be partitioned in several ways. For example, Partition A corresponds to 2-second chunks, while Partition B corresponds to 5-second chunks. A partition is represented by a series of boundary points across a group of elementary streams.
SAP type	Defines the properties of a Stream Access Point as specified in Annex I of ISO-BMFF [4] SAP Types 1 and 2 correspond to what is known in some coding schemes as a "Closed GOP random access point".
Segment	A chunk with boundaries aligned to include all component streams in the source multiplex across the target presentation time range. Segment boundaries are typically explicit for only one main component (video, for example), and other component boundaries are implicitly derived from this main component. A segment is typically used when packaging content in HLS.
Stream Access Point (SAP)	Enables random access into a media stream as defined in Annex I of ISO-BMFF [4].
Time Sync	Indicates identical AUs across representations have the same presentation time stamp.
Timeline Descriptor **Timeline Structure** **Timeline Data**	af_descriptor to indicate Timeline of PES frames. These can contain “wall clock” timeline of the stream Structure of the Timeline af_descriptor Data in the Timeline af_descriptor
Transcoder	A subsystem that converts a packetized elementary stream of one bitrate to one or more lower bitrate streams by changing coding parameters, including media resolution. In a broader sense, it can change from one codec format to another.
Virtual Segment	A section of an MPEG elementary stream that is defined by boundary af_descriptors [11]

4 ABBREVIATIONS AND ACRONYMS

This document uses the following abbreviations and acronyms.

AAC	Advanced Audio Coding
AAC-LC	Low Complexity AAC
ABR	Adaptive Bit Rate
ASC	Audio Specific Config
ATS	Adaptive Transport Stream
AU	Access Unit
AVC	Advanced Video Coding
DASH	Dynamic Adaptive Streaming over HTTP
DRC	Dynamic Range Control
DTS	Decoding Time Stamp
EBP	Encoder Boundary Point
ESAM	Event Signaling and Management
ENC	Encoder
EPT	Earliest Presentation Time
ES	Elementary Stream
GOP	Group of Pictures
HDS	HTTP Dynamic Streaming (Adobe) [17]
HLS	HTTP Live Stream [18]
HRD	Hypothetical Reference Decoder
HSS	HTTP Smooth Streaming (Microsoft)[20][21]
MBR	Multi-Bit Rate
MPD	Media Presentation Description
NTP	Network Time Protocol
PCR	Program Clock Reference
PID	Program Identifier or Packet Identifier per 13818-1[11].
PES	Packetized Elementary Stream
PMT	Program Map Table
POIS	Placement Opportunity Information System
PS	Parametric Stereo
PTS	Presentation Time Stamp
RAP	Random Access Point
SAP	Stream Access Point
SATS	Segmented Adaptive Transport Stream
SBR	Spectral Band Replication
SD	Source Description
SDI	Serial Digital Interface
SPTS	Single Program Transport Stream
TS	Transport Stream
VES	Video Elementary Stream
VOD	Video on Demand

5 ATS METADATA OVERVIEW

ATS Metadata refers to a description of the ATS bitstream. Metadata can be categorized as:

- static metadata, where the description remains the same over the entire bitstream; for example, structural metadata describing the number of ATS streams and their relationships
- time-variant metadata, where the description can vary and change at different points of relative time in the bitstream, (either stream or wall clock time), for example EBP structures describing chunk boundaries
- event metadata, such as ad insertion points or program start/end.

ATS supports several mechanisms to carry this metadata. In the elementary stream, metadata can be carried through the EBP structure, located in its transport MPEG packet header, which can convey time variant metadata such as location of chunk boundaries in the stream, types of chunks, and chunk labels as well as wall-clock media time at chunk boundaries. In the transport stream, the PMT descriptors can describe metadata at the transport level and elementary stream level and can describe PID, stream type (video descriptor, audio descriptor, captioning, language, bandwidth), and supported chunk cadences through the EBP PMT descriptor. The PMT information is usually static until a major event that changes the media experience occurs in the stream. Lastly, event metadata can be conveyed through the SCTE 35 descriptors and triggers (see Section 8.5) or through a private metadata mechanism on a separate pid (see Section 7.14.1). The information carried is used to indicate alternate content such as ad insertion, campaigns, and blackout events. Additionally, static metadata is carried out of band or alternatively inband in the ATS Source Description (see Section 0).

6 ATS SOURCE DESCRIPTION

6.1 Introduction

An ATS Source Description is used to describe the properties of a set of related ATSs. The primary use of the ATS Source Description is to provide information about the location of the source ATSs, for example a set of multicast addresses for a group of ATSs that are intended to be combined into a DASH AdaptationSet. It carries static metadata in addition to content source location such as program service metadata, MPEG-2 TS structure and attributes, etc. This metadata *may* be used by downstream devices to configure packaging or recording tasks.

The ATS Source Description follows the MPD schema of MPEG DASH [14] with enhancements to describe sources that are streams rather than files.

The information conveyed by an ATS Source Description *should* be created while or before the media transcoding task is started. ATS Source Description *may* be updated at a programming boundary. The publication and delivery of ATS Source Description is beyond the scope of this specification.

All restrictions and extensions defined in SCTE 214-1 [8] and SCTE 214-2[9] ***shall*** apply.

Extensions and restrictions specified in section 0 of this apply to an MPD otherwise compliant to SCTE 214-1 [8] and SCTE 214-2 [9]. ATS source description is thus based on the DASH/TS profile as defined in SCTE 214-1[8] and SCTE 214-2[9] but is not a proper subset of it. Hence there *should* be no expectation on behalf of DASH/TS client to be able to present it.

6.2 Identification

ATS Source Description document is a DASH MPD document with MPD@profiles value "urn:scte:ats:source:2015".

6.3 Non-HTTP URIs

6.3.1 General

ISO/IEC 23009-1 limits BaseURL values to HTTP(S) URLs only.

ATS Source Description require support for `file://` and `udp://` schemes in addition to HTTP(S).

When multiple URI schemes are used simultaneously, BaseURL ***shall not*** be present at any level except for Representation.

6.3.2 MPEG-2 TS over UDP

A URI with scheme value "udp " identifies an MPEG-2 TS packets embedded into UDP.

The udp:// scheme syntax is as follows:
`udp://[<localaddress>@]<destination|multicastgroup>[:<port>]`

Each UDP packet typically contains 1 to 7 complete 188 byte TS packets with no additional headers or footers. This is consistent with the definition of the scheme given by http://www.iana.org/assignments/uri-schemes/prov/udp.

NOTE: `local address` corresponds to the multicast source

Each UDP packet payload contains only complete MPEG-2 TS packets

6.3.3 File

File URI format is defined in Uniform Resource Identifier (URI): Generic Syntax [7].

6.4 Anchor addressing

6.4.1 Introduction

The fragment part of a URI used in this specification is one or more ampersand-separated key-value pairs.

Each key-value pair can be viewed as a filter – a client is supposed only to include subset of MPEG-2 TS identified by a combination of parameters in the fragment part of the URI.

6.4.2 MPEG-2 TS Addressing

The following parameters *may* be used for any resource consisting of a sequence of MPEG-2 TS packets

Table 1- Parameters for MPEG-2 TS Anchors

Key	Value	Semantics
`pid`	PID value (decimal)	Packets with PID value specified in this parameter ***shall*** be kept.
`program`	Program number (decimal)	Packets with PID values associated with this program ***shall*** be kept. In context of this parameter "associated" PIDs include PIDs 0..4, PMT PID (only for the relevant program) and all PIDs mentioned in the PMT.

Packet Identifiers (PID) in the MPEG-2 TS indicate Video, Audio, PAT, or PMT information. Additional information a PID could indicate is SCTE 35, EBIF (ETV, EISS) data.

Anchoring Example:

Video: www.example.com/stream.ts#pid=481

6.4.3 EBP Addressing

The following parameters can be used to address parts of adaptive transport stream. Note that they can be used in combination with MPEG-2 TS parameters defined in 6.4.2.

Table 2- Parameters for EBP Anchors

Key	Value	Semantics
`ebp_pid`	PID carrying EBP structures for a selected partition	***Shall*** be used if EBP parameters are used and more than one EBP PID is present, optional otherwise.
`partition_id`	ID of EBP partition addressed	ID of partition referred to. This parameter ***shall*** appear any time a subset of ATS is addressed.
`Sequence_number`	number (decimal)	Chunk number, as specified in the EBP structure. Needed for use with $Number$ template addressing. Corresponds to a value of sequence number in Boundary Descriptor
`abs_time`	EPT of a chunk (decimal)	Chunk time, as specified in the EBP structure. Corresponds to media time in TEMI. (Number of seconds according to EPOCH time format. Use W3C media Fragment+ add reference http://www.w3.org/2008/WebVideo/Fragments/WD-media-fragments-spec/#naming-time)

6.5 Carriage of ATS Source Description in MPEG-2 Transport Streams

If an ATS Source Description is present in the MPEG-2 TS, then a DASH MPD Update event (as defined in

ISO/IEC 23009-1:2014 COR1 5.10.4.4) ***shall*** be used for inband transport of source description. Use of this event is allowed if

(a) MPD@type='dynamic',

(b) there is an AdaptationSet.InbandEventStream element announcing it, and

(c) the event is aligned (as defined in ISO/IEC 23009-1:2014 [14]).

Only complete event messages *may* be contained between two EBP structures.

NOTE: This means that devices processing ATS streams, such as an packager, will scan the input stream for PID 0x0004 (used per ISO/IEC 13818-1 for carriage of DASH events [11]), and look for event message with URN "urn:mpeg:dash:event:2012".

This event will span over multiple transport stream packets which are not necessarily consecutive, however the reading a single chunk is sufficient to recover the complete message.

7 ATS SPECIFICS

7.1 Format

The ATS (Adaptive Transport Stream) format allows for streaming/storage of adaptive streaming content in a generic manner without restricting this to a particular adaptive streaming technology (HSS/HLS/HDS). The EBP structure plays a significant role of signaling the stream/file characteristics to allow for adjusting the stream/file to a specific adaptive streaming technology at the downstream packager/encapsulator device. For a set of MBR streams to be in an ATS format, there are several constraints.

The TS structure of each stream ***shall*** comply with 13818-1 [11] including the HRD buffering. For AVC/H.264, each TS stream ***shall*** follow SCTE 128-2 [3]. As a result, the video and audio AUs are skewed with audio following video.

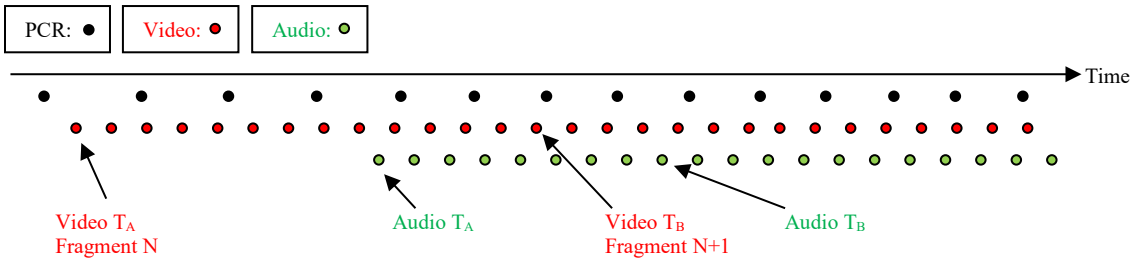


Figure 5 - Example of Audio and Video in a TS stream following HRD model and SCTE 128

7.2 General Makeup

An ATS is a single-program compliant MPEG-2 Transport Stream following restrictions as indicated in SCTE 128-2 [3] or SCTE 215-2 [10]. There are multiple interleaving options allowable as described below; however, all devices that process ATS streams ***should*** support Method 1 interleaving option.

Video and Audio data ***shall*** have been conditioned for ABR purposes as described in the previous sections. However, an interleaved ATS is not segmented like SATS. Chunk boundaries on video data and optionally audio data ***shall*** be marked in the transport stream by using the Encoder Boundary Points (EBP) structure.

7.3 Interleaving options

This section describes the base interleave options for an ATS stream.

7.3.1 Method 1 – All representations carry all audio streams

An ATS ***shall*** contain a single video representation and all audio representations intended to be used with the video representation. While different audio representations (e.g., different languages, different encoding formats) do not have to be in time sync with one another, the same audio representation (language and bitrate) across all ATS ***shall*** be in time sync.

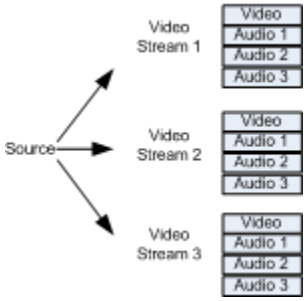


Figure 6 - Interleave Method 1

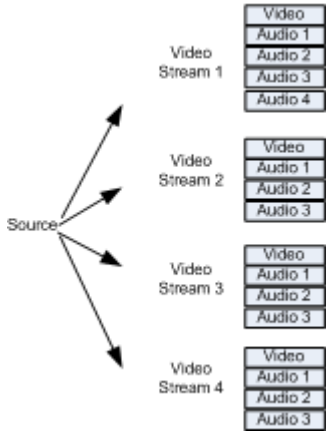


Figure 7 - Interleave Method 1 with Audio 4 used by one Video representation

In Figure 7, audio 4 is only needed for Stream 1. The end use of Streams 2-4 does not use audio 4. Thus according to Method 1, audio 4 does not need to be present in these other streams.

7.3.2 Method 2 – All representations carry a common subset of audio streams

An ATS ***shall*** contain a single video representation and ***shall*** contain at least one common audio representation. While different audio representations do not have to be in time sync with one another, if the same audio representation is present in more than one ATS, they ***shall*** be in time sync with each other.

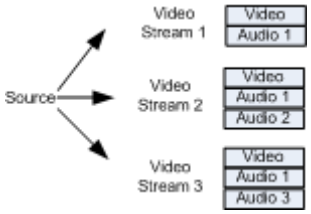


Figure 8 - Interleave Method 2

7.3.3 Method 3 - Audio carried in separate Representations

An ATS *may* contain a single video representation and/or one or more audio representations. Thus, video-only and audio-only ATS streams are supported. While different audio representations do not have to be in time sync with one another, if the same audio representation is present in more than one ATS, they ***shall*** be in time sync with each other. This is the most efficient bandwidth and storage-wise, as duplication is gated solely on redundancy needs. Non-interleaved encapsulation formats have no need for interleaved ATS assets.

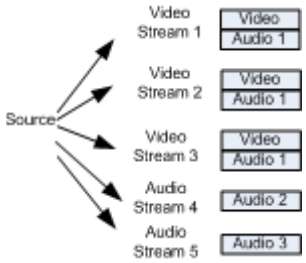


Figure 9 - Interleave Method 3, Hybrid of Method 2

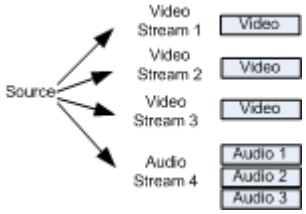


Figure 10 - Another Interleave Method 3 Embodiment

Video-only streams of Method 3 ***shall*** be playable and include PCR on the video PID. Audio-only streams of Method 3 ***shall*** include a PCR.

7.4 Late Binding Packaging Implications

Each stream in Method 1 contains all audio representations intended to be used with the video representation. This is not the case with Methods 2 and 3, and thus downstream encapsulation/packaging *may* need to source multiple source streams in order to bind the desired audio(s) to the desired video component. In the late binding case, there is a potential of multiple PCR tracks that could be associated with different elementary streams due to the need to create compliant transport streams. The PCR track associated with the Main Elementary Stream, usually Video, ***shall*** be the PCR track used for the bound package.

7.5 Encoder Boundary Point (EBP)

The EBP data contained within an ATS stream is carried as data descriptors in the public adaptation field of an MPEG-2 TS packet for video or audio. It can be applied to each video packetized elementary stream (PES) and audio PES packet, resolved down in many cases to a single Video or Audio access unit (AU) or a group of audio AUs. The EBP data contains a set of af_descriptors to indicate Boundary, Labeling, and Timeline information.

There are several ways to refer to an EBP:

Encoder Boundary Point- An indicated point in the stream that is assigned to a PES packet containing one or more access units. Often indicates the frame of a chunk or segment.

EBP Structures- This refers to a collective set of af_descriptor data structures (Boundary Descriptor, Labeling Descriptor, Timeline Descriptor) that is located in the public adaptation field of an MPEG packet that contains the PES Header Information. These structures indicate within the stream boundary

markers to create different size chunks through partitions, labeling of those chunks, and time information indicators. When referring to the collective structure, it is usually referring to data positioning information within the stream.

EBP Data- Referring to the collective data contained in the EBP structure (boundary markers, labeling, timing) without the need to referring to each data descriptor within the stream.

7.5.1 EBP AF_Descriptor Tags

The af_descriptors are public adaptation field data descriptors that have been accepted as part of the MPEG-2 TS systems specification [11]. The descriptors are part of the MPEG-2 TS packets that indicate the start of a PES frame. The information conveyed in these descriptors can indicate boundary point, labeling, and timeline information that are used by packagers to prepare MPEG-2 TS streams for ABR delivery. Collectively the information is called EBP Data and requires the use of the Boundary Descriptor, Labeling Descriptor, and Timeline Descriptor.

***Table 3** – AF Descriptor Tags*

AF Descriptor Tag	Identification
0x00-0x03	Rec. ITU-T H.222.0 | ISO/IEC 13818-1 Reserved
0x04	Timeline Descriptor
0x0B	Boundary Descriptor
0x0C	Labeling Descriptor

7.5.2 EBP Structure Placement in MPEG-2 TS Elementary Streams

The Encoder Boundary Point Structure is a signaling mechanism for Boundary Descriptor, Labeling Descriptor, and Timeline Descriptor information in the public field of the adaptation field of an MPEG-2 TS packet for video or audio. This structure can be applied to each video PES and audio PES packet although typically it only exists on chunk boundaries. It indicates a hinting mechanism for taking the continuous stream conditioned for adaptive streaming (the ATS) and creating ABR format specific chunks (fragments, segments, etc.) from them. There are bits in the EBP structure to indicate discretely different chunk boundaries, for example, fragment boundaries differing from segment boundaries. Additional boundary bits are available to indicate additional boundaries and thus define different partitions. This is to facilitate the different durations of ABR format specific chunks and to allow chunks to occur on different boundaries. A set of continuous chunks for a given media element defines a partition.

In the video domain, however, boundaries of different formats are typically subsets of other boundaries to minimize the bit cost associated with boundary frames. For example, HSS and HDS *may* both have 2-second chunk durations and are ideally aligned. An HLS chunk *may* be 6 seconds in duration and align ideally with HDS and HSS chunk boundaries.

When any elementary stream contains explicit EBP structures and carries PCR, a PCR ***shall*** be inserted in any transport packet carrying a boundary EBP structure according to access unit constraints as indicated by NOTE 2 of section 6.4.2.2 of AVC[12].

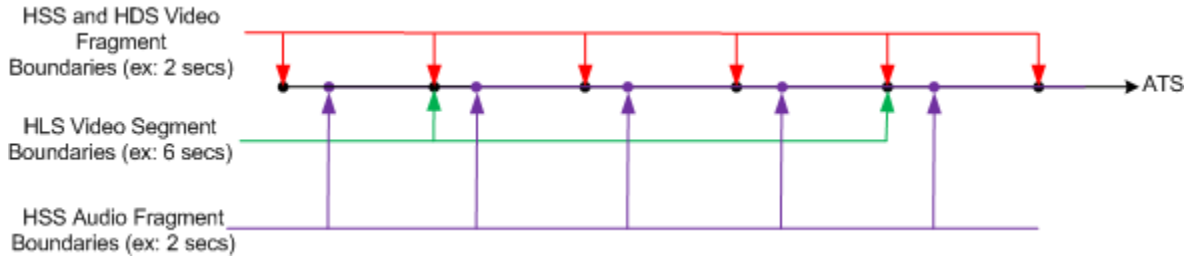


Figure 11 - Boundary Points

When applied to a video stream, a boundary point is assigned to a PES packet containing one or more access units and can be assigned independently from GOP structure (e.g. SAP_type, IDR, P, B, audio AU), but in most cases is loosely bound to the beginning of a GOP

The boundary indicator can also be applied to the one or multiple audio streams that are contained in the same program and PES Frame. Among other purposes, this mechanism *should* be used to indicate chunk boundary points in an adaptive streaming context that can be signaled via the transport stream. At the packager, the information in the EBP data structure is used as a hint for processing.

The EBP data ***shall*** be carried in adaptation field public data descriptors.

- When present, the EBP structure ***should*** be carried in the transport packet that has the Payload Unit Start Indicator set for the PES packet to which it is assigned. The EBP structure should be contained in the first MPEG packet with the payload start indicator and should not exceed 140 bytes. If the EBP structure extends beyond a single MPEG packet, the boundary af_descriptor ***shall*** be carried in the transport packet that has the Payload Unit Start Indicator.
- More than one EBP structure ***shall not*** be sent with each PES packet header.

For Video:

- Each video access unit ***shall*** be completely contained within one PES packet, and the first byte of the PES packet payload ***shall*** be the first byte of the video access unit.
- The video AU that is the start of the chunk ***shall*** be indicated through the EBP structure signaling mechanism.
- The video AU with an EBP structure that starts the chunk *should* be conditioned by a transcoder or encoder to be an intra-coded AU that meets the requirements of SAP type 1 or 2, though it *may* not be the only intra-coded AU in the chunk. If the video AU starts with a closed GOP (most frequent case), then the EBP_SAP_type in the EBP structure *should not* be used to indicate SAP type. For all other types of GOP structures, if EBP structure is present, the EBP_SAP_type ***shall*** be used to indicate SAP type.

For Audio:

- The PES packet *may* contain multiple audio AUs.
- The PES packet ***shall*** contain an integral number of audio AUs.
- More than one EBP structure ***shall not*** be sent with each PES packet header.
- Audio AU with an explicit EBP structure that indicates the start of the audio chunk ***shall*** start the PES packet.
- The audio AU that indicates the start of the fragment *may* be indicated through the EBP structure signaling mechanism.
- The audio AU that starts the chunk *should* be conditioned by a transcoder or encoder to be an intra-coded AU that meets the requirements of SAP type 1 or 2, though it *may* not be the only intra-coded AU in the chunk.
- In absence of audio EBP structure (i.e., an implicit audio EBP), the first audio AU of a chunk ***shall*** be the first AU where the presentation time is greater than or equal to presentation time of the corresponding video boundary point.
- Audio AU ***shall not*** lag the aligned video AU by greater than 3 seconds in the MPEG-2 transport stream.

NOTE: In MPEG-2 TS, it is possible for an MPEG transport packet to contain only payload, or only adaptation field with no payload, or both. As EBP data is carried within the adaptation field, it is possible to carry it in the latter two packet types. If there is no payload in a packet that carries the EBP structure, the structure is interpreted as if it were carried in the nearest following MPEG-2 TS packet with the same PID.

7.5.3 Usage of Timeline AF Descriptor Structure

This specification uses the timeline descriptor in the af_descriptor form to convey UTC time associated with the virtual segments. The timeline descriptor is described in the MPEG Systems specification [11]. Some additional details on connecting the time source to the timeline descriptor (TEMI).

- The clock source to derive the timeline af_descriptor used in linear EBP data is prioritized in the following order: 1) SCTE 35 time descriptor on source signal, 2) TEMI on the source signal, or 3) local UTC source (e.g. an NTP server) interfaced to transcoding device if 1 or 2 do not exist.
- The time source to derive the timeline af_descriptor used in VoD EBP data is generated through a Normal Playback Timeline

Table 4- TEMI Timeline Descriptor
(See Table U-8 TEMI Timeline Descriptor [11])

When N**T**P timestamps (acquisition time) are inserted into EBP structures for linear services or recordings of linear services, the transcoder or encoder ***shall*** acquire UTC time such as NTP originating from an SCTE 35 time descriptor or from a low stratum server (lower than stratum 3), such as a GPS source or other sources with equivalent accuracy (e.g. PTP). When NTP timestamps are inserted into EBP structures for VoD services, the transcoder or encoder shall follow a Normal Playtime (NPT) Timeline.

NTP inserted time ***shall*** use the NTP 64-bit timestamp format NTP [1] a 32-bit unsigned seconds field counting from the prime epoch of 1-Jan-1900 00:00:00 and a 32-bit fraction field.

TEMI parameters that need to be set for use with NTP timestamps are the following:

- ***has_timestamp*** set to "10"
- ***has_ntp*** set to "1"
- ***has_ptp*** set to "0"
- ***has_timecode*** set to "00"

7.5.3.1 Constraints on Carriage of Acquisition Time

The value placed in the acquisition time field is derived by the transcoder or other device from an external time reference and ***shall*** meet the following requirements:

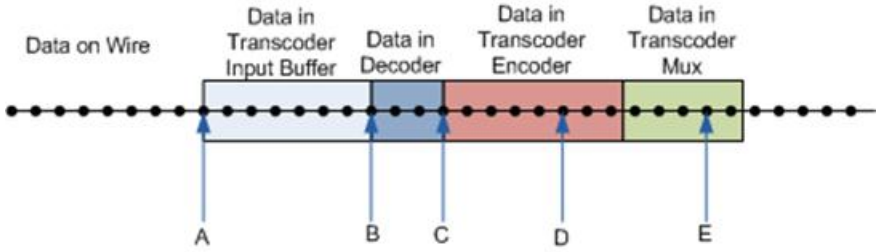
- EBP data acquisition time at the start of chunk ***shall*** be the same across all streams of the MBR being generated from the same source, within the sampling accuracy limit described below.
- EBP data acquisition time ***shall*** be continuous even if PTS of the stream is in rollover mode or has a time discontinuity.
- EBP data acquisition time among different transcoders for the same AU in the same input source ***shall not*** differ by more than +/- 300ms.
- When acquisition times are included in:
  - Chunk boundary point partition EBP structures, they ***shall*** be included in every chunk boundary point partition EBP structure.
  - Other types of EBP structures, they ***shall*** be included at a frequency of at least every 2 seconds.
- The difference of NTP value between two EBP points ***shall*** reflect the temporal distance between those boundary points with jitter no more than +/- 10 ms. For example, if a fragment chunk is two seconds, it is expected that the difference of NTP at the beginning of this fragment and the next fragment will be two seconds with jitter no more than +/- 10 ms.
- In the case where the incoming streams (arriving at the transcoder) *already* contain EBP structures carrying timestamps, the transcoder ***shall*** preserve the incoming timeline. This means that:
  - When outgoing EBP structures remain on the same AU as the incoming, they have the same EBP time value.
  - When outgoing EBP structures do not remain on the same AU (e.g., due to re-encoding), they have EBP data time values derived from the incoming timeline.

- In the case where the transcoder was using the incoming timeline (EBP structures arriving from upstream with acquisition times), but then incoming streams stopped including acquisition times, the transcoder ***shall*** continue to preserve the incoming timeline.
- In the case where the transcoder was using the local timeline (EBP structures arriving from upstream without acquisition times), but then incoming streams started including acquisition times, the transcoder ***shall*** continue to preserve the local timeline.

Generation of adaptive transport streams *may* introduce varying delays among the different formats. In order to provide a consistent value of acquisition time among the different formats and different transcoder implementations, this standard describes where the time ***shall*** be sampled in terms of reference points in a nominal transcoder model that consists of:

- An input buffer to remove network jitter
- A decoder that conforms to the T-STD buffer model
- Video processing and encoding
- Construction of the output multiplex

Figure 12 illustrates these reference points for the nominal transcoder model. It is required that NTP value for each boundary point represents the time at point C when the corresponding access unit exits the decoder buffer of a decoder that conforms to the T-STD buffer model described in 13818-1 [11]. This allows acquisition times to be based on the presentation order of the access units.



A= enters transcoder dejitter buffer, B= AU enters decoder buffer, C=AU exits decoder output buffer, D= video processing and encoding, E=output multiplexer

Figure 12 - Candidate Reference Time Sample Points in Signal Chain in Nominal Transcoder

Transcoders are required to use time values derived from NTP carried in the source signal or an internal clock that is synchronized to NTP or timesource of equivalent accuracy (e.g. PTP). Implementations *may* sample NTP at different points in the signal chain, but they ***shall*** compensate accordingly to reflect time at point C. If the implementation does not sample NTP for each access unit, it ***shall*** derive an interpolated value for each access unit.

Transcoder implementations *may* introduce some amount of dejitter delay between points A and B in the figure above. If a transcoder introduces more than 300 ms of dejitter delay, it ***shall*** adjust the value derived at point C to compensate for the average delay between points A and B.

Derivation of Acquisition Time (Informative) of this standard contains a more detailed discussion of the derivation of acquisition time.

If the transcoder or other processing device is unable to lock to the external reference, it ***shall*** follow NTP [1]. Alternatively, the source encoder *may* insert a time value in an EBP structure on pre-designated video frames during creation of the original content stream.

7.5.4 Boundary AF Descriptor Structure

(see Section U.3.11 Boundary Descriptor [11])

7.5.5 Boundary AF Descriptor Semantics

(see section U.3.12 Semantic definition of fields in Boundary Descriptor [11])

7.5.6 Labeling AF Descriptor Structure

(see U.3.13 Labeling Descriptor [11])

7.5.7 Labeling AF Descriptor Semantics

(see U.3.14 Semantic definition of fields in Labeling Descriptor [11])

7.5.8 Usage of Labeling Descriptor in MPEG-2 TS

The labeling descriptor *may* be used in any PES frame in the MPEG-2 TS and can be used to describe that frame whether it represents a boundary point (part of the EBP data) of a segment or independently placed on any PES frame. The labeling descriptor can carry more multiple descriptions through the label_type parameter. If the stream is part of an MBR set, then the construct ***shall*** be aligned across the MBR set of streams.

In this specification, the purpose for the labeling descriptor is to provide an informational label(s) for the boundary point as indicated in the EBP Data which is most likely a boundary splice point created to indicate an SCTE 35 segmentation_type_id or segment_upid_type [6].

7.5.8.1 Use of segmentation_type_id in the Labeling Descriptor

The label_type format for segmentation_type_id is a byte field bounded from 0-255 in decimal format or 0x00-0xFF in hexadecimal format as listed in Table 9-8 of the SCTE 35 specification [6] but with an added offset of 0x100 (i.e. values may range from 0x100 to 0x1FF).

Note: label_length_code should be "000" since there is no byte payload (label_byte) associated with the label.

The label_type *may* also be used to label an alternate set of boundary points (other than the default set). This *may* be useful for alternate sets of downstream devices reading the EBP structures (but following the alternate sets) or providing a repeated counter.

SCTE splice indicators are inserted into the stream several seconds before the actual splice insertion point. The PTS needs to be determined in order to discover the exact splice point in the stream. Alternatively label_type in the labeling descriptor placed at the beginning of each chunk can be used by packagers to easily identify this point through identifying the segmentation_type_id of the chunk. In this manner, the frame accurate splice point starts a new fragment and an event message in a manifest can indicate the splice beforehand. With the use of segmentation_type_ids, splice points can be used to identify advertisement points, programs, chapters, and blackouts [6].

7.5.8.2 Use of segment_upid_type in the Labeling Descriptor

The label_type parameter can also carry a UPID structure through the segmentation_upid_type as described in Table 9-7 of the SCTE 35 specification [6]. The label type format for segmentation_upid_type is a byte field bounded from 0-255 in decimal format or 0x00-0xFF in hexadecimal format as listed in Table 9-7 of the SCTE 35 specification [6] but with an addedoffset of 0x200 (i.e. values may range from 0x200 to 0x2FF. The segmentation_upid_types carries several types of registered identifiers which can include Ad-ID, EIDR, URN or private identifier. If multiple identifiers are needed to be carried then it should be looped. Using a label_length_code of "7" can be used to explicitly indicate the length of the label_type using the label_length parameter.

Note: The use of the MID segmentation_upid_type is discouraged.

This allows a mechanism for a content identifier to be carried with an MPEG-2 TS stream and passed to other structures such as MPD and ISOBMFF boxes as the content delivery system is transformed. The content identifier can also be used to validate content to a service or guide. These identifiers in the stream can be mapped for example into a DASH Manifest Period@assetIdentfier, MPD@programinformation attributes or alternativeID, utc-time supplemental properties.

7.6 Partitions

Media, such as audio and video, within an ATS is conditioned as described given the considerations in Section 8. An ATS is thus a conditioned stream containing independently decodable sections or **chunks** of media. Being a compliant MPEG-2 Transport Stream, there is no physical (packet) alignment within an ATS of discrete media PIDs as a result of this conditioning. For example, the audio chunk from time TA to TB is likely skewed from the video chunk for the same time period.

Media *may* be conditioned to target multiple end ABR formats. The duration of chunks across formats *may* vary, and therefore a boundary for one chunk of media *may* not be a boundary for a chunk targeting a different duration. An ATS contains one or more **partitions** where each partition defines a set of continuous chunks. For example, one video partition *may* nominally contain chunks of 2-second durations and another video partition *may* nominally contain chunks of 10-second durations. Target ABR formats with identical partitioning duration requirements *may* share a single partition; for example, HSS and HDS with 2-second chunk durations *may* use the same video partition. However, it is allowable to have multiple identical partitions; e.g., a video partition for HSS and a video partition for HDS, each with exactly the same chunk boundaries.

The chunk boundaries within a partition are delineated in EBP structure as described in Section 7.5. For a given PID, a partition can be explicit or implicit. An explicit partition is one which is used in the EBP structures carried on this PID, while an implicit partition is a reference to an explicit partition carried on a different PID (and, possibly, in a different multiplex). Unique video partitions are required to be explicit. A partition *may* be implicit and reference or derive boundaries for its partition media from another partition. HLS audio partitions are typically implicit with the boundaries of audio chunks derived from a video partition.

A typical application of boundary partitions would be to associate one or more ABR formats with a unique' partition_id.

The following reserved values of boundary partition_id are defined:

- partition_id=0 (default)

It is possible to use more partitions than defined above;partitions 1-7 can be used for any purpose.

All non- chunk boundary EBP structures always implicitly belong to partition 0; however, an implementer *may* choose not to use partition 0 and use instead an extended partition for non-boundary use.

The purpose and frequency of a partition are specified in the EBP or virtual segmentation PMT descriptor as defined in section 7.7.1. For boundary partitions, buffering requirements can be derived from the optional max_bitrate_descriptor.

7.7 PMT Descriptors

To describe partitions, an ATS *should* include PMT descriptor data for each elementary stream with explicit and implicit partitions. Such information *may* indicate that, for example, there are two video partitions and three audio partitions with one explicit audio partition and the other two audio partitions are implicitly derived from each of the video partitions. An ATS stream *should* carry the PMT descriptors as described in 7.7.1. If the stream has closed-captioning service, it *should* be signaled as mandated by SCTE 128-2 [3].

An ATS stream *may* also carry the following PMT descriptors:

1) Audio Descriptor for AC-3, E-AC-3, AAC [34] and others.

7.7.1 PMT Descriptors related to EBP Data

7.7.1.1 Descriptors

The PMT structure *should* include one or more descriptors to signal the presence of EBP data in the stream to assist a downstream device in quickly processing a stream that *may* have EBP data information.

7.7.1.2 maximum_bitrate_descriptor

The PMT structure *should* include a maximum_bitrate_descriptor() at the program level and video and audio elementary stream level. This descriptor is specified in 13818-1 [11].

Note: If ATS source description is used, the @bandwidth value on both representation and SubRepresentation level takes precedent over the PMT descriptor on program and elementary stream respectively.

7.7.1.3 Virtual Segmentation Descriptor

(see section 2.6.120 Virtual segmentation descriptor [11])

7.7.1.4 Semantics

7.8 Use of Private Adaptation Descriptors

Prior to the use of public af_descriptors, EBP data was carried in the private adaptation field as delineated in the CableLabs (CL) ATS and EBP specifications [36][37]. The private adaptation field approach to carrying CL-EBP can still co-exist with the newer public af_descriptor approaches which will help with transition strategies. By injecting both types of EBP data formats into the encoder/transcoder output streams, Packagers can transition to the newer EBP data approach at a different pace than transcoder transition strategies. This can be especially helpful to create a smooth operational transition plan where factors of region, asset age, and local network infrastructure need to be considered.

7.9 Video Boundary Points

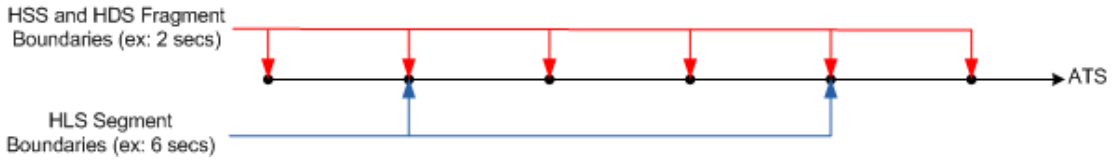


Figure 13 - Boundary Spec Indicating Fragment and Segment Boundaries

Figure 13 illustrates a typical 2-second fragment-chunk (for HSS/HDS purposes) and a 6-second segment-chunk (for HLS purposes). The points on the timeline are video access units and ***shall*** be marked with an EBP structure as defined in section 7.5. The video frame at the boundary point ***shall*** be PES aligned as indicated by SCTE 128-2 [3] or SCTE 215-2 [10].

Per Section 8.6, conditioning video at ad boundaries, both OUT and IN points, ***shall*** occur. For ATS streams with a defined set of segmentation points, a segmentation boundary point ***shall*** be provided at these locations. For HSS/HDS, these technologies have the option to do splicing by manifest manipulation or by client-side dual decoding. In the case of client-side ad insertion systems, it is possible for splicing to occur mid-fragment. A fragmentation boundary point *should* be provided at these locations.it should be noted that audio can lag video , but at an ad splice point both video and audio need to be conditioned. This is illustrated in Figure 14.

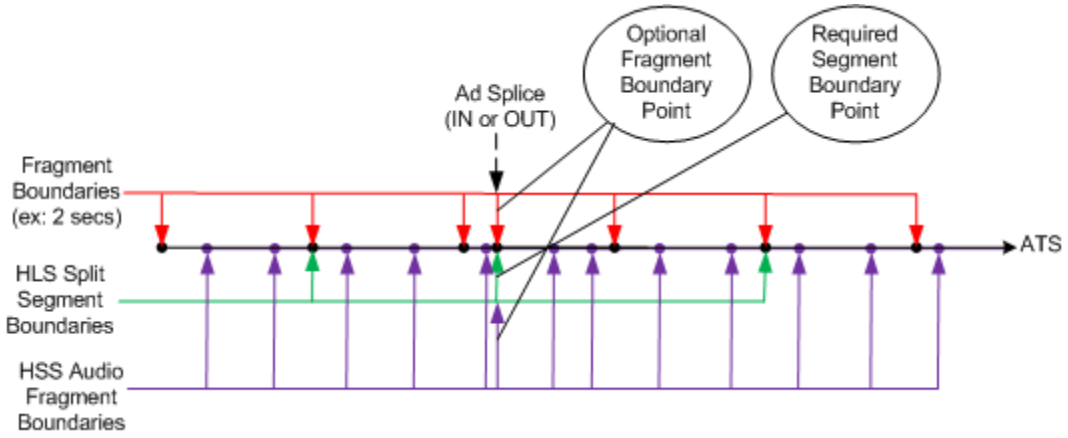


Figure 14 - Boundary Points at Ad Boundaries

7.10 Audio Boundary Points

For HLS and HDS, audio boundaries can be derived from the video boundaries - they have approximately the same start time and end time. In this specification the first audio AU of a derived chunk is the first AU where the presentation time is >= presentation time of the video boundary point for that chunk as defined in the audio portion of section 7.5.2 . This works well for both interleaved and non-interleaved HLS and for HDS. In Figure 15, the yellow audio HLS segment boundaries do not require physical EBP structures. Furthermore, there is no requirement in an ATS to align the physical location of audio packets to some relationship of video for segmentation purposes. This segmentation task is the responsibility of downstream encapsulation processes. The same applies to HDS fragments as shown in Figure 16.

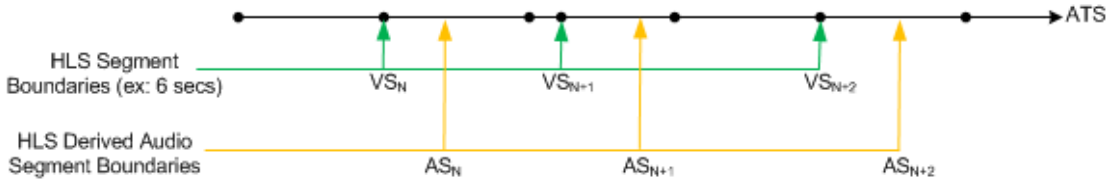


Figure 15 - Implicit / Derived HLS Audio Segment Boundaries

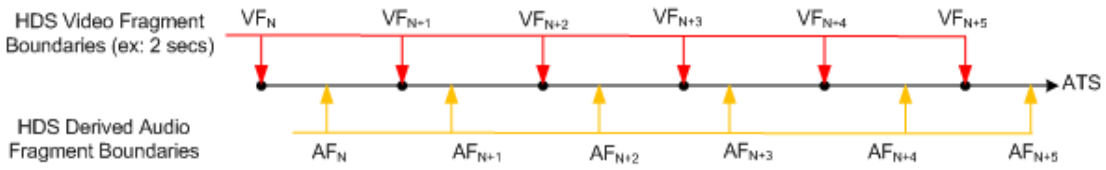


Figure 16 - Implicit / Derived HDS Audio Fragment Boundaries

In some ABR formats, the manifest is optimal if the duration of fragments is constant (i.e., using the t=x, d=y, r=z tags in HSS or DASH time-based segment templates). Using the HLS-derived approach would cause varying fragment durations in the audio domain (there is not a 1:1 relationship between video AUs and audio AUs). So an explicit EBP structure *should* be signaled for audio fragment boundaries so that redundant packagers produce the same audio fragments. Figure 17 shows in purple audio AUs that have been explicitly marked with EBP structure fragment markers. For segmented formats such as HLS, any fragment EBP structures (for audio or video) can be ignored.

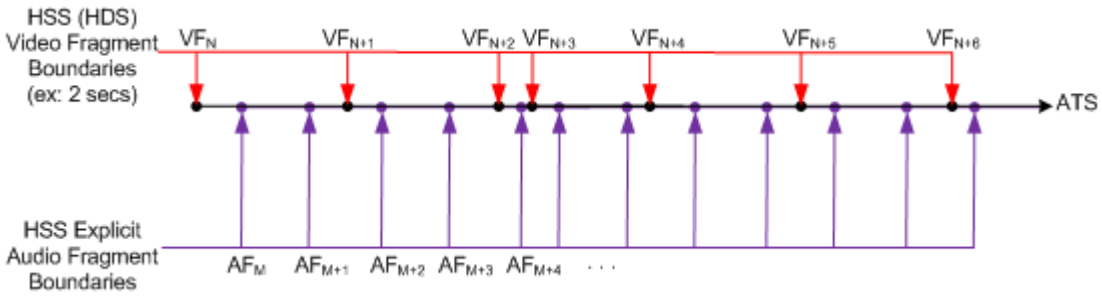


Figure 17 - Explicit Audio Fragment Boundaries

If an audio chunk boundary is indicated via an explicit EBP structure (an explicit audio EBP structure), the audio AU at the boundary *should* be PES aligned and start a new PES packet. If an EBP structure is not provided for the audio chunk boundaries (and implicit audio EBP), there is no requirement for PES alignment. Therefore, it is the responsibility of downstream encapsulation systems to split PES packets at boundary points if converting ATS to SATS.

7.11 Audio / Video Skew

Audio and video skew within the continuous ATS is the same as it would be for a normal TS. Figure 18 shows segments in the video and audio domain in different colors. Video/audio in a given color represents media from approximately the same time interval. There are no explicit audio EBP structures and thus audio boundaries are implicit from the video segments. The presentation time of the first green A is >= to the first green V and the presentation time of the last red A is < the presentation time of the first green V.

VVVVVVVVVVVVVVVVVVVVVVVVVVVVV
AAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Figure 18 - Audio Skew from Video (e.g., HLS Segments)

Figure 19 below is the same diagram but with explicit audio chunks alternately underlined. So in this example, there are 5 audio AUs per chunk. Note that the audio chunks explicitly delineated by EBP structures do not align with the implicit audio chunks that are derived from the video chunk timeline.

VVVVVVVVVVVVVVVVVVVVVVVVVVVVV
AAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Figure 19 - Audio Skew from Video (e.g., HSS Fragments)

7.12 ATS Boundaries in Relation to HLS/HDS

In an ATS, HLS audio segment boundaries are derived from video segment boundaries. HDS fragment boundaries *may* also be derived from video fragment boundaries. Below, HLS segment boundaries are discussed, but the same approach applies to HDS fragment boundaries.

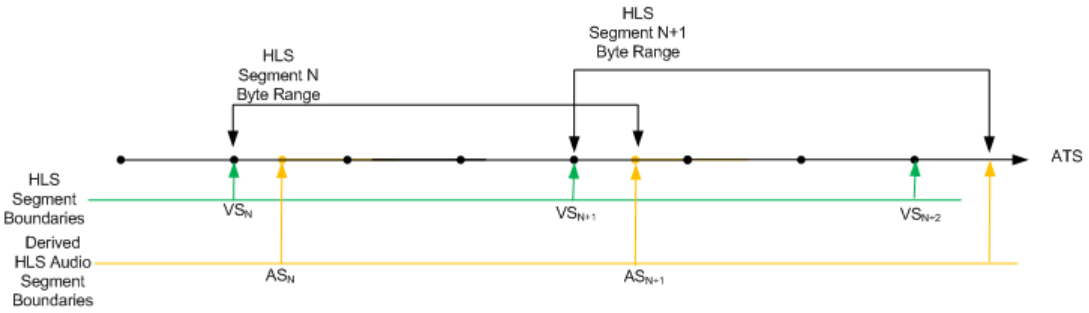


Figure 20 - ATS Segment Byte Ranges

An HLS segment S_N

- has duration TimeStampOf (VS_{N+1}) - TimeStampOf (VS_N)
- starts at byte offset VS_N
- ends at byte offset AS_{N+1}
- size of segment is ByteOffset(AS_{N+1}) - ByteOffset(VS_N)
- byte Ranges for Segment N and N+1 **overlap**
- ByteOffset(S_N+1) != ByteOffset(S_N) + SizeOf(S_N)

In HLS, URLs refer to file assets (or byte ranges within a file asset) which are themselves complete segments. Segment SN has the following properties in the ATS.

Table 5 – Segment SN Properties

Segment	Time Stamp	Duration	Start Offset	End Offset
0	TS(0)	D(0)	VS(0)	AS(1)
1	TS(0) + D(0)	D(1)	VS(1)	AS(2)
2	TS(1) + D(1)	D(2)	VS(2)	AS(3)
3	TS(2) + D(2)	D(3)	VS(3)	AS(4)
4	TS(3) + D(3)	D(4)	VS(4)	AS(5)

7.13 ATS Boundaries in Relation to HSS

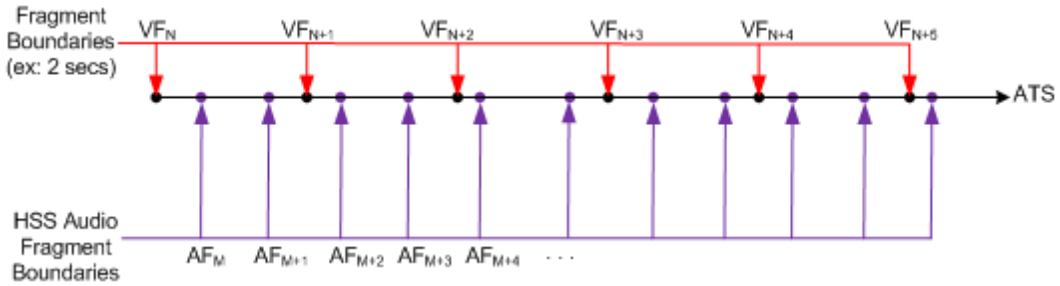


Figure 21 - ATS Video/Audio Fragments

HSS video fragment VF_N

- Has duration TimeStampOf(VF_{N+1}) - TimeStampOf (VF_N)
- Starts at byte offset VF_N
- Ends at byte offset VF_{N+1}

- Size of fragment is ByteOffset(VF_{N+1}) - ByteOffset(VF_N)
- Byte Ranges for Video Fragment N and N+1 **do not overlap**

An HSS audio fragment AF_M

- Has duration TimeStampOf(AF_{M+1}) - TimeStampOf (AF_M)
- Starts at byte offset AF_M
- Ends at byte offset AF_{M+1}
- Size of fragment is ByteOffset(AF_{M+1}) - ByteOffset(AF_M)
- Byte Ranges for Audio Fragment M and M+1 **do not overlap**

For HSS, asset URLs are based on a timestamp.

The TimeStampOf(F_{N+1}) = TimeStampOf(F_N) + DurationOf(F_N)

Table 6- Fragment FN Properties

Fragment	Time Stamp	Duration	Start Offset	End Offset
0	TS(0)	D(0)	S(0)	E(0)
1	TS(0) + D(0)	D(1)	E(0)	E(1)
2	TS(1) + D(1)	D(2)	E(1)	E(2)
3	TS(2) + D(2)	D(3)	E(2)	E(3)
4	TS(3) + D(3)	D(4)	E(3)	E(4)

7.14 Auxiliary data streams

Some applications *may* require the carriage of private, optionally timed, auxiliary data. Thumbnails, Nielsen Metadata, Live Game Stats are such examples. The various ABR formats have specific methods for signaling, formatting and delivering such private data. Clause 2.12 of 13818-1 [11] details various methods for carrying metadata within a transport stream and any could be used for carrying such data.

7.14.1 Auxiliary data in PES Streams

When auxiliary data is carried in a PES stream with timestamps [stream_type 0x06], the alignment and synchronization requirements for audio streams *should* be applied. Any descriptors present for this elementary stream in the PMT *should* be reflected in the ATS Source Description.

A common example of auxiliary data carried in timed PES packets is the Timed Metadata specification [19] which leverages a specific option from 2.12.3 of 13818-1 [11], the synchronous delivery of metadata carried in TS PES packets, for the carriage of timed metadata in the HLS ABR format.

If an elementary stream containing only auxiliary data, which is associated with a region of segments but it is not itself a continuous track, is used, then EBP structures SHOULD NOT be carried on this PID containing this data.

8 ADAPTIVE STREAMING CONDITIONING

8.1 Chunk Conditioning and Synchronization

For any given media content component in an ATS, the chunks delineated, explicitly or implicitly, by EBP structures of a given non-zero partition_id ***shall*** be non-overlapping as defined in section 4.5.2 of 23009-1 [14]. For corresponding media content components in a set of MBR streams, the chunks delineated by EBP structures with the same non-zero partition_id ***shall*** meet the requirements for segmentAlignment=true specified in section 5.3.3.2 of 23009-1 [14].

For video, the EBP structure ***shall*** exist for the start of a chunk, but it *may* exist for any access unit in the video stream. For audio, the EBP structure can be used to indicate audio chunks. When the EBP structure is used to indicate chunk boundary points in the bit stream, it can be used across an aligned set of MBR TS streams (including audio and data) to create specific ABR format fragments and segments used in adaptive streaming technologies. Downstream adaptive streaming packagers/encapsulators use the video EBP structures to help with encapsulation and can optionally use the EBP structure on the audio/data to aid in fragmentation of the audio bitstream. EBP structure is an explicit way of indicating chunk alignment, but dependent elementary streams could alternatively determine alignment in an implicit manner.

NOTE: The duration of the explicit chunk is usually of constant duration for the majority of such chunks. Occasionally the explicit chunk duration can vary, but *should* be recoverable within a few chunks to ensure the cadence of the constant duration chunk is minimally interrupted. This is typically the case with Smooth Streaming Fragments.

8.2 Video Conditioning and Synchronization

As described in the ABR overview section, media content in an ABR stream will be chunked such that seamless switching can occur at the chunk boundaries from one representation to any other. The task of video encoding for ABR *may* be accomplished in a single system with one or more encoders. It *may* be distributed across more than one system. Figure 22 illustrates two transcoding systems, each with multiple encoders.

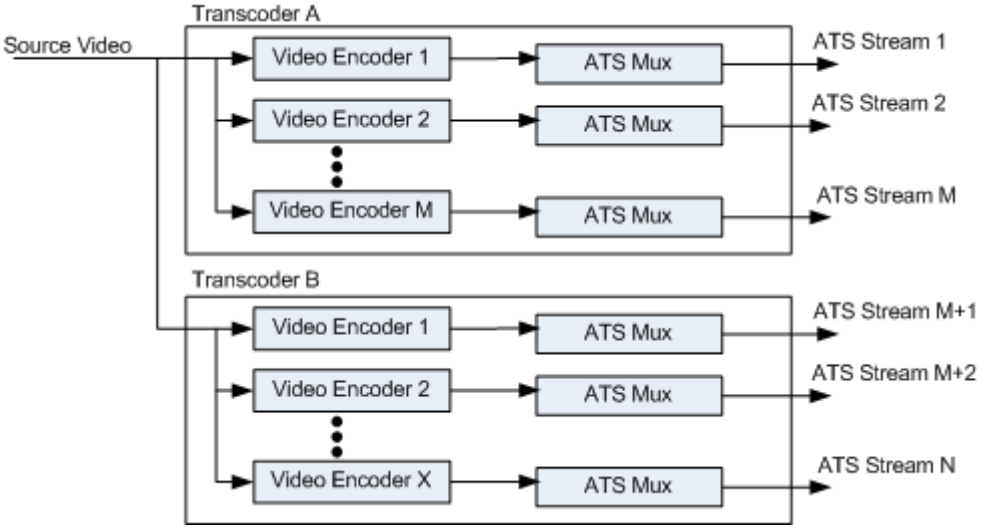


Figure 22 - Video transcoding across one or more systems

8.3 Video Chunk Sync: Start-up Considerations

ABR conditioning for video requires that, regardless of when any given encoder starts, it ***shall*** produce chunk boundaries on the same source frame even across streams of differing frame rates. This is referred to as **Chunk Sync**. Chunks targeted for a specific ABR format ***shall*** be in sync with one another. For example, HSS streams ***shall*** be in fragment-sync with one another. HLS streams ***shall*** be in segment-sync with one another. As described previously, boundaries of longer duration chunks, such as HLS segments, *should* be in sync with boundaries of

shorter duration chunks such as HSS fragments. Since chunk boundaries in the video domain are IDR frames, this cross-format chunk sync minimizes the number of IDR frames, which are bandwidth expensive.

Figure 23 illustrates both fragment and segment chunks. As shown, an encoder cannot arbitrarily start encoding on any given frame. The encoder ***shall*** choose a source frame that is fragment and segment aligned to all other streams for the given presentation. Subsequently, all future fragments/segments ***shall*** also be aligned to the same source frame.

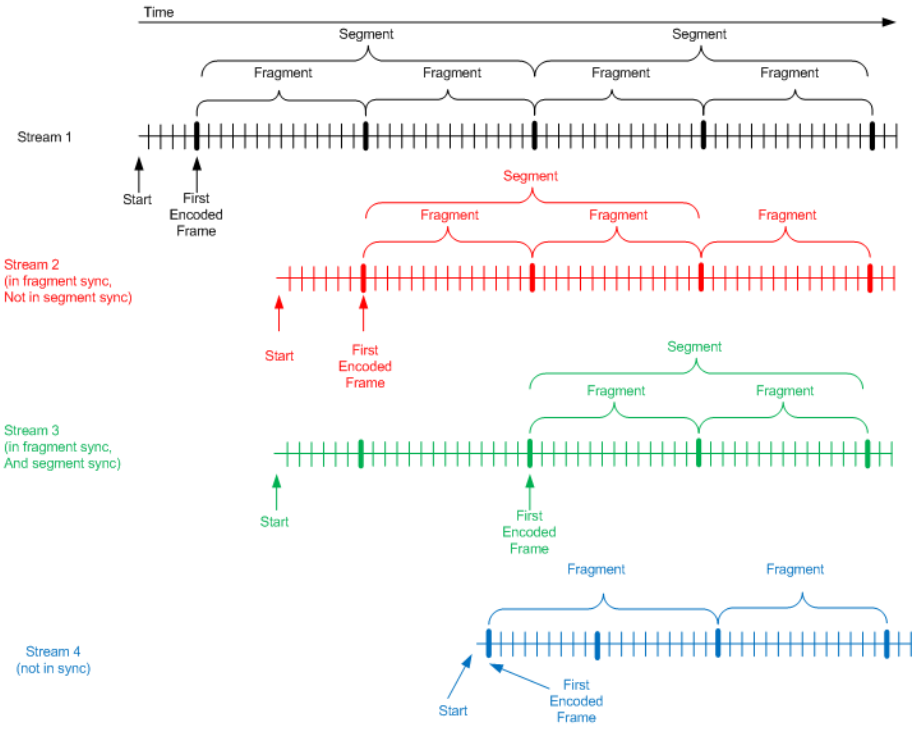


Figure 23 - Conditioning Video Start

8.4 Video Adaptive Sync

When identical AUs across representations have the same timestamp, this is known as being in Time Sync. When Chunk Synced streams are also in Time Sync, the output presentation timestamp of the first chunk boundary ***shall*** be identical across all encoders. Future chunk boundary frames ***shall*** also have identical presentation times. This is referred to as being in Adaptive Sync.

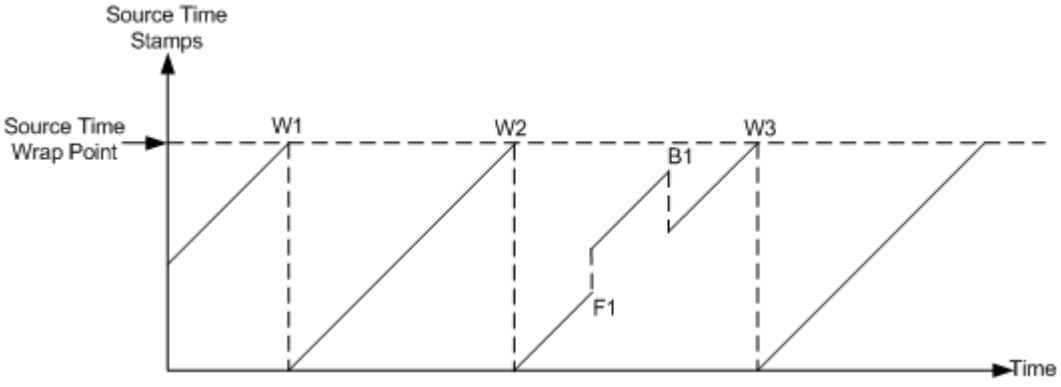


Figure 24 - Source Time Discontinuities

Figure 24 illustrates various source time discontinuities. Many source time bases have wrap points due to the limited precision of the timestamp. For example, a 33-bit MPEG-2 TS PTS will wrap about every 26.5 hours. SDI sources

with timecode *may* wrap every 24 hours. Such wraps are shown as W1, W2, W3. Sources *may* also exhibit forward discontinuities (F1) or backward discontinuities (B1). If two encoders start on either side of any number of discontinuities, not only ***shall*** they be in chunk sync, but the output timestamps of the frames ***shall*** be the same.

8.5 SCTE 35 and Splice Points

The ATS stream *may* contain SCTE 35 descriptor and trigger points that were either carried through on the source stream of the ATS transcoder or inserted at the time of the transcoding of the source stream. SCTE 35 descriptors *may* be identifying parts of the stream. SCTE 35 triggers indicate a point in the stream where conditioning of the ATS stream ***shall*** occur. SCTE 35 triggers can indicate AD insertion points. It can also indicate other types of conditioned inserted content. In order to handle single decoder end client systems, a conditioned point in the ATS stream ***shall*** occur at the beginning of a chunk.

8.6 Ad Breaks

The replacement of ads from source content with alternate ad content *may* take place at downstream ABR transcoders or packagers. The method for some ABR formats, such as HLS, involves the complete replacement of encapsulated chunks with alternate chunks. For this reason, source ad content for ABR formats *should* be chunked discretely. This means that a new chunk will be started at the first frame of an advertising break. Also, a new chunk will be started at the first frame returning from an advertising break.

Figure 25 illustrates a break out of programming (Ad start) and a return In to programming (Ad end). At the boundaries, the previous segment ends and a new one is started. This produces a split in what would have been a single segment.

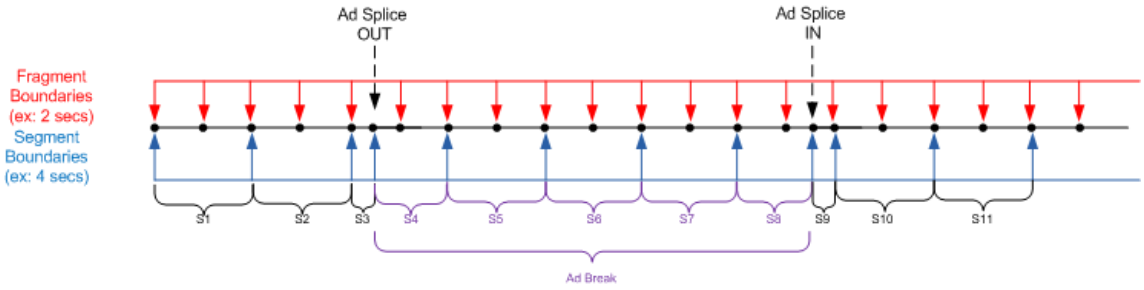


Figure 25 - New Segment at Ad Boundaries

Figure 26 illustrates the same break boundaries. Here, both new fragments and segments were created at the boundaries.

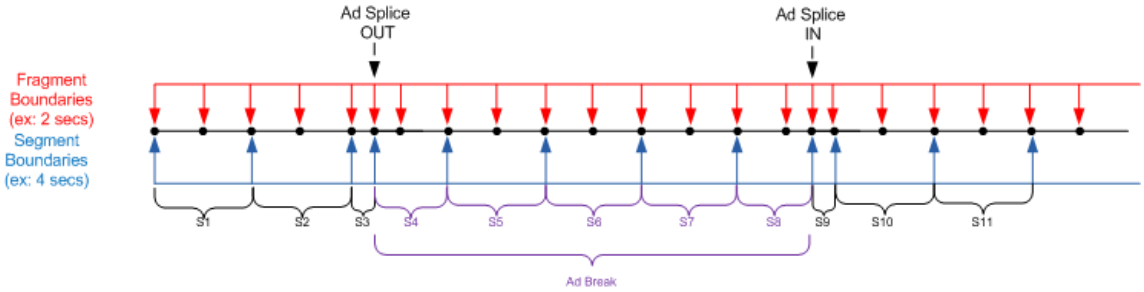


Figure 26 - New Segment and Fragment at Ad Boundaries

An Out point and an In point *may* be co-located; that is, a single boundary *may* serve as both a safe place to leave and a safe place to return. Additionally, an Out point *may* be followed by another Out point. As shown in Figure 27, this allows an ad break to be composed of multiple breaks and to specify at which break boundaries it is safe to return to programming.

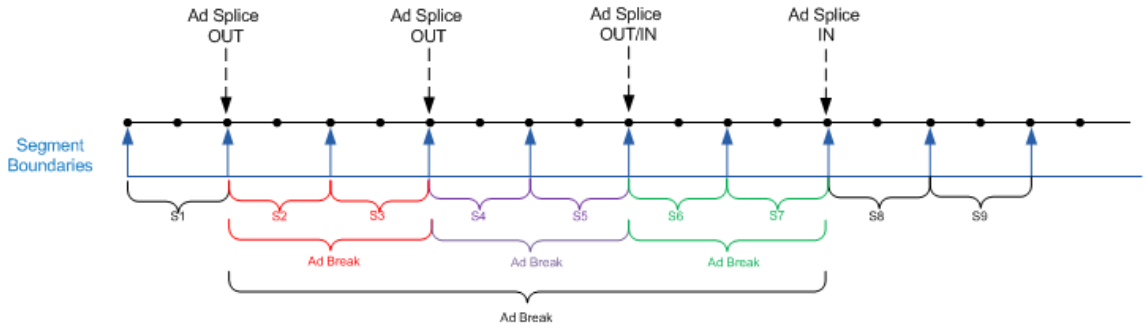


Figure 27 - Multiple Break Boundaries

For more detailed guidance on boundary creation see Appendix IV.

8.7 Data, Audio, Video Stream Alignment Considerations

There is no time alignment expectation between PES encapsulated audio or data AUs with video AUs, i.e., the start time of an audio or data AU is rarely exactly the same as a corresponding video AU. For instance, in the case of 48-KHz AAC-LC audio, there are approximately 47 AU/sec compared to approximately 30 AU/sec for video. Consider Figure 28, which shows a timeline of video and audio AUs (note this is not meant to represent how AUs *may* be interleaved in a container format).

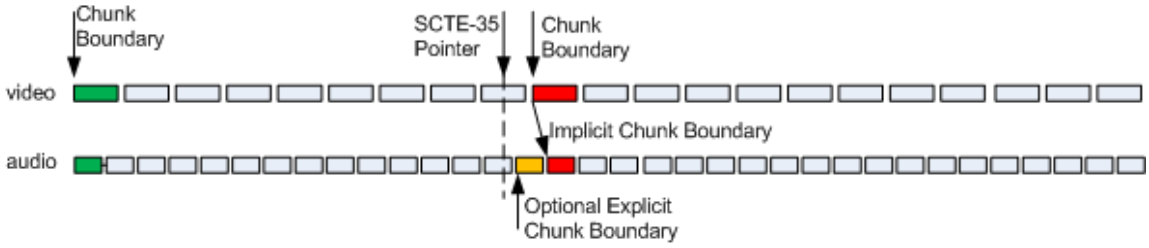


Figure 28 - Splice point and Video and Audio AUs

An advertising splice point is further shown. As described in the previous section, the first video AU >= splice point is chosen as the first AU of the new chunk. This is shown in red.

However, as we'll learn later in this document, the EBP structure details that the first audio AU of an **implicit** chunk is the first AU with sap type 1 or 2 with a PTS >= to the corresponding video AU for the chunk. This is shown as the red audio AU. This is not the first audio AU >= the splice point, shown as the yellow audio AU in the figure.

The first AU of an **explicit** audio chunk does not have to have this relationship to video. Thus the first audio AU of an explicit chunk can be the one >= the splice point, shown in yellow. Ideally, however, an encoder typically chooses to align implicit and explicit audio chunk boundaries at some boundaries, namely advertising and program delineation boundaries. DASH's tendency to create new Periods at such boundaries is one such reason for this ideal alignment.

8.8 Input Frame Loss

If one encoding system experiences input loss that is not experienced by another encoding system at the frame anticipated to be a boundary point, it *may* choose to conceal the loss by some means. Concealment produces a boundary point frame with a timestamp equivalent to what would have been used without the loss. It is highly recommended that concealment is used because it will produce switchable points since all streams will continue to have Time Sync and Chunk Sync. The EBP structure contains a bit-field to indicate when concealment is used on a boundary point.

If concealment is not used, according to 13818-1 [11] the bitstream is to be marked with a time discontinuity in the transport adaptation field. How downstream encapsulation systems handle such non-concealed discontinuities is out of the scope of this document. Such systems *may* choose to further conceal or not; if not, then the chunks *may* not be

obtainable by the client and/or *may* not be switchable (i.e., client *may* not be able to switch to it from another stream).

8.9 Audio Conditioning and Synchronization

For any of the interleave methods described in section 7.3, audio that is used across multiple ABR streams ***shall*** be conditioned described in this section.

In a single transcoding system, audio *may* be encoded once and provided to multiple ATS muxes as shown in Figure 29. In this design case, audio data and their presentation timestamps in each ATS will be identical.

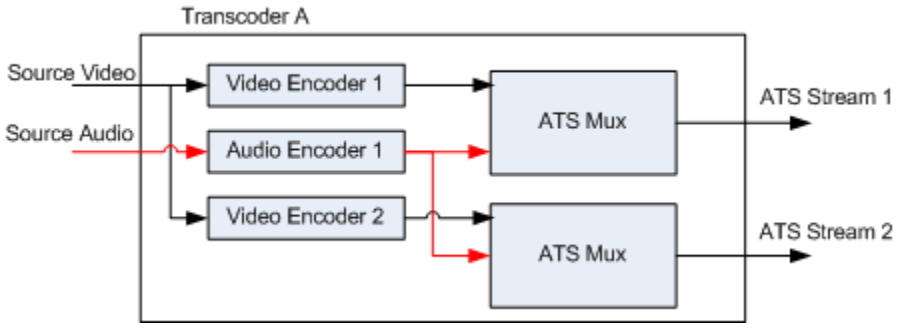


Figure 29 - ATS Transcoder with Audio distributed to Multiple Muxes

However, there *may* be designs where audio is encoded discretely for some set of ATS streams. In the single transcoding system in Figure 30, audio is encoded by two different encoders.

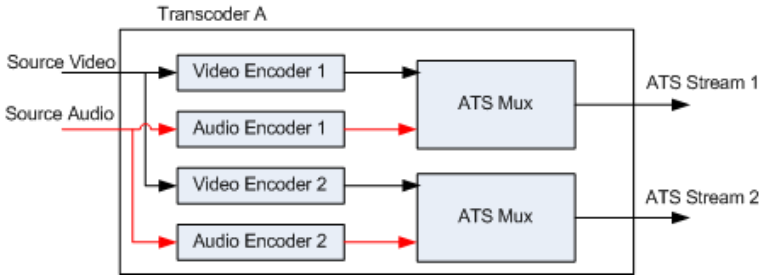


Figure 30 - ATS Transcoder with Audio encoded for each output

In Figure 31, audio is encoded only once in a given transcoder; however, two transcoders are used for stream density or redundancy purposes.

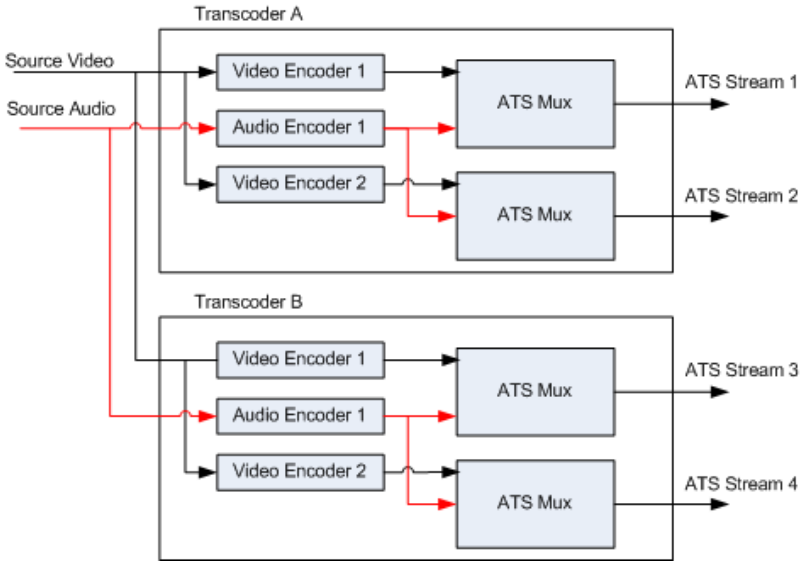


Figure 31 - Audio encoded in multiple ATS Transcoders

For adaptive purposes, audio data ***shall*** be time synchronous across streams and across transcoding systems independently of the time at which transcoding starts.

Uncompressed audio data takes the form of discrete samples. A sample rate of 48,000 samples per second (48 KHz) is common. In contrast to video, audio encoders do not output discrete audio samples. Some audio encoders produce discrete output access units (AUs) given some number of fixed input samples. For example, an AAC audio encoder produces an output AU for exactly 1024 input samples. Thus, the duration of an AAC AU is defined as: 1024 / input sample rate. A 48 KHz audio source flowing into an AAC audio encoder produces AUs of 21.33 millisecond durations (1024/48000), 46.875 AUs per second.

Adaptive Streaming stream-switching requires AUs across streams to be in time sync with one another in order to provide seamless transitioning from one stream to another. In the audio domain, this means that the time stamps of AUs outputted from different audio encoders need to be the same.

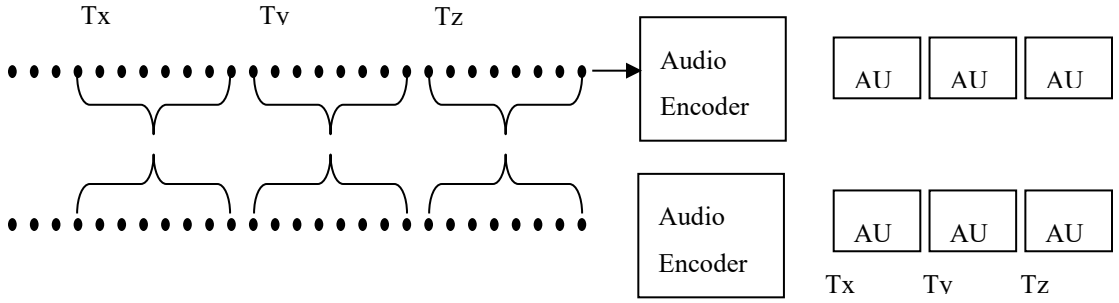


Figure 32 - Audio Access Units: Groups of Samples

If two audio encoders start at exactly the same instant, the set of samples entering the audio encoders will be exactly the same time; more specifically, the initial sample entering each encoder has the same time stamp. In Figure 32, this time stamp is Tx. This initial sample and N-1 further samples will result in the output of a single encoded audio AU. The time stamp of this AU is defined as the time stamp of the first sample it represents, which would again be Tx in this example. Subsequently, another N samples starting at time Ty enter the encoder and another AU with time Ty is generated. It follows that another N samples starting at time Tz enter the encoder and another AU with time Tz is generated.

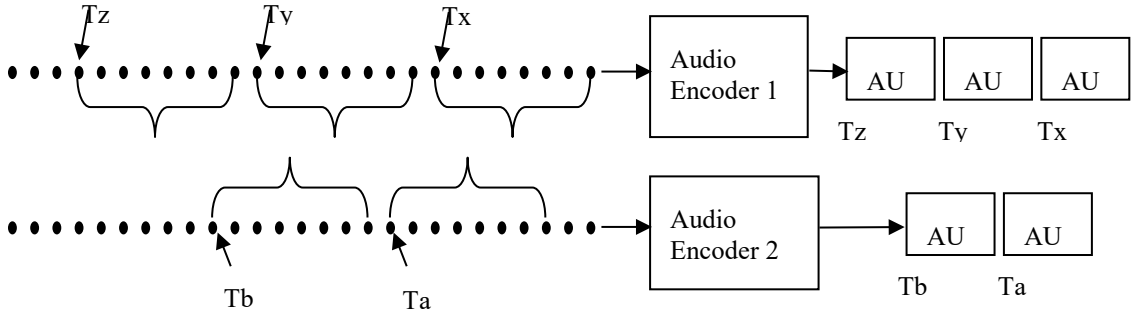


Figure 33 - Nonaligned Audio Access Units

Figure 33 illustrates the case when two encoders do not start at the same time. Encoder 2 started later than encoder 1. The first audio sample entering audio encoder 2, having time stamp Ta, did not align with the first sample of the (depicted) second AU of encoder 1 having time stamp Ty. Because of this discrepancy, the discrete AUs from the encoders are not in sync with one another. If an Adaptive Streaming solution switched from a stream from audio encoder 1 to a stream from audio encoder 2, some number of discrete audio samples would be repeated due to the overlap in AUs. This could cause an audio distortion (pop) or pause, or some other audible manifestation to the end user.

In order to achieve audio synchronization for Adaptive Streaming, the time stamps of audio AUs have to be the same and therefore the audio data represented by the AUs have to be the same. Thus the first audio sample entering an audio encoder cannot be arbitrary. It ***shall*** fall on a boundary such that, independent of when an audio encoder starts, this first sample is synchronous with the first sample of an AU of any other audio encoders, regardless of when these encoders started.

8.10 AAC Family Audio Chunk Boundary requirements

For most audio codecs, each Access Unit is fully decodable and *may* be used as a chunk boundary. However, for AAC family codecs (AAC LC, HE AAC, HE AAC v2), Access Units used as chunk boundaries *should* meet the requirements for AAC Random Access Points as not every AU is required to carry metadata required to reproduce the encoded audio with full fidelity as required by SCTE 193-1 [5].

Audio quality during a switch between Audio streams can be affected by several factors. When switching between audio streams with different encoding parameters, or entering a new stream, the lack of PS and SBR header data

may result in a reduction of audio quality until that header data is acquired. Lack of loudness and downmix metadata *may* result in unacceptable changes in perceived loudness, particularly if the change occurs at an ad insertion point.

8.11 Audio Alignment Across Representations

Audio streams of the same format, language, sample rate and channels, thus differing only in bitrate, ***shall*** be aligned across streams as described in the previous sections.

To enable switching between Audio streams, all Representation within an Adaptation Set ***shall*** meet the following:

1. Identical Audio Encoding parameters are used for corresponding audio components in each Representation.
2. The Access Unit aligned with a chunk boundary in each audio PES packet ***shall*** meet the requirements for an audio RAP (SAP Type = 1)

Appendix I EBP Structure Use Case Examples

This section describes several informative examples of how the EBP structure and the PMT descriptor can be used.

I.1 Indicate a Base Fragment Boundary Point

The EBP structure can be used to indicate fragment boundaries used in AVC video (e.g., 2-second fragments). EBP structure is associated with an IDR AU that starts the fragment. The fragment boundary point *should* be on and other flags and bytes *should* be set according to the table below.

Table 7 - EBP Structure for Fragment Boundary Point

Boundary Descriptor Bytes							
Number of partitions ***minus1***	→	→	***SAP***	→	→	***Concealment Flag***	***Reserved***
"000" → 1 parition	-	-	"001" →SAP is 1	-	-	"0"	"0"
Partition_id	→	→	***Partition_info_flag***	***Reserved***	→	→	→
"010" = 2	-	-	"1"	"0000"	-	-	-
Sequence number_length code	→	**Reserved**	→	→	→	→	→
"11" → 64 bits	-	"000000"	-	-	-	-	-

Additional Bytes
Sequence_number
A 64 bit number

I.2 Indicate a Common Segment Boundary and Fragment Boundary Point

A single EBP data structure can indicate the video IDR AU is the start of both a fragment and/or segment as used in AVC video. This can be used to indicate larger segment boundaries for use in HLS fragments. The same AU can indicate both a segment boundary and fragment boundary (which would be the most common case) by turning on both the fragment and segment flags as indicated in the table below.

Table 8 - EBP Structure for Segment Boundary Point

Boundary Descriptor Bytes & First Partition							
Number of partitions minus1	**→**	**→**	***SAP***	**→**	**→**	***Concealment Flag***	***Reserved***
"001" → 2 partitions	**-**	**-**	"001" → SAP is 1	-	-	"0"	"0"
Partition_id	**→**	**→**	***Partition_info_flag***	***Reserved***	**→**	**→**	**→**
"001" = 1	-	-	"1"	"0000"	-	-	-
Sequence number_length code	**→**	**Reserved**	**→**	**→**	**→**	**→**	**→**
"11" →64 bits	-	"000000"	-	-	-	-	-

Additional Bytes
sequence_number Value
a 64 bit number

Boundary Descriptor Bytes (Cont'd) & 2nd Partition							
Partition_id	**→**	**→**	***Partition_info_flag***	***Reserved***	**→**	**→**	**→**
"010" = 2	-	-	"0"	"0000"	-	-	-

I.3 Indicate a segmentation_upid_type (an identifier) and segmentation_type_id (stream point label) in Labeling Descriptor

Table 9- Carriage of Identifier and Program Start using the Labeling Af_descriptor

Labeling Descriptor Bytes with EIDR (0xOA)							
label_ length_ code	→	→	***label_ type***	→	→	→	→
"100" →12 bytes	-	-	0x20A	-	-	-	-
→	→	→	→	→	→	→	→
-	-	-	-	-	-	-	-

label_byte
EIDR VALUE
12 Byte number

Labeling Descriptor Cont'd with Program Start (0X10)							
label_ length_ code	→	→	***label_type***	→	→	→	→
"000" → 0 bytes	-	-	0x110	-	-	-	-
→	→	→	→	→	→	→	→
-	-	-	-	-	-	-	-

I.4 Indicate NTP Time for Linear Services using TEMI AF_descriptor

NTP time can be indicated through the EBP structure. This time can be applied to a PES header using TEMI the af_descriptor , so it could be put on any video AU or group of audio AUs. The time could be placed at an encoder or transcoder. A transcoder *may* generate its own NTP time from an NTP signal or derive this from the NTP time in the EBP structure of the input content stream, if it exists. In VoD, the NTP timestamps follows a Normal Playtime (NPT) Timeline.

Table 10- Carriage of NTP for Linear Services using the TEMI af_descriptor

TEMI Timeline Descriptor using AF_Descriptor Format and NTP time							
has timestamp	***→***	***has_ntp***	***has_ptp***	***has _timecodes***	***→***	***Forced reload***	***paused***
"00" = 0	-	"1"	"0"	"00" =0	-	x	x
discontinuity	***reserved***	***→***	***→***	***→***	***→***	***→***	***→***
x	"0000000"	-	-	-	-	-	-
Timeline_id	→	→	→	→	→	→	→
0xFF	-	-	-	-	-	-	-
Additional Bytes							
NTP Time							
64 bit number							

Appendix II PMT EBP Virtual_segmentation descriptor use case examples

II.1 Signaling timing and partitions for HSS and HLS

Table 11 displays a use case that shows a video PID with 10-second segments for HLS and 2-second fragments for HSS. Each PMT EBP structure contains an acquisition time.

Table 11 - Example of PMT EBP virtual_segmentation_descriptor() for HLS and HSS video PID

virtual_segmentation_descriptor(){		
...		
num_partitions	010	
timescale_flag	0	
for (i = 0; i<num_partitions; i++)	i = 0	i=1
explicit_boundary_flag	1	1
partition_id	1	2
maximum_durationSAP_type_max	10	2
	1	1

II.2 Explicit and implicit partitions

Table 12 shows a use case with partition 0 for acquisition times and 2-second fragments for HSS, as in the previous example. The example in Table 12 shows an audio PID with explicit 2-second fragments and implicit timing based on video PID 481.

Table 12 - Example of PMT EBP virtual_segmentation_descriptor() for HLS and HSS audio PID

virtual_segmentation_descriptor(){		
...		
num_partitions	010	
timescale_flag	0	
for (i = 0; i<num_partitions; i++)	i = 0	i = 1
explicit_boundary_flag	0	1
partition_id	1	2
boundary_PID	481	n/a
maximum_durationSAP_type_max	n/a	2
	n/a	1

Table 13 and Table 14 show a use case (HDS Method 1) for HDS video fragments using the same partition as HSS video fragments, and HDS audio fragments being implicitly derived from HSS video fragments (PID 481).

Table 13 - Example of PMT EBP virtual_segment_descriptor() for HLS, HSS, HDS (implicit to partition=2) video PID

virtual_segmentation_descriptor(){		
...		
num_partitions	010	
timescale_flag	0	
for (i = 0; i<num_partitions; i++)	i = 0	i=1
explicit_boudary_flag	1	1
partition_id	1	2
maximum_durationSAP_type_max	10	2
	1	1

Table 14 - Example of PMT EBP virtual_segmentation_descriptor() for HLS, HSS, HDS (implicit partition=2) audio PID

virtual_segmentation_descriptor(){			
...			
num_partitions	011		
timescale_flag	0		
for (i = 0; i<num_partitions; i++)	i = 0	i = 1	i = 2
explicit_boundary_flag	0	1	0
partition_id	1	2	2
boundary_PID	481	n/a	481
maximum_duration	n/a	2sec	n/a
SAP_type_max	n/a	1	n/a

Table 15 and Table 16 show a use case (HDS Method 2) where HDS video fragments use their own partition and HDS audio fragments are implicitly derived from this partition.

Table 15 - Example of PMT EBP virtual_segmentation_descriptor() for HLS, HSS, and HDS (explicit partition=3) video

virtual_segmentation_descriptor(){			
...			
num_partitions	011		
timescale_flag	0		
for (i = 0; i<num_partitions; i++)	i = 0	i=1	i=2
explicit_boundary_flag	1	1	1
partition_id	1	2	3
maximum_durationSAP_type_max	10	2	2
	1	1	1

Table 16 - Example of PMT EBP virtual_segmentation_descriptor() for HLS, HSS, and HDS (implicit partition=3) audio PID

virtual_segmentation_descriptor(){ ...			
num_partitions	011		
timescale_flag	0		
for (i = 0; i<num_partitions; i++)	i = 0	i = 1	i = 2
explicit_boundary_flag	0	1	0
partition_id	1	2	3
boundary_PID	481	n/a	481
maximum_durationSAP_type_max	n/a	2??	n/a
	n/a	1	n/a

Appendix III Derivation of Acquisition Time (Informative)

Since the processing required to generate Adaptive Transport Streams in different formats *may* introduce delays that vary by format type and across implementations, implementations *should* ensure that the acquisition time represents a consistent point in the signal processing chain.

Consider the signal chain in an ISO/IEC 13181-1 [11] compliant transcoder model given in Figure 34 below.

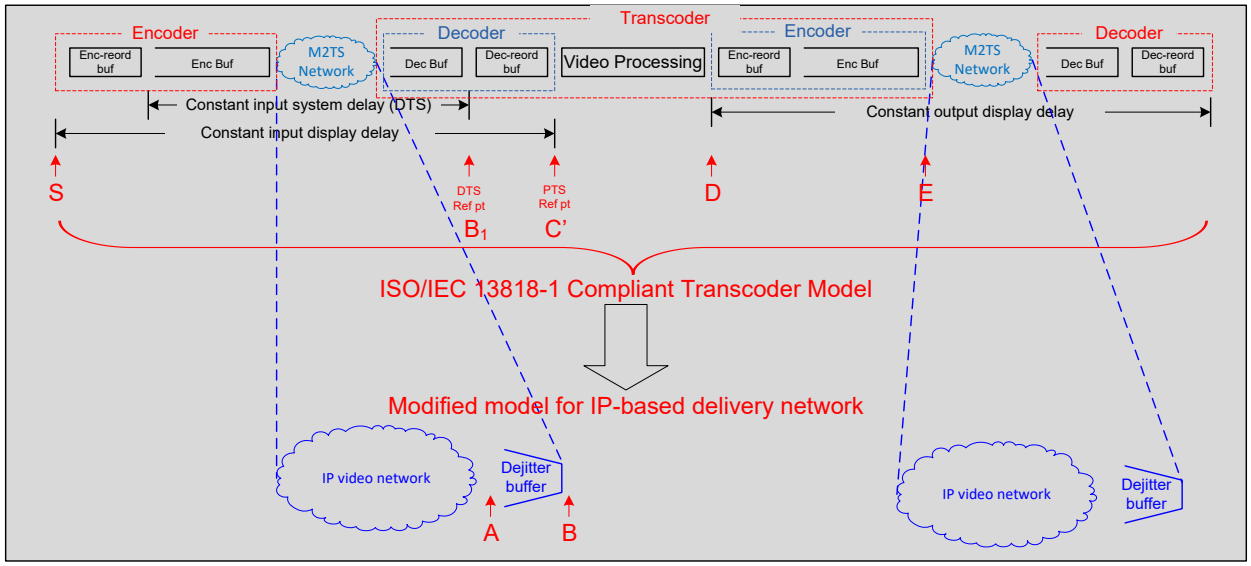


Figure 34 - ISO/IEC 13818-1 compliant transcoder model

The upper half of Figure 34 illustrates a transport network compliant to ISO/IEC 13818-1 [11]. Here, the encoder is modeled with an encoder reordering buffer and an encoder data buffer, and the decoder is modeled with a decoder buffer and a decoder reordering buffer. Both the encoding operation and decoding operation are assumed to be instantaneous. Regardless of implementation details, a transcoder can be modeled with a decoder, an encoder, and optionally a baseband video processing unit. As shown in Figure 34, a transcoder receives compressed video from an upstream MPEG-compliant video encoder and is expected to generate an MPEG-compliant stream as its output.

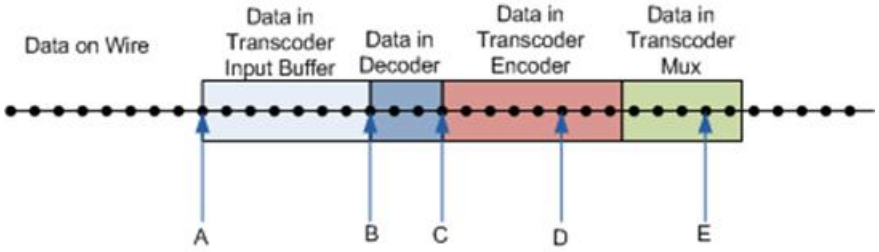
To determine a common sampling point for EBP data acquisition time, several points are marked in Figure 34. Point S is the time when a source video frame is about to enter the reordering buffer at the upstream encoder. Point B_1 is when the decoder within the transcoder decompresses the access unit and moves it into the decoder reordering buffer. This is the time referred to as DTS. Point C is when the access unit is reordered and displayed. This is referred to as PTS. Point D is when the decoded picture is about to enter the encoder reordering buffer at transcoder. Point E is when the re-encoded access unit is about to be multiplexed with audio and other data.

The lower half of Figure 34 extends the traditional MPEG 2-compliant network into an IP-based delivery network, by adding a dejitter buffer at the decoder input. It is expected that TS traffic at the output of a dejitter buffer complies with ISO/IEC 13818-1. For the purpose of discussion, the lower half of Figure 34 introduces two more reference points, A and B. Point A is when the IP packet containing the first byte of an access unit is received at the transcoder side, and point B is when the same TS packet is out of the dejitter buffer.

Among all the points given in Figure 34, point S is obviously the best reference point to sample NTP. However, it is quite impractical to expect sampled NTP at point S due to unknown encoding-decoding delay chosen by the upstream encoder. If jitter induced by the IP network is negligible, point C would be another good reference point, since delay between point S and point C is fixed. However, this is not the case for IP-based transport network. Delay between point S and point C can vary by the amount of network jitter.

Figure 35 illustrates the possible reference points at a nominal transcoder. While it is expected that a transcoder will sample NTP based on its display time referred to as point C, a different decoder *may* choose a different display time. To avoid this potential discrepancy, it is expected that the NTP value for each boundary point is sampled

corresponding to point C signaled by PTS in TS stream. All transcoders with the same input stream are expected to implement this theoretical timing model defined by ISO/IEC 13818-1 [11].



A= enters transcoder dejitter buffer, B= AU enters decoder buffer, C= exits decoder output buffer, D= video processing and encoding, E= gets multiplexed

Figure 35 - Candidate Reference Time Sample Points in Signal Chain in Nominal Transcoder

An implementation of a transcoder *should* sample NTP regularly. It *may* sample NTP at a different point mentioned in Figure 35, but it *should* compensate accordingly to reflect time at point C. If the implementation does not sample NTP for each access unit, it *should* derive the appropriate value for each access unit.

As an example, one can sample NTP at point B and use or derive PCR value and PTS value of the corresponding TS packet that contains the first byte of access unit, according to operation defined in 13818-1[11]. With those data, NTP corresponding to point C can be derived with (sampled NTP + (PTS – PCR)). PTS is equal to (DTS + reordering delay). (DTS – PCR) is known as vbv_delay defined in the ISO/IEC 13818-2 [32] or cpb_removal_delay in ISO/IEC 14496-10 0. Another implementation *may* choose to sample at point A. In this case, it is expected that NTP values will be filtered so that it reflects NTP values at point B. After this is done, it *should* compensate in a similar way to point B.

Similarly, an implementation that samples NTP at point D of E *should* compensate the values back to point C before inserting into EBP structure data field.

It is possible that the transcoder system will introduce extra delay inherent in its deployment architecture. For the purpose of EBP definition, this delay is counted as part of the +/- 300 ms jitter budget. However, a specific implementation *may* choose to compensate this extra delay to its sampled NTP value to improve accuracy of NTP value.

Appendix IV Determining Network Drift Effects on Arrival Times of ATS sets

Whether on an M2TS network or an IP video network, it is important to ensure the arrival times of each transport stream in the ATS set to a downstream receiver occur at approximately the same time. Downstream receivers can be packagers, additional transcoders, manifest creation devices, or STBs. Delays in arrival times of corresponding virtual segments in any of the streams in the ATS set can lead to more complex changes to the CIF manifest description of the ATS set, buffering delays in downstream receivers, simulated intermittent connection errors, fragment switching errors, or playback errors in live content. Some delays are caused by network operations such as encapsulating data into network Ethernet frames or network jitter. It is important to constrain network drift effects on the ATS set to within an acceptable tolerance.

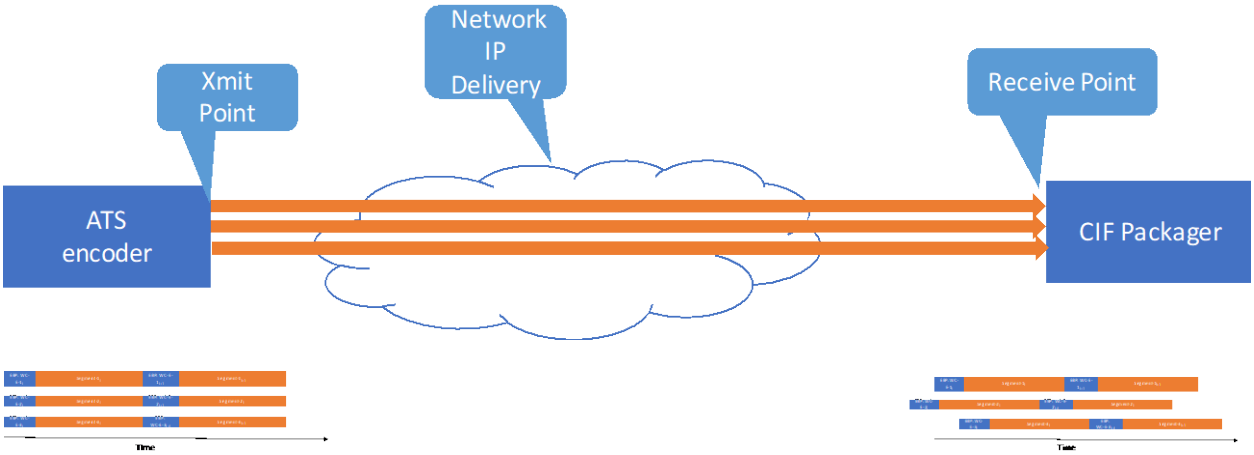


Figure 36 – Effects on Network Drift on Arrival Times

Note: The output timing of the representations at the encoder may in reality not be always perfectly aligned. Encoders still should maintain temporal alignment across the representations that allows for synchronized timings of representations while following HRD restrictions.

The drift between arrival times of streams in the ATS set can be calculated by measuring the clock times between corresponding PCR packets or corresponding EBP timestamps between each of the streams in the ATS set using an NTP clock at the point of measurement. The network drift between arrival times of ATS streams in the set can be determined by measurements at both the transmit point and the receive point. The network drift between arrival times of streams in an ATS set at a packager receiver point shall be no more than 80 ms.

Appendix V Guidelines For Boundary Creation

V.1 Adjusting Chunk Durations

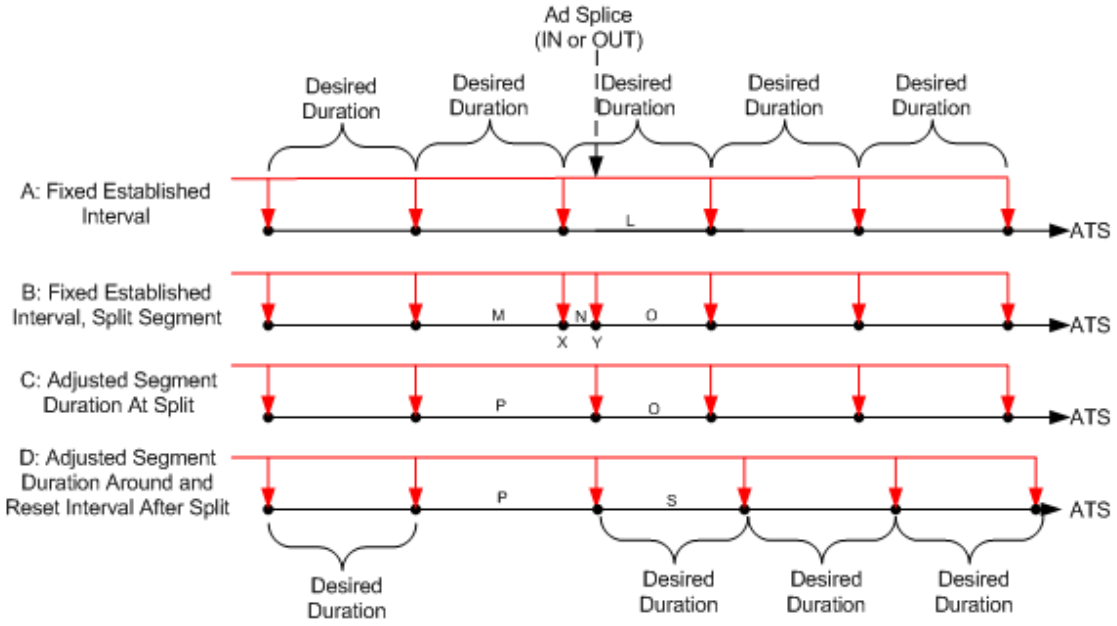


Figure 37 - Adjusting Durations Around/After Splice Points

For Figure 37, Timeline A shows an ATS stream with chunk boundaries of a fixed duration. An advertising splice point occurs within a chunk (L). Per Section 8.5, a new chunk needs to start at this point. This is shown initially in timeline B. Prior to the splice point (Y), a new chunk was started at X due to the creation of chunks at fixed established intervals. The result of starting a new chunk at Y splits what was chunk L in timeline A into two smaller chunks, N and O.

When the size (duration, number of frames) of N is small, a transcoder can choose to adjust the size of a chunk as shown in timeline C. In this example, both a longer (P) and shorter (O) chunk is created around the splice point. However, the duration of the splice point chunk (O) is set so that future chunks align with the original timeline A.

In timeline D, only a single chunk (P) is adjusted (longer in this case) to align with the splice point. The duration of the splice point chunk (S) is set to the desired target chunk duration. Because of this, the boundaries of future chunks no longer align with timeline A.

In Figure 38, a similar example is shown but with a splice point leading up to (versus trailing in as in Figure 37) a chunk boundary. Timeline E is similar to the approach of Timeline C, creating a shorter (T) and longer (U) chunk around the splice point and maintaining chunk boundaries with timeline A. Timeline F is similar to the approach of Timeline D, creating a shorter chunk (T), in this case around the splice point, but then maintaining desired chunk durations afterwards, subsequently becoming unaligned with timeline A.

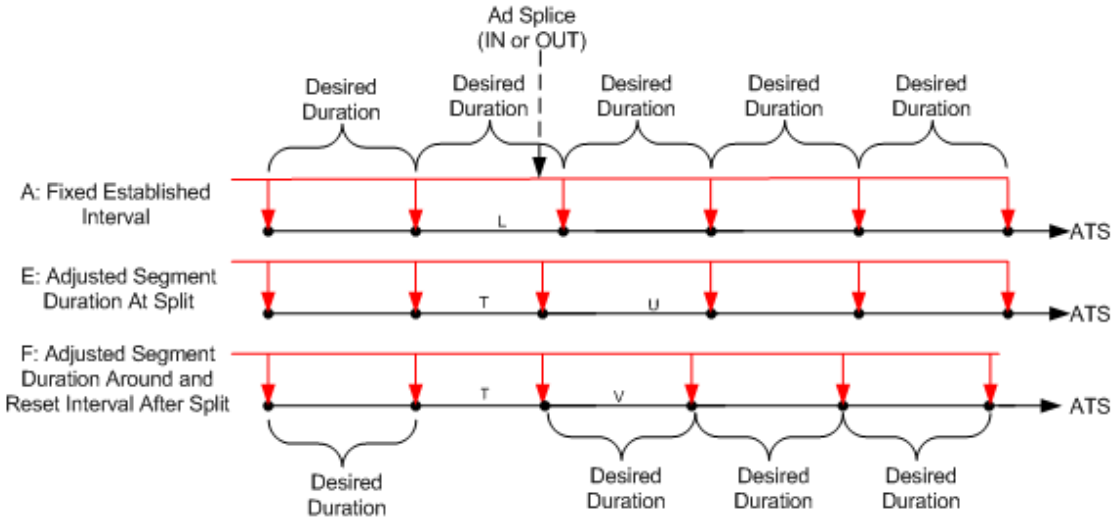


Figure 38 - Example 2, Adjusting Durations Around/After Splice Points

These techniques can apply to scene changes or other conditions that force the placement of an Intra picture in close proximity to a desired boundary position.

All approaches are acceptable. However, if N is less than 1/2 second, then approaches C/E or D/F are highly desired. Furthermore, DASH-IF IOP [15] specifies a +/- 50% tolerance on target durations except for the last chunk of a period, which has no duration restriction. An ATS follows this restriction in most circumstances. For example, the smallest chunk in an ATS partition with a 2-second target duration is 1.0 seconds and the largest chunk is 3.0 seconds. Because of this restriction, depending on the location of a splice point, it could not always be possible to achieve approaches C/D above where alignment to the original timeline past the splice point occurs at the next boundary, although alignment to the original timeline would be possible at the subsequent boundary. Refer to Figure 39.

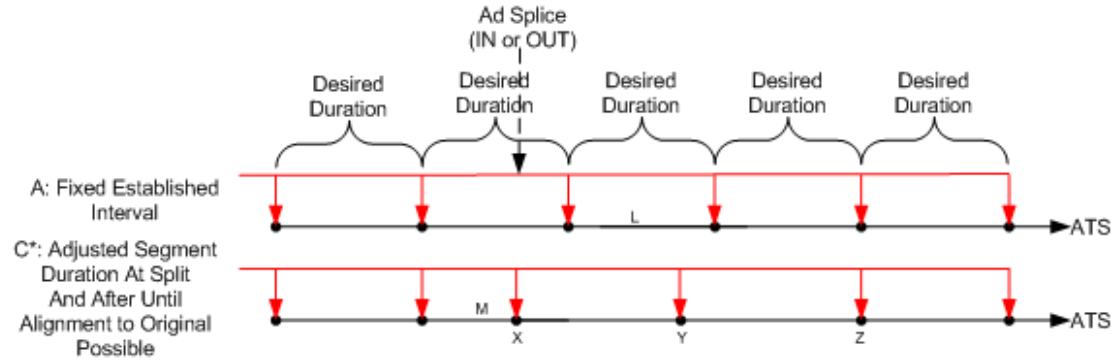


Figure 39 - Adjusting Durations with DASH-IF IOP 50% Tolerance to Align to Original Timeline

V.2 Frame Rate Decimation

In order to maintain alignment across representations with differing frame rates, restrictions on chunk sizes need to be imposed. For example, consider a 25 fps source encoded to both a 25 fps stream and a 12.5 fps stream where every other source frame is skipped.

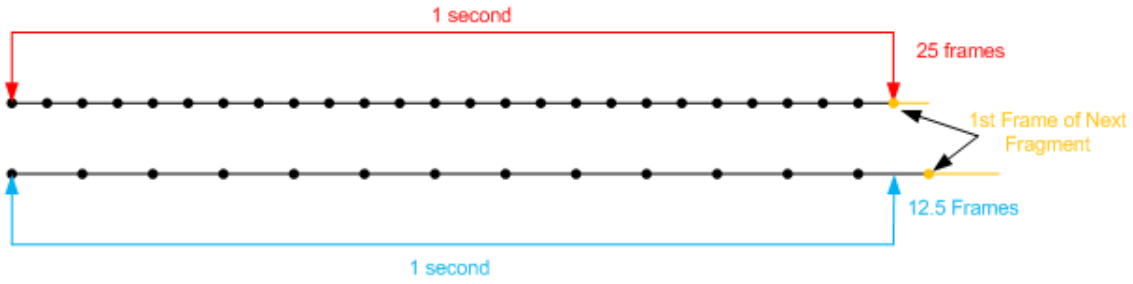


Figure 40 - One Second of Frame Rate Decimation (25 to 12.5)

As Figure 40 illustrates, there is not a common frame between these two streams at the 1-second mark. If a chunk boundary is made after the 1st second as shown above, the two streams would not be in chunk sync nor time sync with one another.

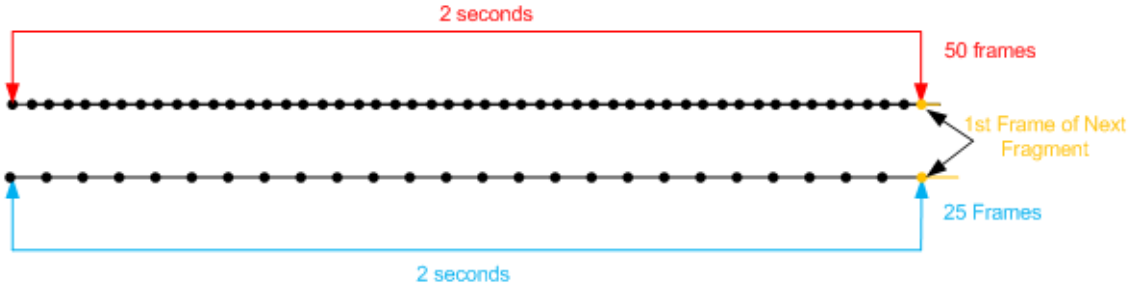


Figure 41 - Two Seconds of Frame Rate Decimation (25 to 12.5)

If we extend the timeline to two seconds as shown in Figure 41, we can see there is an alignment across the two representations at the two-second mark. As this example illustrates, an ABR representation with multiple frame rates is restricted in possible chunk sizes.

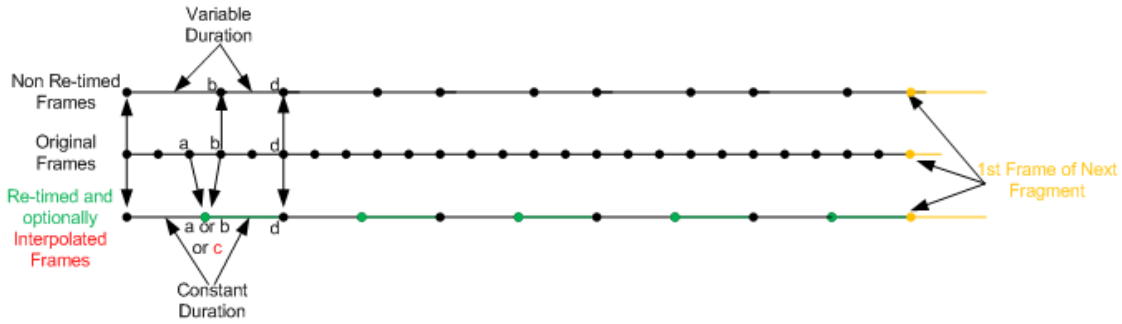


Figure 42 - Non integer frame rate reduction

Figure 42 illustrates non-integer frame rate decimation (i.e., where source frame rate is not divided by an integer). Here the source frame rate is 25 fps and an output frame rate is 10 fps. An encoder can choose multiple approaches to producing frames and the times of these frames.

- Select a source frame and use its timestamp. This is shown in the top timeline. Using this approach will produce variable frame durations, but the decimated frame timestamps will match non-decimated frame timestamps. Note that even with this approach, two different frame rate reduced streams will not have corresponding matching frame times.
- Select a source frame and interpolate its timestamp. This is shown in the bottom timeline. The green samples are re-timed but the frames themselves (a or b) are identical to one of the closest source frames.
- Interpolate an output frame from multiple source frames and interpolate its timestamp. This is also shown in the bottom timeline. The green samples are re-timed but the frames themselves (c) are interpolated.

V.3 Boundaries Desired at Unaligned AUs

The previous section describes creating aligned chunks across multiple frame rates. In Figure 43, a naturally occurring chunk boundary is illustrated. Four different streams are represented; the first (1x) has every AU represented. The second is at 1/2 the frame rate and thus each AU is 2x the duration of the first stream. The third is at 1/3 the frame rate and thus each AU is 3x the duration of the first stream. The fourth is at 1/4 the frame rate and thus each AU is 4x the duration of the first stream. In this example the least common multiple for alignment is 12 frames and thus the target duration for chunks would be a multiple of 12 frames.

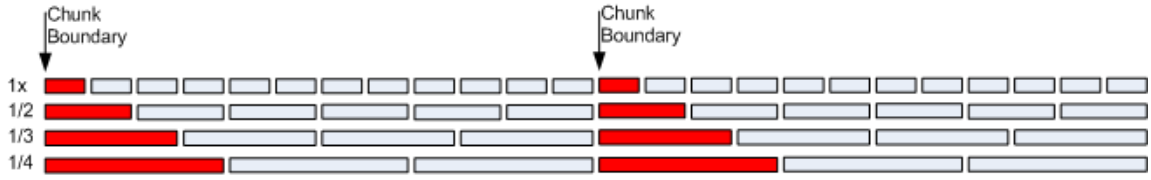


Figure 43 - Naturally Aligned Chunk Boundary

Now consider the need to start a new chunk at an arbitrary AU. This *may* be because of an advertising in or out point, a program delineation point, a scene change in close proximity to an anticipated boundary, etc. Figure 44 shows the same diagram as Figure 43 but with the desire to start a new chunk due to an SCTE 35 [6] marker.

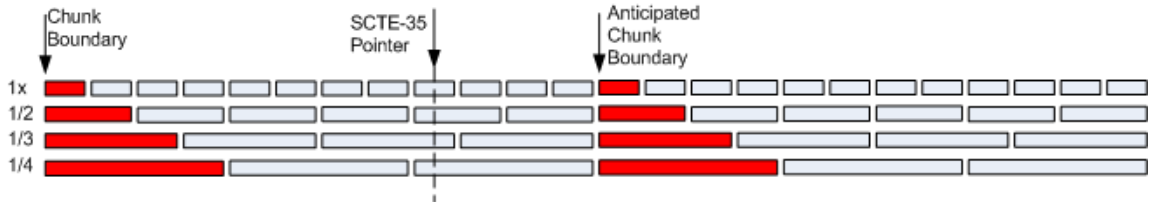


Figure 44 - Desire for new Chunk

In this case, the SCTE 35 metadata is pointing to a time position in the middle of an AU. As a result, the desire is to start a new chunk on the first AU following this point. However, as seen in Figure 45 for this example, this first AU for stream 1 only aligns with an AU from frame rate decimated stream 3. This does not align with AUs with streams 2 and 4.

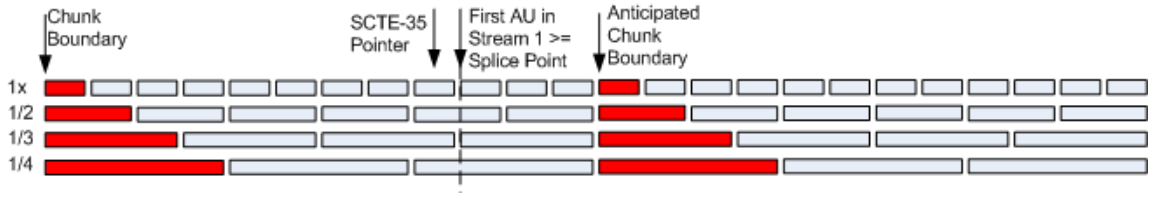


Figure 45 - First AU >= Splice Point

If one were to simply start a new chunk in each stream with that stream's first AU that was >= to the splice point, one *may* end up with something like Figure 46. In this figure we see for each stream a new chunk (in red) starting with that stream's first AU that was >= to the splice point.

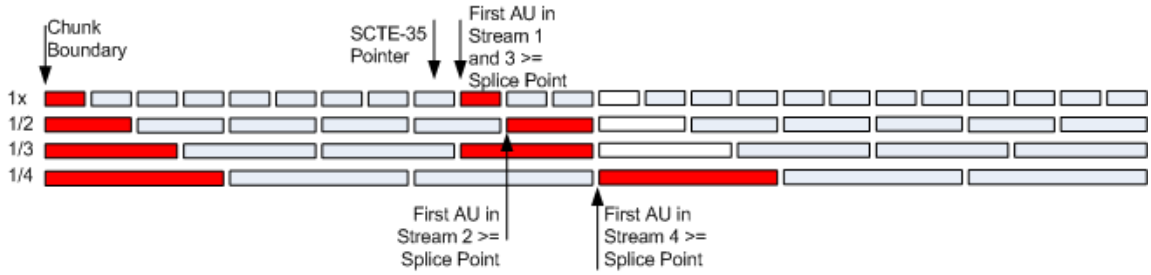


Figure 46 - Chunk simply starting at next frame rate decimated frame AU >= splice point

The issue with this approach is that the streams are not in adaptive sync with one another at this location. Specifically in this example, stream 1 and 3 are still in sync but neither of these is in sync with streams 2 and 4, which in turn are not in sync with each other. To address this problem, the first source AU prior to frame decimation that is >= the splice point needs to be used as the first AU for the new chunk in each stream. In order to get alignment across various chunk sizes with differing frame rates, restrictions on chunk sizes needs to be imposed. This is illustrated in four different methods in the following figures. Recall that Stream 1 represents the unaltered or highest frame rate of all representations.

In the following three cases, a new chunk is started in each stream, regardless of frame rate, on the first source AU >= the splice point.

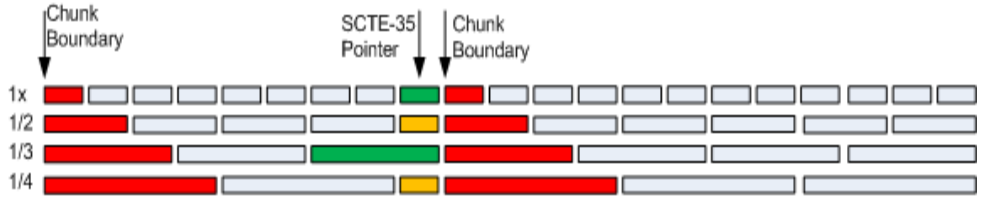


Figure 47 - Using AUs to Complete Chunk

Option 1: In Figure 47, prior to the new chunk boundary, each stream encodes its next AU as it normally would have done. However, in some cases, the duration of this AU in the frame rate decimated streams is shorter than normal. These are shown in yellow (streams 2 and 4). In this example, stream 3's AU, shown in green, is of normal duration prior to the new chunk boundary. Stream 1's AU, the unaltered or highest frame rate stream, will always be of normal duration.

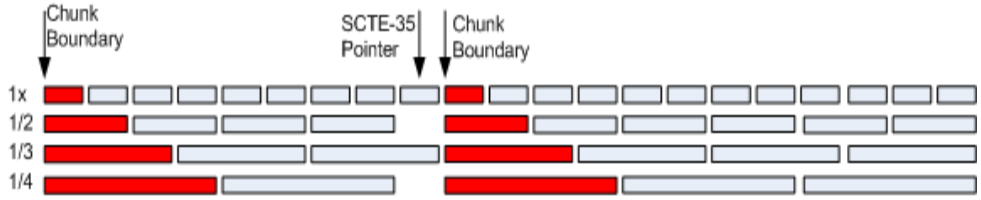


Figure 48 - Skipping AUs leading up to the next Chunk

Option 2: In Figure 48, prior to the new chunk boundary, each stream does not encode its next AU if the duration of this AU would extend beyond the new chunk boundary. This would result in a time discontinuity in some streams. In this example, this occurs in streams 3 and 4. The figure represents this with gaps in the timeline.

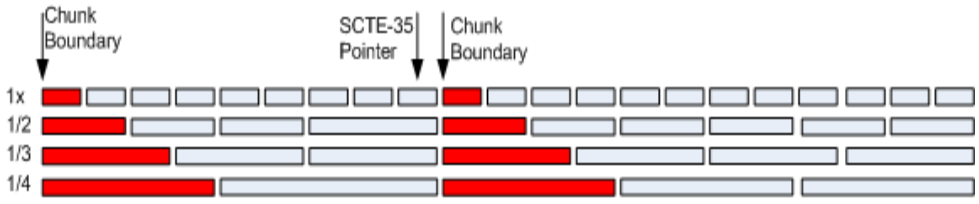


Figure 49 - Skipping AUs leading up to the next Chunk and Extending previous AU's Duration

Option 3: Figure 49's method is similar to Figure 48, content identical: prior to the new chunk boundary, each stream does not encode its next AU if the duration of this AU would extend beyond the new chunk boundary. However, the duration of the AU prior to this skipped AU is extended to the new chunk boundary. While the same number of AUs is represented here as in Figure 48, there is no time discontinuity due to extending the duration of AUs. In this example, this occurs in streams 2 and 4, which were both extended by one AU time interval.

Option 4: In this use case, a new chunk is not started on the first source AU >= the splice point. Instead the first AU >= splice point that is aligned across all streams is chosen. In the example, depicted in Figure 50, we see that the first AU that is aligned across all streams does not occur until three frames after the first AU >= splice point.

Figure 50 - Selecting first AU aligned across four bitrates

In the example depicted in Figure 51, there are only two frame rates and thus, in the worst case, the first AU that is aligned across all streams will occur not more than one frame after the first AU >= splice point.

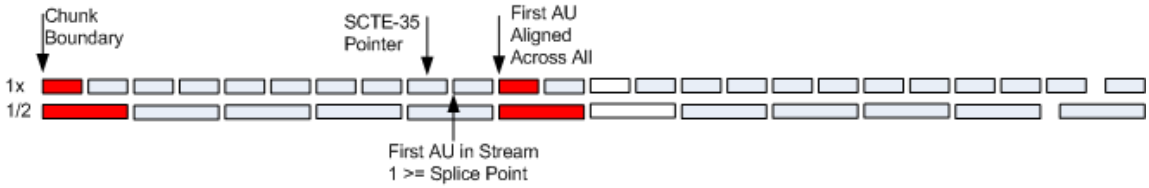


Figure 51 - Selecting first AU aligned across two bitrates

While Option 4 does not create a boundary on the first AU >= splice point, if the number of AUs following this AU, as in Figure 51, is small, this option is ideal because it does not influence the frame rates of the low-rate streams by encoding extra frames as in Option 1 or not encoding frames as in Options 2 and 3. However, it is not anticipated that the SCTE 35 [6] message(s) for this splice point would be modified to account for the new location. If any downstream processing could not accept this, then this option is probably not to be used.

V.4 Boundaries for Scene Changes in Multi Framerate Streams

The examples in the previous section considered an advertising splice point (in or out) but the same considerations and techniques apply to other factors creating new chunks at boundaries that do not naturally align across all frame rate streams. For example, consider Figure 52, which shows a scene change occurring in close proximity to an anticipated chunk boundary.

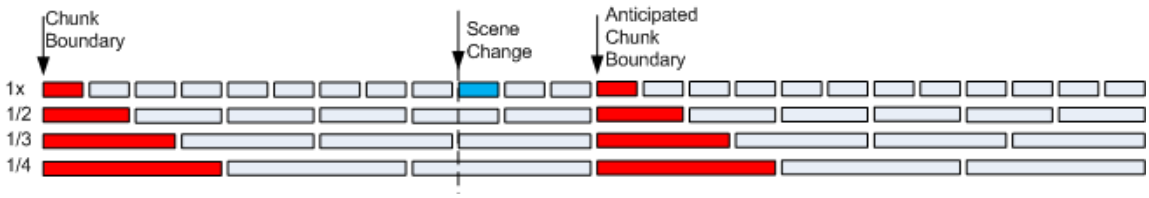


Figure 52 - Scene change in close proximity to anticipated chunk boundary

With no special handling, the resultant encode could produce something like Figure 53, where the scene change frames in each stream (blue) are encoded as I-frames and the boundary frames (red) are encoded as IDR-frames. This approach can impact picture quality due to the excessive bits used to encode the intra encoded frames.

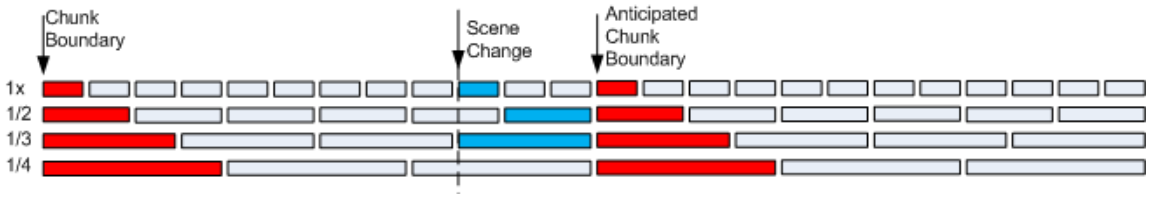


Figure 53 - I (blue) and IDR (red) frames due to Scene Change and Chunk Boundaries

Ideally, the upcoming anticipated boundary is moved up and relocated to the AU of the scene change. The end result could be something like Figure 54, which is similar to the approach shown in Figure 47. Approaches similar to Figure 48 and Figure 49 are also possible.

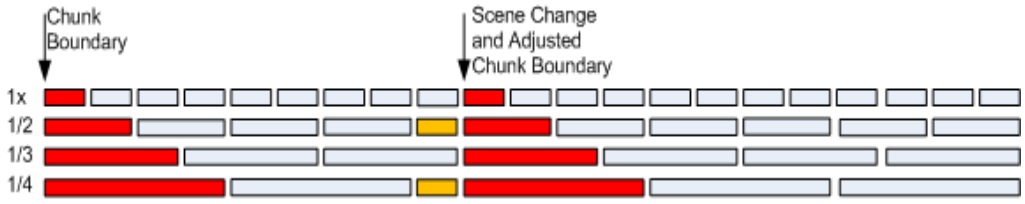


Figure 54 - Adjusted Chunk Boundary at Scene Change