INTERNATIONAL STANDARD

ISO/IEC 23001-13

First edition
2019-07

Information technology — MPEG systems technologies —

Part 13: Media orchestration

Reference number
ISO/IEC 23001-13:2019(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO/IEC 23001 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Audiovisual equipment is pervasive; everyone with a smartphone, a tablet or a laptop has both recording and display equipment at their disposal, usually connected to a local or a public network. This equipment is increasingly sophisticated, with higher resolutions and better lenses (for video), often multiple microphones (for audio), coupled with other sensors (e.g. for geolocation) and increasingly sophisticated processing capabilities. These devices can not only decode in real time, but usually also perform decent encoding. Sensors and displays come in the form of personal smart phones, but also include smart watches, omnidirectional cameras and a variety of virtual reality and augmented reality head-mounted devices. All these devices can either be a source of multimedia content (audio, video or audiovisual) or consume such content. Often, devices play both roles.

The proliferation of these multimedia-capable devices combined with ever-increasing bandwidth, including mobile bandwidth, necessitates better and standardized mechanisms for coordinating such devices, the associated media streams and the available resources, like media processing and transmission. This process is called “orchestration”. Orchestration includes coordination in time (synchronization) and in space, as well as the coordination of computational resources to perform the coordination functions.

This document was developed to address the need for specifying this coordination between capture devices (“sources”) and play-back devices (“sinks”) in a heterogeneous environment. In this context, “heterogeneous” refers to the fact that devices can be of different nature (e.g. a mix of consumer and professional devices) and on different networks.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of patents.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights. The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from:

Electronics and Telecommunications Research Institute

218 Gajeong-ro, Yuseong-gu, Daejeon, 34129 Korea

Koninklijke KPN N.V.

Wilhelminakade 123, 3072 AP Rotterdam, The Netherlands

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — MPEG systems technologies —

Part 13:
Media orchestration

1 Scope

This document specifies an architecture for media orchestration, as well as associated messaging and control, timed metadata, the carriage of that timed metadata, and orchestration data.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 13818-1:2019, *Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems*

ISO/IEC 14496-3:2009, *Information technology — Coding of audio-visual objects — Part 3: Audio*

ISO/IEC 14496-3:2009/Amd 5:2015, *Information technology — Coding of audio-visual objects — Part 3: Audio— Amendment 5: Support for Dynamic Range Control, New Levels for ALS Profile, and Audio Synchronization*

ISO/IEC 14496-12:2015, *ISO Base Media File Format*

ISO/IEC 14496-12:2015/Amd 1:2017, *ISO Base Media File Format — Amendment 1: DRC extensions*

ISO/IEC 23001-10:—[1)], *Information technology — MPEG systems technologies — Part 10: Carriage of timed metadata metrics of media in ISO base media file format*

ISO/IEC 23005-6:2019, *Information technology — Media context and control — Part 6: Common types and tools*

ISO/IEC 23008-1, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 1: MPEG media transport (MMT)*

ETSI TS 103 286 02:2017, *Digital Video Broadcasting (DVB); Companion Screens and Streams; Part 2: Content Identification and Media Synchronization* [online]. Available at http://www. etsi. org/deliver/etsi_ts/103200_103299/10328602/

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax* [online]. Edited by T. Berners-Lee. January 2005. Available at https://www.ietf.org/rfc/rfc3986.txt

IETF RFC 6455:2011, *The WebSocket Protocol* [online]. Edited by I. Fette. December 2011. Available at https://www.ietf.org/rfc/rfc6455.txt

IETF RFC 5905, *Network Time Protocol Version 4: Protocol and Algorithms Specification* [online]. Edited by D. Mills. June 2010. Available at https://tools.ietf.org/html/rfc5905

1) Under preparation. Stage at the time of publication: ISO/IEC DIS 23001-10:2019.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
content
media data
data (3.3) that can be rendered, including audio, video, text, graphics, images, haptic and tactile information

Note 1 to entry: This data can be timed or non-timed.

3.2
controller
elementary function that controls one or more other elements, outputs *control data* (3.5) and can receive control data

3.3
data
bytes of information that are carried, processed or stored

3.4
timed data
data (3.3) that has an intrinsic *timeline* (3.25)

3.5
control data
data (3.3) exchanged for messaging and control

3.6
metadata
data (3.3) about other data that cannot be rendered independently and can affect rendering, processing or orchestration of the associated *media data* (3.1)

3.7
timed metadata
metadata (3.6) that has an intrinsic *timeline* (3.25)

3.8
orchestration data
data (3.3) that orchestrates any number of *timed data* (3.4) streams

Note 1 to entry: Orchestration data itself can also be timed data.

3.9
device
physical entity that can embody a number of elementary functions

3.10
m-processor
media processor
elementary function that can process *media data* (3.1) and/or *metadata* (3.6), that can receive media data and/or metadata, that can optionally receive *orchestration data* (3.8), that can output media data, metadata or both, and that can optionally exchange *control data* (3.5)

3.11
media orchestration
orchestration of media and *metadata* (3.6) capture, processing and presentation involving multiple *devices* (3.9)

3.12
media orchestration session
coherent application context that includes media capture and/or media processing and/or media rendering

3.13
media stream
timed *media data* (3.1)

3.14
mos more
mean opinion score of the subjective quality of a piece of timed *media data* (3.1)

3.15
sink
elementary function that can receive *media data* (3.1), and optionally *orchestration data* (3.8) and/or *metadata* (3.6), can present the media data, can optionally output metadata and can optionally exchange *control data* (3.5)

3.16
source
elementary function that can output *media data* (3.1) and/or *metadata* (3.6) and can optionally exchange *control data* (3.5)

3.17
timestamp
number that is used to identify an audio sample, video frame or *metadata* (3.6) sample

EXAMPLE MPEG-2 TS PTS (presentation timestamp) or ISOBMFF CT (composition time).

Note 1 to entry: The meaning of the term "timestamp" depends on the context. Within the MPEG context, the term is usually used as a single number to identify a point on the timeline of an MPEG-2 transport stream or within an ISOBMFF file. In the context of DVB-CSS (ETSI TS 103 286 02), the term "timestamp" is used to provide an association between a wall clock time and a point on a certain media timeline. This document uses the "dvb-css" prefix to flag where the DVB-CSS (ETSI TS 103 286 02) notion of timestamp is used.

3.18
dvb-css timestamp
pair of two values each of which represents a time value on a timeline such that the two-time values correspond to the same moment in time

[SOURCE: ETSI TS 103 286 02:2017, 3.1]

3.19
dvb-css control timestamp
dvb-css timestamp (3.18) representing the time (in relation to the wall clock) at which a *synchronization client* (3.24) is recommended to present a particular point in time (in relation to the synchronization *timeline* (3.25)) of its *timed data* (3.4)

[SOURCE: ETSI TS 103 286 02:2017, 3.1, modified — «timed content» has been replaced with «timed data» and the note has been deleted]

3.20
dvb-css correlation timestamp
time values on two or more *timelines* (3.25) that correlate with each other

[SOURCE: ETSI TS 103 286 02:2017, 3.1, modified — «ore more" has been added and the note has been deleted]

3.21
dvb-css earliest presentation timestamp
dvb-css timestamp (3.18) representing the earliest time (in relation to the wall clock) at which a *synchronization client* (3.24) believes it can present a particular point in time of its *timed data* (3.4)

[SOURCE: ETSI TS 103 286 02:2017, 3.1, modified — "(in relation to the Synchronization Timeline) of its Timed Content" has been replaced with "of its timed data" and the note has been deleted]

3.22
user
human that uses one or more *device(s)* (3.9) and/or service(s)

Note 1 to entry: A user can be a producer and consumer of media, and both at the same time.

3.23
media synchronization application server
elementary function that coordinates the process of obtaining a shared agreement on the progress of *timelines* (3.25) among all *synchronization clients* (3.24)

Note 1 to entry: This is for the purpose of enabling the synchronization clients to present timed data simultaneously with respect to each other.

[SOURCE: ETSI TS 103 286 02:2017, 3.1]

3.24
synchronization client
elementary function that aligns its presentation of *timed data* (3.4) with other synchronization clients by communicating with a *media synchronization application server* (3.23)

[SOURCE: ETSI TS 103 286 02:2017, 3.1, modified — "wishes to align" has been replaced with "aligns" and "timed content" has been replaced with "timed data"]

3.25
timeline
reference frame for describing time, represented as a linear scale against which time can be measured for a particular system

Note 1 to entry: This could manifest in various ways, such as: as a local oscillator, the progress of a broadcast or the time position within an item of media content.

[SOURCE: ETSI TS 103 286 02:2017, 3.1]

3.26
timeline correlation
correlation between two *timelines* (3.25)

[SOURCE: ETSI TS 103 286 02:2017, 3.1]

3.27
wall clock
linear monotonic clock that is not assumed to represent real date and time

Note 1 to entry: The wall clock is intended for sharing between two or more entities for the purposes of having a common synchronized time reference frame.

Note 2 to entry: DVB-CSS (ETSI TS 103 286 02) wall clock is NOT a "UTC-type date&time". It has a random offset and it has no leap seconds or leap years. It is a clock that ticks like a metronome, and it has only a local meaning in the context of media synchronization between devices. This is a different notion for a timestamp than what is commonly used across MPEG. This document uses the definition from ETSI TS 103 286 02 to maintain consistency with that specification.

[SOURCE: ETSI TS 103 286 02:2017, 3.1, modified — Note 2 to entry has been added]

3.28
wall clock synchronization
achieving a shared agreement on the progress of time of a *wall clock* (3.27)

Note 1 to entry: This is a process that takes place between devices.

[SOURCE: ETSI TS 103 286 02:2017, 3.1]

4 Abbreviated terms, relationships between terms and mnemonics

4.1 Abbreviated terms

The following abbreviated terms are used in this document:

CSA	Companion screen application
CT	Composition time
DROP	Distinctive region or pattern
GPS	Global positioning system
MORE	Media orchestration
MORE-TS	MORE timeline synchronization (interface)
MORE-WC	MORE wall clock (interface)
MSAS	Media synchronization application server
URL	Uniform resource locator
PTS	Presentation timestamp
SC	Synchronization client
SNR	Signal-to-noise ratio
WCC	Wall clock client
WCS	Wall clock server

4.2 Relationships between defined terms

Figure 1 sketches relationships between some of the defined terms, including synonyms.

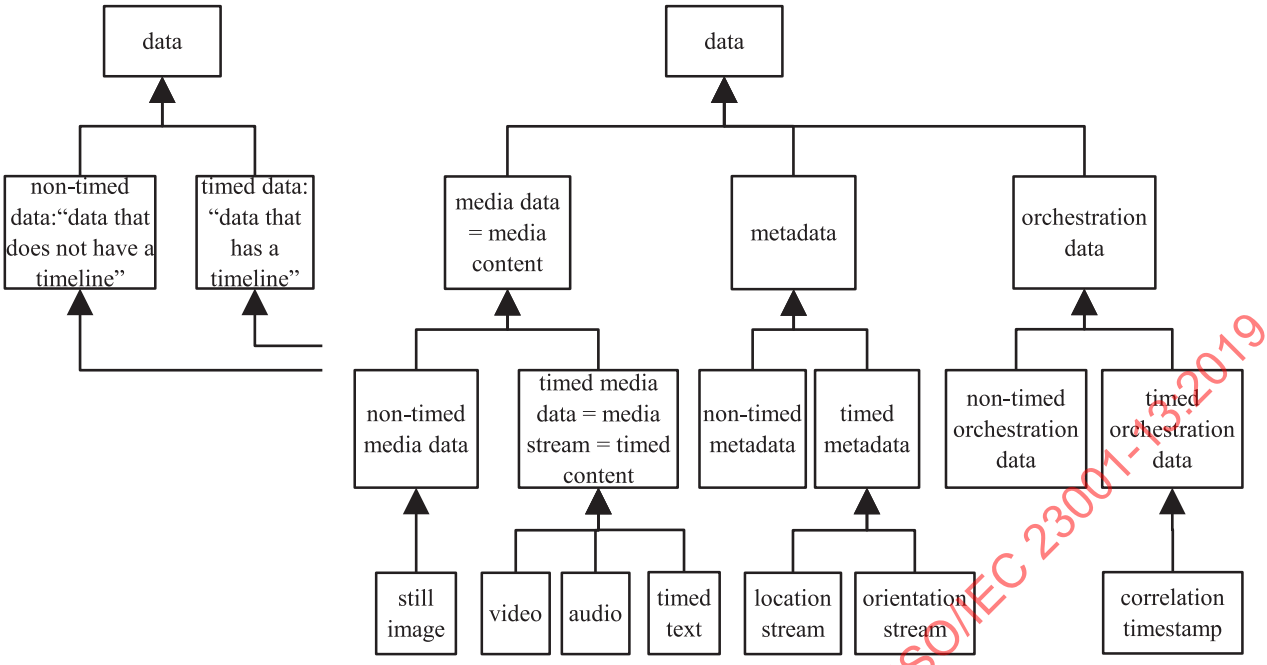


Figure 1 — Relationships between defined terms

4.3 Mnemonics

rpchof remainder polynomial coefficients, highest order first

bslbf bit string, left bit first

fsbf float, sign bit first

uimsbf unsigned integer, most significant bit first

5 Architecture

5.1 General

The term "media orchestration" denotes the orchestration of media and metadata capture, presentation and processing involving multiple devices.

Orchestration of media capture refers to metadata and control signalling to determine which device captures what, when and how. **What** to capture is about what media to capture and which part of that media should be captured. **When** to capture is about capture synchronization with other devices, as well as start and stop of capture. **How** to capture is about location, orientation, capture capabilities, frame rate, resolution, microphone gain, white balance settings, etc. as well as codecs used, metadata delivered and possible processing to be applied.

Orchestration of media presentation refers to metadata and control signalling to determine which device presents what, when and how. **What** to present is about what media to retrieve and which parts of that media should be presented. **When** to present is about presentation synchronization with other devices. **How** to present is about where exactly to play out something, e.g. positioning of a media part in a screen, positioning of an audio object in a room and possible processing to be applied.

Orchestration of processing refers to metadata and control signalling for applying processing to sets of captured media and/or metadata. This includes single-media processing (e.g. media synchronization in case of transcoding), as well as processing of multiple media and/or metadata together, e.g. performing

video stitching, changing arrangements of media in space and time, or automated editing and selection processes.

A **media orchestration session** is a coherent application context that includes media capture and/or media rendering and/or media processing. It is defined by its scope: a concert lasting one evening in one concert hall, a week-long festival in a park, the vacations of a family in Norway over a month, the synchronized presentation of media across multiple screens, the reconstruction of a crime captured over many different cameras…

Media orchestration is independent from media type or encoding or packaging or distribution.

DVB CSS introduces the concept of timed data, see TS 103 286 02, 5.2.1. DVB uses the term "timed content" for this concept. Timed data has an intrinsic timeline. It may have a start and/or end (e.g. content on-demand) or it may be continuous (broadcast). It may be atomic (video-only) or it may be composite (A/V, 3D-audio, multiplex). Classic examples of timed data are video, audio and timed text.

In the context of media orchestration, streams of location and orientation as well as other timed sensor outputs are also timed data; see Figure 2. This can also be called timed metadata, as defined in DVB CSS.

NOTE 1 The notion of time has various applications in media orchestration. It can pertain to the *delivery* or to the media itself, or the *presentation.* It can also pertain to the *capture* of media.

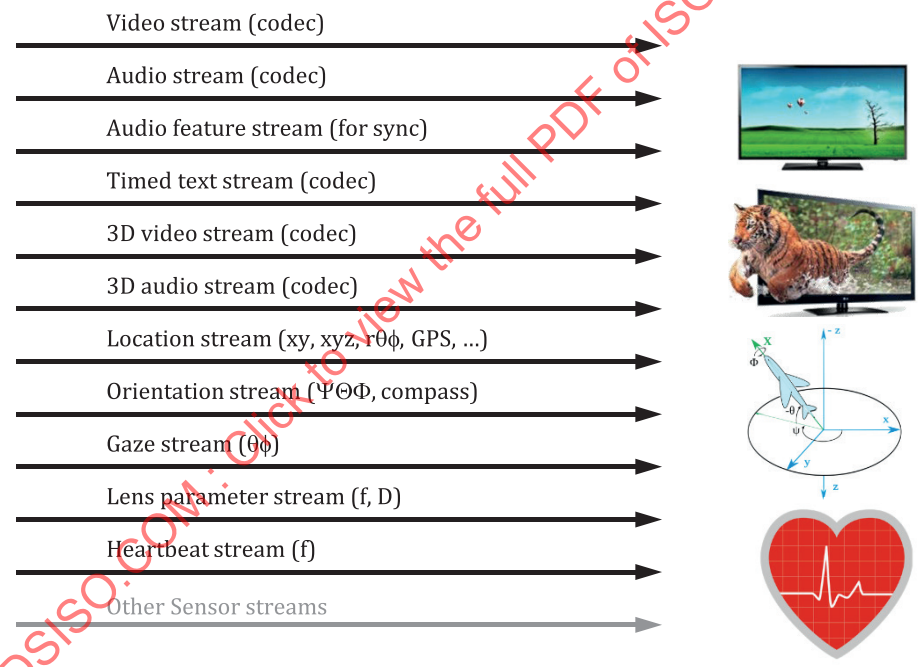


Figure 2 — Examples of timed data in the context of media orchestration

This document distinguishes a number of functional entities for media orchestration, all derived from the same abstract entity. It first describes the entities and then gives an example of how they can be configured for an actual use case.

Table 1 shows the meaning of the arrows used in Figures 3 to 6.

Table 1 — Types of data

	media data (timed or non-timed, unidirectional)
	metadata (timed or non-timed, unidirectional)
	orchestration data (timed or non-timed, unidirectional)

Table 1 *(continued)*

◆----------→	control data for messaging and control (non-timed, possibly bidirectional)

Figures 3 to 6 illustrate the concepts of source, sink, m-processor and controller, respectively, as defined in Clause 3.

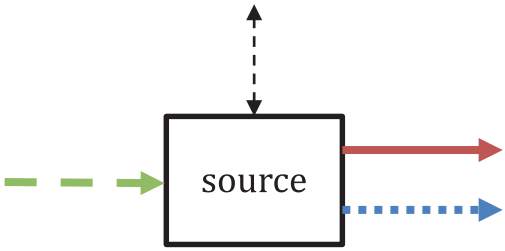


Figure 3 — Source

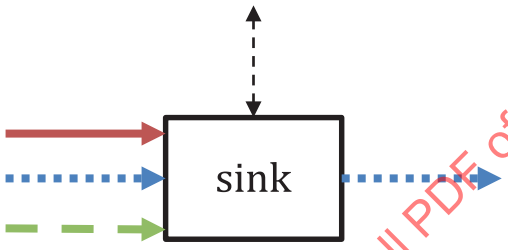


Figure 4 — Sink

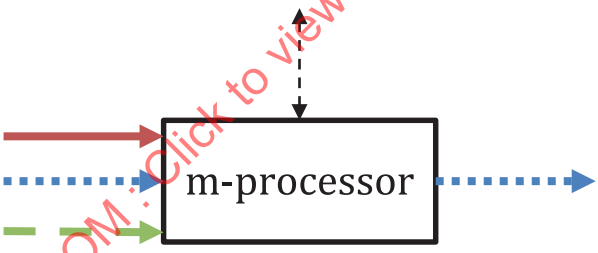


Figure 5 — m-processor

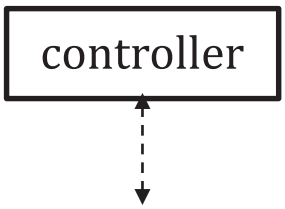


Figure 6 — Controller

A device may have one or more sources. Each source may generate timed media data or timed metadata. When an m-processor is located within the same device, the device can also generate orchestration data.

The timed media data and metadata generated from sources can be delivered to other functional elements, i.e. m-processor and sink. The timed metadata may also be multiplexed with associated media data.

NOTE 2 The following applies:

— a user device can (and very often will) have both source and sink functions;

— an application server can combine controller and m-processing functions;

— m-processing function can include transcoding, adding or changing timelines, multiplexing, demultiplexing, selection (editor), stitching, tiling (e.g. using MPEG DASH spatial relationship description), translation, extracting timed metadata (e.g. CDVS metadata) and more.

An example instantiation of this model is as follows:

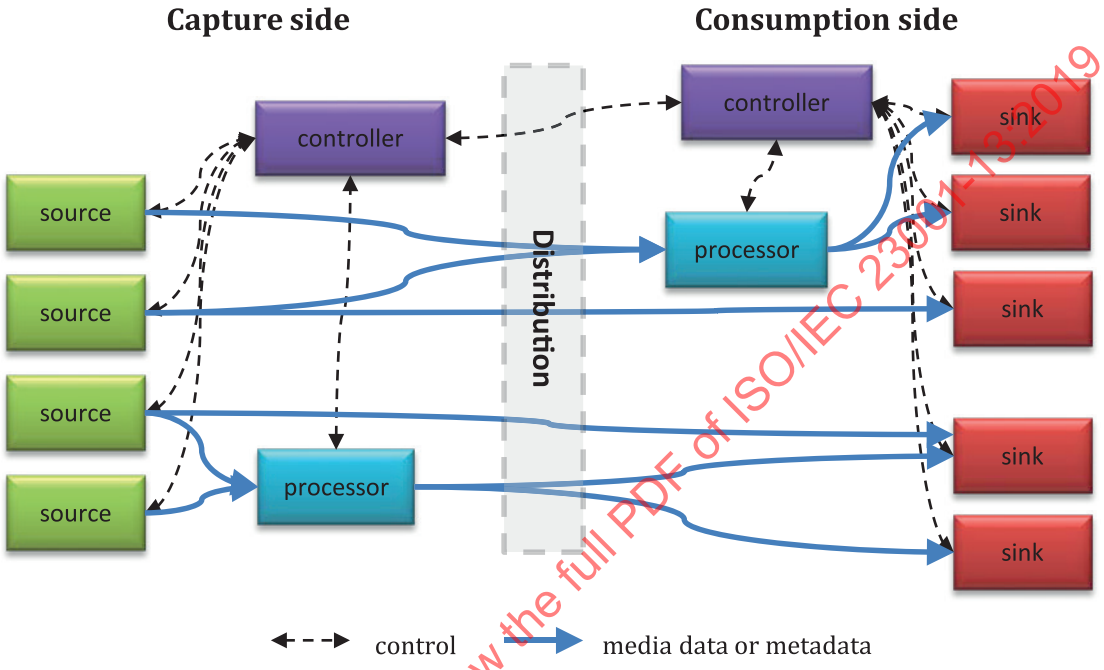


Figure 7 — Examples of timed data in the context of media orchestration

This model shows a complex scenario where many sources contribute to an experience with many orchestrated sinks.

Figure 7 is broadcast-oriented, with a clear distinction between the production side and the consumption side. The architecture also supports mixed cases, for example shared experiences, where sinks are used for broadcast media consumption in orchestration with sources and sinks used for communication between users.

Annexes B and C provides illustrations of control information and its exchanges.

5.2 Architecture for temporal orchestration

5.2.1 General

Temporal orchestration, or media synchronization, is the process of time alignment of timed media data and metadata. When media data and metadata are captured at different source devices, their timelines will generally be different. Audio samples, video frames and metadata samples are dvb-css timestamped against the clock of the source device. Such dvb-css timestamps identify audio samples, video frames and metadata samples by their presentation timestamp (PTS, MPEG-2 transport stream), composition time (CT, MPEG-4 ISOBMFF) or external timeline (e.g. TEMI, MPEG-2 timeline for external data). Clocks ("wall clocks") of source devices are typically free running and have clock skew, clock drift and clock drift variation. Moreover, transcoders and multiplexers may change the time base of timed media data and metadata. Finally, there are typically significant differential delays between distribution chains

to different sink devices. As a consequence, timed media data and metadata can arrive severely out of sync at sink devices, and temporal orchestration is needed to get them synchronized again.

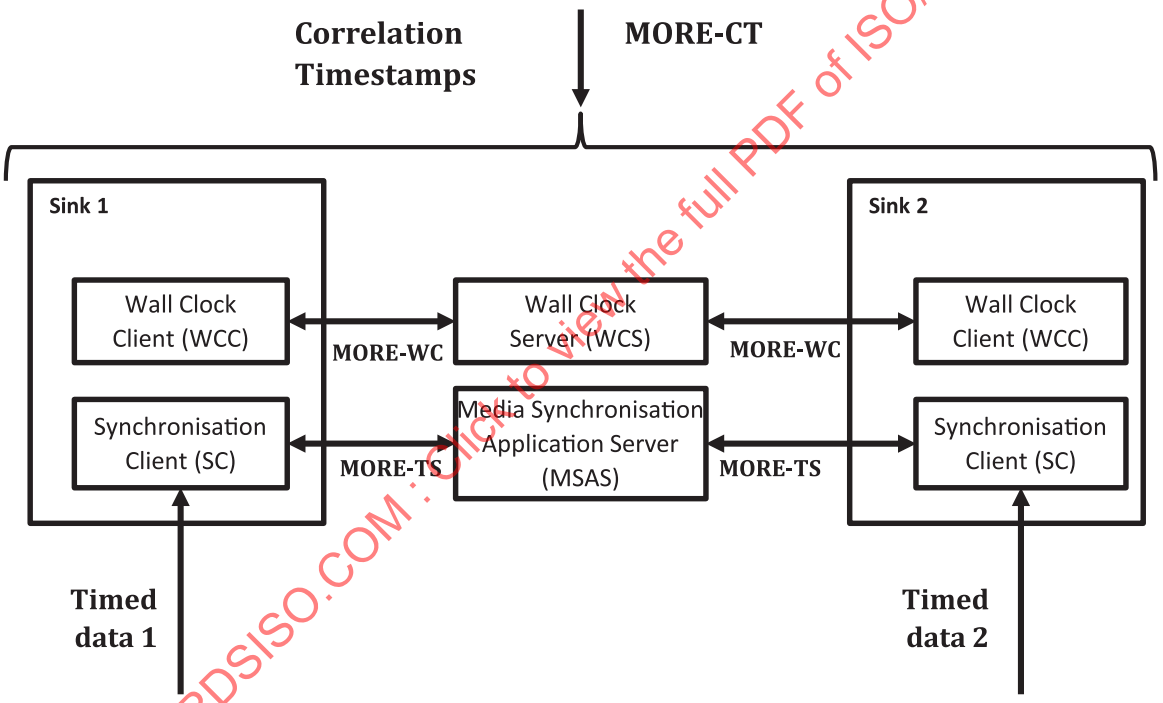
NOTE Clock skew is an offset. Clock drift is where two clocks have a nominal relationship to each other that is not observed in practice (e.g. an audio clock that claims to be 48 KHz but does not actually sample at that rate). Clock drift variation is when the actual sampling rate is not only off, but also not constant.

The introduction clause of ETSI TS 103 286 02:2017 provides a useful introduction into sink-side media synchronization, including use cases, the need for content identification functionality and the need for playback synchronization functionality, as well as issues with timelines and a conceptual model for time-controlled playback.

5.2.2 and 5.2.3 provide architectures for temporal orchestration at the sink and source sides, respectively.

5.2.2 Sink-side architecture for temporal orchestration

Figure 8 provides a sink-side architecture for temporal orchestration, based on ETSI TS 103 286 02. The purpose of this architecture is to synchronize two or more pieces of incoming timed data (timed media, timed metadata or timed orchestration data) within/between one or more sinks.



SOURCE: ETSI TS 103 286 02, reproduced with the permission of DVB

Figure 8 — Sink-side architecture for temporal orchestration

The sink-side architecture has the following elements.

— There are sinks, each of which receives pieces of timed data. Each of those pieces of timed data can be timed media data, timed metadata or timed orchestration data.

— Each sink has a synchronization client (SC) per received piece of timed data (stream), which takes care of the synchronization of the timed data, delaying the presentation/use of the timed data when appropriate.

— An SC coordinates the timeline synchronization of its timed data with a media synchronization application server (MSAS) via the MORE-TS interface.

— Each sink has a wall clock client (WCC), which provides the wall clock time that is used for the dvb-css timestamping in the MORE-TS interface.

— A WCC synchronizes its wall clock against a wall clock server (WCS) via the MORE-WC interface.

— dvb-css correlation timestamps are a form of orchestration data that may be received and processed by SCs and/or MSAS via the MORE-CT interface; see also 9.3. dvb-css correlation timestamps are used to perform timeline translations, in the generic case where the timelines of different received timed data are different.

NOTE 1 Pieces of timed data (e.g. media streams) usually have different timelines, as each source typically has its own local oscillator. A common example is the case where timed data 2 is a transcoded (and remultiplexed) version of timed data 1.

Figure 9 shows a simplified case of the above architecture. This simplified case corresponds with the ETSI TS 103 286 02 inter-device media synchronization standard. The following applies to this example:

— A television device integrates WCC, WCS, SC and MSAS functions.

— Only timed media data is considered, not timed metadata or timed orchestration data.

— The timed media data at the companion screen application (CSA) is enslaved to the timed media data received by the TV device, ensuring that the playout of the timed media data at the TV device is never delayed.

— dvb-css correlation timestamps are received and processed by the CSA (design choice by ETSI TS 103 286 02).

— Timed media data are presented synchronized between TV device and CSA.

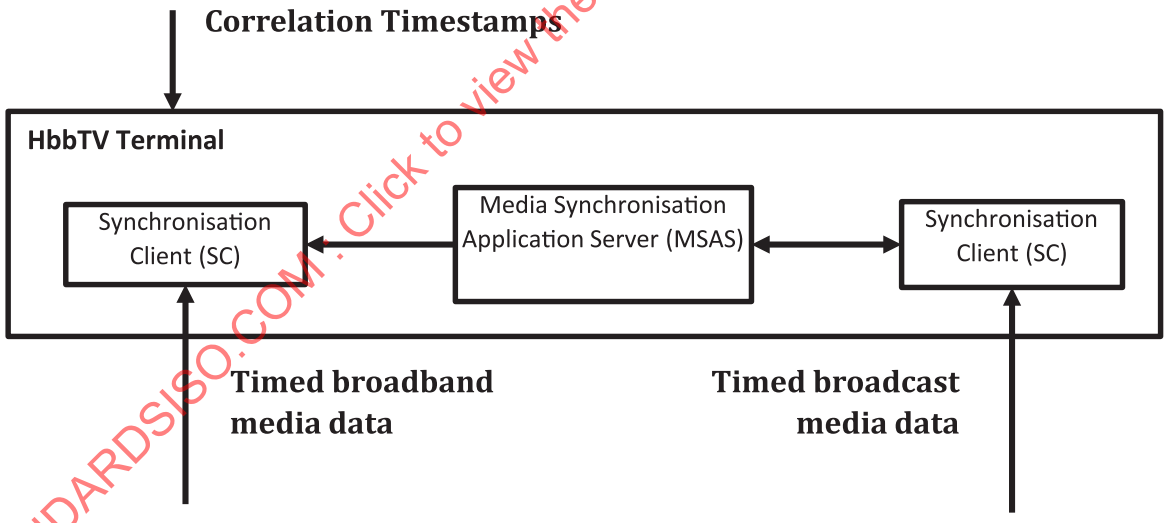


Figure 9 — Simplified case of the sink-side architecture, corresponding to HbbTV 2.0 ETSI TS 102 796

Figure 10 illustrates temporal orchestration at the sink side.

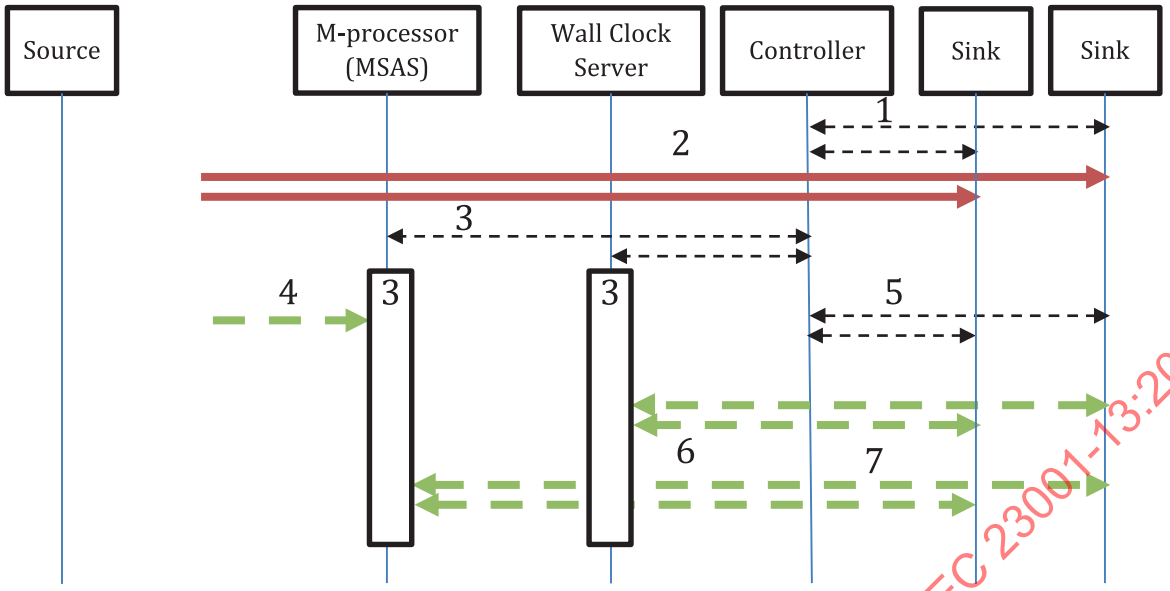


Figure 10 — Orchestration of timeline synchronization at the sink side

The sequence diagram of Figure 10 has the following steps.

1. Discovery/registration processes, during which one or more sinks are associated with a controller for a specific media orchestration session.
2. The sinks start receiving timed media data and/or timed metadata. The sources of these timed media data and/or timed metadata have independent and free-running (not genlocked) clocks, hence necessitating sink-side timeline synchronization.
3. The controller initiates instances of a wall clock server and a media synchronization application server.

   NOTE 2 The control signalling for this is out of scope of this document.

4. The media synchronization application server starts receiving dvb-css correlation timestamps.

   NOTE 3 This example assumes that it is the MSAS that calculates or processes the dvb-css correlation timestamps In other examples, it can be the sinks that process the dvb-css correlation timestamps, e.g. TV-to-companion synchronization in DVB-CSS ETSI TS 103 286 02.

5. The controller provides the sinks with the end-point addresses of the wall clock server and the media synchronization application server; see also below.
6. The sinks synchronize their wall clocks with the wall clock server using the wall clock protocol specified in 6.1.
7. The sinks start exchanging dvb-css presentation timestamps and dvb-css control timestamps with the media synchronization application server using the sink-side timeline synchronization protocol specified in 9.1. The media synchronization application server uses the dvb-css correlation timestamps to translate dvb-css presentation timestamps between the time bases (timelines) of the different timed media data and/or timed metadata and calculate dvb-css control timestamps, which sinks may use to delay and achieve synchronous playout.

   NOTE 4 The source is drawn only for illustration purposes. It is not involved in this orchestration example.

The wall clock server end-point address is represented as a URL of the form "udp://<ip-address>:<port-number>", where <ip-address> and <port-number> are the IP address and port number, at which the wall clock server is providing its service endpoint; see also ETSI TS 103 286 02:2017, 8.4.

The media synchronization application server end-point address is a websocket URI as specified in IETF RFC 6455:2011, Clause 3, at which the media synchronization application server is providing its service endpoint; see also ETSI TS 103 286 02:2017, 9.3.

5.2.3 Source-side architecture for temporal orchestration

Figure 11 provides a source-side architecture for temporal orchestration. It is a mirror image of the sink-side architecture of Figure 8. The purpose of this architecture it to enable the future synchronization of two or more pieces of timed data (timed media, timed metadata or timed orchestration data) that are produced by one or more sources. The word "future" implies that the actual synchronization takes place at playout with the sink(s). To enable this, sources report about the production timing of their produced timed data.

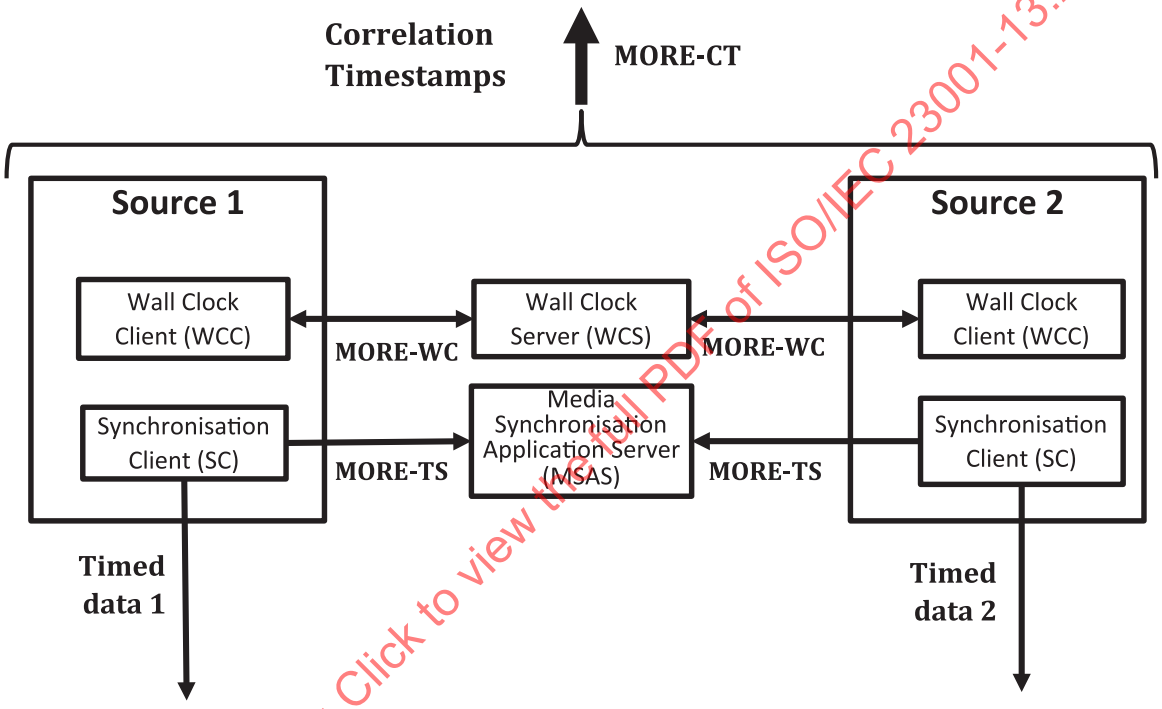


Figure 11 — Source-side architecture for temporal orchestration

The source-side architecture has elements similar to the sink-side architecture, with the following differences.

— The direction of timed data (timed media data, timed metadata and timed orchestration data) is outgoing.

— The SCs report to the MSAS, but they do not need to delay the produced timed data.

— The dvb-css correlation timestamps (MORE-CT interface, see also 9.3) are outgoing, to be used for the future synchronization of timed data at the sink side.

NOTE 1 A media-orchestrated system can have multiple independent wall clock servers (WCS) and media synchronization application servers (MSAS). For example, an MSAS used at the source/production side can be different from an MSAS used at the sink/consumption side.

Simplified cases are possible, similar to the ones described for the sink-side architecture; for example, WCS and MSAS may be integrated with a source.

A single source device may integrate multiple sources, producing multiple pieces of timed data (streams).

Figure 12 provides an example of the simplified case that integrates a source and a sink function. This example is an m-processor (e.g. a transcoder or a CDN node) that remultiplexes and possibly delays the timed media data, resulting in that the timeline for the incoming timed media data is different than for outgoing timed media data. The m-processor produces dvb-css correlation timestamps in order to enable future synchronization between the two pieces of timed media data.

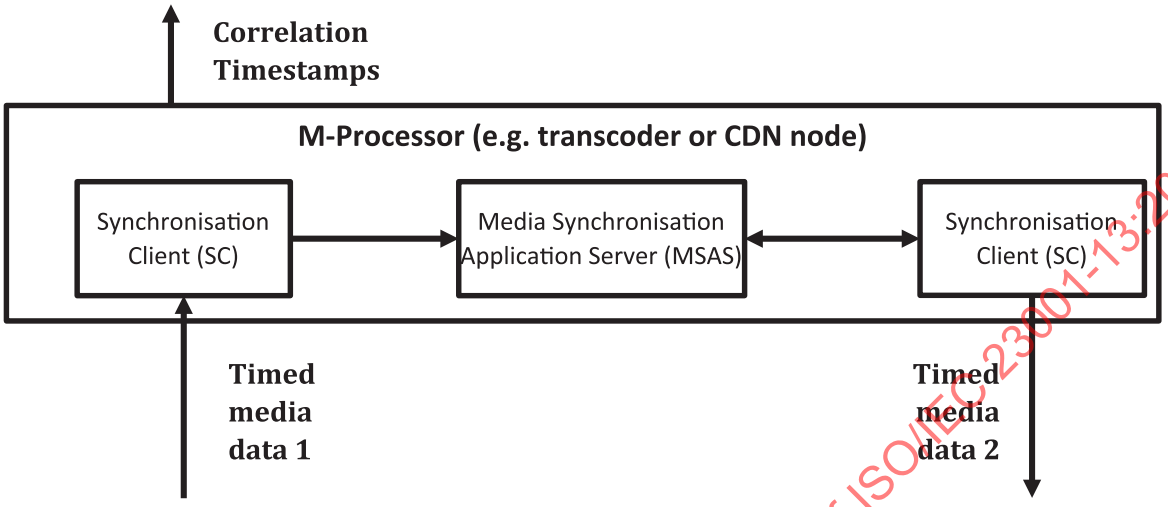


Figure 12 — Simplified source-side architecture for temporal orchestration

Figure 13 illustrates temporal orchestration at the source side.

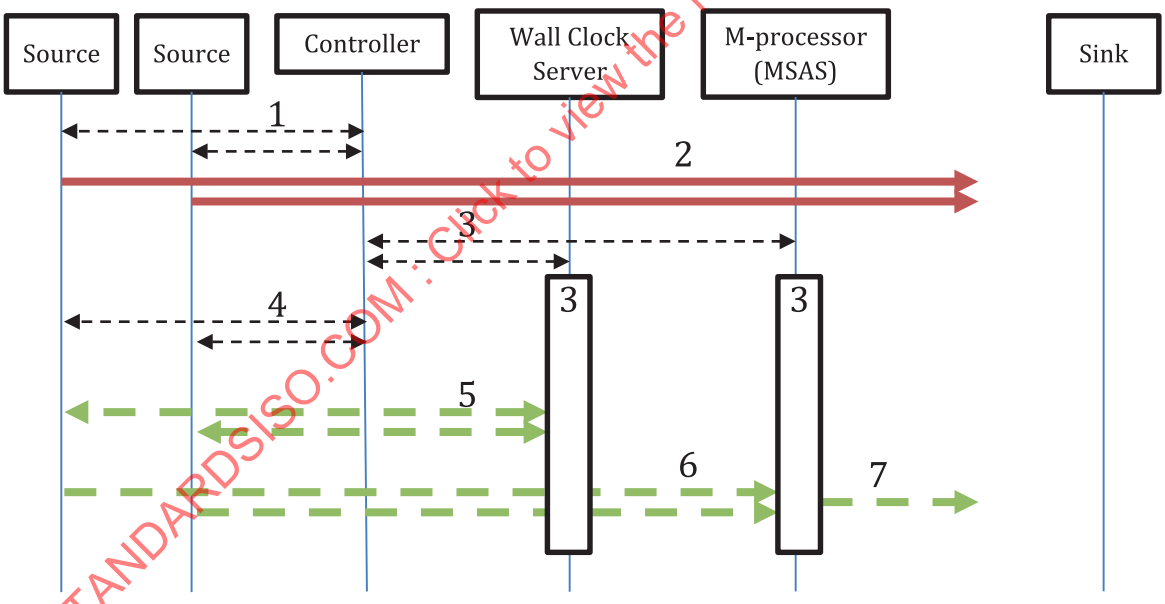


Figure 13 — Orchestration of timeline synchronization at the source side

The sequence diagram of Figure 13 has the following steps.

1. Discovery/registration processes, during which one or more sources are associated with a controller for a specific media orchestration session.
2. The sources start sending timed media data and/or timed metadata. The sources have independent and free-running (not genlocked) clocks, hence necessitating source-side timeline synchronization.
3. The controller initiates instances of a wall clock server and a media synchronization application server.

NOTE 2 The control signalling for this is out of scope of this document.

4. The controller provides the sources with the end-point addresses of the wall clock server and the media synchronization application server, see also below.

5. The sources synchronize their wall clocks with the wall clock server using the wall clock protocol specified in 6.1.

6. The sources start sending dvb-css capture timestamps to the media synchronization application server using the source-side timeline synchronization protocol specified in 9.2 and Annex A.

7. The media synchronization application server starts calculating and sending dvb-css correlation timestamps based on the received dvb-css capture timestamps.

NOTE 3 The sink is drawn only for illustration purposes. It is not involved in this orchestration example.

5.3 Architecture for spatial orchestration

5.3.1 General

Spatial orchestration is the process of recognizing the spatial relationships between various media data (or elements like source, sink) using various types of metadata, and describing these relationships using orchestration data. There are various sources and sinks in a MORE session. Those sources capture something and send captured media data to m-processor. The m-processor needs to know which media data is related to what media data and how they are related. These relation data can be achieved with various metadata, such as GPS, position, orientation, camera parameters, image features, depth map, etc. Those relation data can be spatial orchestration data and spatial orchestration data can be helpful to carry out some processes of m-processor, such as stitching, tilting, etc. and furthermore, video summarization, user selection, generating VR contents, etc. On the sink side, when media data are consumed by sinks, sinks can communicate with m-processor with various types of metadata and receive spatial orchestration data for more interactive and immersive services.

5.3.2 M-processor side architecture for spatial orchestration

Figure 14 provides an m-processor side architecture for spatial orchestration. The purpose of this architecture is letting m-processor to gather media data with associated metadata from sources, recognise relationship between media data or sources, and generate orchestration data. Those metadata can be orientation (gaze) data, capture mask, absolute and/or relative locations of sources, etc. Also, m-processor can generate metadata which can be helpful to recognise relations between media data, such as image features, depth map, etc., by itself. All of those metadata can be used for spatial orchestration and generating spatial orchestration data.

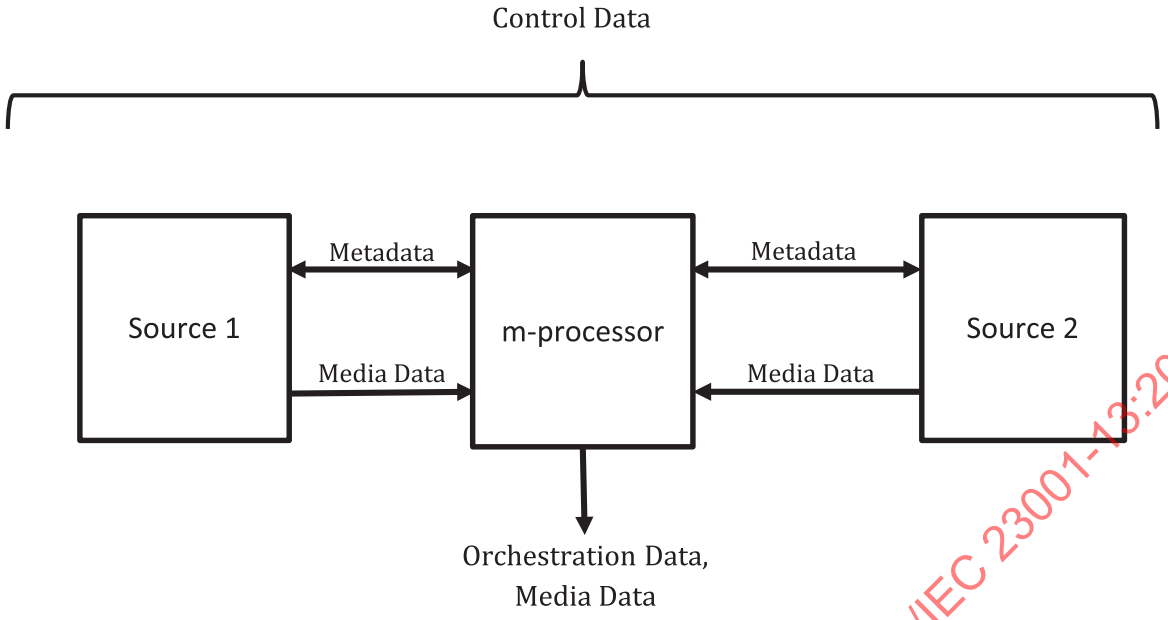


Figure 14 — A simple structure of end-to-end system in terms of spatial orchestration

Figure 15 illustrates spatial orchestration at the m-processor side.

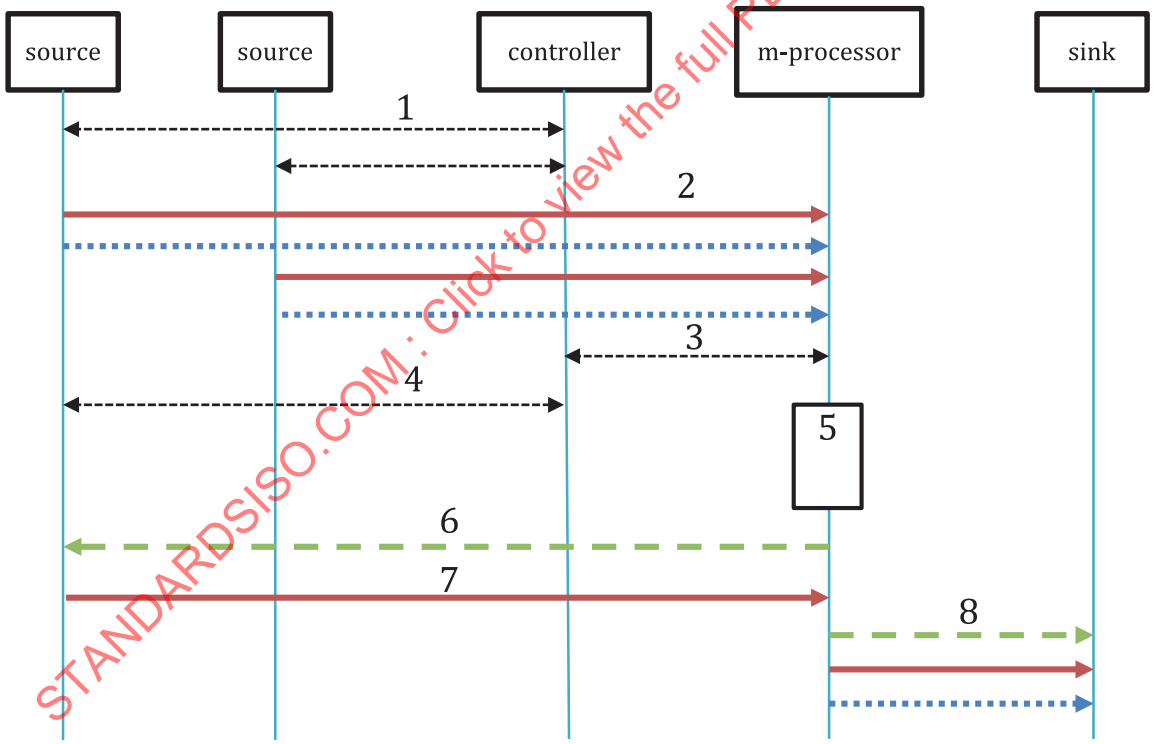


Figure 15 — Spatial orchestration at the m-processor side

The sequence diagram of Figure 10 has the following steps.

1. Discovery/registration processes, during which one or more sources are associated with a controller for a specific media orchestration session.

2. The sources start sending media data and/or metadata. The sources can have their own sensors, which can generate metadata associated with media data they generate.

3. The controller initiates an instance of an m-processor.

   NOTE 1 The control signalling for this is out of scope of this document.

4. The controller provides the sources with the end-point addresses of the m-processor.

5. The m-processor analyses the media data and metadata received from the sources and generates spatial orchestration data.

6. The m-processor sends the orchestration data to request appropriate media data and/or metadata, such as DROP, to source(s).

   NOTE 2 This process is optional. Thus, it can be skipped if unnecessary.

7. Source(s) sends appropriate media data and/or metadata as a reply for requests from the m-processor.

   NOTE 3 This process is optional. Thus, it can be skipped if unnecessary.

8. The m-processor writes orchestration data and starts to send them to sinks with associated media data and/or metadata.

   NOTE 4 The sink is drawn only for illustration purposes. It is not involved in this orchestration example.

5.3.3 Sink-side architecture for spatial orchestration

Figure 16 provides a sink-side architecture for spatial orchestration. The purpose of this architecture is to let the sink communicate with the m-processor by exchanging metadata and receive orchestration data. Those metadata can be orientation (gaze) data, presenting device order, region of interest (ROI), absolute and/or relative locations of sinks, etc. The m-processor can use the spatial metadata from the sinks to determine what media data to provide to those sinks.

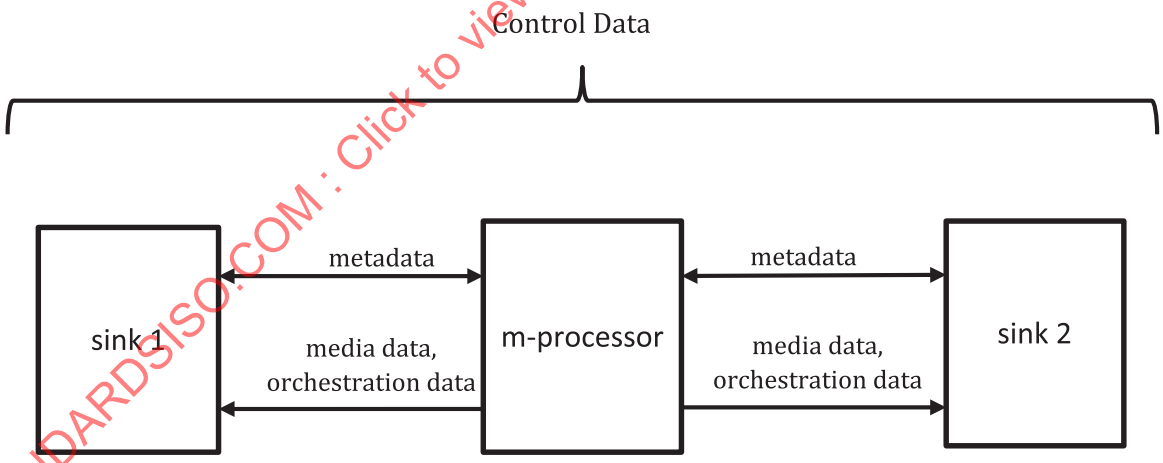


Figure 16 — Sink-side architecture for spatial orchestration

The sink-side architecture has elements similar to the m-processor architecture with the following differences.

— The direction of media data is outgoing, which means the sink receives media data from the m-processor instead of sending them.

— The m-processor generates spatial orchestration data and sends them to the sink with associated media data and/or metadata.

— Sources can communicate with the m-processor using various types of metadata.

Figure 17 illustrates spatial orchestration at the sink side.

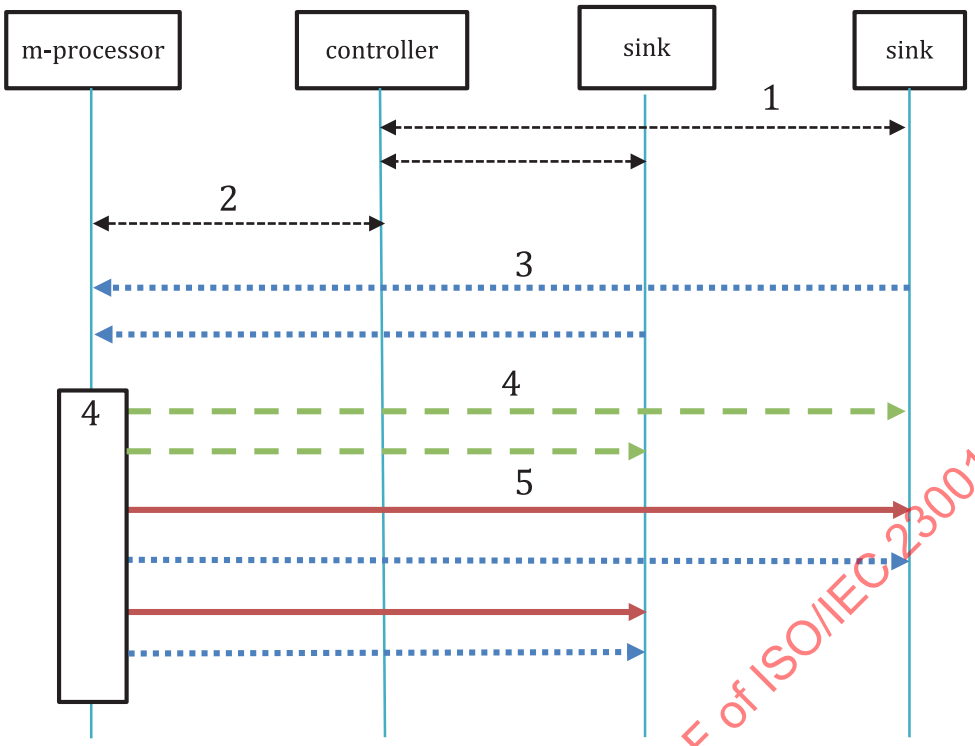


Figure 17 — Spatial orchestration at the sink side

The sequence diagram of Figure 17 has the following steps:

1. Discovery/registration processes, during which one or more sinks are associated with a controller for a specific media orchestration session.
2. The controller initiates instances of an m-processor.

   NOTE Discovery and instantiating an m-processor are outside the scope of this document.
3. Sinks start sending metadata to the m-processor.
4. The m-processor starts sending orchestration data to sinks.
5. Sinks start receiving media data and/or metadata from the m-processor.

6 Messaging and control

6.1 Wall clock

Sources and sinks involved in temporal orchestration with other sources or sinks shall support the wall clock protocol as specified in ETSI TS 103 286 02:2017, Clause 8.

6.2 MORE communication channel

6.2.1 General

The MORE communication channel is used for the exchange of messages and control (control data) between a controller and a controlled object, i.e. a source, a sink or an m-processor. The following are example cases for which the MORE communication channel may be used:

— If new sources become available, starting and stopping capture or updating capture masks (i.e. instructions on what to capture) may need to be performed directly.

— The timing correlation between 2 timed media data changes, e.g. because one media timestamp is doing a wrap-around or because a media source performs a clock reset. If the correlation is not immediately passed on, media playback will become unsynchronized.

— In a multi-display playback, more displays are added and thus the content to be played is divided differently across displays. New displays will retrieve their initial configuration, but existing displays need to retrieve an update.

— Control over PTZ (pan tilt zoom) functions of a video source.

The MORE communication channel is modelled after the SAND communication channel of MPEG DASH SAND, ISO/IEC 23009-5:2017. Figure 18 illustrates the establishment of a MORE communication channel.

1. Details of the MORE communication channel are signalled to a controlled object, for example via the retrieval of orchestration data. The details include an end-point address and the to-be-used protocol.
2. The controlled object sets up a MORE communication channel to a controller using the provided details.
3. The established MORE communication channel is used to asynchronously send control data from the controller to the controlled object and/or vice versa.

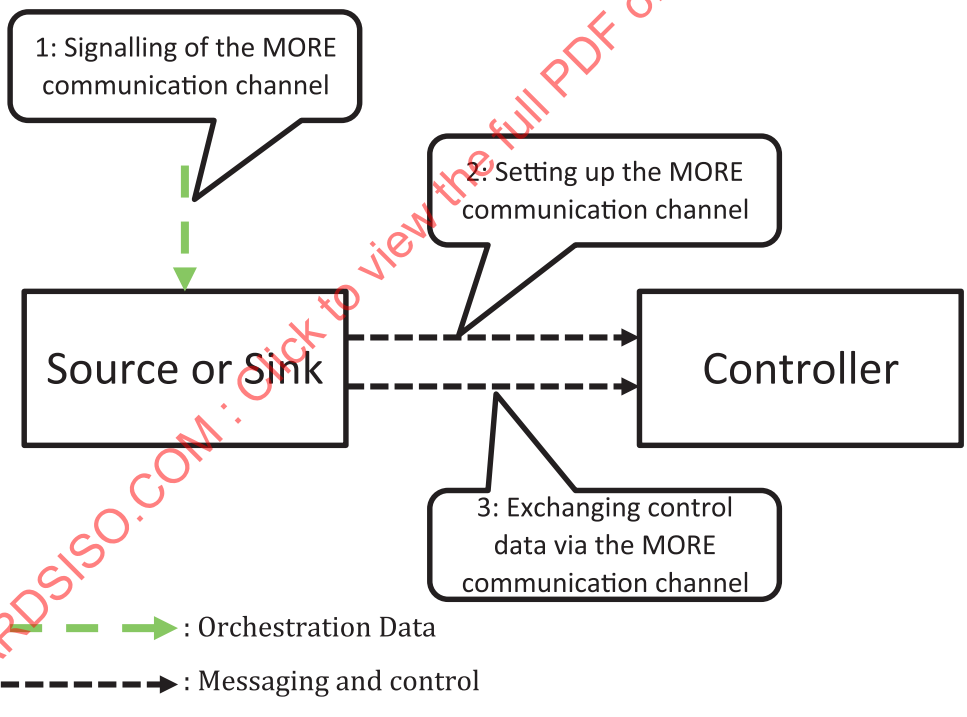


Figure 18 — MORE communication channel

6.2.2 Signalling of the MORE communication channel

This subclause describes the signalling mechanism to signal the transport protocol and end-point address for the exchange of MORE messages. In order to signal the MORE communication channel to sources and sinks, the presence of a MORE communication channel is signalled via the orchestration data using the samo:Channel element (samo stands for server-assisted media orchestration) defined in the "urn:mpeg:more:schema:samo:2016" namespace. The namespace prefix used in this document is "samo:". Table 2 specifies the samo:Channel element.

Table 2 — samo:Channel element

Element or attribute name	Use	Description
Channel		Provides information about a MORE channel
@id	O (string)	Specifies an identifier for this MORE channel
@schemeIdUri	M	Identifies the channel scheme. The channel scheme defines the protocol the recipient of the MORE channel shall support.
@endpoint	O (string)	Provides the endpoint to the MORE channel. The endpoint shall conform to the URI specification, IETF RFC 3986.
NOTE The conditions only hold without using xlink:href. If linking is used, then all attributes are "optional" and <minOccurs=0>.		

The `samo:Channel@schemeIdUri` specifies which protocol a controlled object shall use with this MORE channel. Table 3 lists the mandatory protocols.

Table 3 — Protocols for MORE communication channels

`@schemeIdURI`	Description
urn:mpeg:more:samo:channel:websocket:2016	The identifier indicates that the source or sink shall use the WebSocket Protocol as specified in Clause 6.3. In this case, the @endpoint of the samo:Channel shall be a valid WebSocket URI as specified in 3 WebSocket URIs of IETF RFC 6455.

6.3 Websocket protocol

This subclause defines the exchange of MORE control data messages over the websocket protocol as specified in IETF RFC 6455.

Data frame messages of the websocket protocol shall be set to the text type and the content shall be UTF-8 encoded as specified by the websocket protocol. Each websocket message shall contain a valid MORE message compliant with the MORE message XML schema.

7 Metadata

7.1 Timed metadata

Clause 3 defines metadata as the data that cannot be rendered independently and can affect rendering, processing or orchestration of the associated media data. Like media data, this metadata can be timed data. For example, when a tracker (that is able of tracking location and orientation) is attached to a camera, the location and orientation of the camera are continuously tracked as the camera captures and moves. With the captured video stream, the stream of location and orientation (gaze) of the camera is also generated. This location stream can be timed metadata about associated video stream since it has an intrinsic timeline.

ISO/IEC 23005 (all parts) specifies tools for describing sensing capability of individual sensors (ISO/IEC 23005-2) and detected information from these sensors (ISO/IEC 23005-5). Since most timed metadata (e.g. position, orientation) is generated from sources that include one or more sensors, this tool is used to describe the sensing ability of each source and timed metadata as detected information from these sources in the context of media orchestration. For that purpose, ISO/IEC 23005-6 also specifies a profile of sensing capabilities and sensed information datatypes that are useful for media orchestration applications.

Therefore, this clause refers the existing binary syntaxes according to ISO/IEC 23005-6 and also describes other timed metadata types which are not covered by ISO/IEC 23005-6.

7.2 Position metadata

7.2.1 General

Position can be acquired from different types of sources (sensors). A global navigation satellite system (e.g. GPS) sensor detects global position information and optionally altitude. The other position sensors detect position information, i.e. values along x, y, and z-axes.

ISO/IEC 23005-6 specifies a separate description for detected altitude information from the description of global position information. When a source detects both global position and altitude together, then the two fields should be put together in samples.

7.2.2 Global position

The syntax and semantics of the detected global position information for a source with global positioning system (GPS) shall be used as specified for `GlobalPositionSensorType` in ISO/IEC 23005-6:2019, 5.1.3.3; see also Table 4. That description includes the number of levels of longitude and latitude that the sensor can perceive and its accuracy of detected information.

Table 4 — Binary syntax of sensed global position information as defined in ISO/IEC 23005-6

Syntax	No. bits	Mnemonic
GlobalPositionSensorType{		
SensedInfoBaseType	7	SensedInfoBaseTypeType
crsLength	8	vluimsbf5
crs	0	UTF-8
Latitude	32	fsfb
Longitude	32	fsfb
}		

7.2.3 Altitude

The syntax and semantics of detected altitude information shall be used as specified for `AltitudeSensorType` in ISO/IEC 23005-6:2019, 5.1.3.4; see also Table 5. The description of the sensing capability of a source with altitude sensor includes the number of value levels, the range of values and its accuracy.

Table 5 — Binary syntax of sensed altitude information as defined in ISO/IEC 23005-6

Syntax	No. bits	Mnemonic
AltitudeSensorType{		
SensedInfoBaseType	7	SensedInfoBaseTypeType
crs	0	UTF-8
Altitude	32	fsfb
}		

7.2.4 Relative position

The syntax and semantics of detected relative position information shall be used as specified for `PositionSensorType` in ISO/IEC 23005-6:2019, 5.1.3.5; see also Table 6. The sensing ability of a source with position sensors is described as the maximum and minimum value that the position sensor can perceive along the x, y, and z-axis in the unit of meter. To ensure accurate orchestration of multiple sources, a calibration should have been performed to determine a single origin and the origin for all sources shall be the position after the calibration.

Table 6 — Binary syntax of sensed position information as defined in ISO/IEC 23005-6

Syntax	No. bits	Mnemonic
PositionSensorType {		
UpdateMode = 0	1	bslbf
if(UpdateMode ==0){		
PositionSensorNormal		PositionSensorNormalType
}else{		
PositionSensorUpdate		PositionSensorUpdateType
}		
}		
PositionSensorNormalType{		
positionFlag	1	bslbf
unitFlag	1	bslbf
SensedInfoBaseType	7	SensedInfoBaseTypeType
if(positionFlag) {		
Position		Float3DVectorType
}		
if(unitFlag) {		
Unit		unitType
}		
}		

NOTE UpdateMode is set as '0' (Only PositionSensorNormal is used in the context of this document).

When this position metadata describes the position of captured objects, the usage of the attribute, 'sensorIdRef' should be known to all stakeholders to indicate the metadata is sensed for captured objects. In this case, the origin should be the position of the source which is capturing the object.

7.3 Orientation metadata

The syntax and semantics of detected orientation information shall be used as specified for `OrientationSensorType` in ISO/IEC 23005-6:2019, 5.1.3.6; see also Table 7. The sensing capability of sources with orientation sensors includes the range of yaw, pitch, roll that the orientation sensor can perceive in the unit of degree.

Table 7 — Binary syntax of sensed orientation information as defined in ISO/IEC 23005-6

Syntax	No. bits	Mnemonic
OrientationSensorType {		
UpdateMode = 0	1	bslbf
if(UpdateMode ==0){		
OrientationSensorNormal		OrientationSensorNormalType
}else{		
OrientationSensorUpdate		OrientationSensorUpdateType
}		
}		

Table 7 *(continued)*

Syntax	No. bits	Mnemonic
OrientationSensorNormalType{		
orientationFlag	1	bslbf
unitFlag	1	bslbf
SensedInfoBaseType	7	SensedInfoBaseTypeType
if(orientationFlag) {		
orientation		Float3DVectorType
}		
if(unitFlag) {		
Unit		unitType
}		
}		

NOTE Updatemode is set as '0' (Only OrientationSensorNormal is used in the context of this document).

7.4 Regions of interest

7.4.1 General

An overview of timeline representation in media orchestration is as depicted in Figure 19. The orchestration data includes a universal timeline in which multiple timed media data (media streams) are represented. The composition of an orchestrated single video experience includes the use of "edit lists" to represent the availability of a specific timed media data. Similarly, if the timed media data is considered a fragmented movie, there is explicit provision for defining "empty time". Each timed media data (media stream) is supported with a complementary ROI timed metadata consisting of timed metadata of a bounding box.

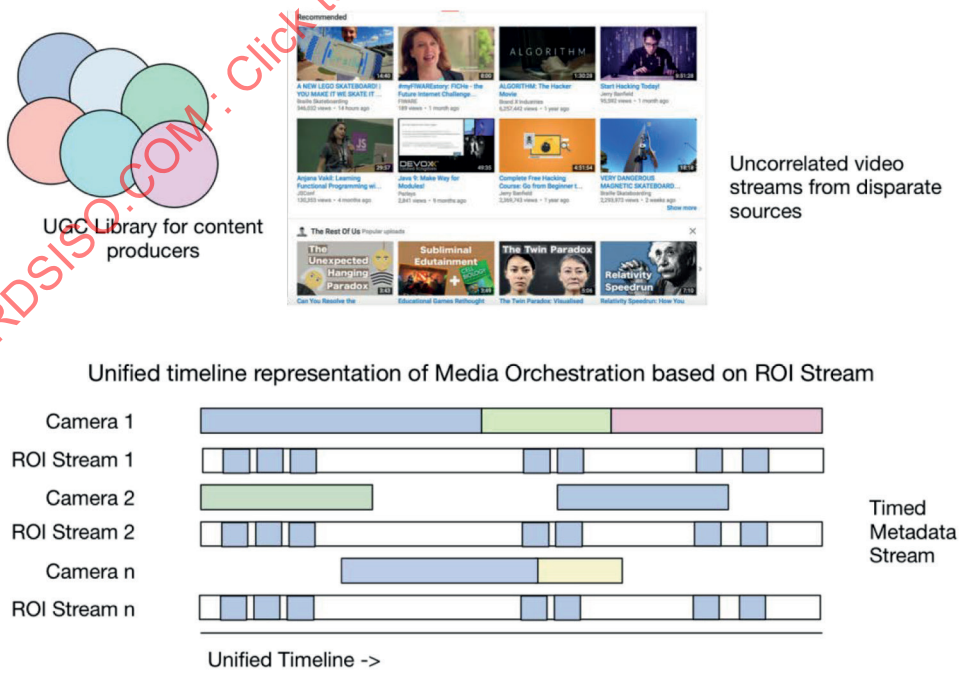


Figure 19 — Timeline representation for media orchestration

The representation of the ROI shall use ISO/IEC 23001-10 for pointers to 2DCartesianCoordinateSamples, the 2DCartesianCoordinateSampleEntry, their timing and their relationship to video samples in the ISOBMFF file.

ISO/IEC 14496-12:2015/Amd 1:2017, 6.2.2.1 specifies the syntax of a 2DCartesianCoordinateSample, which consists of 9 bytes in byte (8-bit) alignment.

ISO/IEC 14496-12:2015/Amd 1:2017, 6.2.1.1 shall be used for the syntax of a 2DCartesianCoordinateSampleEntry, which is a MetadataSampleEntry of the type '2dcc'.

7.4.2 Example implementation using DROP features

The extraction of ROI is presented as below and examples are given in the context of a surveillance application. A region of a video that consists of unique information is represented as a DROP (distinctive region or pattern). A set of examples of these DROP regions is presented in Figure 20. When choosing a DROP, the selected part of the image should be a region whose qualifying metrics of a region to be DROP are likely unique and distinctive, as presented in Figure 21.

Figure 20 — Examples of DROP from LASIE dataset

Figure 21 — Identification of DROP region in a CCTV footage

The extraction of the DROP feature is out of scope of this document. It can be based on the use of MPEG – 7 CDVS or it can rely on Haar-like features. The workflow of the query-based ROI timed metadata

synchronization is presented in Figure 22. To address the challenges of DROP identification, extraction and tracking, a Haar-like feature extraction is implemented in this example. As colour is considered a psychological motivator for the determination of the "DROP" by human operators, a special emphasis is placed on the determination of dominant colour descriptor while extracting Haar-like features. The extraction process is presented in Figure 23. The search process of the DROP regions is based on the learned pattern from the query is presented Figure 24.

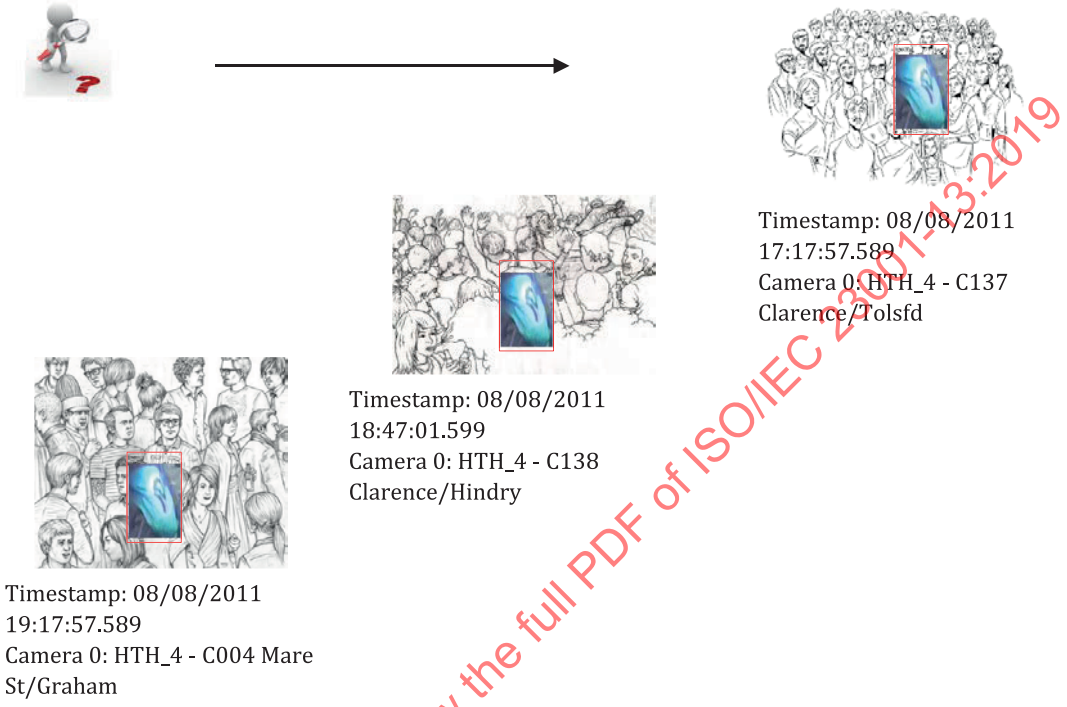


Figure 22 — Workflow of DROP utilisation based on visual query input

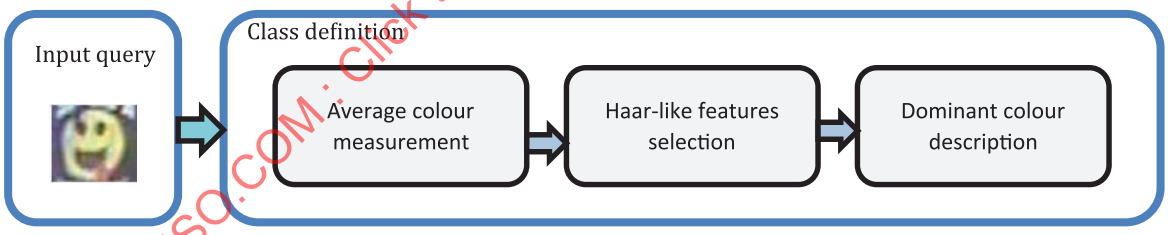


Figure 23 — DROP definition process

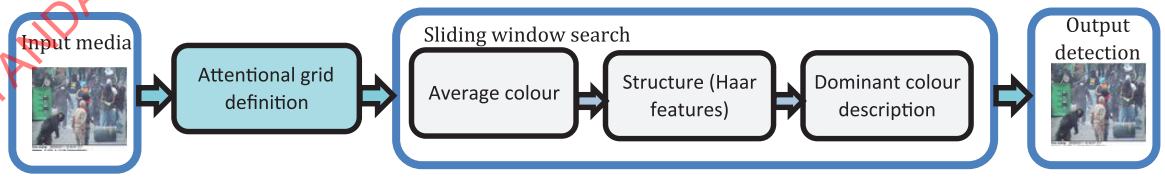


Figure 24 — DROP search process

7.5 Quality metadata

7.5.1 General

Quality metadata is timed metadata that provides information on captured timed media data, either in real time or on demand. One of the applications of timed quality metadata is the selection between

multiple sources. Examples of aspects influencing the perceived subjective quality are steadiness (isn't the camera moving around too much), occlusion (are there backs of people in the way blocking the view), contrast quality and colour quality; see Figure 25.

Figure 25 — Multiple sources capturing the same event

The perceived subjective quality can be expressed as a mean opinion score (MOS); see also ISO/IEC 23001-10:—, 4.3.6. This is the average score that a group of people would give to the capture, expressed on a scale from 1 to 5, where 1 is the lowest perceived quality and 5 is the highest perceived quality.

The quality assessment may be performed by an m-processor that receives one or more timed media data (media streams). Alternatively, a source may perform the quality assessment itself.

The perceived subjective quality of the multimedia capture may vary per source, and it may vary over time. An m-processor may include quality metadata to make a time-dependent selection between chunks of timed media data from different source; see Figure 26.

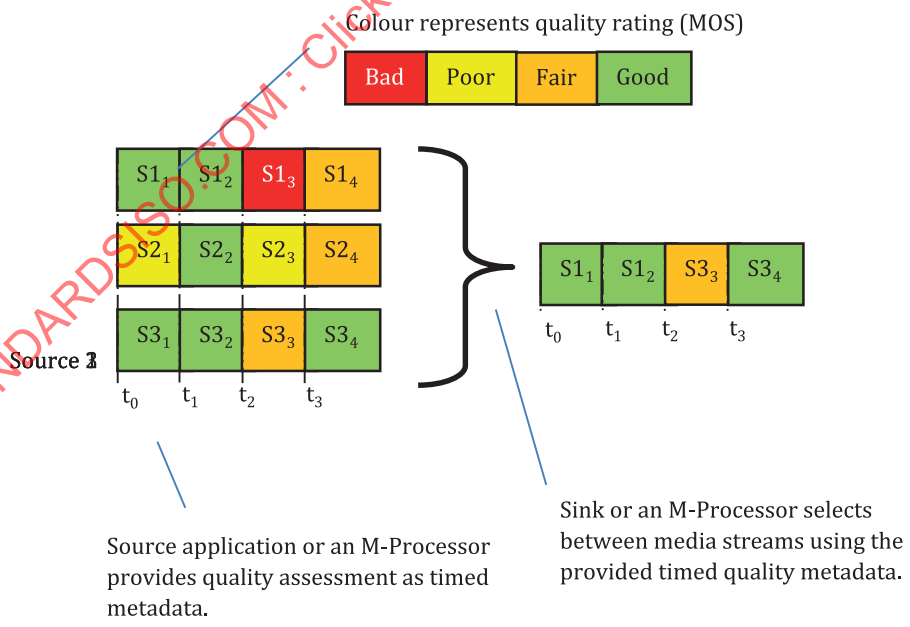


Figure 26 — M-processor selects between sources based on subjective quality

The quality assessment or selection by the m-processor can be used in various ways. For example:

— The m-processor provides quality metadata to a sink as an ISOBMFF or MPEG-2 TS timed metadata, or in an MPEG DASH MPD. The sink then uses this to select a timed media data (media stream) to be played out.

— The m-processor performs the selection itself and assembles a timed media data (media stream) that is sent to a sink.

— The selection is fed back to one or more sources, e.g. via a controller function, requesting not-selected clients to hold off sending their media data and hence save network bandwidth.

The algorithm that a source uses to determine the perceived subjective MOS of the captured media is not defined. The algorithm may be dependent on the application scenario and the specific type of content (e.g. soccer match, concert, etc.). In an application scenario, the algorithm may be part of the application that the user downloads to the source device that they use for the media capture.

7.5.2 Quality metadata

Quality metadata is specified using a MOS MORE (mean opinion score for media orchestration) score.

Syntax

A MOS MORE value is a value in the range 1 to 5. The lowest value ("1.0") corresponds to the worst possible perceived subjective quality, whereas "5.0" corresponds to the best possible.

— Binary encoding: 8-bit integer value x ranging from 0 to 250, which is decoded using the equation MOS MORE = ceil((real) x/50).

— String encoding: encoded as digit, followed by a decimal point, and followed by another digit. For example, "1.0", "3.5", "4.2" or "5.0".

Semantics

The semantics of the MOS MORE parameter is the perceived subjective quality of a time interval of a video. The subjective evaluation may consider camera steadiness, occlusion, contrast quality, colour quality and other aspects that may make a user prefer one video over another video for that time interval.

The present specification does not specify an algorithm for determining MOS MORE values. It is noted that the algorithm may depend on the context, e.g. the type of content.

This document does not specify the time interval. The time interval may be as short as a single video frame, as long as the whole video, or something in between, e.g. a DASH period; see clause 8.4.2.

7.6 Stream monitor

7.6.1 General

Timeline synchronization is challenging in environments where timed-data timestamps may be altered, sources are not synchronized, and/or timeline correlation is unavailable. The following are some examples of this.

— An m-processor (e.g. transcoder or multiplexer as in existence today) typically changes the timestamps of the carried timed data. Also, persistent timestamps like MPEG-TEMI (ISO/IEC 13818-1:2019, Annex U) may be stripped.

— Different sources have different timelines that are typically not correlated.

The former problem can be solved by having the m-processor generate dvb-css correlation timestamps (MORE-CT) as sketched in Figure 12. The latter problem can be solved by synchronizing the sources

(MORE-WC), by having each source report dvb-css timestamps (MORE-TS) and by subsequently generating dvb-css correlation timestamps (MORE-CT) as sketched in Figure 11. However, these approaches fail in case of legacy transcoders/multiplexers that cannot provide dvb-css correlation timestamps, and in case of sources that are not synchronized at the time of recording.

Stream monitor (SM) metadata is timed metadata that provides timestamped features of a timed media data (video, audio) to aid timeline synchronization. These features may be extracted and timestamped at the media source and/or within the network. Features may be watermarks that have been purposely introduced in the timed media data, or fingerprints that are properties of the timed media data itself. Both fingerprints and watermarks are supposed to be persistent, surviving recording at the source as well as transcoding/multiplexing operations. Figure 27 sketches three scenarios that use stream monitor metadata. All scenarios have at least two stream monitors, as the required synchronization is always between at least two timed media data (media streams).

— Scenario 1: Transcoder with two stream monitors

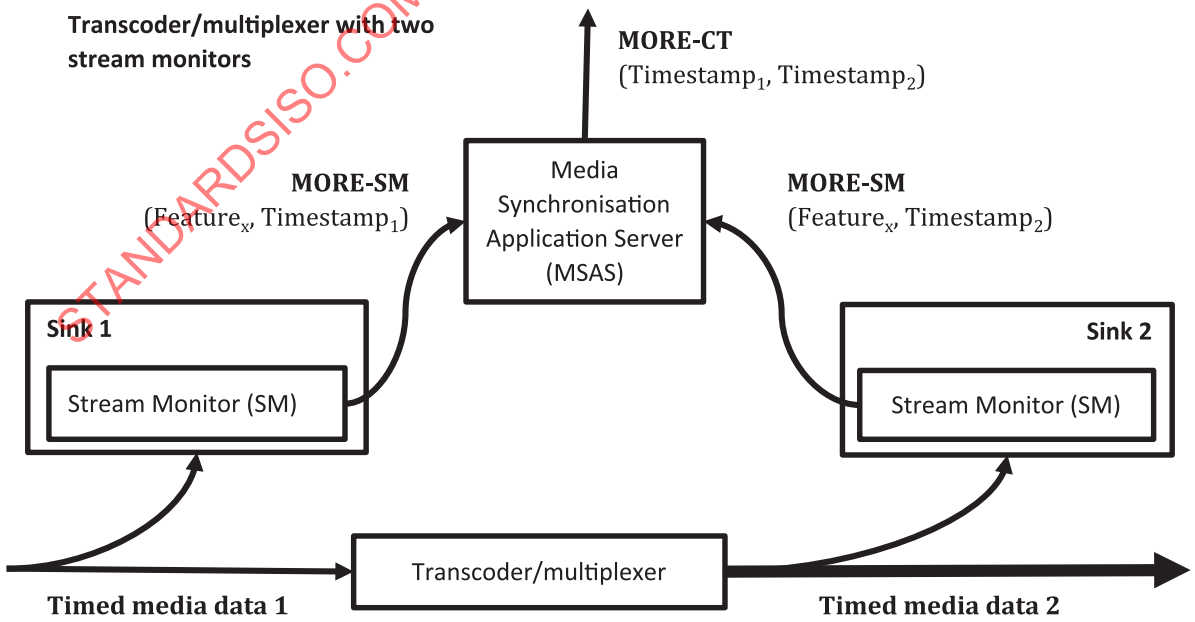
  In this scenario, one SM monitors timed media data going into a transcoder/multiplexer, and another SM monitors the outgoing timed media data. Each SM reports timestamped features via a MORE-SM interface, i.e. the combination of a feature and the media or metadata timestamp (e.g. PTS, CT or TEMI) associated with that feature. An MSAS receives both stream monitor metadata and correlates the two, producing dvb-css correlation timestamps.

— Scenario 2: Two sources with two steam monitors

  In this scenario, two SMs monitor timed media data produced by two sources, reporting timestamped features via a MORE-SM interface, i.e. the combination of a feature and the media or metadata timestamp (e.g. PTS, CT or TEMI) associated with that feature. Again, an MSAS receives both stream monitor metadata and correlates the two, producing dvb-css correlation timestamps.

— Scenario 3: Two sinks with two steam monitors

  In this scenario, two SMs monitor timed media data received at two sinks that are wall clock synchronized (MORE-WC), reporting timestamped features via a MORE-SM interface; in this case, that is the combination of a feature and a wall clock time [e.g. a wall clock time carried in TEMI, ISO/IEC 13818-1:2019, Annex U, at the timestamping reference point (see also subclause 9.1)]. The MSAS receives this information and it can use this to calculate timing differences, in order to instruct the delay of the media data at a sink or in the network to achieve synchronous playout.



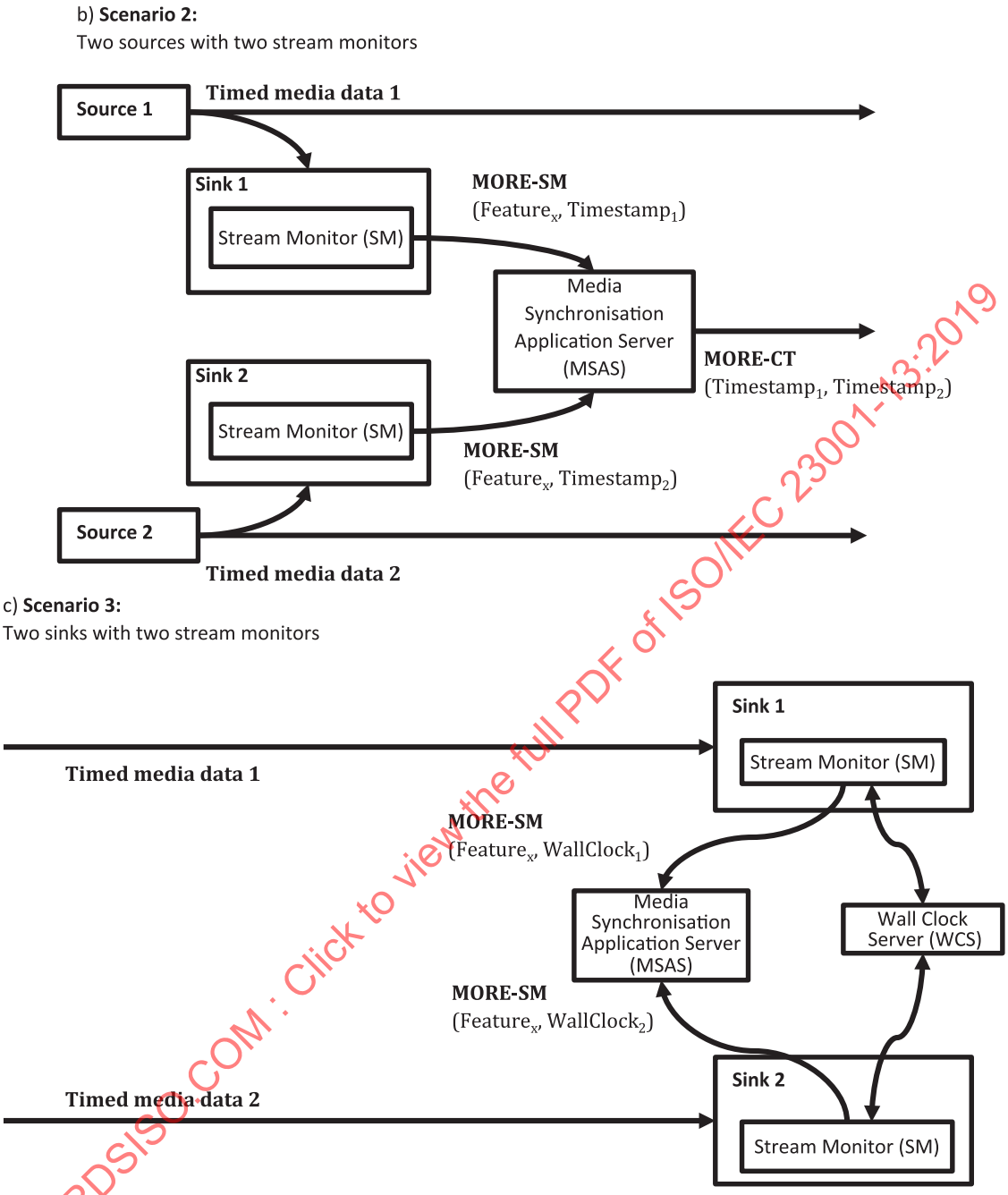


Figure 27 — Example scenarios with stream monitors

There are many variations possible to these scenarios, for example the following.

— The first stream monitor does not need to be close to the transcoder. It can also be located close to the source of the timed media data.

— Similarly, the second stream monitor does not need to be close to the transcoder. It can also be located at the downstream end of a distribution network.

— A stream monitor may be located inside a source, monitoring the recorded raw media data.

— The stream monitor process does not need to take place in real time, and it may be performed on a recorded timed media data for a posteriori synchronization.

— Stream monitor metadata may be multiplexed with timed media data.

— Stream monitor metadata may be multiplexed with timed orchestration data, e.g. with MORE-CT.

— Stream monitor metadata may be received and processed by a sink for timeline synchronization of received timed media data.

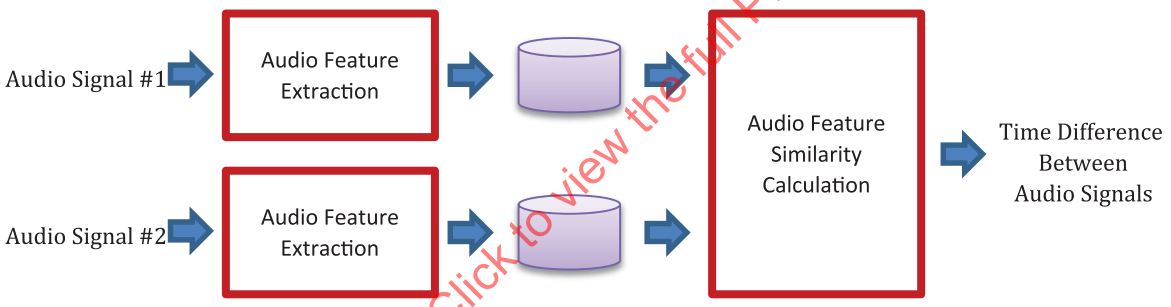
NOTE 1 Additional scenarios with stream monitors are provided in ETSI TS 103 286 02:2017, B.5 and B.6.

NOTE 2 The determination of a timestamp associated with a feature can require extrapolation. Such extrapolations are simple linear algebraic calculations. Examples of such calculations are provided in ETSI TS 103 286 02:2017, 5.4 and C.4 to C.7, and ISO/IEC 13818-1:2019, Annex U.

The MORE-SM interface is specified in more detail in the following clauses. It constitutes the combination of features and associated timestamps (e.g. PTS, CT or TEMI), carried by ISOBMFF or MPEG-2 TS.

7.6.2 MPEG-Audio-Sync-based stream monitor metadata

MPEG-Audio-Sync (ISO/IEC 14496-3) specifies an audio synchronization system that uses audio feature extraction; see Figure 28. An audio feature is a coded binary digit sequence extracted from an audio signal. The system consists of an audio feature extraction tool and an audio feature similarity calculation tool. The audio feature extraction tool generates timed metadata with audio features for synchronization from a time domain audio signal. The audio feature similarity calculation tool compares two audio feature timed metadata to find the time difference between the audio signals. ISO/IEC 14496-3 normatively specifies an audio feature extraction algorithm, as well as the syntax and semantics of an extracted audio feature frame. An audio feature frame has 128 bits and is determined with 8 ms intervals for a sliding-window 32 ms frame length.



SOURCE ISO/IEC 14496-3

Figure 28 — Audio synchronization system

A typical scenario of MPEG-Audio-Sync multiplexes 1 to 15 consecutive audio features extracted from a 1st screen device (e.g. a main media device receiving a television broadcast) into the timed media data for the 2nd screen device (e.g. tablet device, companion screen receiving audio/video content over an IP network). The microphone of the 2nd screen device captures audio from the 1st screen device, it continuously extracts audio features from this audio, it compares it with the received audio features, it determines the time difference and it synchronizes the playout at the companion device with that of the main device. In this scenario, no common clock covering the 1st and 2nd screen devices is required, nor an exchange of time-stamps between the devices.

Audio feature extraction may also be used for the synchronization scenarios presented in subclause 7.6.1, using the MORE SM interface to transport timestamped audio features. In these scenarios, they are carried stand-alone in an ISOBMFF file or an MPEG-2 Transport Stream.

7.7 Capture mask

7.7.1 General

A capture mask is a selection of a video frame, which may be used in the context of video stitching. A capture mask is used if an m-processor is not interested in receiving the full video frame, but only

a section of the frame defined by the capture mask. The concept of a capture mask is illustrated in Figure 29. An m-processor receives video streams from multiple sources, on which it performs video analysis and video stitching. It generates an output video that is sent to a sink.

Based on received video data, the m-processor determines a capture mask and provides this to a source as orchestration data. A source receives the capture mask and crops the video frame(s) accordingly.

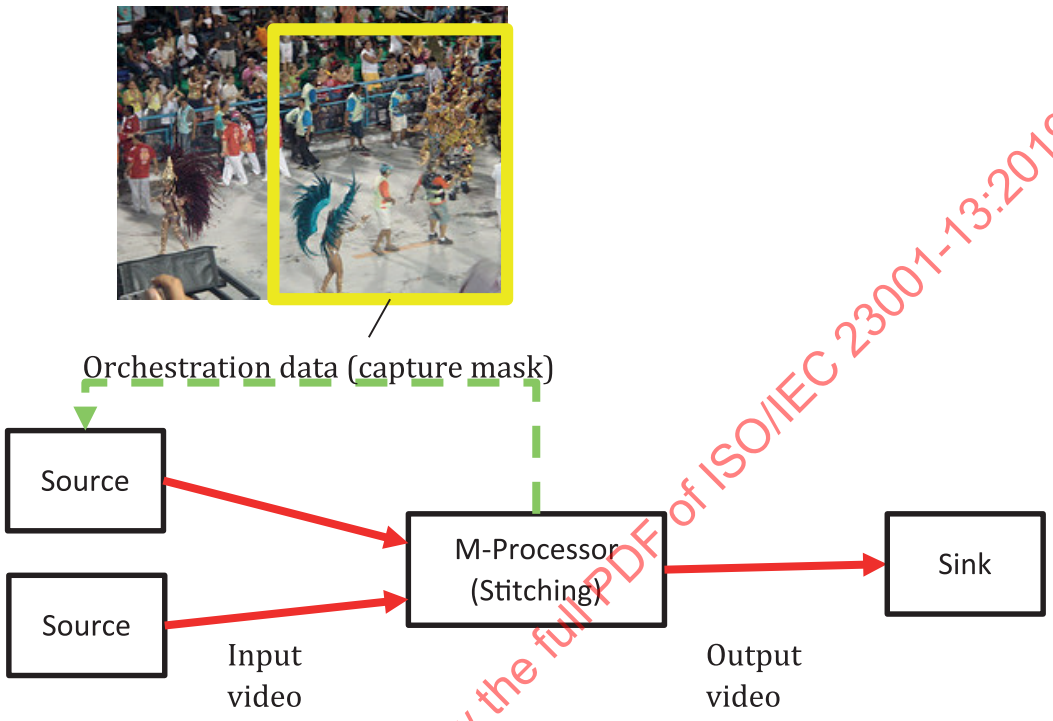


Figure 29 — Use of capture mask in media orchestration

Two cases are relevant:

— non-real-time: the content is pre-recorded;

— live streaming: the content is live.

In the former case, the capture-mask orchestration data may cover the entire recorded video content from the beginning until the end. The capture mask may be different for different periods in the video, and the orchestration data may also indicate that the m-processor is only interested in specific time intervals of the video.

In the latter case, messaging and control may be used by the m-processor to signal the actual capture mask.

7.7.2 Capture-mask metadata

The concept of a capture mask at the source side is very similar to the concept of a region of interest (ROI) at the sink side. ISO/IEC 23001-10 specifies metadata for the identification of a rectangular region of interest inside a rectangular video frame, as well as the carriage of this metadata in ISOBMFF. The syntax and semantics for region of interest are therefore reused to define capture mask.

The size of the reference video frame is specified as a 2D Cartesian coordinates sample entry according to ISO/IEC 23001-10:—, 7.2.1.

The coordinates of a capture mask are specified in 2D Cartesian coordinates samples according to ISO/IEC 23001-10:—, 7.2.2.

7.8 Camera metadata

This clause specifies basic timed metadata associated with media data from a camera source. This metadata describes the absolute position and orientation of a camera source with the range of resolution, focal length, aperture and shutter speed.

The syntax and semantics of camera sensor information shall be used as specified for `CameraSensorType` in ISO/IEC 23005-6:2019, 5.1.3.7; see also Table 8.

Table 8 — Binary syntax of camera sensor information as defined in ISO/IEC 23005-6

Syntax	No. bits	Mnemonic
CameraSensorType {		
CameraOrientationFlag = 0	1	bslbf
CameraLocationFlag = 0	1	bslbf
focalLengthFlag	1	bslbf
apertureFlag	1	bslbf
shutterSpeedFlag	1	bslbf
filterFlag = 0	1	bslbf
SensedInfoBaseType	7	SensedInfoBaseTypeType
if (CameraOrientationFlag == 1){		
CameraOrientation	0	OrientationSensorType
}		
if (CameraLocationFlag == 1){		
CameraLocation	0	GlobalPositionSensorType
}		
if (focalLengthFlag == 1){		
focalLength	32	fsbf
}		
if (apertureFlag == 1){		
aperture	32	fsbf
}		
if (shutterSpeedFlag == 1){		
shutterSpeed	32	fsbf
}		
if (filterFlag == 1){		
filter	0	bslbf
}		
}		

7.9 Depth metadata

This subclause specifies depth timed metadata associated with media data from a camera source. This metadata describes the distance between a source and the objects in the environment, encoded as a matrix. The syntax and semantics of camera sensor information shall be used as specified for `DepthCameraSensorType` in ISO/IEC 23005-6:2019, 5.1.3.8; see also Table 9.

Table 9 — Binary syntax of depth camera sensor information as defined in ISO/IEC 23005-6

Syntax	No. bits	Mnemonic
DepthCameraSensorType {		

Table 9 *(continued)*

Syntax	No. bits	Mnemonic
CameraSensorType		CameraSensorType
RawVideo {		
width	16	uimsbf
height	16	uimsbf
bit_depth	8	uimsbf
stride	32	uimsbf
coding4cc	32	uimsbf
fps	16	uimsbf
use_frame_packing	1	bslbf
RawVideoDataSize = 0	8	vluimsbf5
RawVideoData	0	
}		
RawDepth {		
width	16	uimsbf
height	16	uimsbf
bit_depth	8	uimsbf
stride	32	uimsbf
coding4cc	32	uimsbf
fps	16	uimsbf
use_frame_packing	1	bslbf
RawVideoDataDize		vluimsbf5
RawVideoData		bslbf
}		
}		

8 Transport

8.1 Carriage of timed metadata in ISO base media file format

8.1.1 General

Carriage of timed metadata for media orchestration in ISOBMFF is useful in cases where timed media data and associated timed metadata are stored in files, either together or in separate files. It is also useful for transport over DASH or MMT.

If timed metadata is carried in an ISO Base Media File, it shall be carried in the metadata tracks within that file. The timed metadata track shall use the 'meta' handler type in the handler box of the media box with a null media header ('nmhd').

A metadata track carrying timed metadata can be associated with other media (e.g. video or audio) track. The timed metadata track is linked to the track it describes by means of a 'cdsc' (content describes) track reference. One or more metadata track can be associated with a same media track. The metadata samples that have the same composition time can be consumed for the associated media at the composition the same time.

8.1.2 Carriage of position metadata in ISO base media file format

The sample entry for position metadata (defined in subclause 7.2) is defined as the `PositionMetadataSampleEntry`.

Sample Entry Type: 'pmse'

Container: Sample Description Box ('stsd')

Mandatory: No

Quantity: 0 or 1

The position metadata sample entry shall contain a `PositionMetadataConfigurationBox`, describing one or more position metadata entities that are present in each sample. Each sample is an array of position metadata entities, corresponding one for one to the declared position metadata defined in subclause 7.2.

```
class PositionMetadataSampleEntry() extends MetadataSampleEntry ('pmse') {
      PositionMetadataConfigurationBox();
}

class PositionMetadataConfigurationBox extends
      FullBox('pomc', version = 0, flags = 0){ {
      unsigned    int(8)      entity_count;
      for (i=1; i <= entity_count; i++){
            unsigned int(16)        entity_size_bytes;
            unsigned int(7)         reserved=0;
            unsigned int(1)         sensor_capability_included;
            unsigned int(32)        position_entity_code[i];
            if (sensor_capability_included == '1')
            {
                  PositionSensorCapability(position_entity_code[i]);
      }
      }
}
class PositionSensorCapability(unsigned int(32) position_entity_code) {
      if(position_entity_code == 'GPOS')
      {
            GlobalPositionSensorCapabilityType  gps_cap;
      } else  if(position_entity_code == 'RPOS'){
            PositionSensorCapabilityType        rps_cap;
      } else  if(position_entity_code == 'ALTI'){
            AltitudeSensorCapabilityType        alt_cap;
      }
}
```

Semantics:

`entity_count` specifies the number of position metadata entities in each sample.

`entity_size_bytes` indicates the size (in bytes) of each position metadata entity.

`sensor_capability_included`, equal to 1, indicates that sensing capability information associated to the position entity is present. According to `position_entity_code` value, the following sensing capability information is present.

- 'GPOS ' = global position sensing capability as defined in subclause 7.2.2.
- 'ALTI' = altitude sensing capability as defined in subclause 7.2.3.
- 'RPOS ' = relative position sensing capability as defined in subclause 7.2.4.

`position_entity_code[i]` indicates which position metadata is present in i-th position metadata entity as follows.

— 'GPOS' = i-th position metadata entity includes global position information as defined in ISO/IEC 23005-6:2019, 5.1.3.3.

— 'ALTI' = i-th position metadata entity includes altitude information as defined in ISO/IEC 23005-6:2019, 5.1.3.4.

— 'RPOS' = i-th position metadata entity includes relative position information as defined in ISO/IEC 23005-6:2019, 5.1.3.5.

The position metadata sample shall use the following syntax.

```
class PositionMetadataSample(){
      for(i=1; i<= entity_count; i++){
            if(position_entity_code[i] == "GPOS") {
                  GlobalPositionSensorType      global_position;
            } else  if(position_entity_code[i] == "RPOS"){
                  PositionSensorType            relative position;
            } else  if(position_entity_code[i] == "ALTI"){
                  AltitudeSensorType            altitude;
            }
      }
}
```

Semantics:

`global_position` includes global position information as defined in ISO/IEC 23005-6:2019, 5.1.3.3.

`altitude` includes altitude information as defined in ISO/IEC 23005-6:2019, 5.1.3.4.

`relative position` includes relative position information as defined in ISO/IEC 23005-6:2019, 5.1.3.5.

8.1.3 Carriage of orientation metadata in ISO base media file format

Orientation metadata (defined in subclause 7.3) track uses the `OrientationMetadataSampleEntry`.

Sample Entry Type: 'omse'

Container: Sample Description Box ('stsd')

Mandatory: No

Quantity: 0 or 1

The orientation metadata sample entry shall contain an OrientationMetadataConfigurationBox, describing orientation metadata that are present in each sample. Each sample is an array of orientation metadata entities, corresponding one for one to the declared orientation metadata defined in subclause 7.3.

NOTE For forward compatibility, the syntax accounts for the possibility of having multiple orientation information types per sample. However, the current version of the specification provides only the single type referenced above.

```
class OrientationMetadataSampleEntry() extends MetadataSampleEntry('omse'){
      OrientationMetadataConfigurationBox();
}

class OrientationMetadataConfigurationBox extends
      FullBox('ormc', version=0, flags=0){
      unsigned    int(8)      entity_count;
      unsignedint(16)entity_size_bytes;
      for(i=1;i<=entity_count;i++){
```

```
            unsignedint(7)        reserved=0;
            unsignedint(1)        sensor_capability_included;
            if(sensor_capability_included==‘1’) {
                  OrientationSensorCapabilityTypeorin_cap;
            }
      }
}
```

Semantics:

`entity_count` specifies the number of orientation metadata entities in each sample.

`entity_size_bytes` indicates the constant size (in bytes) of orientation metadata entity.

`sensor_capability_included`, equal to 1, indicates that sensing capability information associated to the orientation metadata entity, as defined in subclause 7.3, is present.

The orientation metadata sample shall use the following syntax.

```
class OrientationMetadataSample(){
    for(i=1; i<= entity_count; i++){
      OrientationSensorType   orientation;
    }
}
```

Semantics:

Orientation includes orientation information as defined in ISO/IEC 23005-6:2019, 5.1.3.6.

8.1.4 Carriage of quality metadata in ISO base media file format

Quality metadata is defined in subclause 7.5.

Sample Entry Type: ‘mosm’

Container: Sample Description Box (‘stsd’)

Mandatory: No

Quantity: 0 or 1

The MosMoreSampleEntry identifies the MosMoreSamples; see also subclause 7.5.

Syntax of MosMoreSampleEntry

```
aligned(8) class MosMoreSampleEntry
   extends MetadataSampleEntry (‘mosm’) {
}
```

Semantics of MosMoreSampleEntry

N/A

Syntax of MosMoreSample

```
aligned(8) class MosMoreSample {
   unsigned int(8) mosMoreValue;
}
```

Semantics of MosMoreSample

`mosMoreValue` is the binary encoded MOS MORE value, providing a measure of the subjective quality of the associated media data, as specified in subclause 7.5.2.

8.1.5 Carriage of audio features in ISO base media file format

The present subclause specifies how audio features shall be carried in accordance with ISO/IEC 14496-12:2015 for the MORE-SM interface.

AudioSyncFeature sample entry

Sample Entry Type: `'asfe'`

Container: Sample Description Box ('stsd')

Mandatory: No

Quantity: 0 or 1

The AudioSyncFeatureSpecificConfig sample entry provides the configuration of audio sync features.

Syntax of AudioSyncFeatureSampleEntry

```
aligned(8) class AudioSyncFeatureSampleEntry
   extends MetadataSampleEntry ('asfe') {
   AudioSyncFeatureConfiguratonBox();
}

class AudioSyncFeatureConfigurationBox extends
   FullBox('asfc', version=0, flags=0){
   bit(32) audioSyncFeatureSpecificConfig();
}
```

Semantics of AudioSyncFeatureSampleEntry

`audioSyncFeatureSpecificConfig` is a set of 32 bits, corresponding with the AudioSyncFeatureSpecificConfig() provided in ISO/IEC 14496-3:2009, Table 13.1. The semantics of this is provided in ISO/IEC 14496-3:2009, 13.4.

NOTE 1 The configuration information includes, among others, the type of the audio sync features, their frame length and their time resolution. Its length is 32 bit (4 bytes).

AudioSyncFeatureFrameSample format

An audio_sync_feature is the binary feature for audio synchronization for a single frame ISO/IEC 14496-3.

Syntax of AudioSyncFeatureFrameSample

```
aligned(8) class AudioSyncFeatureFrameSample {
   bit(128) audioSyncFeatureFrame();
}
```

Semantics of AudioSyncFeatureFrameSample

`audioSyncFeatureFrame` is a set of 128 bits, corresponding with the AudioSyncFeatureFrame() provided in ISO/IEC 14496-3:2009, Table 13.2. The semantics of this is provided in ISO/IEC 14496-3:2009, 13.4.

NOTE 2 Audio_sync_feature samples are typically 128 bit (16 bytes), which is the default frame length as specified in the AudioSyncFeatureSpecificConfig.

8.1.6 Carriage of capture-mask metadata in ISO base media file format

Capture mask metadata (defined in subclause 7.7) shall be carried in ISOBMFF, ISO/IEC 14496-12:2015, as 2DCartesianCoordinatesSample with reference to a 2DcartesianCoordinatesSampleEntry; see subclause 7.7.2. The capture-mask metadata is carried in a timed metadata track in ISOBMFF.

Each 2DcartesianCoordinatesSample has an associated decoding timestamp, defined in the decoding time to sample box ('stts'). Composition timestamps (identical to decoding timestamps, so no 'ctts' box is needed) are used to map a specific capture mask to a specific frame of the source video.

Preferably, the source and the m-processor use the same types of timestamp type (namely composition time) and the same time base. In that case, there is a one-to-one mapping between a capture mask and a video frame (to be cropped for video stitching). However, if the source and the m-processor use different types of timestamps or a different time base, then correlation timestamps (see subclause 5.1) need to be used.

The first and last decoding timestamps of a timed metadata track containing capture-mask metadata correspond with the first and last frame of the time interval for which the m-processor is interested in receiving the (cropped) source video.

In case of non-real-time content, the m-processor provides the source with an ISOBMFF file with capture-mask metadata that covers the full content.

In case of live streaming content, the m-processor provides the source with an ISOBMFF file with a single 2DcartesianCoordinatesSample.

8.1.7 Carriage of dvb-css correlation timestamps in ISO base media file format

A dvb-css correlation timestamp is a data object that associates two or more timestamps of two or more timed data with each other. A typical example is the association between a presentation timestamp (PTS) of an MPEG-2 transport stream and a composition time of an ISOBMFF file. Its meaning is that the identified timed media data samples (audio sample, video frame) and/or timed metadata samples (location sample, orientation sample, etc.) have been captured at the same time and that they should be rendered at the same time.

Sample Entry Type: ‘coti’

Container: Sample Description Box (‘stsd’)

Mandatory: No

Quantity: 0 or 1

The CorrelationTimestampSampleEntry provides metadata about the CorrelationTimestampSamples.

Syntax of CorrelationTimestampSampleEntry

```
aligned(8) class CorrelationTimestampSampleEntry
     extends MetadataSampleEntry (‘coti’) {
     CorrelationTimestampConfiguratonBox();
}
```



```
class CorrelationTimestampConfigurationBox extends
      FullBox('cotc', version=0, flags=0){
      unsigned int(8)numberOfTimelines;
      for (timeline=0;timeline<number_of_timelines;timeline++) {
           string timedDataID;
           string timelineSelector;
           unsigned int(32) unitsPerTick;
           unsigned int(32) unitsPerSecond;
           unsigned int(32) accuracy;
           unsigned int (2) temi_timestamp_length_index;
           bit(6) reserved;
      }
}
```

Semantics of CorrelationTimestampSampleEntry

`numberOfTimelines` is the number of timelines covered by the CorrelationTimestampSampleEntry. Its value shall be 2 or greater, as the CorrelationTimestampSampleEntry covers at least two timelines, namely the timeline of the carrying ISOBMFF track itself with `timeline=0` and at least one other timeline (`timeline>=1`).

`timedDataID` is a null-terminated UTF-8 string, providing a URL identification of timed data associated with this timeline. Its purpose is to make clear which timeline of a dvb-css correlation timestamp is associated with which timed data. If the value of the string is null, then no URL identification is provided. For example, the carrying ISOBMFF track itself would not need to be explicitly identified.

`timelineSelector` is a null-terminated UTF-8 string according to Table 10 which replicates and extends ETSI TS 103 286 02:2017, Table 8.4.2.1. It indicates what type of timestamps the associated timed media data or timed metadata uses, e.g. MPEG2-TS presentation timestamp (PTS), ISOBMFF composition time (CT), TEMI, MPEG DASH Period relativeTimeline or NTP. The latter, NTP, is signalled with the value `timelineSelector=urn:mpeg:more:timeline:ntp`. This timeline is used to carry a ETSI TS 103 286 02 wall clock time as specified in subclause 6.1.

`unitsPerTick` and `unitsPerSecond` are integer values greater than zero that define the tick rate of the timeline according to ETSI TS 103 286 02:2017, 5.3.2. The tick rate is calculated as ticksPerSecond = unitsPerSecond/unitsPerTick. ETSI TS 103 286 02 uses this construction in order to be able to specify any rational value of ticksPerSecond as the division between two integers. The allowed values of `unitsPerTick` and `unitsPerSecond` depend on the value of `timelineSelector` according to Table 10 via references to ETSI TS 103 286 02. In case of `timelineSelector=urn:mpeg:more:timeline:ntp`, the unitsPerTick of the timeline shall be 1 and the unitsPerSecond shall be 4294967296 (2^32, which shall correspond to the tick rate of the NTP timestamp format of IETF RFC 5905).

`accuracy` provides the accuracy of the timeline, expressed as an integer number of microseconds. Accuracy quantifies the understanding of how accurately time values on this timeline represent the timing of underlying media content. The value should represent the 95 % confidence interval (2 standard deviations) or better. The minimum value is zero, indicating perfect accuracy (see ETSI TS 103 286 02:2017, 5.3.2 and 5.5.9.5).

`temi_timestamp_length_index` provides the length of the timestamp in case of TEMI timestamps (i.e. timelineSelector=urn:dvb:css:timeline:mpd:period:rel:*). Its value shall be identical to the "has_timestamp" parameter of the associated TEMI, see ISO/IEC 13818-1:2019, subclause U.3.6. That is, value 1 means a 32 bit media timestamp is present in the CorrelationTimestampSample; value 2 means a 64 bit media timestamp is present in the CorrelationTimestampSample; value 3 is reserved; and value 0 is not applicable (TEMI without a TEMI timestamp). The value of `temi_timestamp_length_index` shall be 0 for other (non-TEMI) types of timestamp.

`reserved` are reserved bits, and now used for padding. Their value shall be 0.

Syntax of CorrelationTimestampSample

```
aligned(8) class CorrelationTimestampSample
      for (timeline=1;timeline<numberOfTimelines;timeline++) {
            if(timelineSelector=urn:dvb:css:timeline:pts) {
                  unsigned int(48) timeValue;
            }
            else if(timelineSelector=urn:dvb:css:timeline:ct) {
                  unsigned int(32) timeValue;
            }
            else if(timelineSelector=urn:dvb:css:timeline:temi:*) {
                  if(temi_timestamp_length_index=1){
                        signed int(32) timeValue;
                  }
                  else if (temi_timestamp_length_index=2){
                        signed int(64) timeValue;
                  }
            }
            else if(timelineSelector=urn:dvb:css:timeline:mpd:period:rel:*) {
                  signed int(64) timeValue;
            }
            else if(timelineSelector=urn:mpeg:more:timeline:ntp) {
                  unsigned int(64) timeValue;
            }
      }
}
```

The "*" symbol indicates a wildcard. The allowed values of the wildcard are provided below.

Semantics of CorrelationTimestampSample

`numberOfTimelines` is the value provided in the CorrelationTimestampSampleEntry. The number of time values in a CorrelationTimestampSample equals to `numberOfTimelines-1`, as the time values for `timeline=0` are the composition timestamps of the ISOBMFF file itself, carried in the 'ctts' box, or in absence thereof in the 'stts' box. This implies that no `timeValue` is provided for `timeline=0`.

`timeValue` indicates a point on the particular timeline signalled by the CorrelationTimestampSampleEntry, expressed in units of that timeline. Table 10 specifies the detailed semantics of `timeValue` for different values of `timelineSelector`.

Table 10 — Specification of values for timelineSelector

Name	timelineSelector	Details	timeValue
MPEG-TS PTS: presentation time stamp	urn:dvb:css:timeline:pts	ETSI TS 103 286 02:2017, 5.3.4	48-bit presentation time stamp
ISOBMFF: composition time	urn:dvb:css:timeline:ct	ETSI TS 103 286 02:2017, 5.3.5	32-bit composition time

[a] ETSI TS 103 286 02:2017, Table 8.4.2.1 specifies `timelineSelector` values including delimiting quotes, as ETSI TS 103 286 02 carries these as JSON format strings. Delimiting quotes are omitted in Table 10, as these strings are already delimited as null-terminated UTF-8 string.

[b] As specified in ETSI TS 103 286 02 with reference to ISO/IEC 13818-1:2019, subclause U.3.2, *the value of <component_tag> is the PID value of the MPEG-2 Transport Stream media component. The value of <timeline_id> is the timeline_id value in TEMI location descriptor.*

[c] As specified in ETSI TS 103 286 02, 5.3.7.2, *<ticks-per-second> is a base-ten integer number as a string that defines the scale of the Timeline. The unitsPerTick of the Timeline shall be 1. The unitsPerSecond of the Timeline shall be value of <ticks-per-second>. The <period-id> identifies the period to be used as the base period. It consists of the value of the Period@ID attribute of the period.*

Table 10 *(continued)*

Name	timelineSelector	Details	timeValue
TEMI: timeline for external data	urn:dvb:css:timeline:temi: <component_tag>:<timeline_id>[b]	ETSI TS 103 286 02:2017, 5.3.6	64-bit unsigned integer
MPEG DASH: period relative timeline	urn:dvb:css:timeline:mpd:period:rel: <ticks-per-second> urn:dvb:css:timeline:mpd:period:rel: <ticks-per-second>:<period-id>[c]	ETSI TS 103 286 02:2017, 5.3.7	64-bit signed integer NOTE ETSI TS 103 286 02:2017, 5.3.7 specifies timeValue in the range $-2^{63} \leq t < 2^{63}$
NTP: network time protocol	urn:mpeg:more:timeline:ntp	IETF RFC 5905:2010, Clause 6	64-bit NTP timestamp format

[a] ETSI TS 103 286 02:2017, Table 8.4.2.1 specifies `timelineSelector` values including delimiting quotes, as ETSI TS 103 286 02 carries these as JSON format strings. Delimiting quotes are omitted in Table 10, as these strings are already delimited as null-terminated UTF-8 string.

[b] As specified in ETSI TS 103 286 02 with reference to ISO/IEC 13818-1:2019, subclause U.3.2, *the value of <component_tag> is the PID value of the MPEG-2 Transport Stream media component. The value of <timeline_id> is the timeline_id value in TEMI location descriptor.*

[c] As specified in ETSI TS 103 286 02, 5.3.7.2, *<ticks-per-second> is a base-ten integer number as a string that defines the scale of the Timeline. The unitsPerTick of the Timeline shall be 1. The unitsPerSecond of the Timeline shall be value of <ticks-per-second>. The <period-id> identifies the period to be used as the base period. It consists of the value of the Period@ID attribute of the period.*

In case of wrapping of a timeline (see also ETSI TS 103 286 02:2017, 5.5.5 and B.2.2.5.1), it is highly recommended to provide a dvb-css correlation timestamp directly before the wrapping, and one directly after, in order to prevent any ambiguities or discontinuities.

8.2 Carriage of timed metadata in MPEG-2 systems

8.2.1 General

Carriage of timed metadata for media orchestration in MPEG-2 TS is useful in cases of broadcast and/or real-time streaming of media data and associated timed metadata. The code points for carriage of MORE access units and descriptors in MPEG-2 TS shall be used as specified in ISO/IEC 13818-1:2019, subclauses 2.4.4.5 and 2.4.4.10, and below.

8.2.2 Timed metadata access unit

The timed metadata samples (defined in subclause 7.1) are stored in access units as the format described in Table 11 and encapsulated in a PES stream. Since each PES packet is associated with a certain PTS (presentation time stamp), the timing of activation of timed metadata is defined by the associated presentation time.

Table 11 — Syntax of Timed_Metadata_Access_Unit

Syntax	No. bits	Mnemonic
timed_metadata_access_unit () {		
metadata_access_unit_bytes()	8xM	
if (CRC_32_flag=1) {		
CRC_32	32	rpchof
}		
}		

Semantics:

metadata_access_unit_bytes — bytes that contain an access unit of timed metadata associated with one or more video or audio frames. This field is differently specified according to the type of timed metadata as specified in subclause 8.2.3; see Table 12. The length of this field (M bytes) is provided by timed_metadata_access_unit_length (H) in the timed metadata extension descriptor (subclause 8.2.3).

CRC_32_flag — a flag whether CRC_32 is applied; see subclause 8.2.3.

CRC_32 — This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in Annex A after processing the entire program association section.

The length of a timed metadata access units is provided in the timed metadata extension descriptor; see subclause 8.2.3.

Timed metadata access units are encapsulated in MPEG sections identified by stream_type value specified in subclause 8.2.4.

8.2.3 Timed metadata extension descriptor

The timed metadata extension descriptor is signalled using a descriptor in the program map table. This descriptor shall appear in the elementary stream loop of the PID of the timed metadata. The timed metadata extension descriptor describes the type of metadata that is present in each timed metadata access unit, as well as any static associated metadata. The syntax of media orchestration extension descriptor containing static metadata is shown in Table 12.

Table 12 — Timed metadata extension descriptor

Syntax	No. bits	Mnemonic
timed_metadata_extension_descriptor() {		
timed_metadata_code	8	uimsbf
timed_metadata_access_units_have_constant_size	1	bslbf
CRC_32_flag	1	bslbf
Reserved	6	
if(timed_metadata_units_have_constant_size=1)}		
timed_metadata_access_unit_length (H)	8	uimsbf
}		
timed_metadata_descriptor_length (J)	8	uimsbf
timed_metadata_descriptor_bytes()	8xJ	
}		

Semantics:

timed_metadata_code — the 4CC code name of the timed metadata in the timed metadata access units; see Table 13.

timed_metadata_access_units_have_constant_size — a flag whether all timed metadata units have a constant size (1). If not (0), then the size of a timed_metadata_access_unit is determined elsewhere, e.g. within the timed_metadata_access_unit itself.

CRC_32_flag — a flag whether CRC_32 is applied in the timed metadata access units; see subclause 8.2.2.

reserved — reserved bits, all set at 0.

timed_metadata_access_unit_length (H) — the constant size in byte of the timed metadata access units. When applicable, timed metadata units do not have to signal their own length, which saves bandwidth and parsing effort.

timed_metadata_descriptor_length (J) — the size in byte of timed_metadata_descriptor_bytes.

timed_metadata_descriptor_bytes — the bytes of the timed_metadatdata_descriptor; see Table 13.

Table 13 — timed_metadata_code

timed_metadata_code (4CC)	Semantics	timed_metadata_ access_unit	timed_metadata_ descriptor
POSI	Position	8.2.4.1	8.2.4.2
ORIE	Orientation	8.2.5.1	8.2.5.2
STMO	Stream monitor	8.2.7.1	8.2.7.2
CAMA	Capture mask	8.2.6.1	8.2.6.2

8.2.4 Carriage of position metadata in MPEG-2 systems

8.2.4.1 Position metadata access unit

This subclause specifies the format of an access unit of position metadata (defined in subclause 7.2) and the signalling of position metadata to indicate which position metadata is carried in the PES stream.

Each position metadata access unit carries one or more position metadata associated with one or more video/audio frames, i.e. multiple position metadata entity can be applicable for multiple video/audio frames for a given PTS interval, until a new position metadata entity is declared. The position metadata access unit contains the configuration of position information, corresponding one for one to the declared position information in subclause 7.2.

Every position metadata access unit shall be a random-access point. In order to make processing easier, each position metadata access unit should be contained in a single TS packet. A position metadata access unit contains zero or one of each for global position, altitude and relative position.

A position metadata access unit is a timed metadata access unit (8.2.2), of which the metadata_access_units are specified in Table 14.

Table 14 — Syntax of the metadata_access_unit_bytes of a position metadata access unit

Syntax	No. bits	Mnemonic
position_metadata_access_unit_bytes() {		
if (has_global_position_bytes==1) {		
global_position_bytes	8xP	uimsbf
}		
if (has_altitude_bytes==1) {		
altitude_bytes	8xQ	uimsbf
}		
if (has_relative_position_bytes==1) {		
relative_position_bytes	8xR	uimsbf
}		
}		

Semantics:

has_global_position_bytes — this 1-bit flag indicates whether the access unit includes global_position_bytes. Its value is specified in the position metadata descriptor; see subclause 8.2.4.2.

has_altitude_bytes — this 1-bit flag indicates whether the access unit includes altitude_bytes. Its value is specified in the position metadata descriptor; see subclause 8.2.4.2.

has_relative_position_bytes — this 1-bit flag indicates whether the access unit includes relative_position_bytes. Its value is specified in the position metadata descriptor; see subclause 8.2.4.2.

global_position_bytes — includes global position information as defined in ISO/IEC 23005-6. The length of this field (P bytes) is specified by the global_position_bytes_length field in the position metadata descriptor; see subclause 8.2.4.2.

altitude_bytes — includes altitude information as defined in ISO/IEC 23005-6. The length of this field (Q bytes) is specified by the altitude_bytes_length field in the position metadata descriptor; see subclause 8.2.4.2.

relative position_bytes — includes position information as defined in ISO/IEC 23005-6. The length of this field (R bytes) is specified by the relative_position_bytes_length field in the position metadata descriptor; see subclause 8.2.4.2.

NOTE The reason for enabling having global position, altitude and/or relative position in a single access unit is that some position sensors provide samples of these together with the same timestamp. If there are separate sensors with separate timing, then there can be an individual PES for each sensor.

8.2.4.2 Position metadata descriptor

Since one PES stream can carry one or more position metadata, it needs to signal which type of position metadata per event or program.

The position metadata descriptor describes the type of position metadata, and optionally the capability of the sensor which acquires the position data. In order to indicate the type of position metadata carried in PES streams per event, it shall be signalled in the PMT.

The value of the timed_metadata_code shall be "POSI".

timed_metadata_descriptor_bytes of the position metadata descriptor is specified in Table 15.

Table 15 — Syntax of position_timed_metadata_descriptor

Syntax	No. bits	Mnemonic
position_timed_metadata_descriptor_bytes () {		
has_global_position_bytes	1	bslbf
has_altitude_bytes	1	bslbf
has_relative_position_bytes	1	bslbf
reserved	5	
if (has_global_position_bytes){		
global_position_bytes_length (P)	8	bslbf
}		
if(has_altitude_bytes){		
altitude_bytes_length (Q)	8	bslbf
}		
if (has_relative_position_bytes){		
relative_position_bytes_length (R)	8	bslbf
}		
global_position_sensor_capability_included	1	bslbf
altitude_sensor_capability_included	1	bslbf
relative_position_sensor_capability_included	1	bslbf
reserved	3	bslbf
if (global_position_sensor_capability_included){		
global_position_sensor_capability_length (K)	8	uimsbf
global_position_sensor_capability	K	uimsbf
}		
if (altitude_sensor_capability_included){		
altitude_sensor_capability_length (L)	8	uimsbf
altitude_sensor_capability	L	uimsbf
}		
if (relative_position_sensor_capability_included){		
relative_position_sensor_capability_length (M)	8	uimsbf
relative_position_sensor_capability	M	uimsbf
}		
}		

Semantics:

has_global_position_bytes — equal to 1 indicates that the position metadata access unit includes global_position_bytes; see subclause 8.2.5.1.

has_altitude_bytes — equal to 1 indicates that the position metadata access unit includes altitude_bytes; see subclause 8.2.5.1.

has_relative_position_bytes — equal to 1 indicates that the position metadata access unit includes relative_position_bytes; see subclause 8.2.5.1.

reserved — reserved bits, all set at 0.

global_position_bytes_length (P) — the length in bytes of the global_position_bytes field in a position metadata access unit; see subclause 8.2.5.1.

altitude_bytes_length (Q) — the length in bytes of the altitude_bytes field in an altitude metadata access unit; see subclause 8.2.5.1.

relative_position_bytes_length (R) — the length in bytes of the relative_position_bytes field in a position metadata access unit; see subclause 8.2.5.1.

global_position_sensor_capability_included — equal to 1 indicates that global position sensor capability information is present.

altitude_sensor_capability_included — equal to 1 indicates that altitude sensor capability information is present.

relative_position_sensor_capability_included — equal to 1 indicates that relative position sensor capability information is present.

global_position_sensor_capability_length (K) — length of the global_position_sensor_capability field in bytes.

global_position_sensor_capability — global position sensor capability, as specified in ISO/IEC 23005-6 for GlobalPositionSensorCapabilityType.

altitude_sensor_capability_length (L) — length of the altitude_sensor_capability field in bytes.

altitude_sensor_capability — global position sensor capability, as specified in ISO/IEC 23005-6 for AltitudeSensorCapabilityType.

relative_position_sensor_capability_length (M) — length of the relative_position_sensor_capability field in bytes.

relative_position_sensor_capability — relative position sensor capability, as specified in ISO/IEC 23005-6 for PositionSensorCapabilityType.

If used, the value of timed_metadata_access_unit_length (H) of the timed_metadata_ extension_ descriptor that contains the position_timed_metadata_descriptor_bytes is the sum of global_position_ bytes_length (P), altitude_bytes_length (Q) and relative_position_bytes_length (R) (H = P + Q + R).

8.2.5 Carriage of orientation metadata in MPEG-2 systems

8.2.5.1 Orientation metadata access unit

This subclause specifies the carriage of orientation metadata (defined in subclause 7.3) in PES streams. It specifies an orientation access unit containing orientation metadata and how to signal orientation metadata.

Each orientation metadata access unit carries an orientation metadata associated with one or more video/audio frames, i.e. multiple orientation metadata entity can be applicable for multiple video/audio frames for a given PTS interval, until a new orientation metadata entity is declared. Every orientation access unit shall be a random-access point. In order to make processing easier, each access unit should be contained in a single TS packet.

An orientation metadata access unit is a timed metadata access unit (8.2.2), of which the metadata_ access_units are specified in Table 16.

Table 16 — Syntax of the metadata_access_unit_bytes of an orientation metadata access unit

Syntax	No. bits	Mnemonic

orientation_metadata_access_unit_bytes() {
orientation_bytes
}
	8xH	uimsbf

Semantics:

orientation_bytes — includes orientation information as defined in ISO/IEC 23005-6. The length of this field (H bytes) is specified by the timed_metadata_access_unit_length field (H) in the timed_ metadata_extension_descriptor that contains the orientation_timed_ metadata_descriptor_bytes; see subclause 8.2.2 (timed_metadata_units_have_constant_ size=1).

8.2.5.2 Orientation metadata descriptor

The orientation metadata descriptor describes the type of orientation metadata, and optionally the capability of the sensor, which acquires the orientation data. When present, this descriptor shall be signalled in the PMT.

The value of the timed_metadata_code shall be "ORIE".

Timed_metadata_descriptor_bytes() of the orientation metadata descriptor is specified in Table 17.

Table 17 — Syntax of orientation metadata descriptor

Syntax	No. bits	Mnemonic
orientation_timed_metadata_descriptor_bytes () {		
orientation_sensor_capability_included	1	bslbf
Reserved	7	
if(sensor_capability_included){		
orientation_sensor_capability_length (S)	8	uimsbf
orientation_sensor_capability	S	uimsbf
}		
}		

Semantics:

sensor_capability_included — equal to 1 indicates that sensing capability information associated to the orientation entity is present.

reserved — reserved bits, all set at 0.

orientation_sensor_capability_length (S) — indicates the size (in bytes) of sensor capability information associated to the position entity.

orientation_sensor_capability — orientation sensor capability, as specified in ISO/IEC 23005-6 for OrientationSensorCapabilityType.

The value of timed_metadata_units_have_constant_size=1 of the timed_metadata_extension_ descriptor that contains the orientation_timed_metadata_descriptor_bytes.

8.2.6 Carriage of capture-mask metadata in MPEG-2 systems

8.2.6.1 Capture-mask metadata access unit

The capture-mask metadata (defined in subclause 7.7) is stored in access unit and encapsulated in a PES stream. Since each PES packet is associated with a certain PTS (presentation time stamp), the timing of activation of timed metadata is defined by the associated presentation time.

The coordinates of a capture mask are specified in 2D Cartesian coordinates samples according to ISO/IEC 23001-10:—, 6.2.2.

Detailed syntax of capture-mask metadata access unit is same as 2D Cartesian coordinates.

A capture-mask metadata access unit is a timed metadata access Unit (8.2.2).

The syntax of the metadata_access_unit_bytes of capture-mask metadata is as given in Table 18.

Table 18 — Syntax of capture mask metadata access unit

Syntax	No. bits	Mnemonic
capture_mask_metadata_access_unit_bytes () {		
top_left_x	16	uimsbf
top_left_y	16	uimsbf
width	16	uimsbf
height	16	uimsbf
interpolate	1	bslbf
reserved	7	bslbf
}		

Semantics:

top_left_x — the horizontal coordinate of the top-left corner of the rectangle region associated with the media sample of the referenced track.

top_left_y — give the vertical coordinate of the top-left corner of the rectangle region associated with the media sample of the referenced track.

width — the width of the rectangular region associated with the media sample of the referenced track.

height — the height of the rectangular region associated with the media sample of the referenced track.

interpolate — indicates the continuity in time of the successive samples. When true, the application may linearly interpolate values of the ROI coordinates between the previous sample and the current sample. When false, there shall not be any interpolation of values between the previous and the current samples.

8.2.6.2 Capture-mask metadata extension descriptor

Since one PES stream can carry one or more capture-mask metadata, it needs to signal reference width and height of related media data.

The capture-mask metadata descriptor describes the width and height of referred media.

The value of the timed_metadata_code shall be “CAMA”.

The syntax of capture-mask metadata descriptor is same as 2DCartesianCoordinatesSampleEntry of ISO/IEC 23001-10:—, 6.2.1.

The syntax of capture_mask_extension_descriptor_bytes () is as given in Table 19.

Table 19 — Syntax of capture mask extension descriptor

Syntax	No. of bits	Mnemonic
capture_mask_extension_descriptor_bytes(){		
reference_width	16	uimsbf
reference_height	16	uimsbf
}		

Semantics:

reference_width — the width of the reference rectangular space in which all ROI coordinates (top_left_x, top_left_y, width and height) are computed.

reference_height — the height of the reference rectangular space in which all ROI coordinates (top_left_x, top_left_y, width and height) are computed.

These fields allow associating a ROI metadata track with video tracks of different resolutions but representing the same visual source.

8.2.7 Carriage of audio features in MPEG-2 systems

8.2.7.1 Audio feature access unit

An audio feature access unit is a timed metadata unit (8.2.2), of which the metadata_access_unit_bytes are specified in Table 20.

Table 20 — Syntax of the metadata_access_unit_bytes of an audio feature access unit

Syntax	No. bits	Mnemonic

audio_feature_metadata_access_unit_bytes() {
audio_feature_bytes
}
	N	bslbf

Semantics:

audio_feature_bytes — this corresponds with an AudioSyncFeatureFrame sample, of which the syntax and semantics are specified in in ISO/IEC 14496-3:2009, 13.3 and 13.4. The length of this field is specified in the audio feature descriptor.

NOTE Audio_sync_feature samples are typically N=128 bit (16 bytes), which is the default frame length as specified in the AudioSyncFeatureSpecificConfig; see ISO/IEC 14496-3:2009, Table 13.4.

8.2.7.2 Audio feature descriptor

An audio feature descriptor is a timed metadata extension descriptor (8.2.3), of which the timed_metadata_descriptor_value is specified in Table 21.

Table 21 — Syntax of the timed_metadata_descriptor_value of an audio feature descriptor

Syntax	No. bits	Mnemonic

audio_feature_timed_metadata_descriptor_value () {
audio_feature_descriptor_bytes
}
	128	bslbf

Semantics:

audio_feature_descriptor_bytes — this corresponds with AudioSyncFeatureSpecificConfig sample entry, of which the syntax and semantics are specified in ISO/IEC 14496-3:2009, 13.3 and 13.4.

The value of timed_metadata_units_have_constant_size shall be 1.

The value of the timed_metadata_code shall be "STMO".

The value of timed_metadata_access_unit_length shall be identical to the value provided for the audio_sync_feature_frame_length_index (ISO/IEC 14496-3:2009, Table 13.4).

NOTE The default value timed_metadata_access_unit_length is 128. This corresponds with an audio_sync_feature_frame_length_index=0; see ISO/IEC 14496-3:2009, Table 13.4.

8.2.8 Carriage of quality metadata in MPEG-2 systems

Syntax of a quality metadata access unit

Quality metadata is defined in subclause 7.5. A quality metadata access unit is a timed metadata unit (8.2.2), of which the metadata_access_unit_bytes are specified in Table 22 below.

Table 22 — Syntax of the metadata_access_unit_bytes of a quality metadata access unit

Syntax	No. bits	Mnemonic

audio_feature_metadata_access_unit_bytes() {
mos_more_value
}
	8	bslbf

Semantics:

mos_more_value — this is the binary encoded MOS MORE value specified in subclause 7.5.2.

Audio feature descriptor

A quality metadata descriptor is a timed metadata extension descriptor (8.2.3).

The value of the timed_metadata_code shall be "MOSM".

The value of timed_metadata_units_have_constant_size shall be 1.

The value of the timed_metadata_access_unit_length shall be 1 (i.e. access unit is one byte).

The value of timed_metadata_descriptor_length shall be 0.

The field timed_metadata_descriptor_bytes shall be empty (zero bytes).

8.2.9 Carriage of correlation timestamps in MPEG-2 systems

Correlation timestamps for MPEG-2 transport streams are specified in TEMI, ISO/IEC 13818-1:2019, Annex U. TEMI (timeline for external data) provides correlations between presentation time stamps (PTS) of the MPEG-2 TS itself and timelines of other timed data, which may be carried in MPEG-DASH, ISOBMFF or another MPEG-2 transport stream; see ISO/IEC 13818-1:2019, Table U-5. It also provides correlations with NTP timestamps, PTP timestamps and time codes.

One way of using TEMI correlation timestamps is by including TEMI in an MPEG-2 TS that also carries the timed media data [media stream(s)] and/or timed metadata to which the TEMI applies. Another way is to have an MPEG-2 TS that carries TEMI stand-alone without associated timed media data [media

stream(s)] and/or timed metadata. The latter option provides additional flexibility, as illustrated in subclause 5.1.

8.3 Carriage of MORE information in MMT

8.3.1 General

MORE information shall be carried in MMTP, in accordance with ISO/IEC 23008-1, using the following methods. Firstly, MORE information is stored in an MPU, secondly, MORE information is stored in MPEG-2 system.

8.3.2 Carriage of MORE information in MMT

MMT protocol defines three packetization modes, generic file delivery mode, MPU mode and signalling message mode. When MORE information is stored in MPEG-2 system, generic file delivery mode shall be used for carriage of it. When MORE information is stored in ISO base media file format, MPU mode shall be used for carriage of it.

Carriage of MORE information using MPU mode can be achieved following two approaches:

— Timed metadata is stored in a video MPU with timed metadata track, which means the MPU consists of 3 tracks which are video track, timed metadata track, hint track. Timed metadata samples are stored in 'mdat' box after video samples. 'moof' also contains timed metadata 'traf'. When transmitting MPU through network, MPU shall be fragmented into MFU. Basically, MFU consists of its header, which is usually hint sample for media sample, and media sample. In case that transporting timed metadata, MFU contains timed metadata sample without media sample.

— Timed metadata can be in asset that is different from video and audio asset, which means the MPU consists of a single track which is timed metadata track, or more if necessary. Timed metadata samples are stored in 'mdat' box. 'moof' also contains timed metadata 'traf'.

8.4 Carriage of MORE information in DASH

8.4.1 General

Carriage of timed metadata for media orchestration in DASH MPD is useful in semi-static cases where timed metadata can be associated to DASH periods.

8.4.2 Carriage of quality metadata in MPEG DASH MPD

The quality metadata (defined in subclause 7.5) scheme allows media presentation description authors to express subjective-quality metadata of MPEG DASH media segments. A quality metadata object is represented by either an adaptation set or a sub-representation. The SupplementalProperty and/or EssentialProperty descriptors with @schemeIdUri equal to "urn:mpeg:dash:quality:2016" may be used to provide quality metadata associated with a segment. Quality metadata shall be contained exclusively in these two MPD elements (AdaptationSet and SubRepresentation).

The @value of the SupplementalProperty or EssentialProperty elements using the quality metadata scheme is a non-negative integer in decimal representation, ranging from 0-250. Its semantics is specified in subclause 7.5.2, which references ISO/IEC 23001-10:—, 4.3.6.

9 Orchestration data

9.1 General

Several approaches can be considered for the arrangement, coordination and management of media and devices using the functional elements — be it by a user or in a (semi) automated fashion. For example, a