ISO/IEC JTC 1/SC 29/WG 03 N1121

**ISO/IEC JTC 1/SC 29/WG 03  
MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Text of ISO/IEC 23090-2 Draft AMD 1 Server-side dynamic adaptation

**Status:** Approved

**Date of document:** 2024-02-18

**Source:** ISO/IEC JTC 1/SC 29/WG 03

**No. of pages:** 1 (with cover page)

**Email of Convenor:** young.L @ samsung . com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 03 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 03 N1121**

**February 2024, Online**

|  |  |
| --- | --- |
| **Title** | **Text of ISO/IEC 23090-2 Draft AMD 1 Server-side dynamic adaptation** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **23488** |

Information technology — Coded representation of immersive media — Part 2: Omnidirectional media format — Amendment 1: Server-side dynamic adaptation

Add new Annex H after Annex G:

Annex H

**(Normative)**

**Server-side dynamic adaptation**

H.1 General

This annex provides an overall architecture and adaptation parameters to support server-side dynamic adaptation. These parameters can be used as part of HTTP requests for OMAF related media segments in URL parameters as specified in ISO/IEC 23009-1, in the forms of

* URL query parameters, and/or
* HTTP header parameters.

It is expected that, when received at dynamic adaptation server, OMAF content (e.g., in the form of streaming segments) will be dynamically selected and/or adapted, according to the received adaptation parameters, and delivered back to an OMAF player.

H.2 Overall architecture

Figure H.1 shows a typical content flow process for an omnidirectional media application.

NOTE 1 In this content flow process, Dynamic adaptation service is a different server process than content generation process.

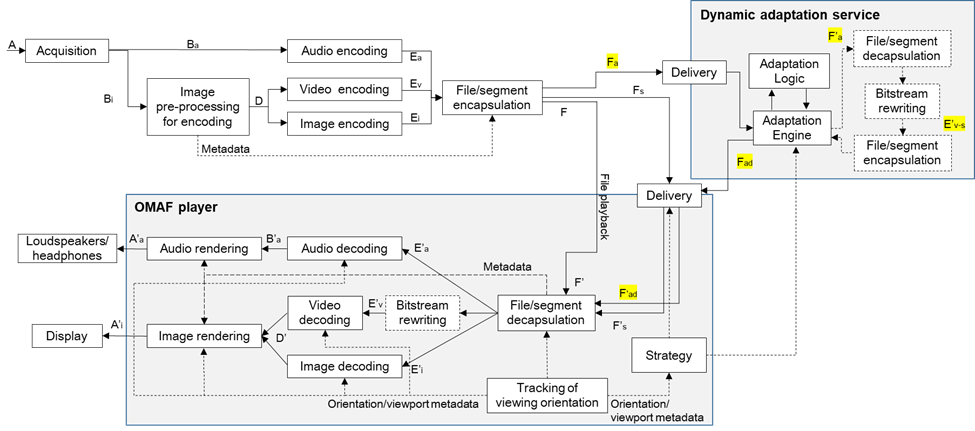


Figure H.1 — Content flow process for omnidirectional media with projected video

The following interfaces are specified in this document:

* E'a, E'v, E'i: audio bitstream, video bitstream, coded image(s), respectively; see Clause10.
* E'v-s: video bitstream, see Clause 10.
* F/F': media file; see Clause7. Moreover, media profiles specified in Clause 10 include the specification of the track formats for F/F', which may contain constraints on the elementary streams contained within the samples of the tracks.
* Clause 8 specifies the delivery related interfaces for DASH delivery.
* Clause 9 specifies the delivery related interfaces for MMT delivery.

The other interfaces in Figure 1 are not specified in this document.

NOTE 2 While the syntax and semantics of the bitstreams Ea, Ev, and Ei are the same as those for E'a, E'v, E'i, respectively, the input interface to the file/segment encapsulation module is not specified.

NOTE 3 While the syntax and semantics of the bitstreams Ev, is the same as for E'v-s the input interface to the file/segment encapsulation module is not specified.

A real-world audio-visual scene (A) is captured by audio sensors as well as a set of cameras or a camera device with multiple lenses and sensors. The acquisition results in a set of digital image/video (Bi) and audio (Ba) signals. The cameras/lenses typically cover all directions around the centre point of the camera set or camera device, thus the name of 360-degree video.

Audio may be captured using many different microphone configurations and stored as several different content formats, including channel-based signals, static or dynamic (i.e., moving through the 3D scene) object signals, and scene-based signals (e.g., Higher Order Ambisonics). The channel-based signals typically conform to one of the loudspeaker layouts defined in ISO/IEC 23091-3. In an omnidirectional media application, the loudspeaker layout signals of the rendered immersive audio program are binauralized for presentation via headphones.

For audio, no stitching process is needed, since the captured signals are inherently immersive and omnidirectional.

This document specifies the following types of omnidirectional video and images, which differ in the architecture in the image pre-processing for encoding and in the image rendering processing blocks:

* Projected omnidirectional video/images:
* Image pre-processing for encoding: The images (Bi) of the same time instance are stitched, possibly rotated, and projected onto a 2D picture coordinates using a mathematically specified projection format. Optionally, the resulting projected pictures may be further mapped region-wise onto a packed picture. Either projected pictures or packed pictures are subject to video or image encoding.
* Image rendering: Either regions of the decoded packed pictures (if region-wise packing has been applied) or the entire projected picture (otherwise) is mapped onto a rendering mesh suitable for the projection format in use.
* Fisheye omnidirectional video/images:
* Image pre-processing for encoding: Circular images (Bi) captured by fisheye lenses are arranged onto a 2D picture, which is then input to video or image encoding.
* Image rendering: The decoded circular images are stitched using the signalled fisheye-specific parameters.
* Mesh omnidirectional video:
* Image pre-processing for encoding: A 3D mesh consisting of mesh elements is generated, where mesh elements can be either parallelograms or regions of a sphere surface. The images (Bi) of the same time instance are stitched, possibly rotated, and projected onto the 3D mesh. Mesh elements are mapped onto rectangular regions of one or more 2D pictures, which are input to video encoding.
* Image rendering: Rectangular regions of the decoded 2D picture(s) are mapped to the 3D mesh, which is used directly as the rendering mesh.

Further details of the architecture for projected, fisheye, and mesh omnidirectional video/images are provided in subclauses4.3, 4.4, and4.5, respectively.

The pre-processed pictures (D) are encoded as coded images (Ei) or a coded video bitstream (Ev). The captured audio (Ba) is encoded as an audio bitstream (Ea). The coded images, video or audio are then composed into a media file for file playback (F) or a sequence of an initialization segment and media segments for streaming (Fs, Fa), according to a particular media container file format. In this document, the media container file format is the ISO Base Media File Format specified in ISO/IEC 14496-12. The file encapsulator also includes metadata into the file or the segments.

The segments Fa are delivered using a delivery mechanism to Dynamic adaptation service.

In the Dynamic adaptation service, the dynamic adaptation parameters, such as bitrate (see Clause H.3), are sent from the Strategy module in OMAF player to the Adaptation engine module in Dynamic adaptation service, which determines which segments to be received based on the dynamic adaptation parameters, according to what Adaptation logic module instructs. The received segments (Fad) carried in tracks determined by delivery mechanism are identical to the segments (Fa) except when bitstream rewriting is needed. Viewport-dependent video may be carried in multiple tracks, which may be processed and extract the coded video streams in File/segment decapsulator. The coded video streams are merged in the bitstream rewriting into a single video bitstream (E'v-s). This single video bitstream (E'v-s) is composed into segments for streaming (Fad) in the File/segment encapsulation module, according to a particular media container file format. The segments Fad are delivered to an OMAF player.

NOTE 4 When the OMAF player uses the Dynamic adaptation service, the OMAF player can perform a reduced process because, instead of Dynamic adaptation service handling the adaptation and bitstream merging function, the OMAF player only needs to send the dynamic adaptation parameters.

NOTE 5 If the segments (Fad) are contained in a file and delivered to an OMAF player, the file is a file that contains the adapted segments, and is not an entire duration file containing content longer than the segments.

The segments Fs are delivered using a delivery mechanism to an OMAF player.

The file that the file encapsulator outputs (F) is identical to the file that the file decapsulator inputs (F'). A file decapsulator processes the file (F') or the received segments (F's and F’ad) and extracts the coded bitstreams (E'a, E'v, or E'i) and parses the metadata. Viewport-dependent video may be carried in multiple tracks, which may be merged in the bitstream rewriting into a single video bitstream E'v prior to decoding. The audio, video or images are then decoded into decoded signals (B'a for audio, and D' for images/video). In the image rendering block, the decoded pictures (D') are projected onto the screen of a head-mounted display or any other display device based on the metadata parsed from the file. Likewise, decoded audio (B'a) is rendered, e.g., through headphones, according to the current viewing orientation. The current viewing orientation is determined by the viewing orientation tracking functionality. When a head-mounted display is in use, the viewing orientation tracking can involve head tracking and possibly also eye tracking. When sphere-relative overlays are in use, the viewing orientation tracking functionality can include or be complemented by viewing position tracking and rendering of overlays with background visual media can take both the viewing position and the viewing orientation into account. Besides being used by the renderer to render the appropriate part of decoded video and audio signals, the current viewing orientation may also be used by the video and audio decoders for decoding optimization. In viewport-dependent delivery, the current viewing orientation is also passed to the strategy module in OMAF player, which determines the video tracks to be received based on the viewing orientation.

The process described above is applicable to both live and on-demand use cases.

H.3 Signalling of server-side dynamic adaptation information

Server-side dynamic adaptation (SSDA) as complementary to client-side dynamic adaptation (CSDA) suggests that some dynamic adaptation can be performed at the server side, instead of at the client side, as shown in the Figure H.2 below.



Figure H.2 — Example streaming system using server-side dynamic adaptation

It should be noted that in an SSDA scheme, the Streaming Client can still make some static selection (such as those related to video codec profile, screen size and encryption algorithm), and only leave dynamic adaptation to the server, by collecting and passing dynamic adaptation parameters needed for Adaptation Logic to the server as part of (HTTP) segment requests. Moreover, in a hybrid mode, SSDA and CSDA schemes can be jointly implemented to share dynamic adaptation tasks split between the Streaming Client and Server.

The segment URL parametrization scheme identified by URN "urn:mpeg:dash:urlparam:2014" or extended parametrization scheme identified by URN "urn:mpeg:dash:urlparam:2016" may be present at Adaptation Set level or at the Representation level.

In addition, an **EssentialProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:omaf:2022:serverSideDynamicAdaptation" is referred to as a server-side dynamic adaptation (SSDA) descriptor. When an SSDA descriptor exists in AdaptationSet or Representation, the OMAF player making an SSDA request shall add the query parameter(s) to the segment file URL or alternatively to the HTTP header.

In addition, Adaptation Sets and Representations which have SSDA descriptor shall either have or inherit the following DASH profile description in @profiles DASH attribute:  
 urn:mpeg:mpegI:omaf:dash:profile:serverSideDynamicAdaptation.

The list of parameters provided in the Table H.1 is for the purpose of track selection or switching.

**Table H.1** — **Parameters for track selection or switching**

|  |  |  |
| --- | --- | --- |
| Query parameter name | Header name | Query parameter value definition |
| scsw | SSDA-Request | Indicates the width of screen in units of pixels.  The value is the same as the width field of VisualSampleEntry. |
| scsh | SSDA-Request | Indicates the heights of screen in units of pixels.  The value is the same as the height field of VisualSampleEntry. |
| bitr | SSDA-Request | Indicates the bitrate (kbps)  The value is calculated by total bit count of the samples in the track divided by (the total duration of the samples of the segment \* 1000). |
| frar | SSDA-Request | Indicates the frame rate (fps)  The value is calculated by number of samples in the segment divided by the total duration of the samples of the segment. |
| nvws | SSDA-Request | Indicate the number of views in the track |

With these parameters in DASH HTTP requests for segments related to a track selection and switching, it is expected to return an HTTP response containing:

* a segment from a track matching the requirements from the parameters, selected or produced with server-side dynamic adaptation.

The list of parameters provided in the Table H.2 is for the purpose of viewport related adaptation.

**Table H.2** — **Parameters for viewport related adaptation**

|  |  |  |
| --- | --- | --- |
| Query parameter name | Header name | Query parameter value definition |
| azim | SSDA-Request | Specifies the azimuth of the centre point of the sphere region in units of 2−16 degrees. |
| elev | SSDA-Request | Specifies the elevation of the centre point of the sphere region in units of 2−16 degrees. |
| tilt | SSDA-Request | Specifies the tilt angle of the sphere region, in units of 2−16 degrees. |
| azrg | SSDA-Request | Specifies the azimuth range of the sphere region through the centre point of the sphere region in units of 2−16 degrees. |
| elrg | SSDA-Request | Specifies the elevation range of the sphere region through the centre point of the sphere region in units of 2−16 degrees. |
| styp | SSDA-Request | Specifies the shape type of the sphere region, as specified in 7.7.2.3. |

With these parameters in DASH HTTP requests for segments related to a viewport, it is expected to return an HTTP response containing:

* a viewport segment, selected or produced with server-side dynamic adaptation. Here, the viewport segment covers the viewport of at least the same quality as the background, and the viewport segment may include some margin to the viewport.

H.4 Extensions to the ISOBMFF for server-side dynamic adaptation

**H.4.1 Region-wise packing sample group**

**H.4.1.1 Definition**

The 'rwpk' grouping\_type for sample grouping specifies the mapping between packed regions and the corresponding projected regions and specifies the location and size of the guard bands, if any.

If the region-wise packing sample group exist, the region-wise packing box shall not exist.

**H.4.1.2 Syntax**

class ResionWisePackingEntry() extends VisualSampleGroupEntry ('rwpk'){

RegionWisePackingStruct()region\_wise\_packing\_struct;

}

**H4.1.3 Semantics**

Subclause 7.5.3 applies with the following additional constraint:

* packed\_picture\_width and packed\_picture\_height shall have such values that packed\_picture\_width is an integer multiple of width and packed\_picture\_height is an integer multiple of height, where width and height are syntax elements of the VisualSampleEntry containing this box.

H.5 Segment format for server-side dynamic adaptation

When any sample of a media segment is mapped to the 'rwpk' sample group, all samples of the media segment shall be mapped to the same 'rwpk' sample group description entry.