**ISO/IEC 23008-1:2023(E)**

ISO/IEC JTC 1/SC 29/WG 03

Date: 2025-10

**Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 1: MPEG media transport (MMT) — Amendment 2: Additional asset descriptor**

CD stage

**Warning for WDs and CDs**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

© ISO 20XX

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO’s member body in the country of the requester.

ISO copyright office

CP 401 • Ch. de Blandonnet 8

CH-1214 Vernier, Geneva

Phone: +41 22 749 01 11

Email: copyright@iso.org

Website: www.iso.org

Published in Switzerland

Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 1: MPEG media transport (MMT) — Amendment 2: Additional asset descriptor

*Add the following new subclause after 10.5.13.4 and before 10.6:*

### Alternative descriptor

* + - 1. **General**

This descriptor signals alternative relationship information among assets. The assets signalled by this descriptor can be used instead of the asset including this descriptor in its asset descrioptor loop.

* + - 1. **Syntax**

Table 95 shows the syntax of the alterantive asset descriptor.

Table 95 — Alternative asset descriptor

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Values** | **No. of bits** | **Mnemonic** |
| Alternative\_asset\_descriptor () { |  |  |  |
| **descriptor\_tag** |  | **16** | **uimsbf** |
| **descriptor\_length** |  | **8** | **uimsbf** |
| **number\_of\_alternatives**  for (i=0 ; i<N ; i++){  asset\_id()  } | **N** | **8** | **uimsbf** |
| } |  |  |  |

* + - 1. **Semantics**

number\_of\_alternatives specifies the number of assets which have alternative relationship.

asset\_id()provides the identifier of the asset referenced by the asset this descriptor is associated with as an alternative.

### MUR descriptor

* + - 1. **General**

This descriptor contains information of the edit id and related MPUs. When there is a need for the content customized transmission and presentation, the MUR\_descriptor() is written as a part of asset descriptor in MPT.

* + - 1. **Syntax**

Table 96 shows the syntax of the MUR descriptor.

Table 96 — MUR descriptor syntax

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Values** | **No. of bits** | **Mnemonic** |
| MUR\_descriptor () { |  |  |  |
| **descriptor\_tag** |  | **16** | **uimsbf** |
| **descriptor\_length** |  | **8** | **uimsbf** |
| for (i=0 ; i<N ; i++){  **mpu\_sequence\_number**  **edit\_id**  } |  | **32**  **8** | **uimsbf**  **uimsbf** |
| } |  |  |  |

* + - 1. **Semantics**

descriptor\_tag - a tag value indicating the type of a descriptor.

descriptor\_length - indicates the length in bytes counting from the next byte after this field to the last byte of the descriptor.

mpu\_sequence\_number - enumerates the sequence number of all the MPUs which share the same edit\_id.

edit\_id - indicates the identification of MPU within specified content version.

The content provider would give different edit\_id to specific MPU at the same time when it is generated, or look up the mapping relation in the config information at server side. During the delivery of the content, the client side consumed the targeted content based on the edit\_id provided in the MUR\_descriptor() while the presentation time of the selected MPUs would be presented in the MPU\_timestamp\_descriptor().

During the presentation, under the situation of packet loss and the buffer is not enough to wait for retransmission, the client is unable to get the duration time of each MPU inside MPU box. Meanwhile, considering about consumers’ interactions such as starts, pauses, seeks and stops, the cost re-generating and re-transmitting of the existing MPU\_timestamp\_descriptor which containing absolute presentation time of the MPUs is too high. So MPU\_presentation\_descriptor() is used in MP table to indicate the version and correspondent MPU duration information which is located at chapter 10.5.16.

### MPU presentation descriptor

* + - 1. **General**

This descriptor contains information about the duration of specified MPUs. A MPT may contain an MPU\_presentation\_descriptor() for each selected media version within an asset.

* + - 1. **Syntax**

Table 97 shows the syntax of the MPU presentation descriptor.

Table 97 — MPU presentation descriptor syntax

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Values** | **No. of bits** | **Mnemonic** |
| MPU\_presentation\_descriptor () { |  |  |  |
| **descriptor\_tag** |  | **16** | **uimsbf** |
| **descriptor\_length** |  | **8** | **uimsbf** |
| for (i=0 ; i<N ; i++){  **mpu\_sequence\_number**  **mpu\_duration**  } |  | **32**  **64** | **uimsbf**  **uimsbf** |
| } |  |  |  |

* + - 1. **Semantics**

descriptor\_tag - a tag value indicating the type of a descriptor.

descriptor\_length - indicates the length in bytes counting from the next byte after this field to the last byte of the descriptor.

mpu\_sequence\_number - enumerates the sequence number of all the MPUs which share the same edit\_id.

mpu\_duration - indicates the duration time of specified MPU.

Specifically, based on the selected video version, the client analyses duration time of each MPU within the corresponding video version and calculates the actual presentation time. The client would modify the presentation timeline dynamically especially for repeated consumptions within the same asset.

In the condition of packet loss, the client would request for the losing media unit when the frame cache is not empty. If the cache is running out, the client would analyse the MPU\_presentation\_descriptor() to get the corresponding duration time of the losing MPU and skip on the presentation timeline. Therefore, the client successfully keep the synchronization of the whole service and guarantee smooth presentation procedure.

### MUR Live descriptor

* + - 1. **General**

This descriptor contains information of the edit id and corresponding MPUs. When there is a need for the content differentiated transmission in live streaming services, the MUR\_Live\_descriptor() is written as a part of asset descriptor in MPT.

* + - 1. **Syntax**

Table 98 shows the syntax of the MUR Live descriptor.

Table 98 — MUR Live descriptor syntax

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Values** | **No. of bits** | **Mnemonic** |
| MUR\_Live\_descriptor () { |  |  |  |
| **descriptor\_tag** |  | **16** | **uimsbf** |
| **descriptor\_length** |  | **8** | **uimsbf** |
| for (i=0 ; i<N ; i++){  **mpu\_sequence\_number**  **edit\_id**  } |  | **32**  **8** | **uimsbf**  **uimsbf** |
| } |  |  |  |

* + - 1. **Semantics**

descriptor\_tag - a tag value indicating the type of a descriptor.

descriptor\_length - indicates the length in bytes counting from the next byte after this field to the last byte of the descriptor.

mpu\_sequence\_number - enumerates the sequence number of all the MPUs which share the same edit\_id.

edit\_id - indicates the identification of MPU within specified content version.

‘edit\_id’ is a newly added attribute to identify specific MPUs within a set of media source. It may be of assigned based on the particular condition of the client side. Live broadcasting requires real-time. There is no time for the server to analyze the related information and generate signaling messages. Meanwhile, the user is not able to notify the server about some personalized requirement, especially it may cause extra waste of power.

Therefore, under low power circumstance, the content provider would give different edit\_id to specific MPU at the same time when it is generated. During the delivery of the content, the client side consumed the targeted content based on the edit\_id provided in the MUR\_Live\_descriptor() while the presentation time of the selected MPUs would be presented in the MPU\_timestamp\_descriptor() at the same time.

*Add the following new subclause after 10.4.27 Service List (SL) information message and before 10.5:*

### POST Request/Response (PRR) Message

* + - 1. **Introduction**

POST Request/Response Message (PRR message) should be used when a MMT client needs to send a POST request to the MMT server. And the server will send the client a corresponding response.

* + - 1. **Syntax**

Table 80 shows the syntax of the POST Request/Response Message.

Table 80 — PRR message syntax

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Values** | **No. of bits** | **Mnemonic** |
| PRR\_message () {  **message\_id**  **version**  **length**  message\_payload {  **reserved**  **PRR\_type**  if(PRR\_type == 0) {  **POST\_serial\_number**  **mime\_type()**  **PRR\_data\_length**  for(j == 0; j < N1; j++) {  **PRR\_data\_byte**  }  }  else if(PRR\_type == 1) {  **Response\_serial\_number**  **status\_number**  if(status\_number == 0x02){  **mime\_type()**  **PRR\_data\_length**  for(j == 0; j < N2; j++) {  **PRR\_data\_byte**  }  }  }  }  } | **‘111 1111’**  **N1**  **N2** | **16**  **8**  **32**  **7**  **1**  **8**  **16**  **8**  **8**  **8**  **16**  **8** | **uimsbf**  **uimsbf**  **uimsbf**  **bslbf**  **bslbf**  **uimsbf**  **uimsbf**  **uimsbf**    **uimsbf**  **uimsbf**  **uimsbf**  **uimsbf** |

* + - 1. **Semantics**

message\_id - indicates the identifier of the PRR messages.

version - indicates the version of the PRR messages.

length - a 32-bit field for conveying the length of the PRR message in bytes, counting from the beginning of the next field to the last byte of the PRR message. The value ‘0’ is not valid for this field.

PRR\_type - indicates the type of the PRR messages as shown in Table 81.

Table 81 — value of PRR\_type

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | POST Request |
| 1 | POST Response |

POST\_serial\_number - indicates the POST serial number the client send. It shall be modulo-256 incremented per number change.

mime\_type() - provides mime type as defined in 10.6.4.

PRR\_data\_length - the length of the POST request or response data in bytes.

PRR\_data\_byte - specifies a byte in the data.

Response\_serial\_number - indicates the serial number of response from the server. It shall be modulo-256 incremented per number change. The request and response match up when the value is equal to the POST\_serial\_number.

status\_number - indicates number of response status as shown in Table 82.

Table 82 — value of PRR\_type

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0x00 | POST request failed, the POST request is not received in a specified time. |
| 0x01 | POST request succeeded. |
| 0x02 | POST request succeeded, response data is in this message. |
| 0x03~0x7F | Reserved for ISO |
| 0x80~0xFF | Reserved for private use |