ITUPublications
Recommendations

International Telecommunication Union
Radiocommunication Sector

# Recommendation ITU-R BT.2074-2

**(11/2023)**

BT Series: Broadcasting service (television)

# Service configuration, media transport protocol, and signalling information for MMT-based broadcasting systems

## Foreword

The role of the Radiocommunication Sector is to ensure the rational, equitable, efficient and economical use of the radio-frequency spectrum by all radiocommunication services, including satellite services, and carry out studies without limit of frequency range on the basis of which Recommendations are adopted.

The regulatory and policy functions of the Radiocommunication Sector are performed by World and Regional Radiocommunication Conferences and Radiocommunication Assemblies supported by Study Groups.

## Policy on Intellectual Property Right (IPR)

ITU-R policy on IPR is described in the Common Patent Policy for ITU-T/ITU-R/ISO/IEC referenced in Resolution ITU-R 1. Forms to be used for the submission of patent statements and licensing declarations by patent holders are available from http://www.itu.int/ITU-R/go/patents/en where the Guidelines for Implementation of the Common Patent Policy for ITU-T/ITU-R/ISO/IEC and the ITU-R patent information database can also be found.

### Series of ITU-R Recommendations

(Also available online at https://www.itu.int/publ/R-REC/en)

| Series | Title |
|---|---|
| **BO** | Satellite delivery |
| **BR** | Recording for production, archival and play-out; film for television |
| **BS** | Broadcasting service (sound) |
| **BT** | **Broadcasting service (television)** |
| **F** | Fixed service |
| **M** | Mobile, radiodetermination, amateur and related satellite services |
| **P** | Radiowave propagation |
| **RA** | Radio astronomy |
| **RS** | Remote sensing systems |
| **S** | Fixed-satellite service |
| **SA** | Space applications and meteorology |
| **SF** | Frequency sharing and coordination between fixed-satellite and fixed service systems |
| **SM** | Spectrum management |
| **SNG** | Satellite news gathering |
| **TF** | Time signals and frequency standards emissions |
| **V** | Vocabulary and related subjects |

***Note**: This ITU-R Recommendation was approved in English under the procedure detailed in Resolution ITU-R 1.*

*Electronic Publication*
Geneva, 2023

# RECOMMENDATION ITU-R BT.2074-2

## Service configuration, media transport protocol, and signalling information for MMT-based broadcasting systems

(2015-2017-2023)

### Scope

This Recommendation defines the service configuration, media transport protocol, and signalling information required for broadcasting and broadband systems using ISO/IEC 23008-1 (MPEG Media Transport). It specifies the constraints to ISO/IEC 23008-1 for MMT-based broadcasting systems.

**Keywords**

Transport, MMT, SMT, multiplexing, IP-based broadcasting, hybrid delivery, UHDTV

The ITU Radiocommunication Assembly,

*considering*

*a)* that multimedia services consist of various media components such as audio, video, closed captions, and other data;

*b)* that various media components for multimedia services may be delivered in broadcasting channels and broadband networks;

*c)* that multimedia services have also been introduced in broadband networks where IP packets are used;

*d)* that an IP-friendly media transport protocol is desirable for multimedia broadcasting systems to enable harmonization of broadcasting and broadband;

*e)* that synchronized presentation of various media components over various delivery channels is required for multimedia broadcasting applications;

*f)* that efficient and reliable transport of various media components is required over broadcasting channels;

*g)* that ISO/IEC 23008-1 "MPEG Media Transport (MMT)" specifies an encapsulation format of media components, delivery protocol, and signalling information for various applications including broadcasting applications;

*h)* that a common MMT protocol packet syntax has been specified in ISO/IEC 23008-1;

*i)* that practical implementation of broadcasting systems may require certain constraints on ISO/IEC 23008-1;

*j)* that it is desirable for such constraints to be in common with MMT-based broadcasting systems for development and deployment of systems including receiver terminals;

*k)* that GB/T 33475-6 "Smart Media Transport (SMT)" specifies an extension of MMT while preserving the basic architecture of MMT,

*recommends*

1 that broadcasting systems using MPEG Media Transport as per ISO/IEC 23008-1 should be designed on the basis of the system structure and service configuration described in Annex 1;

2 that the broadcasting systems using MPEG Media Transport should comply with the media transport protocol and signalling information described in Annex 2.

NOTE – Attachment 1 to Annex 1 shows the extension of MMT by Smart Media Transport (SMT). Attachment 1 to Annex 2 shows additional signalling information specified in ARIB systems.

## References

Normative references:

– ISO/IEC 23008-1: 2023: Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 1: MPEG media transport (MMT)

– ISO/IEC 23009-1:2022 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats

– ISO/IEC 14496-12:2020 Information technology – Coding of audio-visual objects – Part 12: ISO base media file format

Informative references:

– Recommendation ITU-T H.222.0 | ISO/IEC 13818-1: 2022: Information technology – Generic coding of moving pictures and associated audio information: Systems

– IETF RFC 768: User Datagram Protocol, Aug. 1980

– IETF RFC 791: Internet Protocol, Sep. 1981

– IETF RFC 2460: Internet Protocol, Version 6 (IPv6) Specification, Dec. 1998

– IETF RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification, June 2010

– Recommendation ITU-R BT.1869 – Multiplexing scheme for variable-length packets in digital multimedia broadcasting systems

– Recommendation ITU-T H.265 | ISO/IEC 23008-2 (2020) – Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding

## Abbreviations

| | |
|---|---|
| AAC | Advanced audio coding |
| ADC | Asset delivery characteristic |
| AIT | Application information table |
| AL-FEC | Application layer forward error correction |
| ALS | Audio lossless coding |
| AMT | Address map table |
| BIT | Broadcaster information table |
| CA | Conditional access |
| CAS | Conditional access system |
| CDT | Common data table |
| CRI | Clock relation information |
| CEU | Common encapsulation unit |
| DCI | Device capability information |
| DCM | Download control message |

| | |
|---|---|
| DMM | Download management message |
| ECM | Entitlement control message |
| EIT | Event information table |
| EMM | Entitlement management message |
| EPG | Electronic programme guide |
| GFD | Generic file delivery |
| GOP | Group of pictures |
| HEVC | High efficiency video coding |
| HRBM | Hypothetical receiver buffer model |
| IP | Internet Protocol |
| IRAP | Intra random access point |
| LAOS | Low overhead audio stream |
| LATM | Low overhead audio transport multiplex |
| LCT | Layout configuration table |
| LDT | Linked description table |
| MFU | Media fragment unit |
| MMT | MPEG media transport |
| MMTP | MMT protocol |
| MPI | MMT presentation information |
| MPT | MMT package table |
| MPU | Media processing unit |
| NIT | Network information table |
| NPT | Normal play time |
| NTP | Network time protocol |
| PA | Package access |
| PLT | Package list table |
| SDT | Service description table |
| SDTT | Software download trigger table |
| SMT | Smart media transport |
| SMTP | SMT protocol |
| TLV | Type length value |
| UDP | User datagram protocol |

# Annex 1

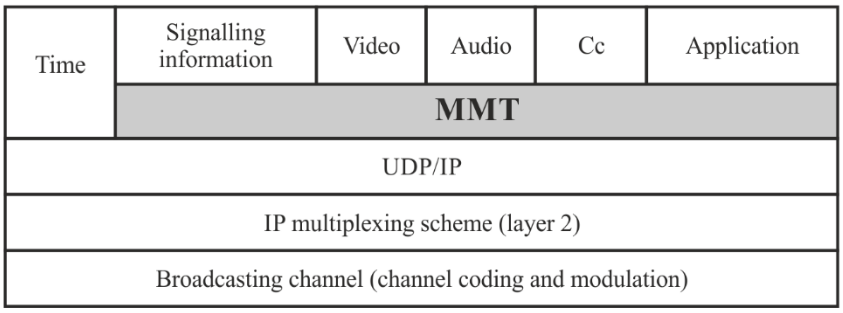
## System structure and service configuration

### 1 System structure

This section describes the general structure of MMT-based broadcasting systems. Figure 1 shows the protocol stack of MMT-based broadcasting systems.

FIGURE 1

**Protocol stack of MMT-based broadcasting systems**


Time
Signalling information
Video
Audio
Cc
Application
MMT
UDP/IP
IP multiplexing scheme (layer 2)
Broadcasting channel (channel coding and modulation)

BT.2074-01

In these systems, media components, such as video, audio, and closed captions (cc), constituting a TV programme are encapsulated into media fragment units (MFUs)/media processing units (MPUs). They are carried as MMT protocol (MMTP) payloads of MMTP packets and delivered in IP packets. Data applications that are related to a TV programme are also encapsulated into MFUs/MPUs, carried in MMTP packets, and delivered in IP packets.

IP packets generated like this are multiplexed over broadcasting channels with an IP multiplexing scheme, also referred to as a layer 2 (L2) protocol, e.g. the TLV multiplexing scheme described in Recommendation ITU-R BT.1869.

The systems also have MMT signalling information (MMT-SI). MMT-SI is signalling information on the structure of a TV programme and associated information on TV services like the electronic programme guide (EPG). MMT-SI is carried in MMTP packets and delivered in IP packets.

In order to provide Coordinated Universal Time (UTC) in broadcasting systems for receiver terminals to synchronize with broadcast station, time information is also delivered in IP packets.

### 2 Service configuration

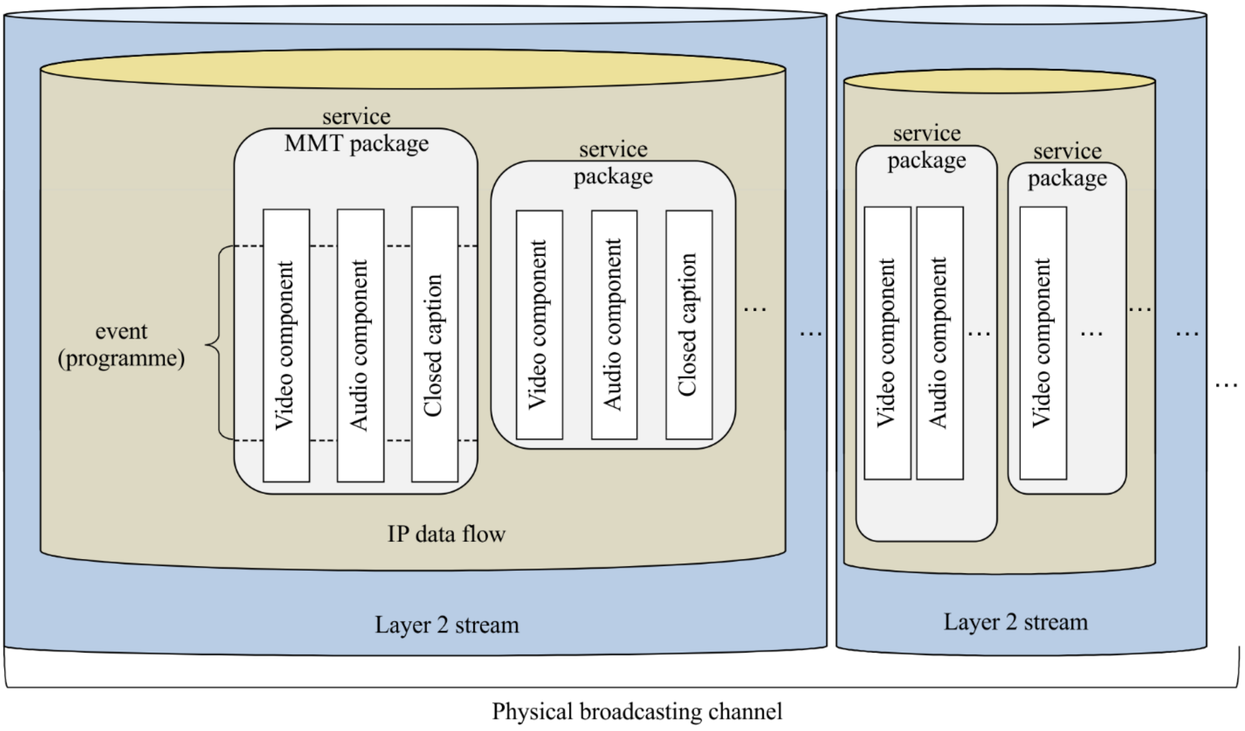
### 2.1 Services in a broadcasting channel

ISO/IEC 23008-1 specifies the MMT package as a logical structure of content. The MMT package includes presentation information and associated Assets that constitute content.

A broadcasting service is generally a series of TV programmes. In MMT-based broadcasting systems, one MMT package corresponds to one broadcasting service. The relationship between the broadcasting service and the MMT package is shown in Fig. 2. As shown in the Figure, one TV programme is distinguished from the rest of the service by its start and end times and corresponds to one event.

FIGURE 2

**Relationship between a broadcasting service and MMT package in a broadcasting channel**



BT.2074-02

In ISO/IEC 23008-1, an Asset is defined as a media component. An Asset is equivalent to a series of MPUs. In MMT-based broadcasting systems, one TV programme is an MMT package including one or more Assets and signalling information. A package access (PA) Message is an MMT-SI, and the MMT package table (MPT) carried in the PA message identifies Assets constituting the TV programme.

Multiple MMT packages can be delivered in one IP data flow, as shown in Fig. 2. Here, an IP data flow is defined as a sequence of IP packets of which the source IP address, destination IP address, protocol, source port number, and destination port number are the same combination. There may be other IP data flows carrying content for download services or extended services in addition to IP data flows carrying MMT packages.

Multiple IP data flows might be multiplexed into one layer 2 stream. The layer 2 stream includes signalling information for demultiplexing IP packets from broadcasting signals.

## 2.2 Services in broadcasting channels and broadband networks

ISO/IEC 23008-1 has been developed to support delivery of media data over heterogeneous networks including broadcasting channels and broadband networks. In the MMT specifications, broadcasting channels and broadband networks can be treated in the same way for delivery of content. Figure 3 shows a service configuration using both broadcasting channels and broadband networks.

In Fig. 3, video component 1, audio component 1, and closed caption 1 are delivered on broadcasting channels. In addition to these components, video component 2, audio component 2, and closed caption 2 are delivered on broadband networks.

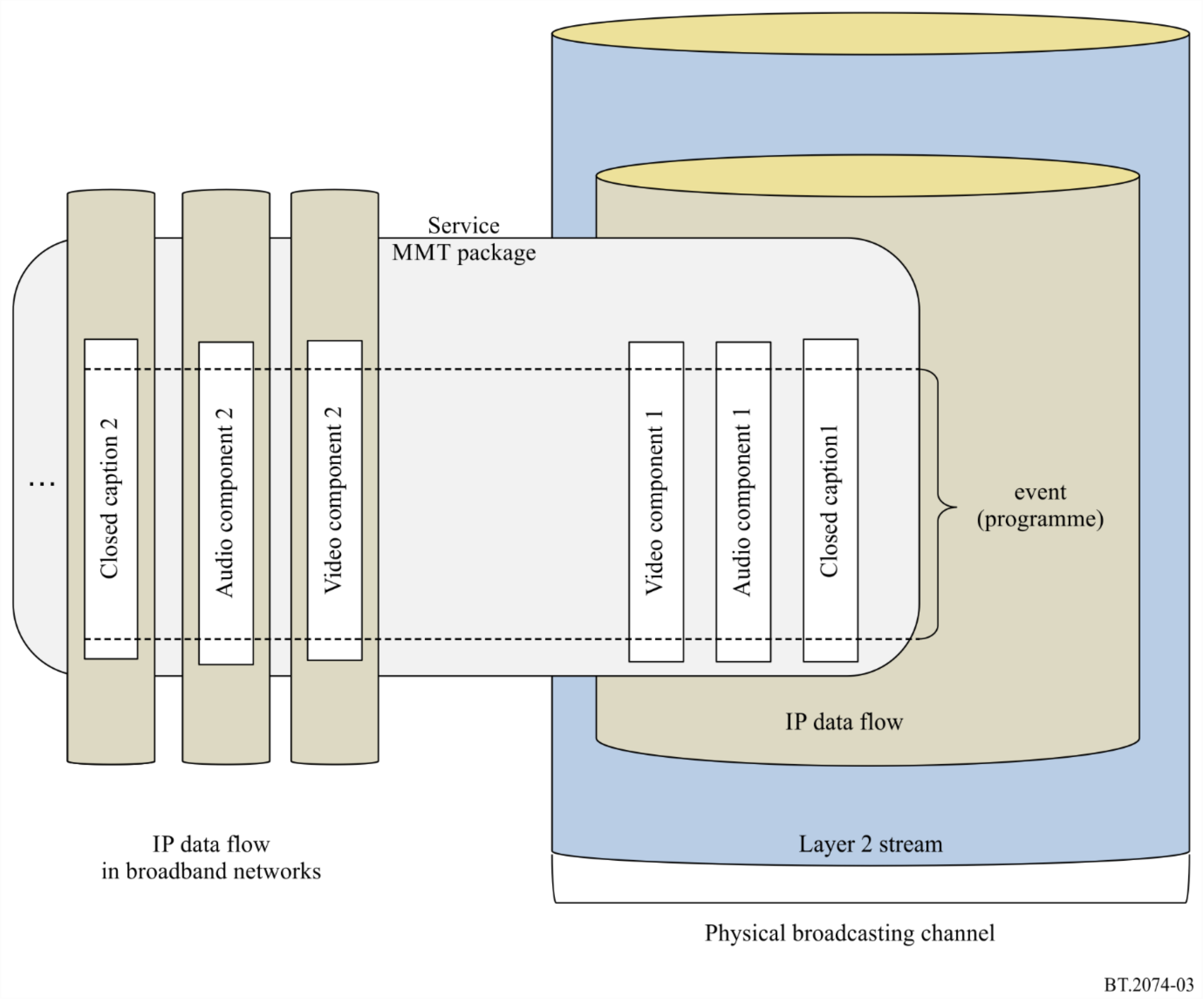
In the broadcasting channels, the three components are multiplexed into one IP data flow and delivered in one layer 2 stream, since all transmitted information is delivered to all receiver terminals.

On the other hand, in the broadband networks, components are delivered as a separate IP data flow, since each component is delivered to the receiver terminal requesting it.

In MMT-based broadcasting systems, media components delivered in different channels can easily be included in one MMT package. MMT-based broadcasting systems support hybrid delivery of multimedia content.

FIGURE 3

**Service configuration over both broadcasting channels and broadband networks**



BT.2074-03

# Attachment 1
# to Annex 1
# (normative)

## Extension of MMT by SMT

### Summary

SMT extends MMT by using private extension methods MMT has provided while preserving the basic architecture of MMT.

## Bibliography

GB/T 33475-6 “Smart Media Transport (SMT)”: It specifies an encapsulation format of media components, delivery protocol, signalling information, media presentation, and adaptive forward error correction mechanism for various applications including broadcasting applications.

## 1 Service configuration extension by SMT

The multimedia services consist of various media components such as audio, video, closed captions, and other data which may be delivered in broadcasting channels and broadband networks. A common media transport protocol is desirable for multimedia broadcasting systems to enable harmonization of broadcasting and broadband.

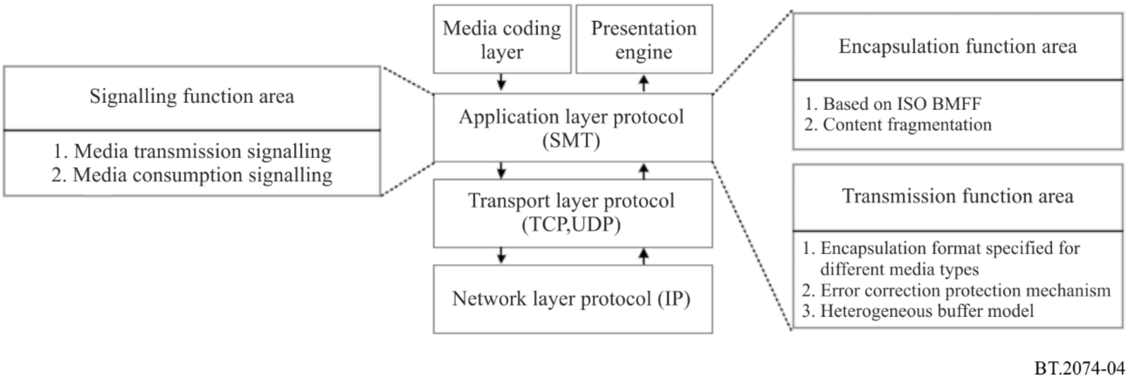
SMT specifies an IP-based media protocol over both broadcasting networks and broadband networks, covering the media data encapsulation, media data transport, signalling information, media presentation. SMT extends MMT by using private extension methods MMT has provided while preserving the basic architecture of MMT. Some encapsulation format of media components, delivery protocol, signalling information and adaptive forward error correction coding for various applications are defined to enable smart and efficient transport various media formats and components.

## 2 System structure extension by SMT

This section describes the general structure of SMT-based heterogeneous systems. Figure 4 shows the protocol stack of SMT-based heterogeneous systems.

FIGURE 4

**Protocol stack of SMT-based heterogeneous systems**


Media coding layer
Presentation engine
Signalling function area
1. Media transmission signalling
2. Media consumption signalling
Application layer protocol (SMT)
Transport layer protocol (TCP,UDP)
Network layer protocol (IP)
Encapsulation function area
1. Based on ISO BMFF
2. Content fragmentation
Transmission function area
1. Encapsulation format specified for different media types
2. Error correction protection mechanism
3. Heterogeneous buffer model
BT.2074-04

In these systems, any multimedia data, such as video, audio, or web pages, to be used for building a multimedia presentation are encapsulated into Common Encapsulation Units (CEUs). Media data in the CEU is extracted and put on an SMTP payload by the SMT sending entity at transport time.

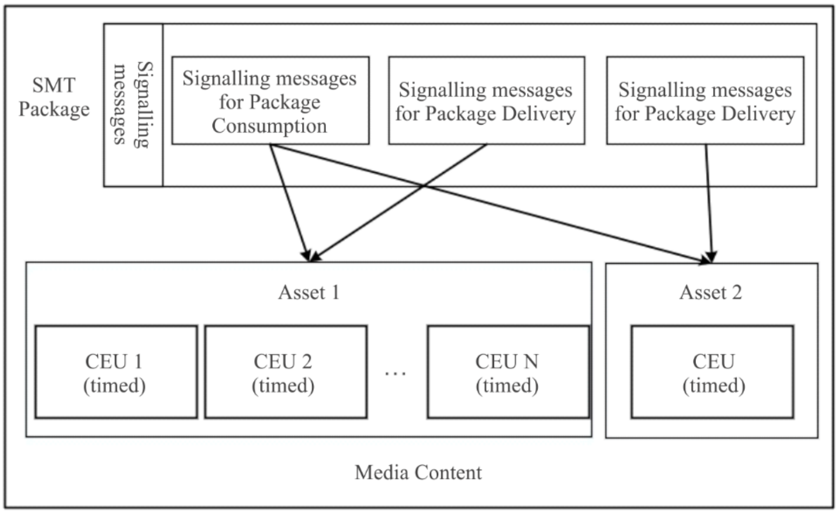
SMT logical packages can be serialized as SMT files to support media file-style storage, transmission and download; they can also be packaged as SMT transport packages to support streaming of media. Due to the high correlation in content between file format and transport packet format, SMT supports easy conversion of both for relay service.

As shown in Fig. 5, a Package is a logical entity. The SMT package is a logical entity and can be considered as a service, which mainly consists of signalling description file and media content. The signalling file includes forward signalling and feedback signalling, which can be divided into two types: signalling messages for package consumption and signalling messages for package delivery.

The signalling messages for consumption mainly contains the description information of the service, such as the composition, storage location, type and presentation policy of the media content; the signalling messages for transmission mainly contains the management information in the transmission process, such as QoS parameters and buffer setting information, etc. Media content can also be divided into two types: one is timed media, such as video and audio content; the other is non-timed media, such as text and picture information. In order to support the effective transmission of media content under heterogeneous network conditions and the dynamic configuration of content during transmission, a generic encapsulation unit for SMT media content is designed, which shall be fragmented, self-contained and unified, thus supporting the needs of dynamic organization of content and dynamic adaptation of transmission.

FIGURE 5

**Data model**


SMT Package
Signalling messages
Signalling messages for Package Consumption
Signalling messages for Package Delivery
Signalling messages for Package Delivery
Asset 1
CEU 1 (timed)
CEU 2 (timed)
…
CEU N (timed)
Asset 2
CEU (timed)
Media Content

BT.2074-05

## 3 Encapsulation of SMT

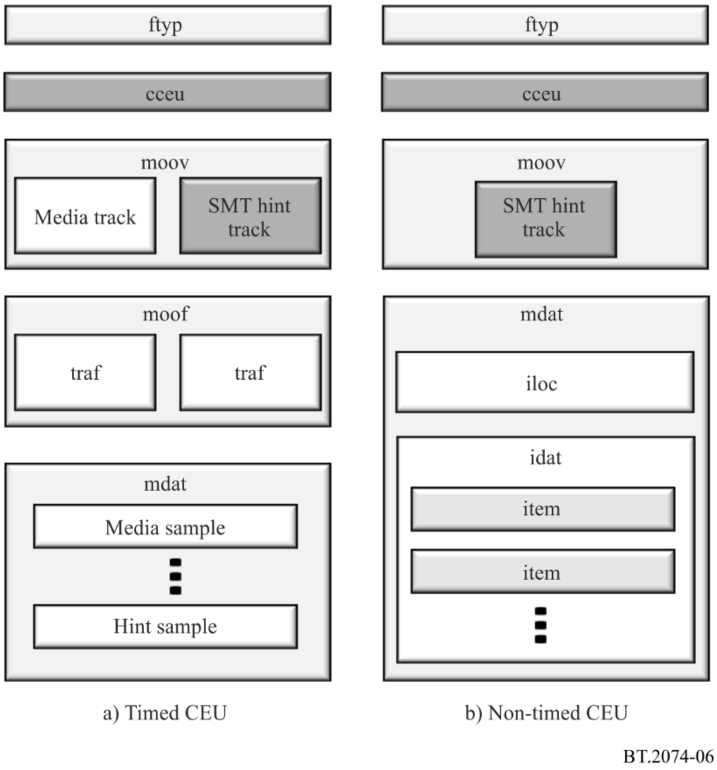
### 3.1 General

A Common Encapsulation Unit (CEU) is a conformant ISO BMFF file generated according to rules in § 3.2. Asset ID, the sequence number of the CEU and related information are provided in the 'cceu' box to uniquely identify the media data encapsulated in the CEU file. The 'moov' box contains all codec configuration information for decoding and presentation of the media data.

Timed media data are stored as track of the ISO BMFF (a single media tracks is allowed in CEU). Non-timed media are stored as part of the metadata in an ISO BMFF. Figure 6 depicts two examples of SMT encapsulation, one for timed and the other for non-timed media. For packetized delivery of CEU, an SMT hint track provides the information to convert encapsulated CEUs to SMTP payloads and SMTP packets.

FIGURE 6

**CEU package structure**



### 3.2 CEU brand definition

The brand 'ceuf' (CEU file) defined in this part identifies files that conform to the encapsulation rules for CEU. The 'ceuf' brand requires the support of the 'isom' brand. Support to the other brands such as 'dash’ brand (defined in ISO/IEC 23009-1) can also be indicated separately.

A CEU file is composed of a set of metadata boxes that enables the CEU to be self-contained. A CEU file shall contain an 'ftyp', 'cceu', 'moov' box, and shall optionally contain a 'sidx' box, all of which are part of the CEU metadata. Other boxes are allowed but will be ignored if the parser does not recognize them.

The 'moov' box shall contain at most one media track and shall contain SMT hint tracks that identify the smallest media fragment units in the transport format. The tracks in the 'moov' box shall contain no samples to help ensure small overhead (i.e. the entry_count in the 'stts', 'stsc' and 'stco' boxes shall be set to 0). The 'mvex' box shall be contained in the 'moov' box for the file storing a CEU with timed media data to indicate that the movie fragment structure is used. The 'mvex' box also sets the default values for the tracks and samples of the following 'moof' boxes.

Additionally, a 'cceu' box shall occur at the file level and the following rules shall be applied, including orders of the boxes.

a) If present, the 'cceu' box shall be placed right after the 'ftyp' box.

b) For timed media data, zero or more 'sidx' boxes shall be present in the file. If present, they shall index the 'moof' boxes that build the current CEU.

In addition to the box orders, the following restrictions shall also be applied to the 'ceuf' brand.

a) The maximum number of independent (e.g., an empty 'tref' box) media tracks in this file shall be one. Additionally, tracks with non-empty 'tref' box (e.g., hint tracks) shall be available.

b) For timed media data, the file shall have at least one 'moof' box.

c) For non-timed media data, one 'meta' box shall be present at the file level and shall contain non-timed media items of the CEU.

d) If present, an Edit List Box ('elst') shall only provide an initial offset.

e) Runs of sample data shall be placed in the 'mdat' box, in decoding order and without any other data between them.

f) Any sample auxiliary data, as described by 'saio' and 'saiz', shall be placed at the beginning of the 'mdat' boxes, before any sample data.

g) Any hint data shall be placed after sample data in the 'mdat' (or in another 'mdat' placed after the sample data) so that the sample offsets are not changed before and after transmission.

The 'tfdt' box shall be present inside the 'traf' box of each 'moof' box to provide the decode time of the first sample of the movie fragment in decoding order.

If any 'elst' box is available, the indicated offset shall be applied to the composition time of the first sample in presentation order of the CEU in addition to the presentation time provided by any presentation information.

Timed media data are stored as a track of the ISO BMFF and indexed by the 'moov' and 'moof' boxes in the fully backward-compatible manner. An SMT hint track guides the SMT sending entity in converting the encapsulated CEUs into a packetized media stream to be delivered using a transport protocol such as the SMT protocol.

Non-timed media data are stored as metadata items that are described by a 'meta' box. The 'meta' box shall appear at the file level. Each file of the non-timed media data shall be stored as a separate item in the 'meta' box. The entry point to the non-timed media shall be marked as the primary item of the 'meta' box (see ISO/IEC 14496-12).

### 3.3 CEU box

The Common Encapsulation Unit ('cceu') box contains the Asset identifier of the Asset to which the current CEU belongs and other attribute information for the current CEU. The Asset identifier is used to uniquely identify the Asset globally The CEU information includes the sequence number of the CEU in the Asset and the related attribute information.

The media processing unit ('mmpu') box used in MMT additionally contains “is_adc_present” flag to indicate whether it is required to store the asset delivery characteristics (ADC) together with MPU. SMT uses Asset Delivery Characteristic (ADC) signalling message to describe the QoS requirements and statistics of Asset for delivery and their associated QoE quality information.

### 3.4 SMT hint track

For transportation purposes, an SMT hint track provides an SMT sending entity with hints for the fragmentation of a CEU into the smallest fragment units of the media in transport format. The smallest fragment unit is media-aware and is used to build the SMTP payload, i.e. media data in the CEU is extracted and put on an SMTP payload by the SMT sending entity at transport time.

The SMT hint track also provides hints for extracting and reconstructing CEUs from SMTP payloads, which shall contain either CEU metadata, fragment metadata, or one or more smallest fragment units in the transport format. CEU metadata shall contain '`ftyp`', '`sidx`', '`cceu`', and '`moov` ' boxes.

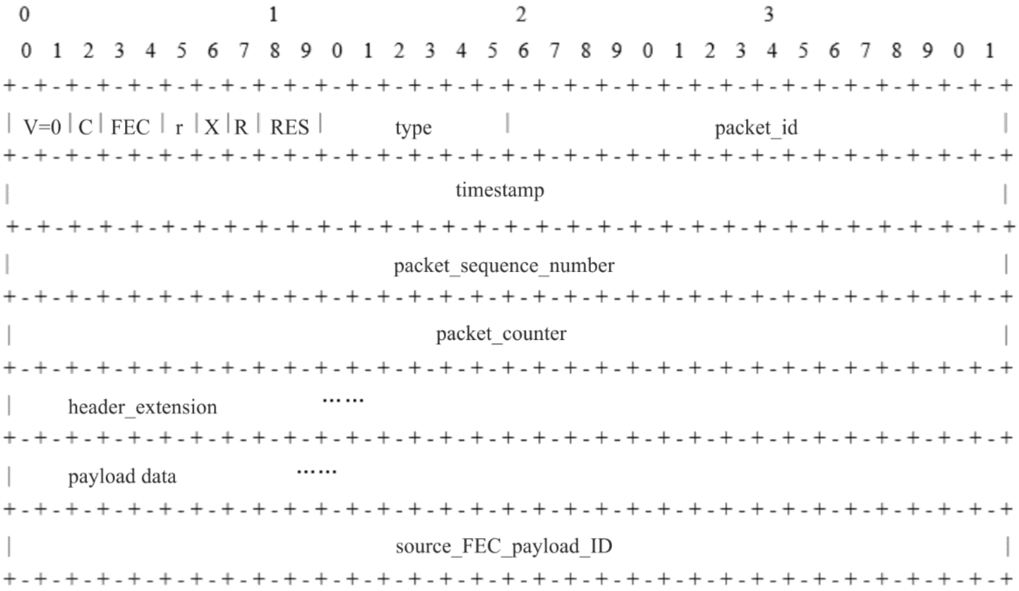
In SMT Hint Sample Entry, “is_fragment” flag is used to indicate whether the CEUs are fragmented into MFUs or not. While in MMT Hint Sample Entry, “has_mfus_flag” is used to indicate whether the MPUs are fragmented into MFUs or not.

### 3.5 SMT package structure

Figure 7 illustrates the structure of the SMTP packet at V = 0. It is the same as MMTP packet at V = 0.

FIGURE 7

**Package structure (V = 0)**



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| V=0 |C| FEC | r |X|R| RES |      type     |       packet_id       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            timestamp                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     packet_sequence_number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         packet_counter                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   header_extension          ……
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   payload data           ……
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      source_FEC_payload_ID                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

BT.2074-07

# Annex 2

## Media transport protocol and signalling information

### 1 Media transport protocol

### 1.1 Introduction

MMT-based broadcasting systems use the syntax and semantics of the MMTP payload and MMTP packet specified in ISO/IEC 23008-1. The extensions described below are intended for broadcast applications.

### 1.2 Header extension of MMTP packets

ISO/IEC 23008-1 specifies a header extension in the MMTP packet. The header extension has three fields: extension_type, extension_length, and header_extension_value. Although the header extension can be used for various purposes, it contains only one piece of information. The multi-type header extension described below enables it to contain multiple pieces of information.

**header_extension_value** – When the extension_type field is set to 0x0000, this field has the structure shown in Table 1.

TABLE 1

**Structure of multi-type header extension**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| Header_extension_value { | | |
|     for (i=0; i<N; i++) { | | |
|         **hdr_ext_end_flag** | 1 | bslbf |
|         **hdr_ext_type** | 15 | uimsbf |
|         **hdr_ext_length** | 16 | uimsbf |
|         for (j=0; j<M; j++) { | | |
|             **hdr_ext_byte** | 8 | bslbf |
|         } | | |
|     } | | |
| } | | |

**hdr_ext_end_flag** – When this flag is set to “1”, this multi-type header extension is the end of the header extension. When this flag is set to “0”, this multi-type header extension is not the end of the header extension.

**hdr_ext_type** – This field specifies the type of multi-type header extension.

**hdr_ext_length** – This field specifies the number of bytes of the following hdr_ext_byte field.

**hdr_ext_byte** – This field provides information on multi-type header extension.

## 2 Encapsulation of multimedia data

### 2.1 Introduction

In order to improve the interoperability of MMT-based broadcasting systems, the following constraints apply to carriage of multimedia data in MMTP packets.

### 2.2 Encapsulation of video data

### 2.2.1 MFU format for HEVC stream

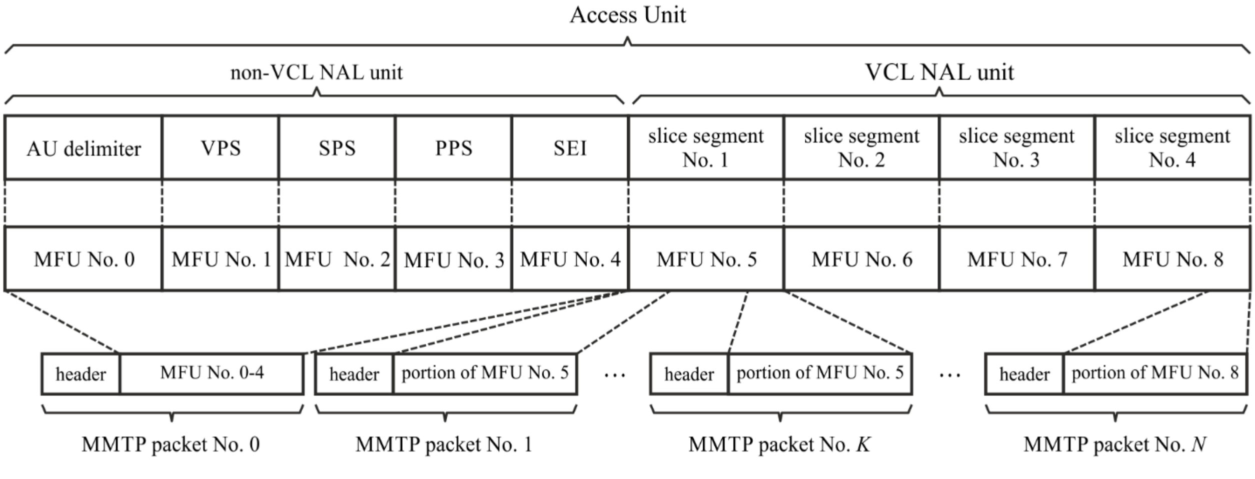
When a high efficiency video coding (HEVC) stream is carried in the MMT protocol, input to the MMT process is a sequence of network abstraction layer (NAL) units. A NAL unit is encapsulated into an MFU when an HEVC stream is carried in the MMT protocol.

If an HEVC encoder generates the byte stream format specified in Recommendation ITU-T H.265 | ISO/IEC 23008-2 Annex B, one start code prefix (0x000001) followed by one NAL unit is replaced with 32-bit length information of the NAL unit (unsigned integer format). Namely, the NAL unit together with the length information are encapsulated into one MFU.

Figure 8 shows an overview of generating MMTP packets and MFUs from a sequence of NAL units output from an HEVC encoder.

FIGURE 8

**Overview of packetization of NAL units of HEVC streams**



BT.2074-08

The duration of video MPU greatly influences the channel change time at the receiver terminal, since the video stream is decoded and presented at the receiver terminal on a per MPU basis. In order to reduce the channel change time, the MPU of an HEVC stream is constructed in intra random access point (IRAP) intervals.

### 2.2.2 Encapsulation of HEVC bitstream subsets

HEVC supports temporal sub-layer coding. One example is that when 120-Hz[1] video is encoded, two streams can be generated: one is a sub-bitstream for 60-Hz[2] video; the other is a bitstream subset for 120-Hz video. At the receiver terminal, 60-Hz video can be decoded from the sub-bitstream, and 120-Hz video can be decoded from both the sub-bitstream and the bitstream subset. The same process can be used for 100-Hz video.

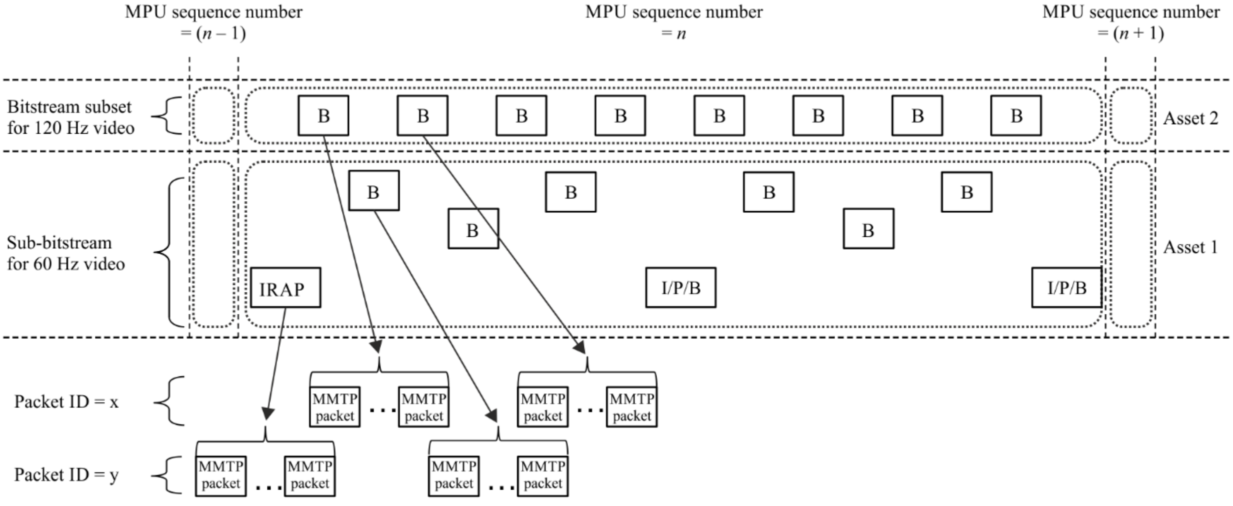
Figure 9 shows an overview of encapsulation of HEVC bitstream subsets. Note that this figure shows display order frame sequence. When an MMT package is made up of various media components, the sub-bitstream and the bitstream subset are encapsulated into separate Assets. In Fig. 9, the sub-bitstream is encapsulated into Asset 1 and the bitstream subset is encapsulated into Asset 2. Since they are separate Assets, the access units of Asset 1 and Asset 2 are carried in MMTP packets that have different packet IDs.

[1] Also includes 120/1.001 Hz.

[2] Also includes 60/1.001 Hz.

FIGURE 9

**Overview of encapsulation of HEVC sub-bitstream and bitstream subset for temporal sub-layer coding**


MPU sequence number = (n – 1)
MPU sequence number = n
MPU sequence number = (n + 1)
Bitstream subset for 120 Hz video
Asset 2
Sub-bitstream for 60 Hz video
Asset 1
B
IRAP
I/P/B
Packet ID = x
Packet ID = y
MMTP packet

BT.2074-09

The sequence number of an MPU that the access units of the bitstream subset belong to is identical to the sequence number of an MPU that the access units of the sub-bitstream belong to in the same time period. Assigning the same sequence number to both MPUs enables receiver terminals to easily identify the MPUs that include the corresponding access units in the same GOP.

In the example shown in Fig. 9, the decoding of Asset 2 depends on Asset 1. A Dependency Descriptor stating that Asset 2 depends on Asset 1 is inserted in the asset_descriptors_byte field of the MP Table. In addition to the Dependency Descriptor, an MPU Timestamp Descriptor and MPU Extended Timestamp Descriptor are inserted in the asset_descriptors_byte fields of both Asset 1 and Asset 2.

### 2.3 Encapsulation of audio data

### 2.3.1 MFU format for MPEG-4 AAC and MPEG-4 ALS

When an MPEG-4 advanced audio coding (AAC) stream or MPEG-4 audio lossless coding (ALS) stream is carried in the MMT protocol, input to the MMT process is in the form of either a LATM/LOAS stream or a data stream.

The low overhead audio transport multiplex (LATM) includes an audio channel configuration and provides multiplexing functions for audio data. The low overhead audio stream (LOAS) provides synchronization for audio data. When an audio encoder generates a LATM/LOAS stream, one AudioMuxElement () specified in ISO/IEC 14496-3 is encapsulated into one MFU.

When an audio encoder generates a data stream, a Raw Data Stream is encapsulated into one MFU.

## 3 Signalling information

### 3.1 Introduction

There are three kinds of MMT-signalling information: Message, Table, and Descriptor. Some of the signalling information specified in ISO/IEC 23008-1 is not used in broadcasting systems. This section summarizes the signalling information for broadcasting systems.

## 3.2 MMT signalling information Messages

### 3.2.1 List of MMT-signalling information Messages

Table 2 shows the list of messages.

TABLE 2

**List of messages**

| Message name | Message_id assignment | Description | Specified in ISO/IEC 23008-1 | Use in MMT-based broadcastin g systems | Use in SMT-based systems |
|---|---|---|---|---|---|
| PA message | 0x0000 | Is the entry point of MMT-signalling information. Conveys one or more tables | X | X | X |
| Media presentation information (MPI) message | 0x0001 – 0x000F | Conveys a presentation information document | X | | |
| MPT message | 0x0010 – 0x001F | Conveys a whole or a subset of an MP table | X | | X |
| Clock relation information (CRI) message | 0x0200 | Conveys clock related information to be used for mapping between the NTP timestamp and MPEG-2 STC | X | | X |
| Device capability information (DCI) message | 0x0201 | Conveys information on required device capabilities for the package consumption | X | | X |
| Application layer-forward error correction (AL-FEC) message | 0x0202 | Conveys configuration information of an AL-FEC scheme to be used to protect Asset | X | | X |
| Hypothetical receiver buffer model (HRBM) message | 0x0203 | Conveys information on end-to-end transmission delay and memory requirement to a receiving terminal | X | | X |
| Asset delivery characteristics (ADC) message | 0x0209 | Conveys information on QoS requirements and statistics of Asset for delivery, and their associated QoE quality information | X | | X |

TABLE 2 (*end*)

| Message name | Message_id assignment | Description | Specified in ISO/IEC 23008-1 | Use in MMT-based broadcastin g systems | Use in SMT-based systems |
|---|---|---|---|---|---|
| M2section message | 0x8000 | Conveys the MPEG-2 Section-format Table. Tables and descriptors in MPEG-2 TS based conventional broadcasting systems can be reused by this message | | X | |
| Resource Request/Response message | 0xE000 | Conveys information of the request and response format between SMT server and client | | | X |
| Interaction feedback message | 0xE001 | Conveys interaction feedback between server and client during immersive media consumption | | | X |
| Session control message | 0xE002 | Conveys information on the function of session start, stop and jump | | | X |
| Synchronization request message | 0xE003 | Conveys information on current network delay and available bandwidth for synchronization control | | | X |
| Synchronization response message | 0xE004 | Conveys information to inform the user of the playback time | | | X |

### 3.2.2 Detailed specifications of messages

#### 3.2.2.1 PA message

The syntax and semantics of the PA message are specified in ISO/IEC 23008-1.

#### 3.2.2.2 MPT message

The syntax and semantics of the MPT message are specified in ISO/IEC 23008-1.

#### 3.2.2.3 Clock relation information (CRI) message

The syntax and semantics of the CRI message are specified in ISO/IEC 23008-1.

#### 3.2.2.4 Device capability information (DCI) message

The syntax and semantics of the DCI message are specified in ISO/IEC 23008-1.

#### 3.2.2.5 Application layer-forward error correction (AL-FEC) message

The syntax and semantics of the adaptive FEC signalling message are specified in ISO/IEC 23008-1. SMT applies "Adaptive FEC coding structure" which is one of the AL-FEC coding structures defined in ISO/IEC 23008-1 and "RaptorQ AD code" which is one of the FEC code algorithms defined in MMT FEC Codes ISO/IEC 23008-10.

#### 3.2.2.6 Hypothetical receiver buffer model (HRBM) message

The syntax and semantics of the HRBM message are specified in ISO/IEC 23008-1.

#### 3.2.2.7 Asset delivery characteristics (ADC) message

The syntax and semantics of the ADC message are specified in ISO/IEC 23008-1.

#### 3.2.2.8 M2section message

Table 3 shows the syntax of the M2section message.

TABLE 3

**Syntax of M2section message**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| M2section_Message () { | | |
| **message_id** | 16 | uimsbf |
| **version** | 8 | uimsbf |
| **length** | 16 | uimsbf |
| **table_id** | 8 | uimsbf |
| **section_syntax_indicator** | 1 | bslbf |
| '1' | 1 | bslbf |
| '11' | 2 | bslbf |
| **section_length** | 12 | uimsbf |
| **table_id_extension** | 16 | uimsbf |
| '11' | 2 | bslbf |
| **version_number** | 5 | uimsbf |
| **current_next_indicator** | 1 | bslbf |
| **section_number** | 8 | uimsbf |
| **last_section_number** | 8 | uimsbf |

TABLE 3 (*end*)

| Syntax | No. of bits | Mnemonic |
|---|---|---|
|     for (i=0; i<N; i++) { | | |
|         signalling_data_byte | 8 | bslbf |
|     } | | |
|     **CRC_32** | 32 | rpchof |
| } | | |

The semantics of each field of the M2section message are as follows:

**table_id** – This field identifies the table to which the section belongs.

**section_syntax_indicator** – This field determines whether a normal or extension format is used. This field is always set to “1” to indicate the extension format.

**section_length** – This field identifies the number of data bytes following this field.

**table_id_extension** – This is a field extending the table identifier.

**version_number** – This field contains the table version number.

**current_next_indicator** – This field contains “1” when the table is currently used and “0” when the table cannot be used at present, but can be used next.

**section_number** – This field contains the number of the first section comprising the table.

**last_section_number** – This field contains the number of the last section comprising the table.

**CRC_32** – This field complies with Recommendation ITU-T.

### 3.2.2.9 Resource Request/Response Message

The Resource Request/Response Message (3R_message) provides the request and response format between the SMT server and the client. This message is used when the client and the server need to interact with request and response information for a session.

The syntax of the 3R_message is defined in Table 4.

TABLE 4

**Syntax of 3R_message**

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
| 3R_message(){ | | | |
|     **message_id** | | 16 | uimsbf |
|     **version** | | 8 | uimsbf |
|     **length** | | 16 | uimsbf |
|     message_payload { | | | |
|         reserved | ‘1111111’ | 7 | bslbf |
|         **method** | | 1 | bslbf |
|         else if (method == REQUEST) { | | | |
|             **request_serial_number** | | 8 | uimsbf |
|             **mime_type()** | | | |
|             **data_length** | N1 | 16 | uimsbf |
|             for (j = 0; j < N1; j++) { | | | |
|                 **data_byte** | | 8 | uimsbf |
|             } | | | |
|         } | | | |
|         else if (method == RESPONSE) { | | | |
|             **response_serial_number** | | 8 | uimsbf |
|             **status_number** | | 8 | uimsbf |
|             if (status_number == 0x02) { | | | |
|                 **mime_type()** | | | |
|                 **data_length** | N2 | 16 | uimsbf |
|                 for (j = 0; j < N2; j++) { | | | |
|                   **data _byte** | | 8 | uimsbf |
|                 } | | | |
|       } | | | |
|     } | | | |
|    } | | | |
| } | | | |

**message_id** – The message identifier of the real-time interaction message, which identifies this message is used to request or respond to the interaction data.

**version** – The version of the resource request/response message.

**length** –The length of the resource request/response message.

**method** – Two bits of the original reserved field are used to mark whether the current user is making a request or response, see Table 5 for the value and description.

TABLE 5

**Value of method and description**

| Value | Description |
|---|---|
| 0 | “REQUEST” |
| 1 | “RESPONSE” |

**request_serial_number** – This field indicates the serial number of the request sent by the client. It is incremented by mod 256 when the message is sent.

**mime_type** – This field identifies the generalized MIME type to instruct the server or client to parse the data in the appropriate file format.

**data_length** – This field identifies the length of the payload.

**data_byte** – This field identifies the number of bytes in the payload.

**response_serial_number** – This field indicates the serial number of the response message sent by the server. It is incremented by mod 256 when the message is sent. When this value is equal to the request_serial_number value, the request and response messages match each other.

**status_number** – This field describes the status returned by the server. See Table 6 for its value and description.

TABLE 6

**Values and description of status_number**

| Value | Description |
|---|---|
| 0x00 | Request failed, the requested resource was not found on the server, or the upload of data failed |
| 0x01 | The request was successful |
| 0x02 | The request was successful and the response header or data body desired by the request will be returned with this response |
| 0x03~0x7F | Reserved for ISO |
| 0x80~0xFF | Reserved for private use |

### 3.2.2.10 Interaction feedback message

Interaction feedback message provides interaction feedback between the server and the client during immersive media consumption.

The interaction feedback message consists of three parts: interaction target, interaction type, and interaction content. As the user's interaction behaviour changes, the encapsulated content changes dynamically, so no specific interaction content is specified in this section.

The syntax of the interaction feedback messages is defined in Table 7.

TABLE 7

**Syntax of interaction feedback messages**

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
| Interaction_feedback_message() { | | | |
|     **message_id** | | 16 | uimsbf |
|     **version** | | 8 | uimsbf |
|     **length** | | 32 | uimsbf |
|     message_payload { | | | |
|         **message_source** | | 1 | bool |
|         **reserved** | | 7 | uimsbf |
|         **asset_id()** | | | |
|         **interaction_num** | N | 8 | uimsbf |
|         for (i=0; i<N; i++) { | | | |
|             **timestamp** | | 32 | uimsbf |
|             **interaction_target** | | 8 | uimsbf |
|             **interaction_type** | | 8 | uimsbf |
|             interaction_content{ | | | |
|                 **interaction_content_length** | | 32 | uimsbf |
|             } | | | |
|         } | | | |
|     } | | | |
| } | | | |

**message_id** – This field identifies the ID of the interaction feedback message.

**version** – This field identifies the version of the interaction feedback message. The information carried with the new version shall overwrite information carried with any previous old version.

**length** – This field identifies the length of the interaction feedback message in bytes, i.e. the length from the next field until the last byte of the interaction feedback message. A '0' value is not valid in this field.

**message_source** – This field indicates the source of the message, value 0 means the interaction feedback message is sent from the client to the server, value 1 means the interaction feedback message is sent from the server to the client.

**asset_id** – This field identifies the asset_id of the media content currently consumed by the client.

**interaction_num** – This field indicates the number of interactions contained in the current message.

**timestamp** – This field indicates the time when the current interaction is generated, using UTC time.

**interaction_target** – This field indicates the target of the current interaction of the client, including the current state of the helmet device, the user’s current region of interest, the current state of the user, etc.; see Table 8 for the values of the interaction target.

TABLE 8

**Interaction target values**

| Type | Value | Descriptor |
|---|---|---|
| Null | 0 | The interaction target is empty, i.e. there is no specific interaction target |
| HMD_status | 1 | The interaction target is the current state of the helmet device |
| Object of interests | 2 | The interaction target is the user's current region of interest |
| User_status | 3 | The interaction target is the current state of the user |

**interaction_type** – This field indicates the type of interaction that the client performs the current interaction_target, the values of this field are related to the interaction_target, including tracking, gazing, touching, etc. The values of the interaction type are shown in Table 9.

TABLE 9

**Interaction type values**

| Type | Value | Description |
|---|---|---|
| Null | 0 | The interaction type is empty, i.e. there is no specific interaction type |
| Tracking | 1 | The interaction type is tracking |
| Gaze | 2 | The interaction type is gazing |
| Touch | 4 | The interaction type is touching |

**interaction_content_length** – This field indicates the length of the client's interaction content for the current interaction. This field is in bytes and is the length from the next byte in the field to the end of the current interaction.

### 3.2.2.11 Session control message

The SMT receiver can use SMT messages to establish and control the session, and the SMT receiver sends SC messages to the sender to control the media transmission. This message mainly provides the function of session start, stop and jump. The syntax of the session control message is defined in Table 10.

TABLE 10

**Syntax of the session control message**

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
| SC_message () { | | | |
| **message_id** | | 16 | uimsbf |
| **version** | | 8 | uimsbf |
| **length** | | 16 | uimsbf |
| message_payload { | | | |
| **command_code** | | 32 | uimsbf |
| if (command_code == 0x01){ | | | |
| **start_time** | | 32 | uimsbf |
| } | | | |
| else if (command_code == 0x02){ | | | |
| } | | | |
| else if (command_code == 0x03){ | | 64 | |
| **current_presentation_time** | | 64 | uimsbf |
| **seek_time** | | 32 | simsbf |
| **progress_point** | | | uimsbf |
| } | | 8 | |
| **number_of_assets** | N | | uimsbf |
| for (i=0;i<N;++i) { | | 8 | |
| **packet_id** | | | uimsbf |
| } | | | |
| } | | | |
| } | | | |

**message_id** – This field identifies the message ID of the SC message. The length of this field is 16 bits.

**version** – This field identifies the version of the SC message.

**length** – This field identifies the length of the SC message. The length of this field is 16 bits.

**command_code** – This field identifies a session control operation, whose value and corresponding description are shown in Table 11.

TABLE 11

**Values of command_code**

| Value | Description |
|---|---|
| 0x01 | Play |
| 0x02 | Pause |
| 0x03 | Skip |
| 0x04~0xFF | Reserve |

**start_time** – This field identifies the start time of the presentation. The value of this field is set by the request time of the SMT receiving entity. When this field is received by the SMT sending entity, the CEU whose presentation time is closest to the time indicated by this field is selected for sending. When the presentation times of both CEUs are equally close to the time indicated by this field, the CEU with the earlier presentation time is selected to send and a timeline is generated based on start_time and the duration of the CEU. start_time is based on the NTP time. In on-demand applications, start_time is used as the baseline for updating the CEU presentation time.

**current_presentation_time** – This field identifies the current presentation time. It is based on NTP time.

**seek_time** – This field identifies the time from the current time to the skip target time. The length of this field is 64 bits, and its value can be positive or negative. A positive value indicates a forward skip, and a negative value indicates a backward skip.

**progress_point** – This field identifies the percentage of the period from the start time to the current playback time in the whole presentation time. The unit is a percentage.

**number_of_asset** – This field identifies the number of assets controlled by this message.

**packet_id** – This field the packet_id in SMTP.

### 3.2.2.12 Synchronization request message

The client needs to know the current network latency and available bandwidth to calculate the fixed end-to-end latency for synchronization control. Table 12 defines the syntax of the sync_request_message.

TABLE 12

**Syntax of the sync_request message**

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
| Sync_request_message() { | | | |
| **message_id** | | 16 | uimsbf |
| **version** | | 8 | uimsbf |
| **length** | | 16 | uimsbf |
| message_payload (){ | | | |
| **network_delay** | | 16 | uimsbf |
| **network_bandwidth** | | 32 | uimsbf |
| } | | | |
| } | | | |

**message_id** – This field identifies SMT signalling message, used to distinguish different kinds of signalling. message_id corresponds to the signalling message one by one, and its mapping relationship is specified in the SMT protocol.

**length** – This field identifies the length of the user’s private location information in bytes.

**version** – This field identifies the version of the signalling message.

**message_payload** – This field specifies the payload of the signalling message, which means network_delay and network_bandwidth here.

**network_delay** – This field specifies current network delay.

**network_bandwidth** – This field specifies the current available bandwidth information.

$$t_{tmp} = t_0 + CEU\_size/B_b + \Delta t \qquad (1)$$

The synchronization request message contains network_delay and available network_bandwidth. In the above equation, CEU_size is the size of the average CEU sent, $B_b$ is the available bandwidth of the broadband network (network_bandwidth), $\Delta t$ is the broadband network delay (network_delay) in the downstream direction, and $t_{tmp}$ is the calculated moment when the first CEU is received at the client. CEU_size/$B_b$ + $\Delta t$ is the fixed end-to-end delay.

### 3.2.2.13 Synchronization response message

While sending the Assets, the server needs to send a message to inform the client of the serial number of the first independent decomposable Asset, so as to inform the user of the playback time. The synchronization response message is as follows.

Table 13 defines the syntax of the sync_response_message.

TABLE 13

**Syntax of the sync_response message**

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
| Sync_response_message () { | | | |
| **message_id** | | 16 | uimsbf |
| **version** | | 8 | uimsbf |
| **length** | | 16 | uimsbf |
| message_payload (){ | | | |
| **number_of_assets** | N | 16 | uimsbf |
| for (i=0; i<N; i++){ | | | |
| **asset_id** | | 16 | uimsbf |
| **CEU_sequence_number** | | 32 | uimsbf |
| } | | | |
| } | | | |
| } | | | |

**message_id** – This field identifies SMT signalling message, used to distinguish different kinds of messages. message_id corresponds to the signalling message one by one, and its mapping relationship is specified in the SMT protocol.

**length** – This field identifies the length of the user's private location information in bytes.

**version** – This field identifies the version of the message.

**message_payload** – This field identifies the payload of the signalling message, which means CEU_sequence_number here.

**number_of_assets** – This field indicates the number of Assets.

**asset_id** – This field indicates the identification of each Asset.

**CEU_sequence_number** – This field indicates the sequence number of the first CEU sent by the server to notify the client of the playback time.

The client is informed of the time information and the average CEU size based on the sequence number of the first asset in the message, caches the current CEU, waits until the timestamps are aligned, and then plays the media to realize the synchronization of broadband and broadcast channels.

## 3.3 MMT signalling information Tables

### 3.3.1 List of MMT-signalling information Tables

Table 14 shows the list of Tables.

TABLE 14

**List of Tables**

| Table name | Table_id assignment | Description | Specified in ISO/IEC 23008-1 | Use in MMT-based broadcasting systems | Use in SMT-based systems |
|---|---|---|---|---|---|
| PA table | 0x00 | Provides information on all other signalling tables. | X | | X |
| MPI table | 0x01 – 0x0F | Provides a presentation information document. | X | | |
| MP table | 0x20 | Provides configuration information on the MMT package, such as lists and locations of Assets. | X | X | X |
| CRI table | 0x21 | Provides a CRI descriptor. | X | | X |
| DCI table | 0x22 | Provides information on the required device capabilities for consumption of the package. | X | | X |
| Package list table | 0x80 | Provides the IP data flow and packet id of the PA message for the MMT package as a broadcasting service. Also provides a list of IP data flows of other IP services. | | X | |
| Block association table | 0xE0 | Provides information on the relationship between the original video Asset and the block video Asset. | | | X |
| Layer display table | 0xE1 | Provides the details for each layer in the presentation, indicates the basic layout of the display. | | | X |
| Layer display update table | 0xE2 | Provides the presentation layer information that needs to be updated for the presentation. | | | X |

### 3.3.2 Detailed specifications of tables

#### 3.3.2.1 MMT Package Access (PA) table

The syntax and semantics of the PA table are specified in ISO/IEC 23008-1.

#### 3.3.2.2 MMT package (MP) table

The syntax and semantics of the MMT package table are specified in ISO/IEC 23008-1.

#### 3.3.2.3 CRI table

The syntax and semantics of the CRI table are specified in ISO/IEC 23008-1.

#### 3.3.2.4 DCI table

The syntax and semantics of the DCI table are specified in ISO/IEC 23008-1.

#### 3.3.2.5 Package list table

Table 15 shows the syntax of the package list table.

TABLE 15

**Syntax of package list table**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| Package_List_Table () { | | |
|     **table_id** | 8 | uimsbf |
|     **version** | 8 | uimsbf |
|     **length** | 16 | uimsbf |
|     **num_of_package** | 8 | uimsbf |
|     for (i=0; i<N; i++) { | | |
|         **MMT_package_id_length** | 8 | uimsbf |
|         for (j=0; j<M; j++) { | | |
|             **MMT_package_id_byte** | 8 | bslbf |
|         } | | |
|         MMT_general_location_info () | | |
|     } | | |
|     **num_of_ip_delivery** | 8 | uimsbf |
|     for (i=0; i<N; i++) { | | |
|         **transport_file_id** | 32 | uimsbf |
|         **location_type** | 8 | uimsbf |
|         if (location_type == 0x01) { | | |
|             **ipv4_src_addr** | 32 | uimsbf |
|             **ipv4_dst_addr** | 32 | uimsbf |
|             **dst_port** | 16 | uimsbf |
|         } | | |
|         if (location_type == 0x02) { | | |
|             **ipv6_src_addr** | 128 | uimsbf |

TABLE 15 (*end*)

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| **ipv6_dst_addr** | 128 | uimsbf |
| **dst_port** | 16 | uimsbf |
| } | | |
| if (location_type == 0x05) { | | |
| **URL_length** | 8 | uimsbf |
| for (j=0; j<M; j++) { | | |
| **URL_byte** | 8 | char |
| } | | |
| } | | |
| **descriptor_loop_length** | 16 | uimsbf |
| for (j=0; j<M; j++) { | | |
| **descriptor ()** | | |
| } | | |
| } | | |
| } | | |

The semantics of each field of the package list table are as follows:

**num_of_package** – This field identifies the number of Packages whose locations are described in this Table.

**MMT_package_id_length** – This field specifies the number of bytes of the following MMT_package_id_byte field.

**MMT_package_id_byte** – This field identifies the MMT package ID.

**MMT_general_location_info** – This field indicates the location information carrying the PA Message of the identified MMT package.

**num_of_ip_delivery** – This field specifies the number of IP flows whose locations are described in this Table.

**transport_file_id** – This field specifies the identification of a file object.

**location_type** – This field specifies the type of location information. When this field is set to 0x01, the location is an IPv4 data flow. When this field is set to 0x02, the location is an IPv6 data flow. When this field is set to 0x05, the location is a URL.

**ipv4_src_addr** – This field specifies an IPv4 source address. The IPv4 address is fragmented into four fields of 8 bits, where the first byte of this field contains the most significant byte of the IPv4 source address.

**ipv4_dst_addr** – This field specifies an IPv4 destination address. The IPv4 address is fragmented into four fields of 8 bits, where the first byte of this field contains the most significant byte of the IPv4 destination address.

**dst_port** – This field specifies the destination port number of an IP data flow.

**ipv6_src_addr** – This field specifies an IPv6 source address. The IPv6 address is fragmented into eight fields of 16 bits, where the first byte of this field contains the most significant byte of the IPv6 source address.

**ipv6_dst_addr** – This field specifies an IPv6 destination address. The IPv6 address is fragmented into eight fields of 16 bits, where the first byte of this field contains the most significant byte of the IPv6 destination address.

**URL_length** – This field specifies the number of bytes of the following URL_byte field.

**URL_byte** – This field specifies the URL.

**descriptor_loop_length** – This field represents the number of bytes in all descriptors immediately after this field.

### 3.3.2.6 Block association table

Table 16 shows the syntax of the block association table.

TABLE 16

**Syntax of the block association table**

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
| Block_association_table () { | | | |
|     **table_id** | | 8 | uimsbf |
|     **version** | | 8 | uimsbf |
|     **length** | | 32 | uimsbf |
|     table_payload { | | | |
|         **partitioned_asset_number** | N1 | 8 | unimbf |
|         for (i=0; i<N1; i++) { | | | |
|             **asset_id()** | | | |
|             **original_height** | | 16 | uimsbf |
|             **original_width** | | 16 | uimsbf |
|             reserved | | 4 | |
|             **block_number** | N2 | 8 | |
|             for (j=0; j<N2; j++) { | | | |
|                 **block_height_top** | | 16 | uimsbf |
|                 **block_width_left** | | 16 | uimsbf |
|                 **block_height** | | 16 | uimsbf |
|                 **block_width** | | 16 | uimsbf |
|                 **asset_id()** | | | |
|             **}** | | | |
|         **}** | | | |
|     **}** | | | |
| **}** | | | |

**table_id** – This field specifies the identifier of the block video relationship table.

**version** – This field specifies the version of the block video relationship table. The information carried by the new version will overwrite any previous old version.

**length** – This field contains the length of the block video relationship table in bytes, i.e. the length from the next field until the last byte of the block video relationship table. The '0' value is not valid in this field.

**partitioned_asset_number** – This field specifies the number of the original video asset to be blocked.

**asset_id** – This field specifies the asset_id of the original video to be blocked.

**original_height** – This field specifies the height of the original video, in pixels.

**original_width** – This field specifies the width of the original video, in pixels.

**block_number** – This field specifies the number of the block video corresponding to the original video asset.

**block_height_top** – This field specifies the distance between the top edge of the block video CEU and the top edge of the original video CEU, in pixels.

**block_width_left** – This field specifies the distance of the left edge of the block video CEU relative to the left edge of the original video CEU, in pixels.

**block_height** – This field specifies the height of the block video CEU, in pixels.

**block_width** – This field specifies the width of the block video CEU, in pixels.

**asset_id** – This field specifies the asset_id of a block video.

### 3.3.2.7 Layer display table

Table 17 describes the details for each layer in the presentation, indicating the basic layout of the display.

TABLE 17

**Syntax of the layer display table**

| **Syntax** | **Value** | **No. of bits** | **Mnemoni c** |
|---|---|---|---|
| Layer_display_table () { | | | |
|     **table_id** | | 8 | uimsbf |
|     **version** | | 8 | uimsbf |
|     **length** | | 16 | unimbf |
|     **number_of_layer** | N1 | 8 | unimbf |
|     for (i = 0; i <N1;i++) { | | | |
|         **layer_id** | | 8 | unimbf |
|         **device_id** | | 8 | unimbf |
|         **center_x** | | 16 | unimbf |
|         **center_y** | | 16 | unimbf |
|         **width** | | 16 | unimbf |
|         **height** | | 16 | unimbf |
|         **display_order** | | 8 | unimbf |
|         **fitting_type** | | 3 | bslbf |
|         **adjust_enable_flag** | | 1 | bool |
|         **reserved** | ’1111’ | 4 | bslbf |
|         **transparency** | | 8 | unimbf |
|     } | | | |
| } | | | |

**table_id** – This field specifies the identifier of the table.

**version** – This field specifies the version of the table. The updated table has a new version number and can replace the original table.

**length** – This field specifies the length of the table, starting from the next byte of this field to the last byte of the table.

**number_of_layer** – This field specifies the number of presentation layers described by this table.

**layer_id** – This field specifies the label of the currently described presentation layer in this table.

**device_id** – This field specifies the device number corresponding to the presentation layer currently described in the table. A number of '0' means that it is presented on the default device, when it is 1 or other values, it is presented on the secondary device or secondary/low priority device, and the higher the number the lower the priority.

**center_x** – This field specifies the horizontal coordinate of the centre of the area where the media content is presented in the currently described presentation layer in this table. It is calibrated in terms of the percentage of pixels in the centre of the display area among the horizontal pixels of the entire layer.

**center_y** – This field specifies the vertical coordinate of the centre of the region in the currently described presentation layer within this table where the media content is presented. It is calibrated as a percentage of the pixels in the centre of the display area out of the vertical pixels of the entire layer.

**width** – This field specifies the width of the area in the presentation layer currently described in this table where the media content is presented. It is calibrated as a percentage of the pixels in the display area among the horizontal pixels of the entire layer.

**height** – This field specifies the height of the area in the presentation layer currently described in this table where the media content is presented. It is calibrated as a percentage of the pixels of the display area among the vertical pixels of the entire layer.

**display_order** – This field specifies the display order of the currently described presentation layer in this table among all presentation layers. This order of '0' indicates the default layer, with the smaller labeled layer at the bottom and the larger brand layer at the top. The middle number can be vacant, but cannot be repeated.

**fitting_type** – This field specifies the type of screen fitting when playing media content for the presentation layer currently described in this table. The fitting type is '0', which means that the CEU resolution aspect ratio is changed and the whole specified area is spread after stretching. The fitting type is '1', which means that the CEU resolution aspect ratio is not changed and the screen is enlarged from the minimum until the width \ height fits in one direction with the left and right / top and bottom sides of the screen, and the remaining part is filled in black. Fitting type '2' means zooming out, that is, without changing the aspect ratio of the CEU resolution, the screen is reduced from the largest size, until the width \ height fits in one direction with the left and right / top and bottom sides of the screen, and the rest is cropped. Fitting type '3' means original picture, i.e. no change in CEU resolution and aspect ratio, positioned in the centre of the specified area, and if it does not match the display area, the insufficient part is made up in black or the excess part is cropped. The fitting type is '4' for omnidirectional video, that is, according to the requirements of omnidirectional video playback, the omnidirectional video will be adapted to the display layer. The fitting type is not limited to the above five types. See Table 18.

TABLE 18

**Screen fitting types**

| **Value** | **Description** |
|---|---|
| 000 | Spread over |
| 001 | Zooming in |
| 010 | Zooming out |
| 011 | Original picture |
| 100 | Omnidirectional video |
| 101-111 | Reserved |

**adjust_enable_flag** – This field specifies a flag whether the presentation layer currently described in this table is adjustable. When this flag is '0', it means that the layer cannot be adjusted by the user side; when this flag is '1', it means that the user side can adjust the device, display area size, position, transparency, adaptation type, etc. for this layer.

**transparency** – This field specifies describes the degree of transparency of the currently described presentation layer within this table. The value of this field represents the value before the percent sign and is valid from 0 to 100%.

### 3.3.2.8 Layer display update table

Table 19 describes the presentation layer information that needs to be updated for the presentation, indicating the update information for the presentation layout. This table is used when there are minor adjustments to the layout. When there are extensive adjustments to the layout, a new version of the Layer_display_table can be resent to update the layout as a whole.

TABLE 19

**Syntax of layer display update table**

| **Syntax** | **Value** | **No. of bits** | **Mnemonic** |
|---|---|---|---|
| Layer_display_update_table () { | | | |
| table_id | | 8 | uimsbf |
| version | | 8 | uimsbf |
| length | | 16 | unimbf |
| **layer_delete_flag** | | 1 | bool |
| **layer_add_flag** | | 1 | bool |
| **layer_display_order_flag** | | 1 | bool |
| **layer_adjust_flag** | | 1 | bool |
| reserved1 | '1111’ | 4 | bslbf |
| if (layer_delete_flag) { | | | |
| number_of_layer | N1 | 8 | unimbf |
| for (i = 0; i <N1;i++) { | | | |
| layer_id | | 8 | unimbf |
| } | | | |
| } | | | |
| if (layer_add_flag){ | | | |
| number_of_layer | N2 | 8 | unimbf |

TABLE 19 (*end*)

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
|     for (i = 0; i <N2;i++) { | | | |
|       new_layer_id | | 8 | unimbf |
|       device_id | | 8 | unimbf |
|       center_x | | 16 | unimbf |
|       center_y | | 16 | unimbf |
|       width | | 16 | unimbf |
|       height | | 16 | unimbf |
|       display_order | | 8 | unimbf |
|       fitting_type | | 3 | bslbf |
|       adjust_enable_flag | | 1 | bool |
|       reserved2 | '1111’ | 4 | bslbf |
|       transparency | | 8 | unimbf |
|     } | | | |
|   } | | | |
|   if (layer_display_order_flag){ | | | |
|     number_of_layer | N3 | 8 | unimbf |
|     for (i = 0; i <N3;i++) { | | | |
|       layer_id | | 8 | unimbf |
|       new_layer_display_order | | 8 | unimbf |
|     } | | | |
|   } | | | |
|   if (layer_adjust_flag) { | | | |
|     number_of_layer | N4 | 8 | unimbf |
|     for (i = 0; i <N4;i++) { | | | |
|       layer_id | | 8 | unimbf |
|       device_id | | 8 | unimbf |
|       center_x | | 8 | unimbf |
|       center_y | | 8 | unimbf |
|       width | | 8 | unimbf |
|       height | | 8 | unimbf |
|       display_order | | 8 | unimbf |
|       fitting_type | | 3 | bslbf |
|       adjust_enable_flag | | 1 | bool |
|       reserved3 | '1111’ | 4 | bslbf |
|       transparency | | 8 | unimbf |
|     } | | | |
|   } | | | |
| } | | | |

**layer_delete_flag** – This field specifies whether there is a deleted layer. A '0' flag indicates that there is no deleted layer, and a '1' flag indicates that there is a deleted layer. If there is a deleted layer, the layer number of the deleted layer should be given.

**layer_add_flag** – This field specifies whether there is an added layer. A '0' flag means there is no added layer, a '1' flag means there is an added layer. If there is an added layer, the full information of the added layer should be given.

**layer_display_order_flag** – This field specifies whether there are layers that need to change the display order. A '0' flag indicates that no layer needs to change its order, a '1' flag indicates that a layer needs to change its order. If a layer needs to be changed, the layer number of the layer to be adjusted and the new given layer display order should be given.

**layer_adjust_flag** – This field specifies whether there is an adjustment layer. A '0' flag indicates that there is no adjustment layer, a '1' flag indicates that there is an adjustment layer. If there is an adjustment layer, the information about the adjusted parameters of the adjustment layer should be given.

## 3.4 MMT signalling information descriptors

### 3.4.1 List of MMT-signalling information descriptors

Table 20 shows the list of descriptors.

TABLE 20

**List of descriptors**

| **Descriptor name** | **Descriptor_tag value assignment** | **Description** | **Specified in ISO/IEC 23008-1** | **Use in MMT-based broadcasting systems** | **Use in SMT-based systems** |
|---|---|---|---|---|---|
| CRI descriptor | 0x0000 | Provides the relationship between the NTP timestamp and the MPEG-2 STC for synchronization. | X | | X |
| MPU timestamp descriptor | 0x0001 | Provides presentation time of MPU. | X | X | |
| Dependency descriptor | 0x0002 | Provides Asset identifications that depend on other Assets. | X | X | |
| Generic file delivery table (GFDT) descriptor | 0x0003 | Provides one or more CodePoints describing the association of a specific object and object delivery properties. | X | | |
| AT descriptor | 0x000C | Provides the information about time period during which Asset could be available from the server side. | X | | X |
| CEU timestamp descriptor | 0xEC00 | Provides presentation time of CEU. | | | X |
| Asset relationship information descriptor | 0xEC01 | Provides the association relationship information between Assets in the same Package. | | | X |
| MUR descriptor | 0xEC02 | Provides the information on ranking media Asset. | | | X |
| CEU consumption descriptor | 0xEC03 | Provides the layer information that shall be presented by the media content CEU. | | | X |

### 3.4.2 Detailed specifications of descriptors

#### 3.4.2.1 CRI descriptor

The syntax and semantics of the CRI descriptor are specified in ISO/IEC 23008-1.

#### 3.4.2.2 MPU timestamp descriptor

The syntax and semantics of the MPU timestamp descriptor are specified in ISO/IEC 23008-1.

#### 3.4.2.3 Dependency descriptor

The syntax and semantics of the dependency descriptor are specified in ISO/IEC 23008-1.

#### 3.4.2.4 Generic file delivery table (GFDT) descriptor

The syntax and semantics of the generic file delivery table (GFDT) descriptor are specified in ISO/IEC 23008-1.

#### 3.4.2.5 AT descriptor

The syntax and semantics of the AT descriptor are specified in ISO/IEC 23008-1.

#### 3.4.2.6 CEU timestamp descriptor

This descriptor provides presentation time of the first AU of CEU in presentation order after application of any offset. When the corresponding media presentation time has exceeded the actual time, the descriptor shall be ignored. The details of this descriptor are described in Table 21.

TABLE 21

**CEU timestamp descriptor syntax**

| Syntax | Value | No. of bits | Remarks |
|---|---|---|---|
| CEU_timestamp_descriptor () { | | | |
|     **descriptor_tag** | | 16 | uimsbf |
|     **descriptor_length** | | 8 | uimsbf |
|     for (i=0; i<N; i++) { | | | |
|       **ceu_sequence_number** | | 32 | uimsbf |
|       **ceu_presentation_time** | | 64 | uimsbf |
|     } | | | |
| } | | | |

**descriptor_tag** – This field indicates the type of the descriptor.

**descriptor_length** – This field indicates the length in bytes counting from the next byte after this field to the last byte of the descriptor.

**ceu_sequence_number** – This field indicates the sequence number of the CEU corresponding to the descriptor.

**ceu_presentation_time** – This field indicates the presentation time of the first AU in the designated CEU by 64-bit NTP time stamp format.

### 3.4.2.7 Asset relationship information descriptor

The asset group descriptor is to indicate the association relationship between Assets in the same SMT Package to guide the client for proper decoding, adaptive switching and personalized presentation. The details of this descriptor are described in Table 22.

TABLE 22

**Syntax of asset relationship information descriptor**

| **Syntax** | **Value** | **No. of bits** | **Mnemonic** |
|---|---|---|---|
| Asset_relationship_ information_descriptor() { | | | |
|    **descriptor_tag** | | 16 | uimsbf |
|    **descriptor_length** | | 16 | uimsbf |
|    reserved | ‘1111’ | 4 | |
|    **dependency_flag** | | 1 | blsbf |
|    **composition_flag** | | 1 | blsbf |
|    **equivalence_flag** | | 1 | blsbf |
|    **similarity_flag** | | 1 | blsbf |
|    if(dependency_flag) { | | | |
|        **num_dependencies** | N1 | 8 | uimsbf |
|        for (i = 0; i <N1; i++) { | | | |
|            **asset_id()** | | | |
|        } | | | |
|    } | | | |
|    if(composition_flag) { | | | |
|        **num_compositions** | N2 | 8 | uimsbf |
|        for (i = 0; i <N2; i++) { | | | |
|            **asset_id()** | | | |
|        } | | | |
|    } | | | |
|    if(equivalence_flag) { | | | |
|        **equivalence_selection_level** | | 8 | uimsbf |
|        **num_equivalences** | N3 | 8 | uimsbf |
|            for (i = 0; i <N3; i++) { | | | |
|                **asset_id()** | | | |
|                **equivalence_selection_level** | | 8 | uimsbf |
|        } | | | |
|    } | | | |
|    if(similarity_flag) { | | | |
|        **similarity_selection_level** | | 8 | uimsbf |
|        **num_similarities** | N4 | 8 | uimsbf |
|        for (i = 0; i <N4; i++) { | | | |
|            **asset_id()** | | | |
|            **similarity_selection_level** | | 8 | uimsbf |
|        } | | | |
|    } | | | |
| } | | | |

**descriptor_tag** – This field indicates the type of the descriptor.

**descriptor_length** – This field indicates the length in bytes counting from the next byte after this field to the last byte of the descriptor.

**dependency_flag** – This field indicates whether there are other Assets with dependencies in this descriptor. A value of '0' means there are none, and no need to add them.

**composition_flag** – This field indicates if there are other Assets in this descriptor that have a composition relationship. The value '0' means there are none and need not be added.

**equivalence_flag** – This field indicates if there are other Assets with equivalence relationship in this descriptor. Value '0' means no, no need to add.

**similarity_flag** – This field indicates if there are other Assets with similarity in this descriptor. Value '0' means no, no need to add.

**num_dependencies** – This field indicates the number of Assets on which the Asset described by this descriptor depends.

**asset_id** – This field indicates the IDs of the Assets on which the Asset described by this descriptor depends. The order of the Asset IDs provided in this descriptor corresponds to its internal encoding dependency hierarchy.

**num_compositions** – This field indicates the number of Assets that have a combined relationship with the Asset described by this descriptor.

**asset_id** – This field indicates the ID of the Assets that have a combined relationship with the Asset described by this descriptor.

**equivalence_selection_level** – This field indicates the presentation level of the corresponding Asset in the equivalence relation group. A value of '0' indicates that the Asset is rendered by default. When the default Asset cannot be selected, Assets with smaller presentation levels will be selected and presented as alternatives.

**num_equivalences** – This field indicates the number of Assets that have equivalence to the Asset described by this descriptor.

**asset_id** – This field indicates the ID of the Asset that has equivalences with the Asset described by this descriptor.

**similarity_selection_level** – This field indicates the presentation level of the corresponding Asset in the similarity relation group. A value of '0' indicates that the Asset is rendered by default. When the default Asset cannot be selected, Assets with smaller presentation levels will be selected and presented as alternatives.

**num_similarities** – This field indicates the number of Assets that have similarities to the Asset described by this descriptor.

**asset_id** – This field indicates the ID of the Assets that have similarities to the Asset described by this descriptor.

### 3.4.2.8 MUR descriptor

In order to meet the demand of personalized consumption, the MUR (media unit relationship) descriptor is proposed to grade Asset according to the media content type, importance and other characteristics. The details of this descriptor are described in Table 23.

TABLE 23

**Syntax of MUR descriptor**

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
| MUR_descriptor() { | | | |
| **descriptor_tag** | | 16 | uimsbf |
| **descriptor_length** | | 16 | uimsbf |
| **edit_id** | | | |
| **edit_id_number** | N1 | 16 | uimsbf |
| for(j=0; j<N1; j++) { | | | |
| **ceu_sequence_number** | | 32 | uimsbf |
| } | | | |
| } | | | |

**descriptor_tag** – This field identifies the type of the descriptor.

**descriptor_length** – This field identifies the length in bytes counting from the next byte after this field to the last byte of the descriptor.

**edit_id** – This field identifies the flag indicating the Asset level.

**edit_id_number** – This field identifies the number of CEUs contained under this Asset level.

**ceu_sequence_number** – This field identifies the sequence number of the ceu, indicating the ceu corresponding to the edit_id flag.

### 3.4.2.9 CEU consumption descriptor

This descriptor describes the layer information that should be presented by the media content CEU and describes the layer information that can be replaced or copied in the CEU. The details of this descriptor are described in Table 24.

TABLE 24

**Syntax of the CEU consumption descriptors**

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
| CEU_consumption_descriptor() { | | | |
| **descriptor_tag** | | 16 | uimsbf |
| **descriptor_length** | | 16 | uimsbf |
| **number_of_CEUs** | N1 | 8 | unimbf |
| for(i = 0; i <N1;i++) { | | | |
| **CEU_sequence_number** | | 32 | unimbf |
| **number_of_layer** | N2 | 8 | unimbf |
| for(i = 0; i <N2;i++){ | | | |
| **layer_id** | | 8 | unimbf |
| } | | | |
| **layer_exchange_flag** | | 1 | bslbf |
| **layer_copy_flag** | | 1 | bslbf |
| reserved | '111111' | 6 | bslbf |

TABLE 24 (*end*)

| Syntax | Value | No. of bits | Mnemonic |
|---|---|---|---|
| if(layer_exchange_flag){ | N3 | 8 | unimbf |
| **number_of_exchange_layer** | | | |
| for(i = 0; i <N3;i++) { | | 8 | unimbf |
| **exchange_layer_id** | | | |
| } | | | |
| } | | | |
| if(layer_copy_flag){ | N4 | 8 | unimbf |
| **number_of_copy_layer** | | | |
| for(i = 0; i <N4;i++) { | | 8 | unimbf |
| **copy_layer_id** | | | |
| } | | | |
| } | | | |
| } | | | |
| } | | | |

**descriptor_tag** – This field identifies the type of the descriptor.

**descriptor_length** – This field identifies the length in bytes counting from the next byte after this field to the last byte of the descriptor.

**number_of_CEUs** – This field identifies the number of layers that should be presented by the current CEU.

**CEU_sequence_number** – This field identifies the sequence number of the current CEU. The first CEU sequence number in the Asset should be '0', and each subsequent CEU should be increased by '1'.

**number_of_layer** – This field identifies the layer number that the current CEU should be presented.

**layer_exchange_flag** – This field identifies whether the current CEU can exchange presentation layers with CEUs presented in other layers under the premise that it can be correctly presented to a layer. A '0' flag indicates that only the layer number listed in this descriptor shall be presented, while a '1' flag indicates that the user can exchange the CEU with other CEUs.

**layer_copy_flag** – This field identifies whether the current CEU can be copied to other layers under the premise that it can be correctly presented to a layer. A '0' flag indicates that only the layer number listed in this descriptor shall be presented, while a '1' flag indicates that the client can copy and present the CEU to other layers. When copying and presenting, the client should overwrite the CEU content of the original layer.

**number_of_exchange_layer** – This field identifies the number of layers that can be presented in exchange with the current CEU.

**exchange_layer_id** – This field identifies the number of layers that can be presented in exchange with the current CEU.

**number_of_copy_layer** – This field indicates the number of layers that can be copied and presented by the current CEU.

**copy_layer_id** – This field indicates the layer id that can be copied and presented by the current CEU.

### 3.5 Packet identification

ISO/IEC 23008-1 does not specify a fixed value for MMTP packet. However, it is beneficial that certain fixed values are used to identify MMTP packets so that a receiver terminal can easily recognize the information carried by the MMTP packet.

## 4 Start-up procedure of broadcasting service

Figure 10 shows the start-up procedure of a receiver terminal from the moment a user presses a channel change button to the moment the new TV programme begins to be shown on screen. Pressing the channel change button corresponds to identifying the service_id of the desired TV programme.

The first procedure is initiated in the IP multiplexing layer. In the case of the TLV multiplexing scheme, the receiver terminal parses the address map table (AMT) to associate the service_id with the IP data flow. Then, it parses the TLV-network information table (NIT) to acquire the physical channel information, such as the channel frequency carrying the IP data flow. On the basis of the acquired information, it tunes in to the broadcasting channel and receives the desired IP data flow.

After receiving the IP data flow, the second procedure in the MMT layer is initiated. The received IP packets carry the MMTP packets. To retrieve the PA message, the receiver terminal seeks MMTP packets whose packet_id=0. It parses the received PA message and gets the MP table in the PA message.

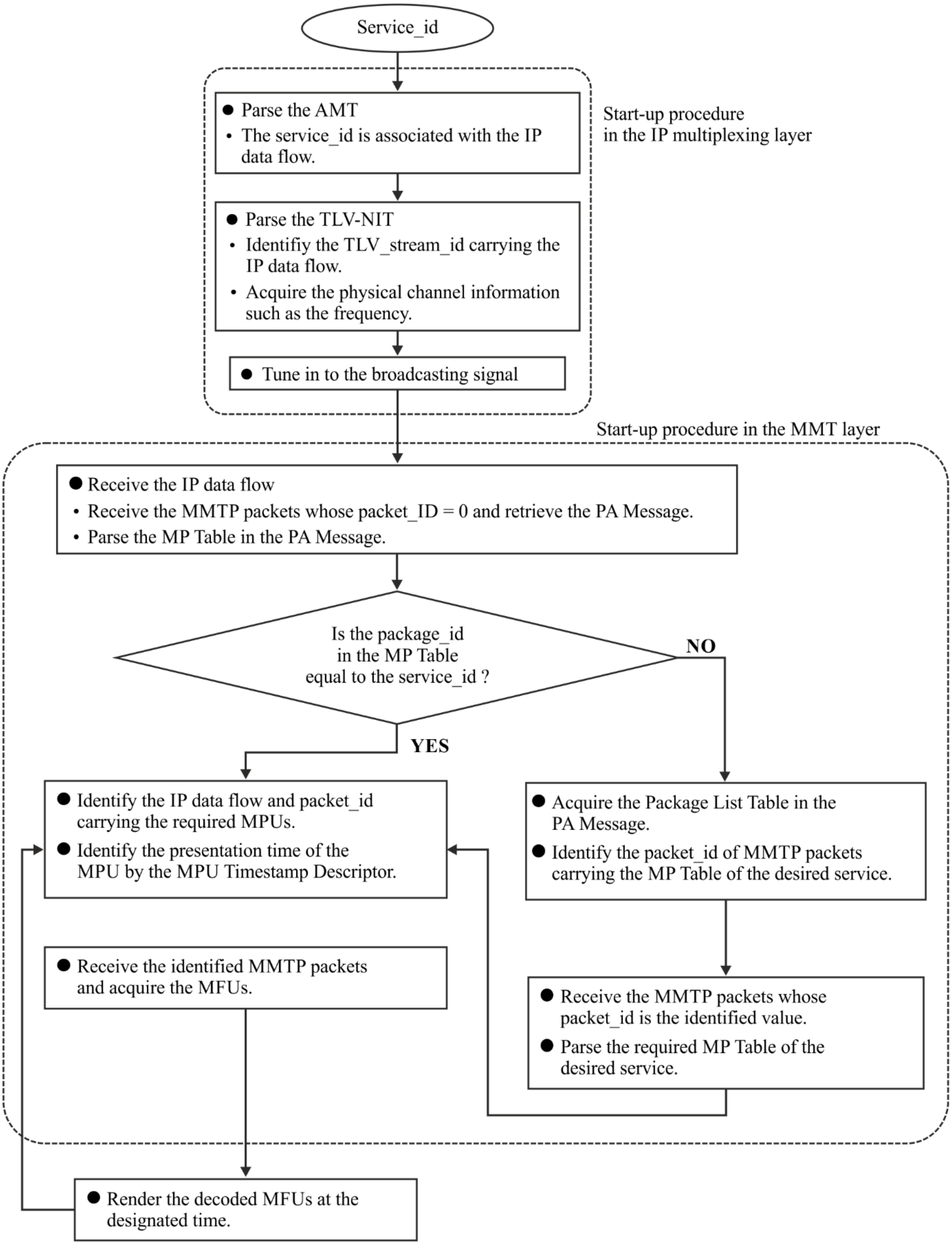
In MMT-based broadcasting systems, multiple services might be multiplexed into one IP data flow, as shown in Fig. 2 of Annex 1. Therefore, the receiver terminal checks whether the package_id of the acquired MP Table is equal to the desired service_id or not. If the package_id of the MP table is not equal to the desired service_id, the receiver terminal acquires the package list table from the PA message. Then, from the package list table, it identifies the packet_id of the MMTP packets carrying the MP table of the desired service.

From the MP table, the receiver terminal identifies the IP data flow and packet_id of MMTP packets carrying the required MPUs in the desired TV programme. It also identifies the presentation time of the MPU by referring to the MPU timestamp descriptor included in the MP table.

Then, the receiver terminal receives the identified MMTP packets carrying media components in the form of MFUs. The MFUs are decoded and rendered at the designated time. The user watches the desired TV programme at this time.

FIGURE 10

**Start-up procedure of broadcasting service**


Service_id
● Parse the AMT
• The service_id is associated with the IP data flow.
● Parse the TLV-NIT
• Identifiy the TLV_stream_id carrying the IP data flow.
• Acquire the physical channel information such as the frequency.
● Tune in to the broadcasting signal
Start-up procedure in the IP multiplexing layer
Start-up procedure in the MMT layer
● Receive the IP data flow
• Receive the MMTP packets whose packet_ID = 0 and retrieve the PA Message.
• Parse the MP Table in the PA Message.
Is the package_id in the MP Table equal to the service_id ?
NO
YES
● Identify the IP data flow and packet_id carrying the required MPUs.
● Identify the presentation time of the MPU by the MPU Timestamp Descriptor.
● Acquire the Package List Table in the PA Message.
● Identify the packet_id of MMTP packets carrying the MP Table of the desired service.
● Receive the identified MMTP packets and acquire the MFUs.
● Receive the MMTP packets whose packet_id is the identified value.
● Parse the required MP Table of the desired service.
● Render the decoded MFUs at the designated time.

Note: This procedure does not include processes related to CAS.

BT.2074-10

# Attachment 1
# to Annex 2
# (informative)

## ARIB signalling information

### 1 Additional signalling information

Additional signalling information is specified by ARIB in its specification STD-B60 "MMT-based media transport scheme in digital broadcasting systems". Tables 25, 26 and 27 list the messages, tables, and descriptors, respectively.

MPEG-2 TS based conventional broadcasting systems have used numerous tables and descriptors. Some of them are reused in MMT-based broadcasting systems. This signalling information has "MH-" at the beginning of its name.

TABLE 25

**List of Messages additionally specified by ARIB**

| **Message name** | **Message_id assignment** | **Description** |
|---|---|---|
| Conditional access (CA) message | 0x8001 | Conveys information on conditional access. |
| M2short section message | 0x8002 | Conveys MPEG-2 short Section-format table. |
| Data transmission message | 0x8003 | Conveys one or more tables related to data transmission. |

TABLE 26

**List of Tables additionally specified by ARIB**

| **Table name** | **Table_id assignment** | **Description** |
|---|---|---|
| Layout configuration table | 0x81 | Assigns layout information for displaying Assets. |
| Entitlement control message | 0x82 – 0x83 | Conveys common information consisting TV programme information (related to TV programmes, descrambling keys, etc.) and control information (instructions on compulsory on/off of the decoder's descramble function). |
| Entitlement management message | 0x84 – 0x85 | Conveys individual information including contract information for each subscriber and work keys to decrypt common information. |
| MH-conditional access table | 0x86 | Conveys one or more descriptors related to conditional access. |
| Download control message | 0x87 – 0x88 | Conveys information related to descrambling keys to descramble channel encryption for download. |
| Download management message | 0x89 – 0x8A | Conveys information related to download keys to decrypt DCM. |

TABLE 26 (*end*)

| **Table name** | **Table_id assignment** | **Description** |
|---|---|---|
| MH-event information table | 0x8B – 0x9B | Conveys information related to TV programmes such as programme name, broadcast date and time, and explanations of them. |
| MH-application information table | 0x9C | Conveys dynamic control information and additional information for executing applications. |
| MH-broadcaster information table | 0x9D | Presents information on broadcasters in the network. |
| MH-software download trigger table | 0x9E | Conveys announcement information about downloads, such as the service id, schedule information, and target receiver terminals. |
| MH-service description table | 0x9F – 0xA0 | Conveys information related to the programme channel, such as the channel name and broadcaster’s name. |
| MH-time offset table | 0xA1 | Indicates the current date and time and provides the time difference between the current time and indicating time for humans. |
| MH-common data table | 0xA2 | Conveys data that are commonly required for receiver terminals and stored in non-volatile memory, such as company logos. |
| Data directory management table | 0xA3 | Provides directory information on files constituting applications. |
| Data asset management table | 0xA4 | Provides the MPU configuration of the Asset and the version of the MPU. |
| Data content configuration table | 0xA5 | Provides configuration information on files that are used as data content. |
| Event message table | 0xA6 | Provides information related to event messages. |

TABLE 27

**List of Descriptors additionally specified by ARIB**

| **Descriptor name** | **Descriptor _tag value assignment** | **Description** |
|---|---|---|
| Asset group descriptor | 0x8000 | Provides the group and priority within a group of Assets. |
| Event package descriptor | 0x8001 | Provides a description on the relationship of events and the MMT packages. |
| Background colour descriptor | 0x8002 | Provides background colour information on the layout configuration. |
| MPU presentation region descriptor | 0x8003 | Provides information on the position of displaying the MPU. |
| Access control descriptor | 0x8004 | Identifies the conditional access method. |

TABLE 27 (*continued*)

| **Descriptor name** | **Descriptor _tag value assignment** | **Description** |
|---|---|---|
| Scramble descriptor | 0x8005 | Identifies the scrambling sub-system. |
| Message authentication method descriptor | 0x8006 | Identifies the message authentication method. |
| MH-emergency information descriptor | 0x8007 | Provides information on and functions for emergency alarm signals. |
| MH-MPEG-4 audio descriptor | 0x8008 | Provides basic information for identifying the coding parameters of MPEG-4 audio streams. |
| MH-MPEG-4 audio extension descriptor | 0x8009 | Provides additional information for identifying the profile and level of MPEG-4 audio streams. |
| MH-HEVC video descriptor | 0x800A | Provides information for identifying the coding parameters of HEVC video streams. |
| MH-linkage descriptor | 0x800B | Provides a description of the relationship with other programme channels. |
| MH-Event group descriptor | 0x800C | Provides a description of grouping information for multiple events. |
| MH-Service list descriptor | 0x800D | Provides a description of programme channels and a list of their types. |
| MH-Short event descriptor | 0x800E | Provides the name and a brief explanation of the TV programme. |
| MH-Extended event descriptor | 0x800F | Provides detailed information about the TV programme. |
| Video component descriptor | 0x8010 | Provides parameters and explanations of video signals. |
| MH-Stream identifier descriptor | 0x8011 | Identifies individual programme element signals of the TV programme. |
| MH-Content descriptor | 0x8012 | Provides a description of the TV programme's genre. |
| MH-Parental rating descriptor | 0x8013 | Provides information on the permitted minimum audience age. |
| MH-Audio component descriptor | 0x8014 | Provides parameters and explanations of audio signals. |
| MH-Target region descriptor | 0x8015 | Provides target region information. |
| MH-Series descriptor | 0x8016 | Provides series information for multiple events. |
| MH-SI Parameter descriptor | 0x8017 | Provides transmission parameters of signalling information, e.g. the retransmission period. |
| MH-Broadcaster name descriptor | 0x8018 | Provides the broadcaster's name. |
| MH-Service descriptor | 0x8019 | Provides descriptions of the programme channel and its company's name. |
| IP Data flow descriptor | 0x801A | Provides information on IP data flows in the broadcasting services. |
| MH-CA Start-up descriptor | 0x801B | Provides information on the start-up of CA programs having conditional access functions. |

TABLE 27 (*continued*)

| **Descriptor name** | **Descriptor _tag value assignment** | **Description** |
|---|---|---|
| MH-Type descriptor | 0x801C | Provides type of files in data transmission. |
| MH-Info descriptor | 0x801D | Provides information related to MPU or item. |
| MH-Expire descriptor | 0x801E | Provides expiry information. |
| MH-Compression type descriptor | 0x801F | Provides the compression type and bytes of an item before compression. |
| MH-Data component descriptor | 0x8020 | Identifies the coding scheme of data. |
| UTC-NPT Reference descriptor | 0x8021 | Provides the relationship between NPT and UTC. |
| Event message descriptor | 0x8022 | Provides general information related to event messages. |
| MH-Local time offset descriptor | 0x8023 | Provides the current local time and indicates whether daylight-savings time is observed. |
| MH-Component group descriptor | 0x8024 | Provides a description of grouping information for multiple components. |
| MH-Logo transmission descriptor | 0x8025 | Provides characters consisting of simple logos and references to CDT-format logos. |
| MPU Extended timestamp descriptor | 0x8026 | Provides a decoding timestamp for access units in the MPU. |
| MPU Download content descriptor | 0x8027 | Provides property information on the download content delivered in the MPU. |
| MH-Network download content descriptor | 0x8028 | Provides property information on the download content delivered in broadband networks. |
| MH-Application descriptor | 0x8029 | Provides a description of an application. |
| MH-Transport protocol descriptor | 0x802A | Provides transmission protocol and location information on applications that depend on transmission protocols. |
| MH-Simple application location descriptor | 0x802B | Provides detailed location information on applications. |
| MH-Application permission descriptor | 0x802C | Provides descriptions of the application boundary and permission information. |
| MH-Autostart priority descriptor | 0x802D | Provides priority information for launch of applications. |
| MH-Cache control info descriptor | 0x802E | Provides cache control information for caching resources constituting applications. |
| MH-Randomized latency descriptor | 0x802F | Provides latency information for application control. |
| Linked PU descriptor | 0x8030 | Provides information on linked presentation units. |
| Locked cache descriptor | 0x8031 | Provides file information that is cached and locked. |
| Unlocked cache descriptor | 0x8032 | Provides file information that is un-cached and unlocked. |

TABLE 27 (*end*)

| Descriptor name | Descriptor _tag value assignment | Description |
|---|---|---|
| MH-Download protection descriptor | 0x8033 | Provides location of download management or download control messages. |
| Application service descriptor | 0x8034 | Provides information on applications related to services. |
| MPU node descriptor | 0x8035 | Provides label of directory to which this MPU belongs. |
| Presentation unit structure descriptor | 0x8036 | Provides list of MPUs for presentation units. |
| MH-Hierarchy descriptor | 0x8037 | Provides information on video components coded with scalable video coding. |
| Content copy control descriptor | 0x8038 | Provides copy control information for services. |
| Content usage control descriptor | 0x8039 | Provides copy control information for programmes. |
| MH-External application control descriptor | 0x803A | Identifies permission of external applications to broadcast resources. |
| MH-Playback application descriptor | 0x803B | Provides information on applications related to recorded content. |
| MH-Simple playback application location descriptor | 0x803C | Provides information on link of applications related to recorded content. |
| MH-Application expiration descriptor | 0x803D | Provides expiration date and time of applications. |
| Related broadcaster descriptor | 0x803E | Provides broadcaster identification for sharing NVRAM. |
| Multimedia service information descriptor | 0x803F | Provides detailed information on each piece of content for multimedia services. |
| Emergency news descriptor | 0x8040 | Signals that emergency news is broadcasted. |
| MH-CA contract information descriptor | 0x8041 | Provides contract information to decide whether end user is able to watch programmes. |
| MH-CA service descriptor | 0x8042 | Provides broadcaster group identification and message control. |
| MH-Linkage descriptor | 0xF000 | Provides links to additional information. |
| MH-Short event descriptor | 0xF001 | Provides short descriptions for events. |
| MH-Extended event descriptor | 0xF002 | Provides detailed descriptions for events. |
| Event message descriptor | 0xF003 | Provides information for event messages. |

## 2 Header extension of MMTP packets

When the extension_type field is set to 0x0000, hdr_ext_type field specifies the type of multi-type header extension. The value of hdr_ext_type is specified in Table 28.

TABLE 28

**hdr_ext_type values**

| Value | Description |
|---|---|
| 0x0000 | Reserved for future use |
| 0x0001 | Reserved for ARIB STD-B61 (scrambling information) |
| 0x0002 | Reserved for ARIB STD-B60 (download_id) |
| 0x0003 – 0x7FFF | Reserved for future use |

## 3 Packet identification assignment

The fixed values are assigned to recognize the information carried by the MMTP packet. These values are listed in Table 29.

TABLE 29

**Packet ID assignments**

| Value | Description |
|---|---|
| 0x0000 | PA Message |
| 0x0001 | Reserved for CA Message |
| 0x0002 | AL-FEC Message |
| 0x0003 – 0x00FF | Reserved for future use |
| 0x0100 – 0x7FFF | Reserved for private use |
| 0x8000 | Reserved for M2section Message carrying MH-EIT |
| 0x8001 | Reserved for M2section Message carrying MH-AIT |
| 0x8002 | Reserved for M2section Message carrying MH-BIT |
| 0x8003 | Reserved for M2section Message carrying MH-SDTT |
| 0x8004 | Reserved for M2section Message carrying MH-SDT |
| 0x8005 | Reserved for M2short Section Message carrying MH-TOT |
| 0x8006 | Reserved for M2section Message carrying MH-CDT |
| 0x8007 | Reserved for Data Transmission Message |
| 0x8008 – 0xEFFF | Reserved for private use |
| 0xF000 – 0xFFFF | Reserved for private use |