INTERNATIONAL STANDARD

ISO 21219-6

First edition
2019-07

Intelligent transport systems — Traffic and travel information(TTI) via transport protocol experts group, generation 2 (TPEG2) —

Part 6: Message management container (TPEG2-MMC)

Systèmes intelligents de transport — Informations sur le trafic et le tourisme via le groupe expert du protocole de transport, génération 2 (TPEG2) —

Partie 6: Conteneur de gestion de message (TPEG2-MMC)

Reference number
ISO 21219-6:2019(E)

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

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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

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

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

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

This document was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*.

This first edition cancels and replaces ISO/TS 21219-6:2015, which has been technically revised.

A list of all parts in the ISO 21219 series can be found on the ISO website.

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

Introduction

History

TPEG technology was originally proposed by the European Broadcasting Union (EBU) Broadcast Management Committee, who established the B/TPEG project group in the autumn of 1997 with a brief to develop, as soon as possible, a new protocol for broadcasting traffic and travel-related information in the multimedia environment. TPEG technology, its applications and service features were designed to enable travel-related messages to be coded, decoded, filtered and understood by humans (visually and/or audibly in the user's language) and by agent systems. Originally, a byte-oriented data stream format, which may be carried on almost any digital bearer with an appropriate adaptation layer, was developed. Hierarchically structured TPEG messages from service providers to end-users were designed to transfer information from the service provider database to an end-user's equipment.

One year later, in December 1998, the B/TPEG group produced its first EBU specifications. Two documents were released. Part 2 (TPEG-SSF, which became ISO/TS 18234-2) described the syntax, semantics and framing structure, which was used for all TPEG applications. Meanwhile, Part 4 (TPEG-RTM, which became ISO/TS 18234-4) described the first application for road traffic messages.

Subsequently, in March 1999, CEN/TC 278, in conjunction with ISO/TC 204, established a group comprising members of the former EBU B/TPEG and this working group continued development work. Further parts were developed to make the initial set of four parts, enabling the implementation of a consistent service. Part 3 (TPEG-SNI, ISO/TS 18234-3) described the service and network information application used by all service implementations to ensure appropriate referencing from one service source to another.

Part 1 (TPEG-INV, ISO/TS 18234-1) completed the series by describing the other parts and their relationship; it also contained the application IDs used within the other parts. Additionally, Part 5, the public transport information application (TPEG-PTI, ISO/TS 18234-5), was developed. The so-called TPEG-LOC location referencing method, which enabled both map-based TPEG-decoders and non-map-based ones to deliver either map-based location referencing or human readable text information, was issued as ISO/TS 18234-6 to be used in association with the other applications of parts of the ISO/TS 18234 series to provide location referencing.

The ISO/TS 18234 series has become known as TPEG Generation 1.

TPEG Generation 2

When the Traveller Information Services Association (TISA), derived from former forums, was inaugurated in December 2007, TPEG development was taken over by TISA and continued in the TPEG applications working group.

It was about this time that the (then) new Unified Modelling Language (UML) was seen as having major advantages for the development of new TPEG applications in communities who would not necessarily have binary physical format skills required to extend the original TPEG TS work. It was also realized that the XML format for TPEG described within the ISO/TS 24530 series (now superseded) had a greater significance than previously foreseen, especially in the content-generation segment and that keeping two physical formats in synchronism, in different standards series, would be rather difficult.

As a result, TISA set about the development of a new TPEG structure that would be UML-based. This has subsequently become known as TPEG Generation 2.

TPEG2 is embodied in the ISO/TS 21219 series and it comprises many parts that cover introduction, rules, toolkit and application components. TPEG2 is built around UML modelling and has a core of rules that contain the modelling strategy covered in ISO 21219-2, ISO 21219-3 and ISO 21219-4 and the conversion to two current physical formats: binary and XML; others could be added in the future. TISA uses an automated tool to convert from the agreed UML model XMI file directly into an MS Word document file, to minimize drafting errors, that forms the annex for each physical format.

TPEG2 has a three-container conceptual structure: message management (ISO 21219-6), application (several parts) and location referencing (ISO/TS 21219-7). This structure has flexible capability and can accommodate many differing use cases that have been proposed within the TTI sector and wider for hierarchical message content.

TPEG2 also has many location referencing options as required by the service provider community, any of which may be delivered by vectoring data included in the location referencing container.

The following classification provides a helpful grouping of the different TPEG2 parts according to their intended purpose. Note that the list below may be incomplete, e.g. new TPEG2 parts may be introduced after publication of this document.

— Toolkit parts: TPEG2-INV (ISO/TS 21219-1), TPEG2-UML (ISO 21219-2), TPEG2-UBCR (ISO 21219-3), TPEG2-UXCR (ISO 21219-4), TPEG2-SFW (ISO 21219-5), TPEG2-MMC (ISO 21219-6), TPEG2-LRC (ISO/TS 21219-7).

— Special applications: TPEG2-SNI (ISO/TS 21219-9), TPEG2-CAI (ISO/TS 21219-10), TPEG2-LTE (ISO/TS 21219-24).

— Location referencing: TPEG2-OLR (ISO/TS 21219-22), TPEG2-GLR (ISO/TS 21219-21), TPEG2-TLR (ISO 17572-2), TPEG2-DLR (ISO 17572-3).

— Applications: TPEG2-PKI (ISO/TS 21219-14), TPEG2-TEC (ISO/TS 21219-15), TPEG2-FPI (ISO/TS 21219-16), TPEG2-TFP (ISO 21219-18), TPEG2-WEA (ISO/TS 21219-19), TPEG2-RMR (ISO/TS 21219-23), TPEG2-EMI (ISO/TS 21219-25), TPEG2-VLI (ISO/TS 21219-26).

TPEG2 has been developed to be broadly (but not totally) backward compatible with TPEG1 to assist in transitions from earlier implementations, while not hindering the TPEG2 innovative approach and being able to support many new features, such as dealing with applications having both long-term, unchanging content and highly dynamic content, such as parking information.

This document is based on the TISA specification technical/editorial version reference: SP10035.

Intelligent transport systems — Traffic and travel information(TTI) via transport protocol experts group, generation 2 (TPEG2) —

Part 6: Message management container (TPEG2-MMC)

1 Scope

This document adds a basic toolkit definition to the ISO 21219 series specifying the Message Management Container (MMC) which is used by all TPEG applications to provide information about the handling of messages on the TPEG client side. The MMC holds administrative information allowing a decoder to handle the message appropriately. This information is not aimed at the end user. The MMC is a toolkit and not a stand-alone application but is included by TPEG applications.

2 Normative references

There are no normative references in this document.

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

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

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

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

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

3.1.1
Message
unit of information that is controlled under a message ID and contains a MessageManagementContainer, MMCMasterMessage or MMCMessagePart component

3.1.2
Monolithic Message Management
message management that allows versioning of messages by updating complete messages only

Note 1 to entry: See 4.1.3.

3.1.3
Multipart Message Management
message management that allows parts of messages being transmitted in packets independently

Note 1 to entry: See 4.1.4.

3.1.4
top level container
any component that is on the same level as the message management container

3.1.5
TPEG Server
device or entity on the sending side of the TPEG transmission chain

Note 1 to entry: May consist, e.g. of a TPEG encoder, a stream encoder, a network transmission unit.

3.1.6
TPEG Client
device or entity on the receiving side of the TPEG transmission chain

Note 1 to entry: May consist, e.g. of a broadcast receiver, a TPEG decoder unit.

3.2 Abbreviated terms

MMC	Message Management Container
LRC	Location Referencing Container
ADC	Application Data Container

4 MMC components and capabilities

4.1 Overview

4.1.1 Structure

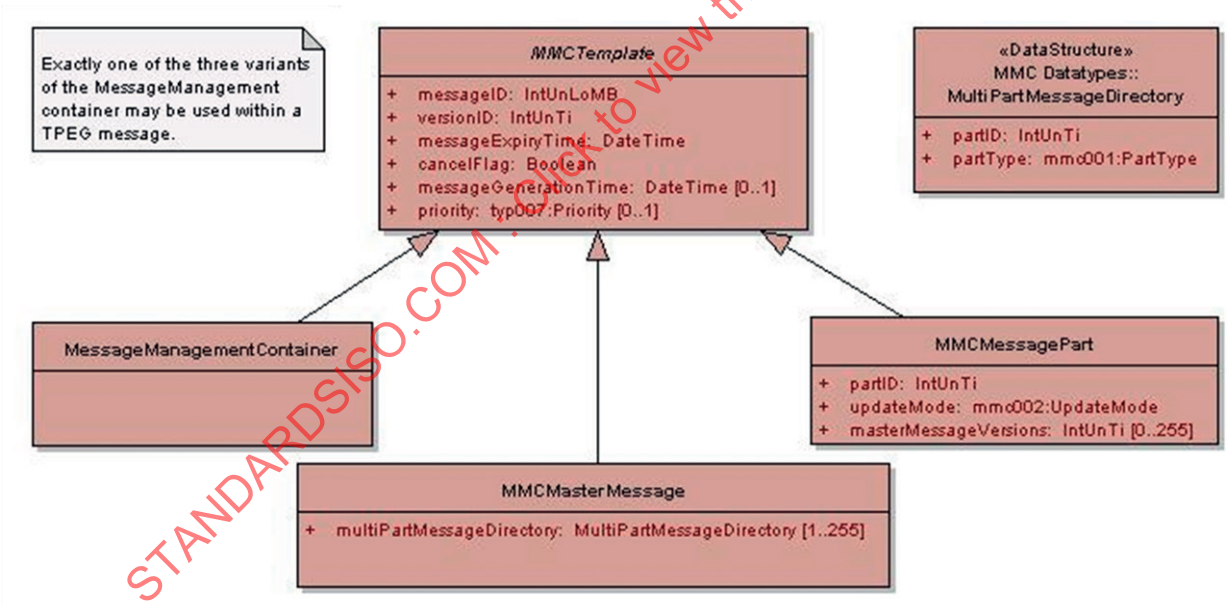

Figure 1 — Structure of the Message Management Container

4.1.2 Capabilities

Any TPEG message typically consists of the following top level containers (see also Figure 2 below):

— Exactly one Message Management Container (MMC);

— Optionally one or several Application Data Container(s) (ADC);

— Optionally one Location Reference Container (LRC).

The MMC contains no data dedicated for the user but only administrative information for the TPEG decoder to handle the message appropriately.

While the ADC part is defined specifically by the related TPEG application specification, the TPEG MMC toolkit is specified by this document for all TPEG applications. The general capabilities and features of the MMC toolkit may be restricted or further detailed by the particular TPEG applications.

The MMC is always stereotyped as an ordered component and shall be the first component in the TPEG Message. The other parts of the message may as well be stereotyped as “UnorderedComponent”.

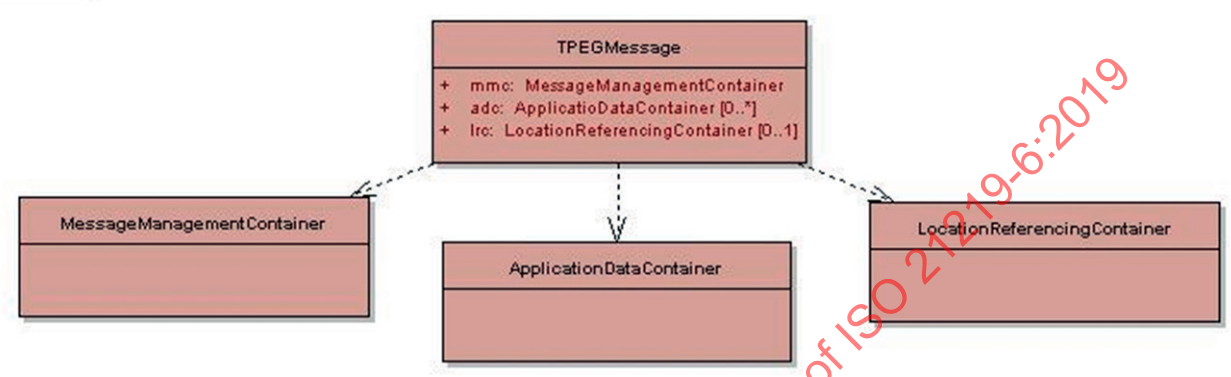

Figure 2 — General structure of a TPEG message

The MMC toolkit includes two basic mechanisms for message management, the Monolithic Message Management and Multipart Message Management, which are described in the following sub-clauses.

Both mechanisms exclude each other, i.e. for a given time there shall either be a message delivered by Monolithic or by Multipart Message Management. Therefore, if a TPEG Client receives a monolithic message and already has a multipart-combined message with the same messageID in its repository it shall remove this message from its repository and shall replace it by the received monolithic message. Conversely, a monolithic message shall be replaced by a new multipart-message as well.

4.1.3 Monolithic Message Management

The usage of monolithic message management enables the replacement of a complete TPEG message by a more recent version of the same message. Thus, this message management method is suitable for TPEG services where most parts of the message are changing during the message updates.

The monolithic message management applies the class ‘MessageManagementContainer’ only which inherits all attributes from the parent class ‘MMCTemplate’ and adds no further ones. The replacement process described above is signalled by using the same value of the attribute ‘messageID’ and an increased value of the attribute ‘versionID’ (see Figure 3 below).

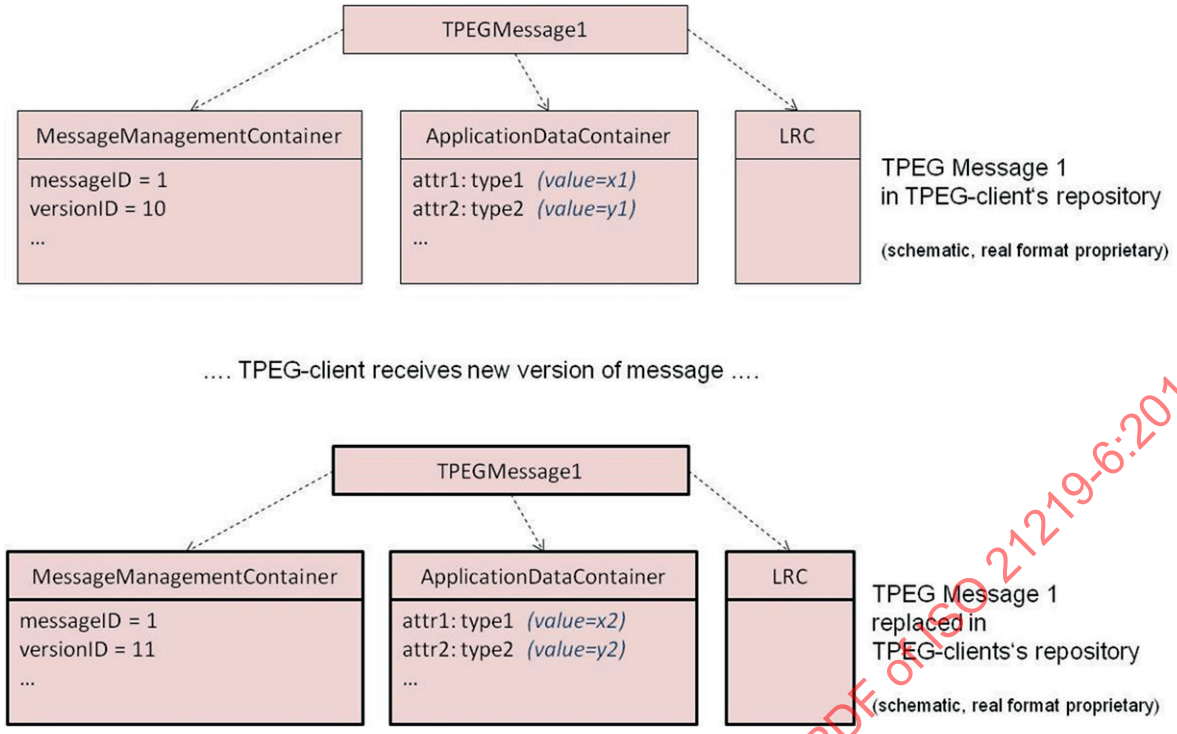

Figure 3 — Replacement of an overall message by Monolithic Message Management

In case of a cancellation, the messageID contained in the MMC indicates the previously received message with the same messageID to be cancelled. If the messageID and versionID is identical to a previously transmitted message this signals that all attached application content (i.e. all ADC and LRC components transmitted in the message) is identical with the previously sent content. In that case a receiver does not need to do a byte-by-byte content comparison of the ADC and LRC content, but nevertheless needs to check the MMC content as that may differ (e.g. a different expiration time).

4.1.4 Multipart Message Management

Multipart Message Management enables the partial update of messages, i.e. it may be used to replace dedicated parts of a message version by version, while other parts remain unchanged. Thus, this method is suitable in particular for TPEG applications where large parts of the message content are static or quasi-static (see also 4.5 and 4.6).

The Multipart Message Management applies the classes 'MMCMasterMessage' and 'MMCMessagePart' (see 4.1.1). Additional to the attributes inherited from the parent class 'MMCTemplate', the 'MMCMasterMessage' container includes a list with the partial messages (attribute 'multiPartMessageDirectory') which shall list all partial messages of the overall/combined message. The master-message shall contain the 'MMCMasterMessage' container only and no ADC or LRC information. The latter containers shall be included in the corresponding partial messages, which are indicated by the 'MMCMessagePart' type MMC.

The master message and each of the related partial messages shall have the same 'messageID' but shall have an independent versioning, i.e. all parts can have a different 'versionID' and the value of this attribute refers to the related message part. Therefore, the TPEG Client has to store the 'versionID' of the most recently received version of each partial message to manage the partial updates of the messages. In particular, the stored combined message shall be updated by a received partial message only if the 'versionID' of the partial message is more recent than the stored 'versionID' of the same partial message.

If the content of the multiPartMessageDirectory has been modified, i.e. the version ID of the 'MMCMasterMessage' is changed, all related partial messages sent before are not valid anymore. A TPEG Client shall then rebuild the message completely by receiving the up-to-date partial messages.

Currently, there is one update feature ('replaceTopLevel') defined for partial messages which is signalled by the 'updateMode' attribute of the 'MMCMessagePart' component. Other update modes may be added in future versions.

4.1.4.1 Update mode 'replaceTopLevel'

This feature may be used to replace or add components in the combined message by the components transmitted by the partial message. The components included by the partial message shall be a sub-tree of the overall message structure, starting with one of the ADC or LDC components as root component. This means that the partial message will update the Application Data or Location Referencing Containers and its related sub-components. Therefore, this feature is an enhancement of the monolithic message management where only an overall message can be replaced/added.

If one or several of the components contained in the partial message are present in the so far stored combined message, these components shall be replaced. If one or several of the components contained in the partial message are not yet present in the so far stored combined message these components shall be added to the combined message.

As only complete components are added/replaced, each of the components contained in the partial message shall include all attributes which are intended to be part of the combined message.

If a combined message contains more than one component of the same component type at a given position within the message structure, all these components shall be replaced by the one(s) contained in the partial message. In particular an array of components (multiplicity [0..*] or [1..*]) shall be replaced completely by the updating message.

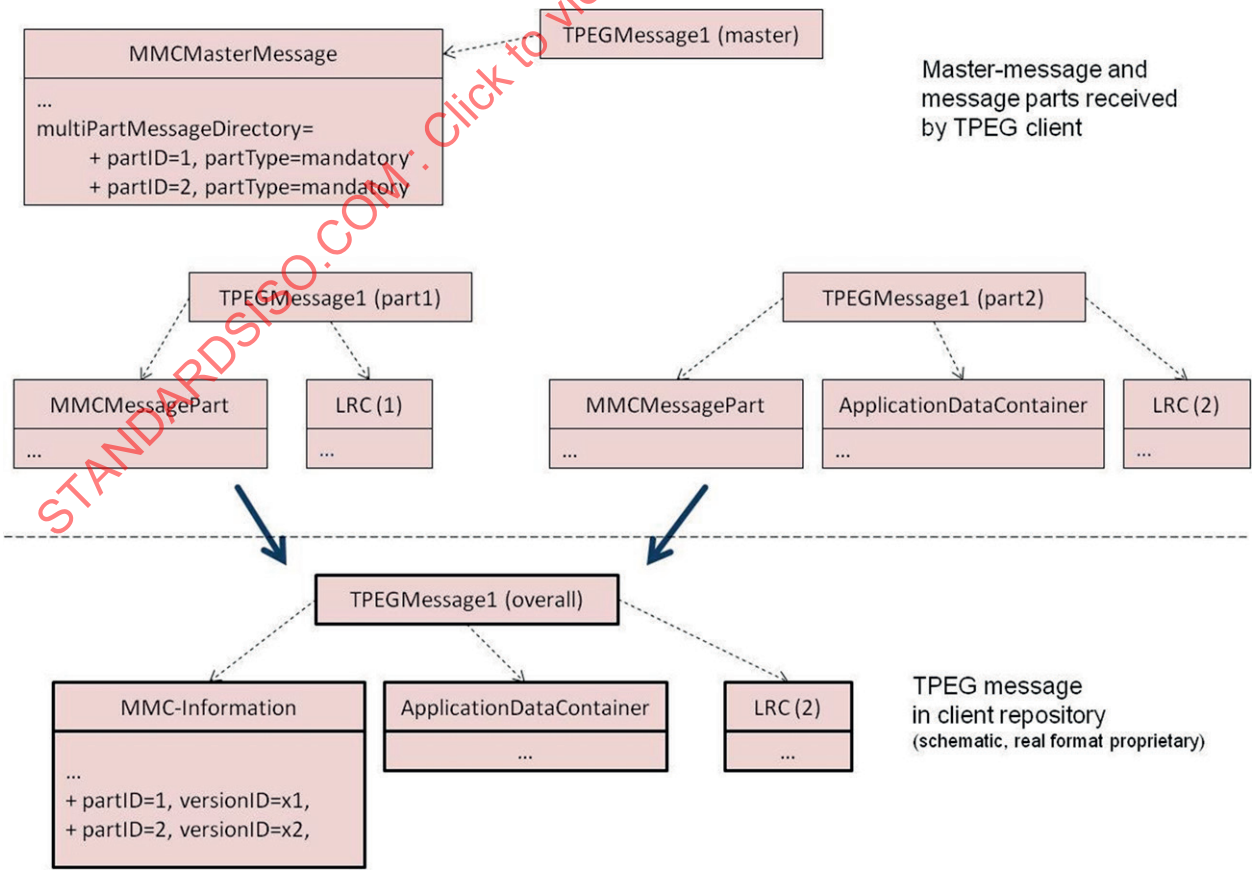

Figure 4 — Example addition/replacement of top level containers by partial messages

4.2 Lifecycle and identification of a TPEG message

TPEG messages or partial messages have the following life-cycle stages which are signalled by the value of the attribute 'versionID' (see also attribute definition in 4.3):

— First transmission: The value of 'versionID' should be 0.

— Message Updates: When the content of a message is updated, i.e. the ADC and/or LRC parts are changing, the versionID value shall be incremented for each new version. This increment should be 1. If the value exceeds 255 it shall be set to a value <255 (wrap around); This value should be 0. In the case of wrap around, the TPEG Server shall ensure that the last usage of the messageExpiryTime for a message with that versionID has expired, before the wrap around appears.

— Cancellation: When a message is not valid anymore the TPEG Server may send a cancellation message with the corresponding 'messageID' value of the message. A cancellation shall result in an incremented versionID with respect to the message it cancels as a cancellation indicates a change in content. The usage of the message cancellation shall be detailed by the particular TPEG application specifications.

— Message expiration: If the current time has progressed beyond the messageExpiryTime of the last received version of the message, the message shall be considered as invalid and not be presented to the user anymore. Message expiration shall be supported both by the client and the server side.

The 'messageID' shall be unique within a dedicated TPEG Service Component. The latter is identified worldwide uniquely by its Application and Content Identification (ACID) which is formed by the TPEG ServiceID, the Content Identification (COID) and Application Identification (AID) as described in the SNI specification (see ISO/TS 18234-3).

A TPEG Client shall not rely on the uniqueness of the messageID and versionID once a message is expired. A TPEG Client shall not rely on the versionID to be incremented by 1 as, e.g. message updates may be lost during transmission.

4.3 MMCTemplate

The abstract component MMCTemplate provides basic attributes for the child components MessageManagementContainer, MMCMasterMessage and MMCMessagePart (see the following subclauses). Applications shall not reference the MMCTemplate component, but one or several of the derived components instead.

Name	Type	Multiplicity	Description
messageID	IntUnLoMB	1	Unique identifier for a message relating to a particular event transmitted in a particular TPEG service component.
versionID	IntUnTi	1	Serial number that distinguishes successive versions of one message in case of message updates. The versionID numbers shall be used incrementally, enabling to track the update progress of a message from first transmission, through updates to cancellation. Wrap around is applied, i.e. the versionID following 255 shall be set to a value <255.

Name	Type	Multiplicity	Description
			The versionID shall change every time the content of a message change. Changes of the message management container only shall not cause a change of the versionID. If for example only the message expiry time in the message management container is changed, the versionID remains unchanged. To avoid ambiguous situations when a wraparound occurs, the following two rules apply: — Not more than one version of a message shall be on air in a service component at the same time within a service with identical service ID. — The message expiry time shall be set in an appropriate way. If a message with the same messageID, but a lower version number is received, a wraparound has occurred if the expiry time of the last received message is later.
messageExpiryTime	DateTime	1	If the current time is later than the messageExpiryTime, the message shall be considered as invalid and the TPEG Client device shall therefore not present it to the user anymore. When the validity of the message contents for the user is to be maintained, the TPEG Server shall send the message, or a newer version thereof before the message expires. Expiry times are transmission specific and may differ dependent on the particular transmission conditions.
cancelFlag	Boolean	1	This flag shall be set (true), if a message, identified by its messageID, is no longer valid and has to be removed in the client device. The message body of messages with a set cancelFlag shall be empty.
messageGenerationTime	DateTime	0..1	Date and time when the message was inserted into the delivery channel. Message generation time provides a system diagnostic tool to determine delays in the transmission chain. End-user devices should ignore this value.
Priority	typ007:Priority	0..1	Messages with higher priority should be decoded or processed prior to other messages if resources in the client device are limited. This is a relative indicator only; it is not directly related to the priority of the content. For example, a new message might have a higher priority than another one with similar content that has been on air for some time.

4.4 MessageManagementContainer

The MessageManagementContainer component inherits all attributes from the abstract component MMCTemplate. No more attributes are added.

Name	Type	Multiplicity	Description
messageID	IntUnLoMB	1	see 4.3
versionID	IntUnTi	1	see 4.3
messageExpiryTime	DateTime	1	see 4.3
cancelFlag	Boolean	1	see 4.3
messageGenerationTime	DateTime	0..1	see 4.3
Priority	typ007:Priority	0..1	see 4.3

4.5 MMCMasterMessage

Where parts of messages are updated dynamically, i.e. where parts are replaced or omitted without retransmission of complete messages, one MMCMasterMessage and a number of partial messages have to be transmitted. The TPEG Server has to be sure to send the MMCMasterMessage within a reasonable time frame, i.e. at least within the message expiry time of the message parts, as no decoding of the message is possible until the MMCMasterMessage has been received.

There shall only be one MMCMasterMessage per messageID in the transmission channel at any time.

The master message connects all MMCMessageParts through the multiPartMessageDirectory listing.

For each entry in the 'multiPartMessageDirectory', one message with a MMCMessagePart container shall be added to the transmission cycle. The messageID of these partial messages shall be identical with the one of the MMCMasterMessage.

Name	Type	Multiplicity	Description
messageID	IntUnLoMB	1	see 4.3
versionID	IntUnTi	1	see 4.3
messageExpiryTime	DateTime	1	see 4.3
cancelFlag	Boolean	1	see 4.3
messageGenerationTime	DateTime	0..1	see 4.3
priority	typ007:Priority	0..1	see 4.3
multiPartMessageDirectory	MultiPartMessageDirectory	1..255	List of MultiPartMessageDirectory entries, referencing all partial messages to be expected and their type indicating whether they are optional or mandatory. There shall be one entry per partial message in the multiPartMessageDirectory

4.6 MMCMessagePart

A message using the MMCMessagePart component is a partial message, which is transmitted separately from the MMCMasterMessage. This message shall not be interpreted before the master message and all mandatory partial messages defined therein have been received.

The messageID shall be identical to the messageID of the master message. All other attributes are completely independent.

Name	Type	Multiplicity	Description
messageID	IntUnLoMB	1	see 4.3

Name	Type	Multiplicity	Description
versionID	IntUnTi	1	see 4.3
messageExpiryTime	DateTime	1	see 4.3
cancelFlag	Boolean	1	see 4.3
messageGenerationTime	DateTime	0..1	see 4.3
priority	typ007:Priority	0..1	see 4.3
partID	IntUnTi	1	Unique ID of the partial message among all messages with the same messageID.
updateMode	mmc002:UpdateMode	1	Defines how the content of the partial message has to be merged with the overall message content.
masterMessageVersions	IntUnTi	0..255	This attribute may be used, if a partial message is only valid for special versions of the master message. If omitted, the partial message shall be applied to any, not expired, version of the master message.

5 MMC Datatypes

5.1 MultiPartMessageDirectory

Datastructure containing a partID and a partType attribute. This pair is used to reference a partial message and to specify whether a partial message has to be received or not before the presentation of the message to the user.

Name	Type	Multiplicity	Description
partID	IntUnTi	1	Unique ID of the partial message.
partType	mmc001:PartType	1	Defines the role of this partial message within the combined message.

6 MMC Tables

6.1 mmc001:PartType

This table holds the message part type instruction for the client device.

Code	Name	Comment
001	mandatory	The partial message is essential to present the overall message to the user.
002	additional	The partial message contains additional information. The overall message may be presented to the user, even if this partial message has not been received, or if it has expired.

6.2 mmc002:UpdateMode

This table holds the update mode instruction for the client device.

Code	Name	Comment
001	replaceTopLevel	See 4.1.4

Code	Name	Comment
002	rfu	Reserved for the 'replaceAttributesWhileKeepingStruture' feature, which may be added in a future version: The structure of the overall message is maintained. All attribute values that are contained in the components of the partial message shall replace the corresponding components attributes in the combined message.
003	rfu-	Reserved for the 'addInformation' feature, which may be added in a future version: The information is independent of the information contained in the overall message. Decoders shall add the contained information components to the combined message unless the message part with this partID has already been added to the combined message. In this case the already added information is replaced by updated versions.

Annex A
(normative)

Management Container, MMC,TPEG-Binary Representation

This Annex defines the TPEG binary representation (see TISA Specification, TPEG2 UML Conversion Rules: TPEG binary format) of the MMC that shall be used by all future TPEG applications.

A.1 Message Components

A.1.1 Generic Component Ids

As the MMC is a toolkit and no application no Generic Component Ids (GCIds) are defined here. The GCIds of the MMC components are assigned by the particular TPEG applications.

A.1.2 MMCTemplate

< *MMCTemplate*(x) > : =	: Abstract class, no instantiation,: see 4.3
< IntUnTi > (x),	: Identifier, is defined by the instance
< IntUnLoMB > (lengthComp),	: Length of component in bytes, excluding the id and length indicator
< IntUnLoMB > (lengthAttr),	: Length of attributes of this component in bytes
< IntUnLoMB > (messageID),	: see 4.3
< IntUnTi > (versionID),	: see 4.3
< DateTime > (messageExpiryTime),	: see 4.3
< BitArray > (selector),	: 1 byte containing 3 switches.
If (bit 0 of selector is set)	
< Boolean > (cancelFlag),	: see 4.3
If (bit 1 of selector is set)	
< DateTime > (messageGenerationTime),	: see 4.3
If (bit 2 of *selector* is set)	
< typ007:Priority > (priority);	: see 4.3

A.1.3 MessageManagementContainer

< MessageManagementContainer(x) < MMCTemplate(x) > > : =	
< IntUnTi > (x),	: Identifier, is defined by the instance
< IntUnLoMB > (lengthComp),	: Length of component in bytes, excluding the id and length indicator
< IntUnLoMB > (lengthAttr),	: Length of attributes of this component in bytes
< IntUnLoMB > (messageID),	: see 4.4
< IntUnTi > (versionID),	: see 4.4
< DateTime > (messageExpiryTime),	: see 4.4
< BitArray > (selector),	: 1 byte containing 3 switches.
If (bit 0 of selector is set)	
< Boolean > (cancelFlag),	: see 4.4

If (bit 1 of selector is set)	
< DateTime > (messageGenerationTime),	: see 4.4
If (bit 2 of *selector* is set)	
< typ007:Priority > (priority);	: see 4.4

A.1.4 MMCMasterMessage

< MMCMasterMessage(x) < MMCTemplate(x) > > : =	
< IntUnTi > (x),	: Identifier, is defined by the instance
< IntUnLoMB > (lengthComp),	: Length of component in bytes, excluding the id and length indicator
< IntUnLoMB > (lengthAttr),	: Length of attributes of this component in bytes
< IntUnLoMB > (messageID),	: see 4.5
< IntUnTi > (versionID),	: see 4.5
< DateTime > (messageExpiryTime),	: see 4.5
< BitArray > (selector),	: 1 byte containing 3 switches.
If (bit 0 of *selector* is set)	
< Boolean > (cancelFlag),	: see 4.5
If (bit 1 of *selector* is set)	
< DateTime > (messageGenerationTime),	: see 4.5
If (bit 2 of *selector* is set)	
< typ007:Priority > (priority),	: see 4.5
< IntUnLoMB > (n),	: Number of entries in array attribute, between 1 and 255.
n * **< MultiPartMessageDirectory >** \	: see 4.5
\(multiPartMessageDirectory) [1..255];	

A.1.5 MMCMessagePart

< MMCMessagePart(x) < MMCTemplate(x) > > : =	
< IntUnTi > (x),	: Identifier, is defined by the instance
< IntUnLoMB > (lengthComp),	: Length of component in bytes, excluding the id and length indicator
< IntUnLoMB > (lengthAttr),	: Length of attributes of this component in bytes
< IntUnLoMB > (messageID),	: see 4.6
< IntUnTi > (versionID),	: see 4.6
< DateTime > (messageExpiryTime),	: see 4.6
< BitArray > (selector),	: 1 byte containing 4 switches.
If (bit 0 of *selector* is set)	
< Boolean > (cancelFlag),	: see 4.6
If (bit 1 of *selector* is set)	
< DateTime > (messageGenerationTime),	: see 4.6
If (bit 2 of *selector* is set)	
< typ007:Priority > (priority),	: see 4.6
< IntUnTi > (partID),	: see 4.6
< mmc002:UpdateMode > (updateMode),	: see 4.6

If (bit 3 of *selector* is set) {	
< IntUnLoMB > (n),	: Number of entries in array attribute, between 0 and 255.
N * **< IntUnTi >** \	: see 4.6
\(masterMessageVersions) [0..255]	
};	

A.2 Datatypes

A.2.1 MultiPartMessageDirectory

< MultiPartMessageDirectory > : =	
< IntUnTi > (partID),	: Unique ID of the partial message.
< mmc002:PartType > (partType);	: Defines the role of this partial message within the combined message.

Annex B
(normative)

Management Container, MMC, TPEG-ML Representation

B.1 General

This annex contains the tpegML physical format (see TISA Specification, TPEG2 UML Conversion Rules: tpegML forma) representation of the Message Management Container.

B.2 XSD schema framing

The schema definition of this toolkit is maintained in an xsd file (see http://www.w3.org/XML/Schema), where the actual definitions are embedded in the following xml framing:

```
< ?xml version = ”1.0” encoding = ”UTF-8” ? >
< xs:schema   xmlns = ”http://www.tisa.org/TPEG/MMC_1_1”
            xmlns:xs = ”http://www.w3.org/2001/XMLSchema”
            targetNamespace = ”http://www.tisa.org/TPEG/MMC_1_1”
            xmlns:tdt = ”http://www.tisa.org/TPEG/TPEGDataTypes_0_0”
            elementFormDefault = ”qualified”
            attributeFormDefault = ”qualified”/>
< /xs:schema >
```

B.3 Element definition

B.3.1 MCCTemplate

```
< xs:complexType name = ”MMCTemplate” abstract = ”true” >
  < xs:sequence >
    < xs:element name = ”messageID” type = ”tdt:IntUnLoMB”/ >
    < xs:element name = ”versionID” type = ”tdt:IntUnTi”/ >
    < xs:element name = ”messageExpiryTime” type = ”tdt:DateTime”/ >
    < xs:element name = ”cancelFlag” type = ”tdt:Boolean”/ >
    < xs:element name = ”messageGenerationTime” type = ”tdt:DateTime” minOccurs = ”0”/>
    < xs:element name = ”priority” type = ”tdt:typ007_Priority” minOccurs = ”0”/>
  < /xs:sequence >
</xs:complexType>
```

This element is never directly instantiated in the xml data, instead one of the derived elements is always used. However, if an application wants to automatically enable multi part message management, the MMCTemplate is used as a data type in the UML, it then appears in the xml with the xs:type attribute containing the name of the actual instantiated child type.

B.3.2 MessageManagementContainer

```
< xs:complexType name = ”MessageManagementContainer” >
  < xs:complexContent >
    < xs:extention base = ”MMCTemplate” >
    < /xs:extention >
  < /xs:complexContent >
</xs:complexType>
```

B.3.3 MMCMasterMessage

```
< xs:complexType name = ”MMCMasterMessage” >
  < xs:complexContent >
    < xs:extention base = ”MMCTemplate” >
      < xs:element name = ”multiPartMessageDirectory” type = ”MultiPartMessageDirectory” maxOccurs = ”255”/>
    < /xs:extention >
  < /xs:complexContent >
</xs:complexType>
```

B.3.4 MMCMessagePart

```
< xs:complexType name = ”MMCMessagePart” >
  < xs:complexContent >
    < xs:extention base = ”MMCTemplate” >
      < xs:element name = ”partID” type = ”tdt:IntUnTi”/ >
      < xs:element name = ”updateMode” type = ”mmc002_UpdateMode”/ >
      < xs:element name = ”masterMessageVersions” type = ”tdt:IntUnTi” minOccurs = ”0” maxOccurs = ”255”/>
    < /xs:extention >
  < /xs:complexContent >
</xs:complexType>
```

B.4 Data types

B.4.1 MultipartMessageDirectory

```
< xs:complexType name = ”MultiPartMessageDirectory” >
  < xs:sequence >
    < xs:element name = ”partID” type = ”tdt:IntUnTi”/ >
    < xs:element name = ”partType” type = ”mmc001_PartType”/ >
  < /xs:sequence >
</xs:complexType>
```

B.5 MMC tables

B.5.1 mmc001_PartType

```
< xs:complexType name = ”mmc001_PartType” >
  < xs:attribute name = ”table” type = ”xs:string” fixed = ”mmc001_PartType” use = ”required”/>
  < xs:attribute name = ”code” use = ”required” >
    < xs:simpleType >
      < xs:restriction base = ”xs:unsignedByte” >
      < xs:minInclusive value = ”1”/ >
      < xs:maxInclusive value = ”2”/ >
      < /xs:restriction >
    < /xs:simpleType >
  < /xs:attribute >
</xs:complexType>
```

B.6 mmc002_UpdateMode

```
< xs:complexType name = ”mmc002_UpdateMode” >
  < xs:attribute name = ”table” type = ”xs:string” fixed = ”mmc002_UpdateMode” use = ”required”/>
  < xs:attribute name = ”code” use = ”required” >
    < xs:simpleType >
      < xs:restriction base = ”xs:unsignedByte” >
      < xs:minInclusive value = ”1”/ >
      < xs:maxInclusive value = ”1”/ >
      < /xs:restriction >
    < /xs:simpleType >
  < /xs:attribute >
</xs:complexType>
```

B.7 tpegML samples

B.7.1 Direct instantiation

This xml snipplet is an example on how an MMC container may look in a TPEG application if a derived class is directly used.

Application XSD example:

```
...
< xs:element name = ”mmt” type = ”MessageManagementContainer” />...
...
```