TECHNICAL SPECIFICATION

ISO/TS 18234-9

First edition
2013-10-15

Intelligent transport systems — Traffic and travel information via transport protocol experts group, generation 1 (TPEG1) binary data format —

Part 9: Traffic event compact (TPEG1-TEC)

Systèmes intelligents de transport — Informations sur le trafic et le tourisme via les données de format binaire du groupe d'experts du protocole de transport, génération 1 (TPEG1)

Partie 9: Événement trafic compact (TPEG1-TEC)

Reference number
ISO/TS 18234-9:2013(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

In other circumstances, particularly when there is an urgent market requirement for such documents, a technical committee may decide to publish other types of normative document:

— an ISO Publicly Available Specification (ISO/PAS) represents an agreement between technical experts in an ISO working group and is accepted for publication if it is approved by more than 50 % of the members of the parent committee casting a vote;

— an ISO Technical Specification (ISO/TS) represents an agreement between the members of a technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/PAS or ISO/TS is reviewed after three years in order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn. If the ISO/PAS or ISO/TS is confirmed, it is reviewed again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.

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.

ISO/TS 18234-9 was prepared by the European Committee for Standardization (CEN) Technical Committee CEN/TC 278, *Road transport and traffic telematics,* in collaboration with ISO Technical Committee ISO/TC 204, *Intelligent transport systems*, in accordance with the Agreement on technical cooperation between ISO and CEN (Vienna Agreement).

ISO/TS 18234 consists of the following parts, under the general title *Intelligent transport systems — Traffic and travel information via transport protocol experts group, generation 1 (TPEG1) binary data format*:

— *Part 1: Introduction, numbering and versions (TPEG1-INV)*

— *Part 2: Syntax, semantics and framing structure (TPEG1-SSF)*

— *Part 3: Service and network information(TPEG1-SNI)*

— *Part 4: Road Traffic Message application (TPEG1-RTM)*

— *Part 5: Public Transport Information (PTI) application*

— *Part 6: Location referencing applications*

— *Part 7: Parking information (TPEG1-PK1)*

— *Part 8: Congestion and travel-time application (TPEG1-CTT)*

— *Part 9: Traffic event compact (TPEG1-TEC)*

— *Part 10: Conditional access information (TPEG1-CAI)*

— *Part 11: Location Referencing Container (TPEG1-LRC)*

Introduction

TPEG technology

TPEG technology uses a byte-oriented data stream format, which may be carried on almost any digital bearer with an appropriate adaptation layer. TPEG-messages are delivered from service providers to end-users and used to transfer information from the database of a service provider to an end-user's equipment.

The brief history of TPEG technology development dates back to the European Broadcasting Union (EBU) Broadcast Management Committee establishing the B/TPEG project group in autumn 1997 with the mandate 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 are 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.

One year later in December 1998, the B/TPEG group produced its first EBU specifications. Two Technical Specifications were released. ISO/TS 18234-2, described the Syntax, Semantics and Framing Structure, which is used for all TPEG applications. ISO/TS 18234-4 (TPEG-RTM) described the first application, for Road Traffic Messages.

Subsequently, CEN/TC 278/WG 4, in conjunction with ISO/TC 204, established a project group comprising the members of B/TPEG and they have continued the work concurrently since March 1999. Since then two further parts were developed to make the initial complete set of four parts, enabling the implementation of a consistent service. ISO/TS 18234-3 (TPEG-SNI) describes the Service and Network Information Application, which should be used by all service implementations to ensure appropriate referencing from one service source to another. ISO/TS 18234-1 (TPEG-INV), completes the series, by describing the other parts and their relationship; it also contains the application IDs used within the other parts. Additionally ISO/TS 18234-5 the Public Transport Information Application (TPEG-PTI) and ISO/TS 18234-6 (TPEG-LRC), were developed.

This Technical Specification adds another powerful application for the ISO 18234 series allowing detailed road event information to be encoded and transmitted to the user. It was developed specifically to satisfy messaging for Navigation System clients and designed to provide cause and effect in the Road Traffic events information domain. This Technical Specification includes new advanced message management and new datatypes as specified in the annexes.

TPEG applications are developed using UML modelling and a software tool is used to automatically select content which then populates this Technical Specification. Diagrammatic extracts from the model are used to show the capability of the binary coding in place of lengthy text descriptions; the diagrams do not necessarily include all relevant content possible.

This Technical Specification describes the binary data format of the on-air interface of the Traffic Event Compact application, (TPEG-TEC) with the technical version number TPEG-TEC_3.0/001.

TEC Model

The basic concept behind the TEC model is that a traffic situation is described by a primary information structure describing the most important information to present to a driver and secondary descriptions of the causes and/or more details. This model enables a traffic editor to describe complex events in a modular way allowing TEC-based system and mobile terminals to present the message as it was intended by the editor. This can be either graphical, textual, voice or a combination of those.

Next to the above mentioned requirement, it is very important to design an efficient coding scheme:

— there will be terminal devices with limited resources;

— to be able to extract those elements early that are relevant to the driver's route, it is key to use the available bandwidth efficiently.

Intelligent transport systems — Traffic and travel information via transport protocol experts group, generation 1 (TPEG1) binary data format —

Part 9:
Traffic event compact (TPEG1-TEC)

1 Scope

This Technical Specification defines the TPEG application Traffic Event Compact (TEC). It has been specifically designed to support information about traffic events, e.g. road works, traffic jams. A specific form of traffic event are local hazard warnings, which as safety-related messages, are sent with high priority to assist a driver in encountering dangerous situations (e.g. black-ice, accident behind curves, obstacles on road) unexpectedly.

Generally, TEC focuses on the following requirements:

— ensuring travel safety for the driver;

— enabling the calculation of alternative routes;

— avoiding delays (e.g. traffic jams);

— warning the driver of obstructions on route;

— informing the driver of infrastructural problems (e.g. closed petrol stations, non-functioning emergency phones).

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/TS 18234-2, *Intelligent transport systems — Traffic and travel information via transport protocol experts group, generation 1 (TPEG1) binary data format — Part 2: Syntax, semantics and framing structure (SSF)*

ISO/TS 18234-11, *Intelligent transport systems — Traffic and travel information via transport protocol experts group, generation 1 (TPEG1) binary data format — Part 11: Location Referencing Container (TPEG1-LRC)*

3 Terms and definitions

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

3.1
local hazard warning
specific form of traffic events which being safety-related messages are sent with high priority to assist a driver from encountering dangerous situations

3.2
location referencing container
concept applied to the grouping of all the location referencing elements of a TPEG-Message

3.3
location referencing
method to provide information which allows a system to accurately identify a location

NOTE The content of a location reference allows the location to be presented in a plain-language manner to the end-user (e.g. text, speech or icons), and also to be used for navigational purposes, for example, for map-based systems.

3.4
startTime
beginning time of a traffic event

3.5
stopTime
end time of a traffic event

4 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

CEN	Comité Européen de Normalisation
EBU	European Broadcasting Union
LRC	Location Referencing Container
OSI	Open Systems Interconnection
RTM	Road Traffic Message (see ISO/TS 18234-4)
TLV	Tag length value; a coding method
TPEG	Transport Protocol Expert Group
WGS 84	World Geodetic System 1984

5 Application framing and signalling

5.1 Application identification

The word 'application' is used in the TPEG specifications to describe specific subsets of the TPEG structure. An application defines a limited vocabulary for a certain type of messages, for example parking information or road traffic information. Each TPEG application is assigned a unique number, called the Application IDentification (AID). An AID is defined whenever a new application is developed and these are all listed in CEN ISO/TS 18234-1.

The application identification number is used within the TPEG-SNI application (ISO/TS 18234-3:2006) to indicate how to process TPEG content and facilitates the routing of information to the appropriate application decoder.

For TPEG TEC, AID has the value five (5).

5.2 Version number signalling

Version numbering is used to track the separate versions of an application through its development and deployment. The differences between these versions may have an impact on client devices.

The version numbering principle is defined in CEN ISO/TS 18234-1.

Figure 1 shows the current version numbers for signalling TEC within the SNI application.

major version number	3
minor version number	0

Figure 1 — Current version numbers for signalling of TEC

5.3 Application framing

TEC makes use of the "Service component frame with severity and message count" according to Annex A, section A.3.2.6.2.4. For explanatory purpose this is repeated here.

< **ServCompFramePrioritisedCountedProtected**>:=	: CRC protected service component frame with group priority and message count
<**ServCompFrameHeader**>(header),	: Component frame header as defined in A.3.2.6.
<**typ007:Priority**>(groupPriority),	: group priority applicable to all messages in the ApplicationContent
<**IntUnTi**>(messageCount),	: count of messages in this ApplicationContent
external <**ApplicationContent**>(content),	: actual payload of the application
<**CRC**>(dataCRC);	: CRC starting with first byte after the header

Within the service component frame, the ApplicationContent is defined as follows:

<**ApplicationContent>:=**	: application content
messageCount * <**TECMessage**>(msg);	: Any number of any TEC message components

5.4 Application specific constraints

TPEG-TEC requires the use of a fixed order of components, unlike other TPEG applications. The order is shown in Figure 2; the first component is *MessageManagement.* If the message is not a cancel message then the *MessageManagement* component shall be followed by the *Event* component and this is followed by the *LocationReferencing* component.

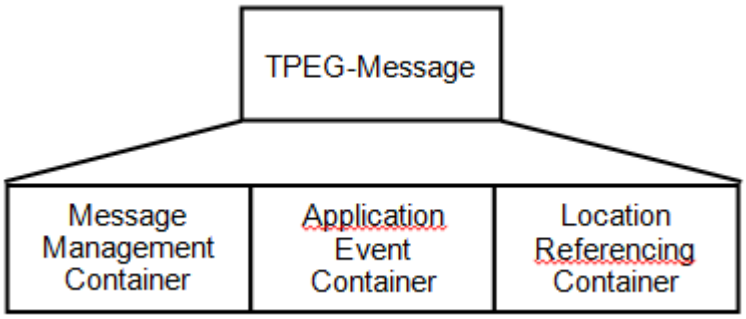

Figure 2 — Every TPEG message is constructed of three containers

Within the *Event* component one or more *Cause* components shall come first, followed by one or more *Advice* components, and so on. Components of the same type shall immediately follow each other, i.e. they shall not be spread over a TECMessage.

Extendibility

The requirement of a fixed component order does not forbid the extension of TEC generally. In case of future extensions, new components may be inserted or existing components may be replaced by new ones without loosing backward compatibility. That means, a TEC decoder must be able to detect and skip unknown components. But, it is not allowed that multiple components of the same type which belong to same upper component are spread over the message.

Example
The *Advice* component is replaced by *BetterAdvice* having an own component id. A *WeatherSituation* component is inserted after *Advice* component.

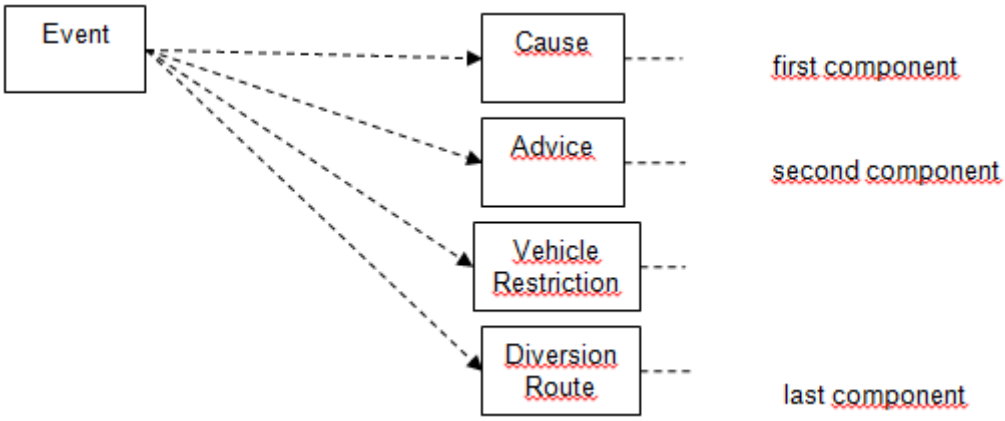

Figure 3 — Example for extension; original component model

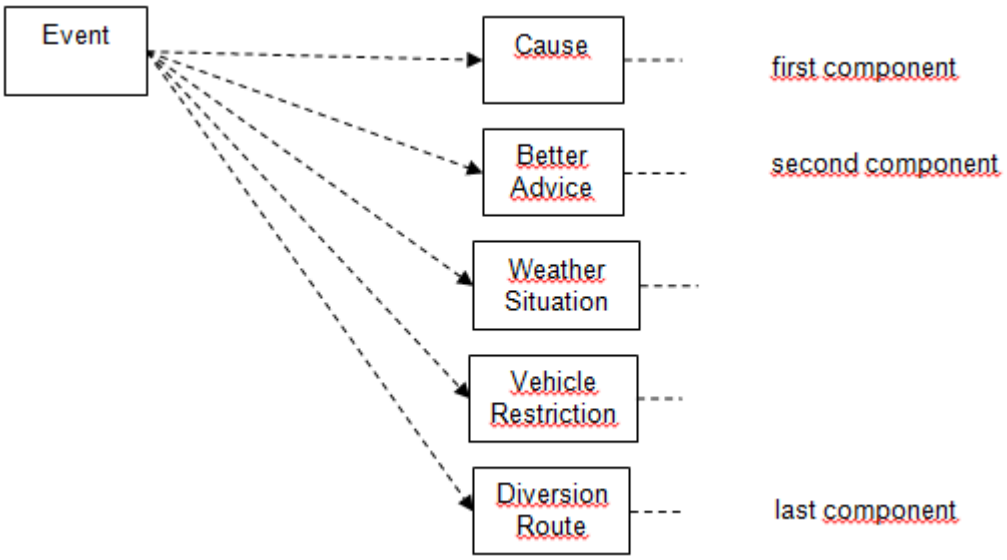

Figure 4 — Example for extension; *Advice* replaced by *BetterAdvice* and *WeatherSituation* added

In any case multiple components of the same type which belong to same upper component shall not be spread over the message.

Example (not allowed)
An *Event* component has two *Cause* components. The first one is followed by an *Advice* component and the last one related to the same *Event* component is a *Cause* component again. This is forbidden.

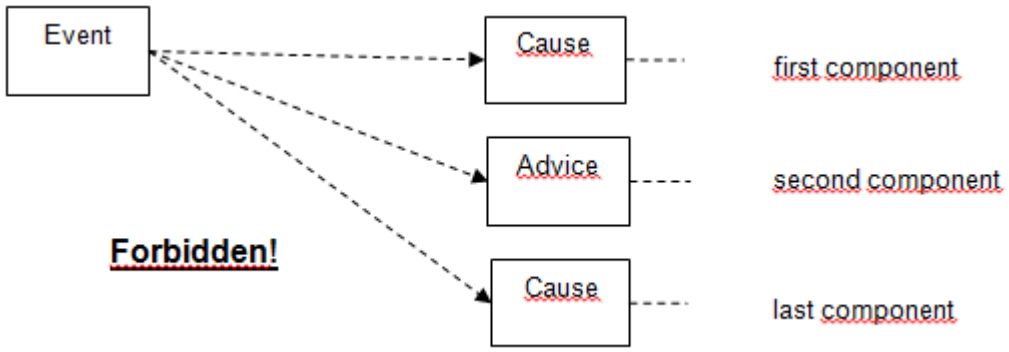

Figure 5 — Forbidden ordering of same components

Note: If general TPEG Toolkit definitions (e.g. ISO 18234 Part 2) deviate from the definition in this Part, the definition given herein takes precedence.

6 Message components

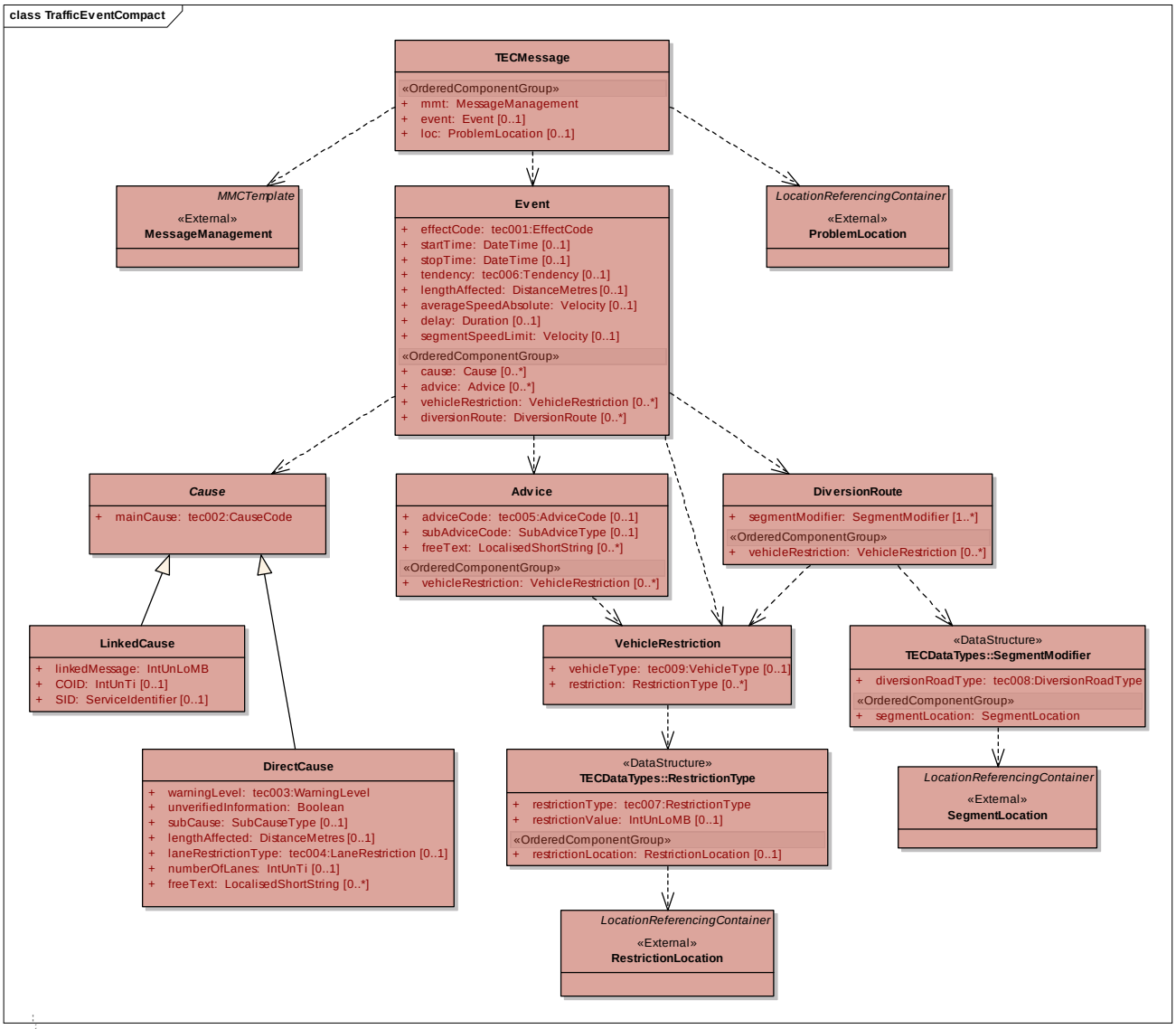

Figure 6 — UML Model of TPEG-TEC

6.1 List of Generic Component Ids

Name	Id
TECMessage	0
MessageManagement	1
ProblemLocation	2
Event	3
DirectCause	4
LinkedCause	5
Advice	6
VehicleRestriction	7
DiversionRoute	8
RestrictionLocation	9
SegmentLocation	10

6.2 TECMessage

A TPEG-TEC Message may include:

— one management container with management information related to the overall message (ID and version, expiry time);

— one event container with one traffic flow effect and optional one or more causes with additional information;

— one location container with the location reference for the overall traffic message.

The message management container is mandatory, the event- and location container are optional.

Event and location container are modelled optionally because cancel messages do not contain these elements: Cancellation messages shall not include an event and location container whereas normal messages (cancelFlag = false) shall include exactly one event and one location container.

<**TECMessage<Component(0)>**>:=	:Traffic Event Compact Message Component
<IntUnTi>(0),	: id is unique within the scope of the application.
<IntUnLoMB>(compLengthInByte),	: length of the component counted in bytes.
<IntUnLoMB>(attributeBlockLengthInByte);	: length of the attribute block in bytes.
<**MessageManagement**>(mmt),	: {mandatory} Message management container
m * <**Event**>(event)[0..1],	: {optional} Event data
m * <**LocationContainer**>(loc)[0..1];	: {optional} Location data

6.2.1 MessageManagement

The MessageManagement component is a placeholder for the MessageManagementContainer (MMC) as specified in Annex B. It assigns the traffic event compact (TEC) application specific local component ID for the MMC container. All component IDs within the MMC container are local to the MMC toolkit. The MMC contains all and only information related to message management.

<**MessageManagement(1)**>:=	: Message management data
external <**MessageManagementContainer**(1)>;	: see MMC specification

Message Generation Systems shall ensure that the information given in the MMC allows unambiguous interpretation over the whole time a message is valid. It is particularly important to recognise that client devices are likely to suffer from non-continuous transmission channels as typically encountered in broadcast systems suffering intermittent RF performance.

TEC shall only use the monolithic message management. Multipart message management must not be used.

6.2.2 Instances of Location Containers

The ProblemLocation**,** RestrictionLocation and SegmentLocation component are instances of the Location Referencing Container (LRC) as specified in ISO/TS 18234-11. It assigns the traffic event compact (TEC) application specific local component ID for the LRC. The different places of the LRC components define different ids to enable *separated evolvement over time* of these components in the specification. All component IDs within the LRC are local to the LRC toolkit.

<ProblemLocation<**LocationReferencingContainer(2)>>: =**	: the information given by the TEC elements (e.g. effect, cause , advice) are related to this ProblemLocation; LRC container
External <**LocationReferencingContainer**(2)>;	: see LRC specification

<RestrictionLocation \ \ <LocationReferencingContainer(9)>>:=	: RestrictionLocation is used by Advice component or DiversionRoute component; LRC container
external <**LocationReferencingContainer**(9)>;	: see LRC specification

<SegmentLocation \ \ <LocationReferencingContainer(10)>>:=	: SegmentLocation is used by DiversionRoute component LRC container
external <**LocationReferencingContainer**(10)>;	: see LRC specification

6.2.3 Event

The Event component with its subordinated component Cause describes in general the impact on the traffic flow and the related cause.
For example: 'Stationary Traffic, (due to) Narrow Lanes

Rules

— For a single event it should be possible to distinguish between the effect that describes an acute impairment of the traffic flow (e.g. stationary traffic) and the cause (e.g. roadworks). The latter can be seen as the reason for the traffic flow effect determined by the attribute effectCode. Furthermore the sub-component Cause can be used to inform or warn the driver for a special situation (e.g. oil on the road). The following combinations are possible:

1) a message contains only the Event component with its attribute effectCode that describes the impairment of the traffic flow directly;

2) the previous described element Event is expanded by one or more sub-components Cause. If a specific traffic flow is not available or shall not be given, the effectCode must set to ‘unknown’.

— A (real) cause can be represented in the message by either a 'DirectCause' or a 'LinkedCause', but not both.

— Causes and subCauses- CauseCodes can be further specified by subCauseCodes. Each cause having more than one assigned subCause has an own subCause list numbered tec1xx where ‘xx’ is the code from tec002. Simple terminals must support at least all causes, terminals with more memory might support subCauses. In the latter case sub-causes should replace the (Main)cause completely. With other words: it is not necessary to combine causes and subCause to a meaningful grammar.

— Advices and subAdvices - The principle for Causes and subCauses are also usedfor Advice, with the exception that the subAdvice tables are numbered tec2xx.

— The component VehicleRestriction can be used to set a filter for different vehicles.

— Table tec001 and tec002 including all subCauseTables (tec1xx) are mainly taken from TMC-Event list (refer to ISO/TS 14819-2:2003).

startTime and stopTime

Describe the beginning and the end of a traffic event. The terminal should use these values to calculate the validity of the event. This validity can be guaranteed by the provider only if the transmission channel is reliable, i.e. the messages expiry time does not elapse. If the message expiry time elapses, the validity can not be guaranteed and it is recommended to ignore the message. Finally, the use of the message is up to the device.

LengthAffected and Length in ProblemLocation (LocationReferencingContainer)

The Length in the location referencing container ProblemLocation, called ProblemLength, describes the overall length of the event.

If LengthAffected is not defined within the Event component or any Cause of the Event Container, the Length of ProblemLocation has to be taken.

If the LengthAffected is given by an Event or Cause component it has to be taken only for the specific component.

In any case, the LengthAffected must not be greater than the ProblemLength.

If, in case of TMC location coding, neither LengthAffected nor ProblemLength is given, the total length between Primary and Secondary Location has to be taken.

The reference point for the EffectCode and all Causes is defined by the ProblemLocation. If *DLR1LocationReference* or *TMCLocationReference* method is used, the reference point is defined as follows:

— the Start Location, in case of TMCLocationReference precise location referencing (refer to ISO/TS 14819-3:2004),

— the Secondary Location, in case of TMCLocationReference without precise location referencing (refer to ISO/TS 14819-3:2004), or

— the first location point, in case of DLR1LocationReference (refer to ISO 17572-3).

Some examples are shown in section “LinkedCause”.Other location referencing methods may also be used even if they are not described here.

Rounding of speed information

Speed information is always given in metres per seconds (m/s) since the general TPEG data type 'Velocity' is used. Depending on customer requirements the terminal has to convert and round these values.

EXAMPLE

m/s	km/h (exact)	km/h (rounded, steps of 5)	mph (exact)	mph (rounded, steps of 5)
0	0,0	0	0,0	0
1	3,6	5	2,24	0
2	7,2	5	4,49	5
3	10,8	10	6,73	5
4	14,4	15	8,98	10
5	18	20	11,22	10
6	21,6	20	13,47	15
7	25,2	25	15,71	15
8	28,8	30	17,96	20
9	32,4	30	20,20	20
10	36	35	22,44	20
11	39,6	40	24,69	25
12	43,2	45	26,93	25
13	46,8	45	29,18	30
14	50,4	50	31,42	30

The following formulae are used to calculate the values listed above in the table:

For steps of 5 km/h (0, 5, 10, 15, 20, …)

— ROUND((v*3,6)/5)*5

For steps of 5 mph (0, 5, 10, 15, 20, …)

— ROUND((v*3,6)/1,604)/5)*5

'ROUND' in this case is a typical mathematical function to round values without fractional digits.

The Event component is coded as follows:

<**Event<Component(3)>>:=**	
<**IntUnTi**>(3),	: Identifier = 3
<**IntUnLoMB**>(lengthComp),	: Length of component in bytes, excluding the id and length indicator
<**IntUnLoMB**>(lengthAttr),	: Length of attributes of this component in bytes
<**tec001:EffectCode**>(effectCode),	: Describes the impairment of the traffic flow
<**BitArray**>(selector),	: 1 byte containing 7 switches.
If (bit 0 of selector is set)	
<**DateTime**>(startTime),	: Date and time at which an event began or is scheduled to begin (used for presentation to the end-user). If startTime is missing first time of reception is used instead.

If (bit 1 of selector is set)	
<**DateTime**>(stopTime),	: Date and time at which an event, or status information, ended or is scheduled to end (used for presentation to the end-user). If stopTime is missing stopTime is set to startTime plus default duration, which is defined per main cause. In case of multilple causes the highest default duration shall be taken.
If (bit 2 of selector is set)	
<**tec006:Tendency**>(tendency),	: Tendency is related to traffic flow. It is not a forecast. compared with TMC it is not related to the jam length.
If (bit 3 of selector is set)	
<**DistanceMetres**>(lengthAffected),	: Length of the event in metres.
If (bit 4 of selector is set)	
<**Velocity**>(averageSpeedAbsolute),	: Average speed in m/s at the given location. It is recommended to use this value for calculation of the route and the referring arrival time.
If (bit 5 of selector is set)	
<**IntUnLoMB**>(delay),	: Delay in minutes. Only applicable to point locations, i.e. at border.
If (bit 6 of selector is set):	
<**Velocity**>(segmentSpeedLimit),	: Maximum speed in m/s. Shall be used as adminstrativ speed limit for re-routing, but not to display or warn the driver because qualitiy cannot be guaranteed.
m * <**Cause**>[0..*],	: A Cause might be only represented by instances of Cause, either as LinkedCause or as DirectCause. Within a single message, the same cause code shall not be used for linked and direct causes. Multiplicity of Cause is then 0…infinity
m * <**Advice**>[0..*],	: m represents the number of occurrences, between 0 and infinity.
m * <**VehicleRestriction**>[0..*],	: m represents the number of occurrences, between 0 and infinity.
m * <**DiversionRoute**>[0..*];	: m represents the number of occurrences, between 0 and infinity.

6.2.3.1 Cause

The cause template specifies the additional interface including a mandatory CauseCode for all instances.

<**Cause(x)<Component(x)>>**:=	: Template for causes
<**IntUnTi**>(id),	
<**IntUnLoMB**>(lengthComp),	: Length of component in bytes, excluding the id and length indicator
<**IntUnLoMB**>(lengthAttr),	: Length of attributes of this component in bytes
<**tec002:CauseCode**>(mainCause);	: Main categorization of the cause according to table tec002

6.2.3.2 DirectCause

The component DirectCause can be used to describe the reason for traffic congestion in general. The main reason for the separation of causes and the effect is that the real traffic situation can be described in meaningful way to the driver.

EXAMPLE Road closed (effectCode=7), (due to) objects on the road (causeCode 10).

<**DirectCause<Cause(4)>>:=**	
<**IntUnTi**>(4),	: Identifier
<**IntUnLoMB**>(lengthComp),	: Length of component in bytes, excluding the id and length indicator
<**IntUnLoMB**>(lengthAttr),	: Length of attributes of this component in bytes
<**tec002:CauseCode**>(mainCause),	: Main categorization of the cause according to table tec002
<**tec003:WarningLevel**>(warningLevel),	: The level informative should be used for all traffic events, which may influence the drivers' route in any way and might require normal attention from the driver. The danger levels 1 to 3 should only be used for really dangerous situations, e.g. ghost-driver. The danger levels 1 to 3 in combination with unverifiedInformation = True should be used in case of an unverified danger. For example, a traffic management centre receives a call from a private "jam buster" that there is a "ghost-driver", but the event is still not confirmed by the police.
<**BitArray**>(selector),	: 1 byte containing 6 switches.
If (bit 0 of selector is set)	
<**Boolean**>(unverifiedInformation),	: If element is set to 1 the given information has not been verified.
If (bit 1 of selector is set)	
<**SubCauseType**>(subCause),	: Carries the value in the sub cause table defined by the mainCauseCode.
If (bit 2 of selector is set)	
<**DistanceMetres**>(lengthAffected),	: Length of the event in metres.
If (bit 3 of selector is set)	

<**tec004:LaneRestriction**>(laneRestrictionType),	: Specifies whether lanes are closed or opened.
If (bit 4 of selector is set)	
<**IntUnTi**>(numberOfLanes),	: Specifies how many lanes are closed or opened. If this element is not given, the plural form shall always be used.
If (bit 5 of selector is set){	
<**IntUnLoMB**>(n),	: Number of entries in array attribute, between 0 and infinity.
n * <**LocalisedShortString**>(freeText)	
};	

6.2.3.3 LinkedCause

Specifies a link to a message providing more details about the cause, e.g. for a divergent location. A link to another message is uniquely specified by the combination of ServiceID, ContentID, ApplicationID, and messageID. The linked message can be found in the service component where those attributes are equal to the values given in that component: linkedMessage, COID, SID. The ApplicationID points always to the TEC Application and is therefore not explicit given in this component.

There are two variants to encode so called 'linked events' where an effect and one or more cause(s) belong together.

— One method is to combine both the effect and the cause by encoding in only one message. This is called a 'DirectCause' and is to be used in situations where no detailed information is given, e.g. about the location or the length of the linked cause.

— The other method is called 'LinkedCause' and it must be used in cases where detailed information are explicitly given. For example: The location of the cause differ from that of the effect and additional speed information is available. Then, complete description of the traffic situation is spread over two or more messages.

Alternative

— The cause and effect occur on the same location (more precisely the most downstream location, refer to example 1 below). In this case the cause will be a DirectCause and described in the same message.

— The causes and effect do not share the same location. For this situation two messages will be created: one message describing the effect, and one describing the cause, with unknown traffic flow effect. The effect message will use the "cause" message's ID to signal its "LinkedCause".

Rules
Effect and cause(s) must be split into two or more messages (linked cause) if:

— they are from different providers, e.g. effect is provided by the police and cause by a private service provider;

— different update rates shall be used;

— the most downstream location (in TMC it is the primary location) of effect and cause is different;

— the same situation requires two traffic flow effects, e.g. different speed limits for cars and lorries.

Further constraints:

— One real cause may either be represented as a linked or as a direct cause. It is not allowed to describe one situation using a direct and a linked cause at the same time.

— If a 'linked cause' is used, it is not allowed to reset causes which have been set in the preceding messages belonging to the same traffic situation.

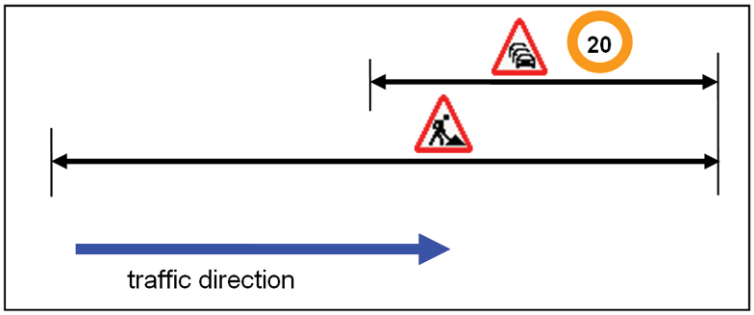

Figure 7 — Example 1 for a traffic situation, which can be coded by use of a DirectCause

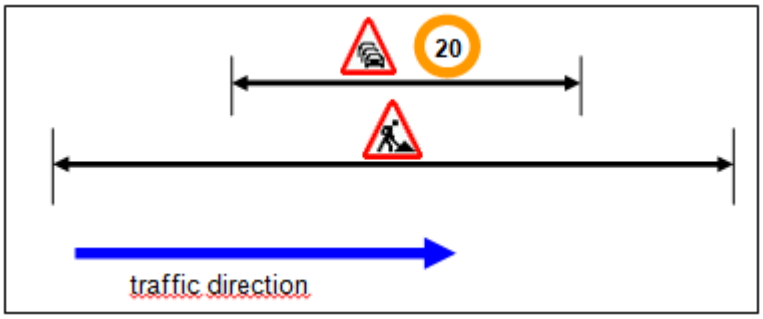

Figure 8 — Example 2 for a traffic situation, where a LinkedCause is necessary

Coding schemes for the given examples (simplified)

EXAMPLE 1

trafficflow_effect	effect_code	“stationary traffic"
	length_affected (m)	5000
	Road_performance	average_speed_abs=20
direct_cause	cause_code	“roadworks"
	length_affected (m)	10000
	severity	Informative

EXAMPLE 2

First message:

trafficflow_effect	effect_code	“stationary traffic"
	length_affected (m)	5000
	Road_performance	average_speed_abs=20
direct_cause	cause_code	“accident"
	severity	Informative
linked_cause	cause_code	“roadworks"
	linked_message	ID of 2nd message

NOTE Direct cause ‘accident’ is added above; in order to point out that a message may consist of direct_cause(s) and linked_cause(s).

Second message:

trafficflow_effect	effect_code	"traffic flow unknown"
	length_affected (m)	10000
	Road_performance	segment_speed=60
direct_cause	cause_code	"roadworks"
	length_affected (m)	10000
	severity	Informative

NOTE Additonal information ‘segment_speed’ is not shown in the figure above.

<LinkedCause<Cause(5)>>:=	
<**IntUnTi**>(5),	: Identifier
<**IntUnLoMB**>(lengthComp),	: Length of component in bytes, excluding the id and length indicator
<**IntUnLoMB**>(lengthAttr),	: Length of attributes of this component in bytes
<**tec002:CauseCode**>(mainCause),	: Main categorization of the cause according to table tec002
<**IntUnLoMB**>(linkedMessage),	: Contains a messageID as pointer to a message. If COID and SID is not given, the linked message is contained in the component stream of the current service.
<**BitArray**>(selector),	: 1 byte containing 2 switches.

If (bit 0 of selector is set)	
<**IntUnTi**>(COID),	: In case of absence, the linked message can be found in the component stream that has the same COID as the linking message.
If (bit 1 of selector is set)	
<**ServiceIdentifier**>(SID);	: In case of absence, the linked message can be found in the same service as the linking message is in.

6.2.3.4 Advice

The advice contains recommendations or prohibitions for the driver either spoken or in computed form e.g. diversion routes.

<Advice<Component(6)>>:=	
<**IntUnTi**>(6),	: Identifier
<**IntUnLoMB**>(lengthComp),	: Length of component in bytes, excluding the id and length indicator
<**IntUnLoMB**>(lengthAttr),	: Length of attributes of this component in bytes
<**BitArray**>(selector),	: 1 byte containing 2 switches.
If (bit 0 of selector is set)	
<**tec005:AdviceCode**>(adviceCode),	
If (bit 1 of selector is set)	
<**SubAdviceType**>(subAdviceCode),	
If (bit 2 of selector is set) {	
<**IntUnLoMB**>(n),	: Number of entries in array attribute, between 0 and infinity.
n * <**LocalisedShortString**>(freeText)	
}	
m * <**VehicleRestriction**>[0..*];	: m represents the number of occurrences, between zero and infinity.

6.2.3.4.1 VehicleRestriction

The given element containing the restriction is restricted to a special vehicle type, e.g. only for trucks. In case that vehicleType is not present the subsequent RestictionType shall be applied to all types of vehicles.

<VehicleRestriction<Component(7)>>:=	
<**IntUnTi**>(7),	: Identifier
<**IntUnLoMB**>(lengthComp),	: Length of component in bytes, excluding the id and length indicator

<**IntUnLoMB**>(lengthAttr),	: Length of attributes of this component in bytes
<**BitArray**>(selector),	: 1 byte containing 2 switches.
If (bit 0 of selector is set)	
<**tec009:VehicleType**>(vehicleType),	
If (bit 1 of selector is set) {	
<**IntUnLoMB**>(n),	: Number of entries in array attribute, between 0 and infinity.
N * <**RestrictionType**>(restriction)	: The VehicleRestriction (even if meaning “all cars”) can be specified further by additional attributes, e.g.: weight in kilograms or height or the given destination with the help of Restriction. For example: “NoWintertyre” means vehicles without winter tyre “width greater 300” means that the event or the closure affects only vehicles with a width greater than 3 metres. “with destination in a given area” means destination filter is selected and the area is attached to this restriction.
};	

6.2.3.5 DiversionRoute

Diversion route specifies one or more diversions valid in case that the location of the message is touched by the undisturbed route.

6.2.3.5.1 Description of Creating and Applying Diversions

A recommendation in conjunction to a traffic problem in its easiest form consists of just an advice like “avoid area“ or “follow signposted diversion“. In addition to that with the DiversionRoute-Component a message can express explicit diversions to the given traffic problem. The possibilities of expression of diversion information in TEC are far more than earlier TMC messages did allow.

TEC defines diversions as being several segments having different values of time delays against normal road going from “closed road“ to “bypass“. This causes parts of the diversion influencing more the routing than other in either reduction or increasing the costs of a segment.

In addition to that it might also be useful to distinguish between different categories of the traffic being targeted by a diversion. This allows e.g. to define different detours for lorries as for cars. The different attributes for differentiation are vehicle type and further restriction types to those vehicles, same as the destination chosen by individual vehicles. The destination is already today in use. Diversions are suggested differently for both road sides for example. This causes expressing to different diversions one e.g. for “northern“ and one for “southern“ (more precise definition follows).

Another use case for diversions is not only to suggest the route to be taken but also to discourage segments because of given circumstances. For example: A road element is blocked due to road works. The road elements before and behind the blockage may only be used by local traffic, which lets a navigation system plan a route by avoiding the parts of the road network also.

The typical way for such diversions would be that a traffic center or service provider defines all necessary diversion information as one strategy being used for this specific traffic problem. Because it is relying on the existing road network, a strategy might be used more than once for this particular place.

In general, the diversion is a multi layer route cost influence for the given parts of the road network (segments). A segment defines one of the following attributes for this purpose:

— traffic problem segments;

— accesses to a possible bypass segments;

— bypass segments;

— not recommended segments.

All segments are in relation to each other; i.e. a vehicle not influenced by the traffic problem shall not be influenced by the bypass delivered, because than too many vehicles would route through the diversion unnecessarily.

6.2.3.5.2 Procedure for Coding a Diversion Strategy

A diversion shall in all cases be provided together with a problem location. If a diversion was provided without a problem location, other vehicles within that area might needlessly be routed to the diversion. This phenomenon is called the “suck effect“ and is undesirable because the diversion may become overloaded.

To prevent suck effects the diversions in a vehicle should only be taken into account if all of the following triggers do apply for the actual situation:

1) the actual route calculated does include at least one part of the given problem location;

2) the filter criteria matches to the conditions of the vehicle.

Another suck effect is caused by a fact that a diversion normally consists of a bypass being in a given distance to the problem location. To guide the traffic from the original route to the bypass mostly some additional streets are concatenated to that part of the bypass being in real parallel to the problem location. These parts of the diversion are called access roads. Access roads should only be taken into account from vehicles following the diversion from the original route. Vehicles being somewhere else, approaching the problem locations road should not take the access roads into account with the same weight but the bypass.

A service provider’s possibility shall be given to be able to code all of the following statements:

— dead end at road works, free for local access only;

— street not recommended because of limited capacity prefer different possibilities if available;

— for a given problem location take this bypass as replacement;

— for a given problem location take this access roads to reach and leave the bypass;

— this road should not be taken into account for routing, unless the destination is not reachable with other ways.

With coding these values no routing cost value and no time delay is given but the relative relation to each other. Because the road network and the routing function are expected to be different in all vehicles, no default value would produce a consistent behavior in a vehicles receiver system. Instead of that, the expected behavior visible to the user is described in addition to the relative route calculation value.

for 1.) LA = Limited Access: The segment shall be used for local traffic only. If possible plan a route avoiding this segment.
-> Routing costs for that segment should be raised by a factor much greater than one

for 2.) NR = Not Recommended: The segment should be avoided if a second alternative is close to it. Service provider's intent is to reduce traffic volume due to reduced capacity of that segment.
-> Routing costs for that segment should be raised by a factor greater than one

for 3.) BP = Bypass: The segment is the part of the road network being in parallel to the problem location. In case that original route uses the problem location, this segment shall be taken into account as subsidiary in such way that also better, local alternatives are not chosen.
-> Routing costs for that segment should be reduced by a factor much smaller than one.

for 4.) AR = Access Road: This segment is part of the access to or away from the bypass. It is conditional to the vehicles position if it would be useful to take this segment into account.
-> Routing costs for that segment should be reduced by a factor smaller than one.

for 5.) CR = Closed Road: The segment should not be taken into account for routing unless there is no other possibility to reach the destination.
-> Routing costs for that segment should be increased by a factor much greater than one and higher than any other possible route reaching the destination.

EXAMPLE 1

Figure 9 shows a combination of diversion segments explained above, causing that a vehicle at position S would chose the closed line route instead of the upper dashed line route as detour.

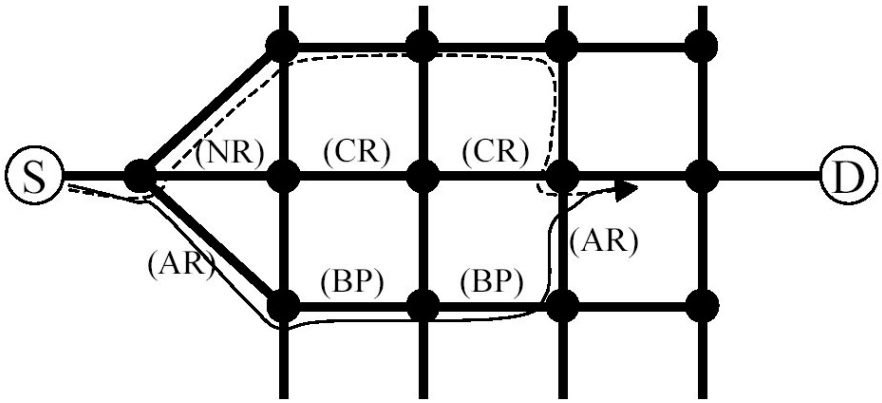

Figure 9 — Example for a simple diversion strategy

As shown in the UML Model (Figure 5) an event can carry a list of DiversionRoute elements which need not to be connected parts of the road network. Each of them can be further discriminated with filtering information in VehicleRestriction and consist of a list of at least one SegmentModifier. The given DiversionRoute-element should only be taken into account for routing if either no filter is given or at least one VehicleRestriction-element completely describes true statements for the vehicles condition.

EXAMPLE 2

A VehicleRestriction element consisting of a destination filter describing the urban area of a given town should only be taken into account for vehicles having a destination in the urban area of that town. Through traffic should not use this DiversionRoute; the service provider has prepared another diversion route for vehicles having a destination outside the urban area.

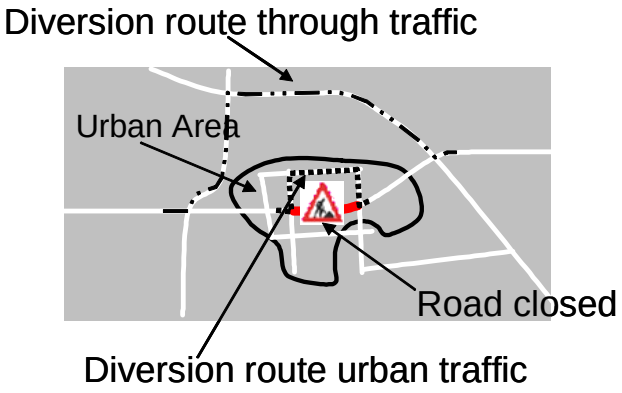

Figure 10 — Example for a simple diversion strategy

The DiversionRoute component is coded as follows:

<DiversionRoute<Component(8)>>:=	
<**IntUnTi**>(8),	: Identifier
<**IntUnLoMB**>(lengthComp),	: Length of component in bytes, excluding the id and length indicator
<**IntUnLoMB**>(lengthAttr),	: Length of attributes of this component in bytes
<**IntUnLoMB**>(n),	: Number of entries in array attribute, between 1 and infinity.
N * <**SegmentModifier**>(segmentModifier)[1..*],	
m * <**VehicleRestriction**>[0..*];	: m represents the number of occurrences, between 0 and infinity.

7 Datatypes

7.1 RestrictionType

<**RestrictionType**>:=	
<**tec007:RestrictionType**>(restrictionType),	
<**BitArray**>(selector),	: 1 byte containing 2 switches.
If (bit 0 of selector is set)	
<**IntUnLoMB**>(restrictionValue),	
If (bit 1 of selector is set)	
<**RestrictionLocation**>(restrictionLocation);	: see Section 6.2.2

7.2 SegmentModifier

<**SegmentModifier**>:=
<**tec008:DiversionRoadType**>(diversionRoadType),
<**SegmentLocation**>; : see Section 6.2.2

7.3 TEC Tables

7.3.1 tec001:EffectCode

Describes the effect to the traffic flow.

Code	CEN English 'Word'	Comment	Example
001	traffic flow unknown	Must be used if traffic flow is unknown. This is often the case for local hazard warnings.	
002	free traffic flow	The traffic is not disturbed.	
003	heavy traffic	Heavy traffic causes problems in the traffic flow.	
004	slow traffic	The traffic is moving slower than normal.	
005	queuing traffic	The traffic is in queues but still slowly floating.	
006	stationary traffic	The traffic is jammed.	
007	no traffic flow	The cause-component may give more information about the reason for "no traffic flow".	- The road has been closed by police as a "regulatory measure" or - The road is blocked by a temporary incident

7.3.2 tec002:CauseCode

Defines various reasons why this message was sent out.

NOTE "Undecodable cause" is to be used by client device unable to read the code provided by a service provider – no code value is transmitted.

Code	CEN English	Comment	Example
001	traffic congestion	In case that the capacity limitation of the street is causing the message.	
002	accident	In case of an accident	
003	roadworks	In case that road works are the reason.	
004	narrow lanes	In case of lanes being smaller as typical for the given country.	
005	impassibility	In case that in general the given part of a road is impassable. Traffic_flow_effect is	

Code	CEN English	Comment	Example
		expected to be no_traffic_flow	
006	slippery road	In case that a slippery road is the reason.	
007	aquaplaning	In case that big areas of water are on the road surface. Deep water being hazardous conditions shall be signalled with hazardous driving condition.	
008	fire	In case that a traffic affecting fire is the reason.	
009	hazardous driving conditions	In case that natural conditions require high caution by the driver. The reason is mostly expected to appear suddenly.	
010	objects on the road	In case that objects impede the drive. The objects would cause an accident or damage of the car.	
011	animals on roadway	In case that animals are on the carriage way.	
012	people on roadway	In case that people are on the carriage way.	
013	broken down vehicles	In case that broken down car lies on the carriage way.	
014	vehicle on wrong carriageway	In case that cars are driving against the one way direction of the carriage way, also known as ghost-driver. (i.e. not parked).	
015	rescue and recovery work in progress	In case that rescue and recovery work is in progress.	
016	regulatory measure	In case that regulatory measure is the reason. (It is expected to be combined with traffic flow effect e.g. ‘ no_traffic_flow’.	
017	extreme weather conditions	In case that extreme weather conditions are the reason.	
018	visibility reduced	In case the reduced visibility needs a speed adaptation.	
019	precipitation	In case that increased precipitation is the reason. This cause is mostly combined with time delays.	
020	reckless persons	In case that reckless persons are the reason.	
021	over-height warning system triggered	In case that an over-height warning system trigger is the reason for e.g. the closure.	
022	traffic regulations changed	In case that changed traffic regulations and therefore high risk of accident are the reason.	

Code	CEN English	Comment	Example
023	major event	In case that a major event is the reason. More close description can be set in free text.	
024	service not operating	In case that a transport service is not operating.	
025	service not useable	In case that a service is not usable although it is operating. (e.g. overcrowded or paused)	
026	slow moving vehicles	In case that slow moving vehicles are the reason.	
027	dangerous end of queue	In case that a dangerous end of queue could cause an accident.	
028	risk of fire	In case that a risk of fire exists. Open fire or glow should be extinguished.	
029	time delay	In case that a time delay exists.	
030	police checkpoint	In case that there is a spot for checking purposes	
031	malfunctioning roadside equipment	In case that a malfunctioning roadside equipment is the reason.	
100	test message	This is a test message used for testing only. Any additional content of this message shall be marked as being a test only.	

7.3.3 tec003:WarningLevel

Defines different levels of danger.

Code	CEN English 'Word'	Comment	Example
001	informative	This level is of standard informative nature.	
002	danger level 1	This level is used for acquiring attention by the driver.	Attention, there is a dangerous obstruction due to fog
003	danger level 2	This level is used for local hazard warnings being dangerous.	Attention, danger due to deer
004	danger level 3	This level is used for local hazard warnings being highly dangerous.	Attention, highest danger due to ghost driver

7.3.4 tec004:LaneRestriction

Defines lanes being restricted with this message.

Code	CEN English 'Word'	Comment	Example
001	lane(s) closed		
002	lane(s) open		
003	right lane(s) closed	Always right lanes are meant; independent where the "normal" driving lane is located.	
004	left lane(s) closed	Always left lanes are meant; independent where the "normal" driving lane is located.	

7.3.5 tec005:AdviceCode

A recommendation or instruction for the driver to do something.

NOTE "Undecodable advice" is to be used by client device unable to read the code provided by a service provider – no code value is transmitted.

Code	CEN English 'Word'	Comment	Example
001	drive to next available parking place		E.g. in combination with time delay at frontier.
002	overtaking not allowed	In case that vehicle should queue up and do not overtake	In case that e.g. ghost-driver approaches.
003	driving not allowed	In case that vehicles should try to stop the vehicle at useful position at the roadway	
004	use hard shoulder as lane	To achieve more capacity for the road.	
005	wait for police patrol	In case that the police controls any continuation of the journey.	
006	wait for improved weather	In case that clearance is already expected.	
007	giving path vehicles coming from behind.	In case that vehicles approaching from behind need to pass.	
008	follow diversion	In case that a diversion is given which shall be followed	
009	no diversion to recommend	In case that no detour is known at this time.	
010	do not divert	In case that no detour shall be taken, because the surrounding road network is crowded already.	
011	follow police instructions	In case that the police regulates the traffic at that spot.	
012	avoid the area	In case that the area is has to be avoided by e.g. high traffic density.	
013	drive carefully	In case that careful driving is required.	

Code	CEN English 'Word'	Comment	Example
014	do not leave your vehicle	In case that driver shall be informed e.g. in a traffic jam not to leave the vehicles.	
015	switch on radio	In case that further important information can be received in car radio.	
016	use toll lanes	In case that toll lanes should be used.	
017	wait for convoy	Due to bad weather conditions, convoy service is required.	

7.3.6 tec006:Tendency

Defines prediction tendencies for the message.

Code	CEN English 'Word'	Comment	Example
001	slightly increasing		
002	increasing		
003	strongly increasing		
004	slightly decreasing		
005	decreasing		
006	strongly decreasing		
007	constant		

7.3.7 tec007:RestrictionType

Defines different types of attributes for filtering the addressed vehicles for this message.

NOTE "Undecodable restriction" is to be used by client device unable to read the code provided by a service provider – no code value is transmitted.

Code	CEN English 'Word'	Comment	Example
001	width less than	cm	
002	width greater than	cm	
003	height less than	cm	
004	height greater than	cm	
005	weight less than	kg	
006	weight greater than	kg	
007	without winter tyre		

Code	CEN English 'Word'	Comment	Example
008	without snow chain		
009	with trailer		
010	with caravan		
011	persons in vehicle less than	num	
012	persons in vehicle more than	num	
013	even number plate		
014	odd number plate		
015	length less than	cm	
016	length greater than	cm	
017	axle load less than	kg	
018	axle load greater than	kg	
019	vehicle fulfils emission standard EURO3	'EURO3' is a specific vehicle emission class according to European Council Directives.	
020	vehicle fulfils emission standard EURO3D4	'EURO3D4' is a specific vehicle emission class according to European Council Directives.	
021	vehicle fulfils emission standard EURO4	'EURO4' is a specific vehicle emission class according to European Council Directives.	
022	vehicle fulfils emission standard EURO5	'EURO5' is a specific vehicle emission class according to European Council Directives.	
023	with petrol-engine		
024	with diesel-engine		
025	with LPG-engine		
026	through traffic		
027	residents traffic		
028	with destination in given area		

7.3.8 tec008:DiversionRoadType

Defines different levels of usability for parts of the diversion.

Code	CEN English 'Word'	Comment	Example
001	bypass	Part of the diversion being mostly parallel to the disturbed road.	
002	access road	Part of the diversion being the access to the bypass	
003	limited access road	Part of the network which is not usable for through traffic.	
004	not recommended road	Part of the network which should be avoided.	
005	closed road	Part of the network which is closed.	

7.3.9 tec009:VehicleType

Defines different types of cars for filtering the addressed vehicles for this message.

NOTE "Undecodable vehicle type" is to be used by client device unable to read the code provided by a service provider – no code value is transmitted.

Code	CEN English 'Word'	Comment	Example
001	car		
002	lorry		
003	bus		
004	taxi		
005	train		
006	motor cycle		
007	vehicle with trailer		
008	motor vehicles		
009	transport of dangerous goods		
010	transport of abnormal load		
011	heavy vehicle		

7.3.10 SubCauseType

The SubCauseType defines the generic type applying different tables according to the different mainCauses. Valid entries for attributes of this type are listed in the tables:

tec101:TrafficCongestion
tec102:Accident
tec103:Roadworks
tec104:NarrowLanes
tec105:Impassibility
tec106:SlipperyRoad
tec108:Fire
tec109:HazardousDrivingConditions
tec110:ObjectsOnTheRoad
tec111:AnimalsOnRoadway
tec112:PeopleOnRoadway
tec113:BrokenDownVehicles
tec115:RescueAndRecoveryWorkInProgress
tec116:RegulatoryMeasure
tec117:ExtremeWeatherConditions
tec118:VisibilityReduced
tec119:Precipitation
tec120:RecklessPersons
tec123:MajorEvent
tec124:ServiceNotOperating
tec125:ServiceNotUseable
tec126:SlowMovingVehicles
tec127:DangerousEndOfQueue
tec128:RiskOfFire
tec129:TimeDelay
tec130:PoliceCheckpoint
tec131:MalfunctioningRoadsideEquipment.

Note: In the case a client device is unable to decode a subcause code the original maincause is to be used.

7.3.10.1 tec101:TrafficCongestion

In case that the capacity of the part of the street caused the message.

Code	CEN English 'Word'	Comment	Example
001	increased volume of traffic	In case that the capacity of the part of the street caused the message.	

7.3.10.2 tec102:Accident

In case of an accident.

Code	CEN English 'Word'	Comment	Example
001	multi-vehicle accident	In case that many cars are involved in the accident	
002	heavy accident	In case of a heavy accident with expected long lasting rescue and recovery work	
003	accident involving lorry	In case of an accident involving a lorry.	
004	accident involving bus	In case of an accident involving a bus.	
005	accident involving hazardous materials	In case of an accident involving hazardous materials.	
006	accident on opposite lane	In case of an accident having happened on the opposite lane.	
007	unsecured accident	In case of an accident being not secured at time of dissemination.	

7.3.10.3 tec103:Roadworks

In case that road works are the reason.

Code	CEN English 'Word'	Comment	Example
001	major roadworks	In case that major road works are the reason.	
002	road marking work	In case that road marking work is the reason.	
003	slow moving road maintenance	In case that a slow moving road maintenance is the reason.	Trimming of the grass on the soft shoulder

7.3.10.4 tec104:NarrowLanes

In case of lanes being smaller as typical for the given country.

Code	CEN English 'Word'	Comment	Example
001	contraflow	In case of lanes being smaller because of road works.	
002	hard shoulder closed	In case that the hard shoulder is closed.	
003	slip lane closed	In case that the slip lane is closed.	
004	crawler lane closed	In case that the crawler lane is closed.	

7.3.10.5 tec105:Impassibility

In case that in general the given part of a road is impassable. Traffic_flow_effect is expected to be no_traffic_flow.

Code	CEN English 'Word'	Comment	Example
001	flooding	In case that flooding water is reason for impassability	
002	danger of avalanches	In case that a danger of avalanches is reason for impassability	
003	blasting of avalanches	In case that a blasting avalanche is reason for impassability in a short period of time	
004	landslips	In case that a landslip is reason for impassability	
005	chemical spillage	In case that chemical spillage is reason for impassability except oil and fuel being expressed as slippery and or danger of fire.	
006	winter closure	In case that a road is impassable due to winter closure.	

7.3.10.6 tec106:SlipperyRoad

In case that a slippery road is the reason.

Code	CEN English 'Word'	Comment	Example
001	heavy frost on road	In case that a slippery road is caused by frost.	
002	fuel on road	In case that a slippery road is caused by petrol on the road.	
003	mud on road	In case that a slippery road is caused by mud on the road.	
004	snow on road	In case that a slippery road is caused by snow on the road.	
005	ice on road	In case that a slippery road is caused by ice on the road.	
006	black ice on road	In case that a slippery road is caused by black ice on the road.	
007	oil on road	In case that a slippery road is caused by oil on the road.	
008	loose chippings	In case that a slippery road is caused by loose chippings on the road.	
009	instant black ice	In case that rain is falling on frozen ground which immediately freezes.	
010	roads salted	In case that roads may be slippery, but are salted.	

7.3.10.7 tec108:Fire

In case that a traffic affecting fire is the reason.

Code	CEN English 'Word'	Comment	Example
001	major fire	In the case of a long lasting major fire.	
002	forest fire	In the case of a long lasting forest fire.	

7.3.10.8 tec109:HazardousDrivingConditions

In case that natural conditions require high caution by the driver. The reason is mostly expected to appear suddenly.

Code	CEN English 'Word'	Comment	Example
001	rockfalls	Danger of falling rocks.	
002	earthquake damage	Damages caused by earthquakes.	
003	sewer collapse	Sewer collapses occur.	
004	subsidence	Caution, partial subsidence occur.	
005	snow drifts	Caution, snow drifts occur.	
006	storm damage	In case that storm damages occur, which are not 'hazardous driving conditions' or 'objects on road'	
007	burst pipe	In case that a burst pipe affects the traffic and cause is not 'sewer collapse' or 'flooding'.	
008	volcano eruption	Danger of volcano eruption.	
009	falling ice	Danger of falling ice.	

7.3.10.9 tec110:ObjectsOnTheRoad

In case that objects impede the drive. The objects would cause an accident or damage of the car.

Code	CEN English 'Word'	Comment	Example
001	shed load	In case that the objects are parts of the load of a lorry.	
002	parts of vehicles	In case that the objects are parts of a vehicle (exhaust pipe).	
003	parts of tyres	In case that the objects are parts of tyres.	
004	big objects	In case that the objects are that big, that a car cannot pass over.	
005	fallen trees	In case that the objects are parts of fallen trees.	
006	hub caps	In case that the objects are hub caps.	
007	waiting vehicles	In case that the objects are waiting vehicles.	

7.3.10.10 tec111:AnimalsOnRoadway

In case that animals are on the carriage way.

Code	CEN English 'Word'	Comment	Example
001	wild animals	In case that wild animals are on the road.	Deer
002	herd of animals	In case that a whole herd is on the road. The driver should expect a full stop while waiting for the herd leaving.	Sheep
003	small animals	In case that small animals, which fit under a car, are on the road.	Frogs, foxes, badgers
004	large animals	In case that big animals, which don't fit under a car, are on the road.	Cows or horses

7.3.10.11 tec112:PeopleOnRoadway

In case that only people or people with unauthorized vehicles are on the carriage way.

Code	CEN English 'Word'	Comment	Example
001	children on roadway	In case that children are walking on the road.	
002	cyclists on roadway	In case that cyclists are driving on the road. (i.e. motorway)	
003	motor cyclist on roadway	In case that motor cyclists are driving on the road. (i.e. motorway)	

7.3.10.12 tec113:BrokenDownVehicles

In case that broken down car lies on the carriage way.

Code	CEN English 'Word'	Comment	Example
001	broken down vehicle burning	In case that the car lying on the road also is burning.	
002	broken down unlit vehicle	In case that the car lying on the road not illuminated.	

7.3.10.13 tec115:RescueAndRecoveryWorkInProgress

In case that rescue and recovery work is in progress.

Code	CEN English 'Word'	Comment	Example
001	emergency vehicles	In case that emergency vehicles are at place.	
002	rescue helicopter landing	In case that a rescue helicopter landing is taking place.	
003	police activity ongoing	In case that police activity is ongoing.	
004	medical emergency ongoing	In case that a medical emergency operation is ongoing.	
005	child abduction in progress	In case that a child abduction is in progress (amber alert)	

7.3.10.14 tec116:RegulatoryMeasure

In case that regulatory measure is the reason. (It is expected to be combined with traffic flow effect e.g. ' no_traffic_flow'.

Code	CEN English 'Word'	Comment	Example
001	security alert	In the case that the reason for the measure is security incident.	An area is closed due to - hostage situation.
002	contagious disease	In the case that the reason for the measure is contagious disease.	A disease of animals such as foot and mouth or blue tongue.
003	environmental	In the case that the reason for the measure is environmental.	An area is closed due to pollution.
004	smog alert	In the case that the reason for the measure is a smog alert.	
005	batch service in progress	In case that batch service is in progress.	Batch service in Tunnels: 5 minutes one way traffic, then 5 minutes the other way.

7.3.10.15 tec117:ExtremeWeatherConditions

In case that extreme weather conditions are the reason.

Code	CEN English 'Word'	Comment	Example
001	strong winds	In case that strong winds, especially cross winds, are the reason.	
002	damaging hail	In case that hail could damage cars.	
003	hurricane	In case the strength of the storm is higher than a given strength.	
004	thunderstorm	In case that a strong thunderstorm affects driving.	
005	tornado	In case the strength of the storm is higher than a given strength.	
006	blizzard	In case the strength of the storm is higher than a given strength combined with lots of snow.	

7.3.10.16 tec118:VisibilityReduced

In case the reduced visibility needs a speed adaptation.

Code	CEN English 'Word'	Comment	Example
001	visibility reduced due to fog	In case that the visibility is reduced by fog.	
002	visibility reduced due to smoke	In case that the visibility is reduced by smoke.	
003	visibility reduced due to heavy snowfall	In case that the visibility is reduced by heavy snow fall.	
004	visibility reduced due to heavy rain	In case that the visibility is reduced by heavy rain.	
005	visibility reduced due to heavy hail	In case that the visibility is reduced by heavy hail.	
006	visibility reduced due to low sun glare	In case that the visibility is reduced by low sun glare.	
007	visibility reduced due to sandstorms	In case that the visibility is reduced by sandstorms.	
008	visibility reduced due to swarms of insects	In case that the visibility is reduced by swarms of insects.	

7.3.10.17 tec119:Precipitation

In case that increased precipitation is the reason. This cause is mostly combined with time delays.

Code	CEN English 'Word'	Comment	Example
001	heavy rain	In case that heavy rain is the reason for e.g. reduced speed limit.	
002	heavy snowfall	In case that heavy snowfall is the reason for e.g. reduced speed limit.	
003	soft hail	In case that hail is the reason for e.g. reduced speed limit.	

7.3.10.18 tec120:RecklessPersons

In case that reckless persons are the reason.

Code	CEN English 'Word'	Comment	Example
001	reckless driver	In case that reckless drivers are the reason. The backward traffic should be given attention.	
002	gunfire on road	In case that a gunfire on the road is the reason.	
003	stone throwing persons	In case that reckless persons throwing stones are the reason. (e.g. from a bridge)	

7.3.10.19 tec124:ServiceNotOperating

In case that a transport service is not operating.

Code	CEN English 'Word'	Comment	Example
001	ferry service not operating	In case that a ferry service not operating is the reason.	
002	plane service not operating	In case that a plane service not operating is the reason.	
003	train service not operating	In case that a train service not operating is the reason.	
004	bus service not operating	In case that a bus service not operating is the reason.	

7.3.10.20 tec123:MajorEvent

Code	CEN English 'Word'	Comment	Example
001	sports event	In case that a major sports event is the reason.	
002	demonstration	In case that a major demonstration is the reason.	
003	demonstration with vehicles	In case that a major demonstration with vehicles is the reason.	
004	concert	In case that a concert is the reason.	
005	fair	In case that a fair is the reason.	
006	military training	In case that a military training is the reason.	
007	emergency training	In case that an emergency training is the reason.	
008	festivity	In case that a festivity is the reason.	
009	procession	In case that a procession is the reason.	

7.3.10.21 tec125:ServiceNotUseable

In case that a service is not usable although it is operating. (e.g. overcrowded or paused)

Code	CEN English 'Word'	Comment	Example
001	fuel station closed	In case that a fuel station is closed for some time.	
002	service area closed	In case that a service area is closed for some time.	
003	service area busy	In case that a service area is busy for some time.	
004	parking full	In case that a parking area is full for some time.	
005	car park closed	In case that a parking area or car park is closed.	

7.3.10.22 tec126:SlowMovingVehicles

In case that slow moving vehicles are the reason.

Code	CEN English 'Word'	Comment	Example
001	slow moving maintenance vehicle	In case that slow moving maintenance vehicles are the reason.	
002	vehicles slowing to look at accident	In case that vehicles slowing down to look at accidents having happened on the opposite side of the road way.	
003	abnormal load	In case that a lorry with abnormal load (without wide load) are on the road.	
004	abnormal wide load	In case that a lorry with abnormal wide load are on the road.	
005	convoy	In case that a convoy is on the road and overtaking is not allowed.	
006	snowplough	In case that a snowplough is on the road.	
007	deicing	In case that a deicing vehicle is on the road.	
008	salting vehicles	In case that a salting vehicle is on the road.	

7.3.10.23 tec127:DangerousEndOfQueue

In case that a dangerous end of queue could cause an accident.

Code	CEN English 'Word'	Comment	Example
001	sudden end of queue	In case that a dangerous end of queue is expected to appear suddenly.	
002	queue over hill	In case that a dangerous end of queue is over the top of a hill.	
003	queue around bend	In case that a dangerous end of queue is around a bend.	
004	queue in tunnel	In case that a dangerous end of queue is inside of a tunnel.	

7.3.10.24 tec128:RiskOfFire

In case that a risk of fire exists. Open fire or glow should be extinguished.

Code	CEN English 'Word'	Comment	Example
001	leakage of fuel	In case that a risk of fire is caused by a leakage of fuel.	
002	leakage of gas	In case that a risk of fire is caused by a leakage of gas.	

7.3.10.25 tec129:TimeDelay

In case that a time delay exists.

Code	CEN English 'Word'	Comment	Example
001	time delay at frontier	In case of a time delay at the frontier.	
002	time delay at ferry port	In case of a time delay at the ferry port.	
003	time delay at vehicle-on-rail terminal	In case of a time delay at the vehicle-on-rail terminal.	

7.3.10.26 tec130:PoliceCheckpoint

In case that there is a spot for checking purposes.

Code	CEN English 'Word'	Comment	Example
001	permanent police checkpoint	In case that a permanent police checkpoint is the reason.	
002	temporary police checkpoint	In case that a sporadic police checkpoint is the reason.	

7.3.10.27 tec131:MalfunctioningRoadsideEquipment

In case that a malfunctioning roadside equipment is the reason.

Code	CEN English 'Word'	Comment	Example
001	road-rail crossing failure	In case that a road-rail crossing is malfunctioning.	
002	tunnel ventilation not working	In case that the tunnel ventilation is malfunctioning.	
003	traffic control signals working incorrectly	In case that traffic control signals are malfunctioning.	
004	emergency telephones not working	In case that emergency telephones are malfunctioning.	
005	automatic payment lanes not working	In case that automatic payment lanes are not working.	

7.3.11 SubAdviceType

The SubAdviceType defines the generic type applying different tables according to the different mainAdviceCodes

Valid entries for attributes of this type are listed in the tables:

tec202:OvertakingNotAllowed,
tec203:DrivingNotAllowed,
tec207:GivingPathVehiclesFromBackward,
tec208:FollowDiversion
tec213:DriveCarefully
tec214:DoNotLeaveYourVehicle.
tec216:UseTollLanes
Note: In the case a client device is unable to decode a subadvice code the original advice is to be used.

7.3.11.1 tec202:OvertakingNotAllowed

In case that vehicle should queue up and do not overtake

Code	CEN English 'Word'	Comment	Example
001	do not use overtaking lanes	In the case that overtaking is not allowed.	A ghost-driver approaches.
002	overtaking not allowed, drive on crawler lane	In the case that additionally the preferred side can be recommended.	A ghost-driver approaches.
003	overtaking not allowed, drive on left most lane	In the case that additionally the preferred side can be recommended.	
004	overtaking not allowed, drive on right most lane	In the case that additionally the preferred side can be recommended.	

7.3.11.2 tec203:DrivingNotAllowed

In case that vehicles should try to stop the vehicle at useful position at the roadway.

Code	CEN English 'Word'	Comment	Example
001	driving not allowed, take next possible place to stop vehicle		An earthquake will happen.

7.3.11.3 tec207:GivingPathVehiclesFromBackward

In case that vehicles approach from backward, which need to pass.

Code	CEN English 'Word'	Comment	Example
001	giving path for rescue vehicle	In case that rescue vehicles approach from behind, which need to pass.	
002	giving path for service vehicles	In case that recovery vehicles approach from backward.	

7.3.11.4 tec208:FollowDiversion

In case that a diversion is given which shall be followed.

Code	CEN English 'Word'	Comment	Example
001	follow diversion signs	In case that a diversion is signposted.	

7.3.11.5 tec213:DriveCarefully

In case that careful driving is required.

Code	CEN English 'Word'	Comment	Example
001	drive carefully, dangerous situation on entry slip road	In case that a road is affected by a dangerous situation on an entry slip road.	
002	drive carefully, dangerous situation on exit slip road	In case that a road is affected by a dangerous situation on an exit slip road.	
003	drive carefully, ice buildup on cable structure	In case that ice is buildup on cable structures close to the road.	

7.3.11.6 tec214:DoNotLeaveYourVehicle

In case that driver shall be informed e.g. in a traffic jam not to leave the vehicles.

Code	CEN English 'Word'	Comment	Example
001	do not leave your vehicle		Due to security reasons.
002	do not leave your vehicle , close windows		Toxic gas coming from a fire could flow into the vehicle.

7.3.11.7 tec216:UseTollLanes

In case that toll lanes should be used.

Code	CEN English 'Word'	Comment	Example
001	use manual payment toll lanes	In case that manual payment toll lanes should be used.	
002	use automatic payment toll lanes	In case that automatic payment toll lanes should be used.	

Annex A
(normative)

Binary SSF and Data Types

A.1 Conventions and symbols

A.1.1 Conventions

A.1.1.1 Byte ordering

All numeric values using more than one byte are coded in “Big Endian” format (most significant byte first). Where a byte is subdivided into bits, the most significant bit (“b7”) is at the left-hand end and the least significant bit (“b0”) is at the right-hand end of the structure.

A.1.1.2 Method of describing the byte-oriented protocol

TPEG uses a data-type representation for the many structures that are integrated to form the transmission protocol. This textual representation is designed to be unambiguous, easy to understand and to modify, and does not require a detailed knowledge of programming languages.

Data types are built up progressively. Primitive elements, which may be expressed as a series of bytes are built into compound elements. More and more complex structures are built up with compound elements and primitives. Some primitives, compounds and structures are specified in this Technical Specifcation, and apply to all TPEG Applications. Other primitives, compounds and structures are defined within applications and are local only to that application.

A resultant byte-stream coded using C-type notation is shown in CEN ISO/TS 18234-2:2006, Annex E.

A.1.1.3 Reserved data fields

If any part of a TPEG data structure is not completely defined, then it should be assumed to be available for future use. The notation is UAV (unassigned value). This unassigned value should be encoded by the service provider as the value 00 hex. This allows newer decoders using a future TPEG Standard to ignore this data when receiving a service from a provider encoding to this older level of specification. A decoder which is not aware of the use of any former UAVs can still make use of the remaining data fields of the corresponding information entity. However, the decoder will not be able to process the newly defined additional information.

A.1.2 Symbols

A.1.2.1 Literal numbers

Whenever literal numbers are quoted in TPEG Standards, the following applies:

123 = 123 decimal

123 hex = 123 hexadecimal

A.1.2.2 Variable numbers

Symbols are used to represent numbers whose values are not predefined within the TPEG Standards. In these cases, the symbol used is always local to the data type definition. For example, within the definition of a

data type, symbols such as “n” or “m” are often used to represent the number of bytes of data within the structure, and the symbol “id” is used to designate the occurrence of the identifier of the data type.

A.1.2.3 Implicit numbers

Within the definition of a data structure it is frequently necessary to describe the inclusion of a component which is repeated any number of times, zero or more. In many of these cases it is convenient to use a numerical symbol to show the component structure being repeated a number of times, but the number itself is not explicitly included within the definition of the data structure. Often, the symbol “m” is used for this purpose.

A.2 Representation of syntax

A.2.1 General

This clause introduces the terminology and the syntax that is used to define TPEG data elements and structures.

A.2.2 Data type notation

A.2.2.1 Rules for data type definition representation

The following general rules are used for defining data types:

— a data type is written in upper camel case letters in one single expression.[1] The data type may contain letters (a-z), number (0-9), underscore "_", round brackets "()" and colon ":"; the first must be a letter;

EXAMPLE 1 IntUnLo stands for Integer Unsigned Long

— a data type is framed by angle brackets “ **< >** ” ;

— the content of a data type is defined by a colon followed by an equal sign “ **:=** ”;

— the end of a data type is indicated by a semicolon “ **;** ”;

— a descriptor written in lower camel case may be added to a data type as one single expression without spaces;

— a descriptor is framed by round brackets “ ( ) ”;

— the descriptor contains either a value or a name of the associated type;

— data types in a definition list of another one are separated by commas “ **,** ”. The order of definition is defined as the order of occurrence in a data stream;

— curly brackets (braces) “ { } ” group together a block of data types;

— control statements (“if”, “infinite”, “unordered” or “external”) are noted in lower case letters. A control statement is followed by a block statement or only one data type:

  3) “if” defines a condition statement. The block’s (or data type’s) occurrence is conditional to the condition statement being valid. The condition statement is framed with round brackets. This statement applies to any data type;

[1] Camel case is the description given to the use of compound words wherein each individual word is signalled by a capital letter inside the compound word. Upper camel case means that the compound word begins with an upper-case (capital) letter, and lower camel case means the compound word begins with a small letter.

4) “infinite” defines endless repetition of the block (or data type). This is only used to mark the main TPEG stream as not ending stream of data;

5) “unordered” defines that the following block contains data types which may occur in any order, not only the one used to specify subsequent data types. This statement applies to components only. (See Clause A2.3.3 - Components);

6) “external” defines that the content of the data type is being defined external to the scope of given specification. The control statement “external” must be followed by only one data. A reference to the corresponding specification should follow in the comment. All types specified in TYP specification are treated as being in scope of any application

EXAMPLE 2

<MMCLink(1)>:=	: externally defined component
external <MessageManagementContainer(1)>;	: id = 1, See Annex B (Message Management Container)

— the expression “ n * ” indicates multiplicity of occurrence of a data type . The lower and upper bound are implicitly from 0 to infinite; other bounds are described in square brackets between two points " .. " and behind the data type descriptor. The " * " stands for no limitation at upper bound

EXAMPLE 3

m * **<IntUnTi>**(Attribute) [1..*] ,	: The “Attribute” must occur once at least and up to infinite.

— a function “ f_n ( ) ” that is calculated over a data type is indicated by italic lower case letters. The comment behind the definition of the function shall explain which function is used;

— any text after a colon “ : ” is regarded as a comment;

— a data type definition can be a *template* (i.e. not fully defined declarative structure) having a *parameter* inside of round brackets "(x)" at the end of the data type name. Templates define structures, whose structural definition is included as a basis for other data type definitions. To declare the given template (making it identifiable) the name of the parameter is repeated as a descriptor in a nested data type of the subsequent definition list. Templates allow for reading the generalised part of different instances i.e. to specify data type interfaces. (See Clause A2.3.2 - Using templates as interfaces for further description)

EXAMPLE 4

<Template(x)> :=	: x defines the template parameter
<IntUnTi>(x);	: descriptor x defines position of setting the parameter in the list

— a data type can *inherit* a template by concatenating the data type name of the template including the square brackets to its own name. The data type itself can again be a template having the "(x)" at its end of name, or it instantiates the inherited template by defining the value of the parameter in the brackets. In the latter case the brackets shall contain the **decimal** number of the identifier and the value shall be set in the subsequent definition list. The structural definition of the inherited template is repeated as the first part of the definition list before new data types are specified. (See Clause A2.3.2 - Using templates as interfaces for further description)

EXAMPLE 5

<AnotherTemplate(x)<Template(x)>>:=	: second template inherits first
<IntUnTi>(x),	: repeated definition from 1st template
<IntUnLi>(n);	: additional structural definition
<Instance<AnotherTemplate(1)>>:=	**: instantiation of the second template**
<IntUnTi>(1),	: definition of parameter in the stream
<IntUnLi>(n),	: structural definition from template
<IntUnTi>(value);	: some more definition

— in the definition list a specific instance of a template (i.e. declarative structure) is described without the brackets. Any inherited data type of this template may occur at that position in the data stream

EXAMPLE 6

<SomeData>:=	
<AnotherTemplate>(anyAnotherTemplate);	: Data stream contains e.g. <**Instance**>

The following additional guidelines help to improve the readability of data type definitions:

— data type names are written in bold;

— nested data type definitions are defined from top to bottom (i.e. higher levels first, then lower levels);

— a box is drawn around a data type definition;

— for clear graphical presentation, lines in a coding box if they are too long to fit, are broken with a backslash "\" followed by a carriage return. The broken line restarts with an additional backslash

EXAMPLE 7

<LongLinesExample>:=	
<DateTimeVeryLongType	: First line
\NameMayBeInSeveralLines>,	: Second line
<DateTime>,	
<ShortString>;	

A.2.2.2 Description of data type definition syntax

A data type is an interpretation of one or more bytes. Each data type has a structure, which may describe the data type as a composition of other defined data types. The data type structure shows the composition and the position of each data element. TPEG defines data structures in the following manner:

<NewDataType>:=	: Description of data type
<DataTypeA>(descriptorA)**,**	: Description of data A
<DataTypeB>(descriptorB);	: Description of data B

This shows an example data structure, which has just two parts, one of type **<DataTypeA>** and the other of **<DataTypeB>**. A descriptor may be assigned to the data type, to relate the element to another part of the definition. Comments about the data structure are included at the right-hand side delimited by the colon ":" separator. Each of the constituent data types may be itself composed of other data types, which are defined separately. Eventually each data type is expressible as one or more bytes.

Where a data structure is repeated a number of times, this may be shown as follows:

<NewDataType>:=	: Description of data type
<DataTypeA>,	: Description of data A
m * **<DataTypeB>**[0..*];	: Description of data B

Often, in such cases it is necessary to explicitly deliver to the decoder the number of times a data type is repeated; sometimes it is not, because other means like framing or internal length coding allows knowledge of the end of the list of the repeated data type. In other cases the overall length of a data structure in bytes needs to be specified. Additionally the constraint on occurrences can be added, which tells how many instances of the data type must be expected by the decoder. The "*" as upper bound means in this case that at this place no restriction is given to the upper bound; in other words, infinite elements may follow.

Where the number of repetitions must be signalled, it may be accomplished using another data element as follows:

<NewDataType>:=	: Description of data type
<IntUnTi>(n)**,**	: An integer representing the value of "n"
n * **<DataTypeA>**[0..255]**,**	: Description of data A
<DataTypeB>;	: Description of data B

In the above example a decoder has to have the value of "n" in order to correctly determine the n'th position of the **<DataTypeB>** in the list. Here as consequence of data type IntUnTi not more as 255 instances of the data type can be coded.

In the following example the decoder uses the value of "n" to determine the overall length of the data structure, and the value of "m" determines that **<DataTypeB>** is repeated m times:

<NewDataType>:=	: Description of data type
<IntUnTi>(n)**,**	: Length, n, of data structure in bytes
m * **<DataTypeA>;**	: Description of data A

This data type definition is used to describe a variable structure switched by the value of x:

<NewDataType>:=	: Description of data type
<IntUnTi>(x)**,**	: Select parameter, x
if (x=1) then **<DataTypeA>,**	: Included if x equals 1
if (x=2) then **<DataTypeB>,**	: Included if x equals 2

A.2.3 Application dependent data types

This clause describes the methodology and syntax by which application data types may be constructed within TPEG Applications. Two basic forms are described: data structures (being non-declarative) and components (being declarative). Components contain an identifier which labels the structure, and which can be used by a decoder to determine the definition of content of the structure. As such, components are used where options are required, or where an application needs to build in 'future proofing'. Data structures do not contain such information, and are used in all other positions.

This Annex does not specify the structures, which are actually used in TPEG Applications. Such specifications are made in the respective parts of the Standard. However examples are given to show how such structures may be built from the primitive elements already described above.

A.2.3.1 Data structures

Data structures are built up from several (i.e. more than one) elements: primitive, compound or other structures (both non-declarative and declarative). As such, any application specific data type definition having no component identifier is per definition a data structure. The term data structure is specifically used for data type definitions having more than one sub element defined.

Examples of data structure might be:

EXAMPLE 1

<Activity>:=	: Activity
<DateTime>,	: Beginning
<DateTime>,	: End
<ShortString>;	: Text

EXAMPLE 2

<Wave>:=	: Sound sample
<IntUnLi>(n)**,**	: Length of samples, n
n * **<IntSiTi>**(sample)[0..8000];	: Between 0 and 8000 occurrences of a sample

Another example making use of a condition within a data type definition is shown below.

EXAMPLE 3 An application could use the example data types above in the following way

<Appointment>:=	: Appointment
<IntUnTi>(at)**,**	: Alarm type
if (at = 1)	
<WaveAlarm>,	: Remind with a sound
if (at = 2)	
<TextAlarm>,	: Remind with a text
<Activity>;	: Let some action follow

<WaveAlarm>:=	: Sound alarm
<DateTime>,	: When to wake up
<Wave>;	: Sound to wake up to!

<TextAlarm>:=	: Text alarm
<DateTime>,	: When to display
<ShortString>;	: Text to display

For optional values a general mechanism is provided, using a bitarray for signalling optional values. In the case that a corresponding bit of the bitarray is set (=1), the optional attribute is stored in the stream. In case the bit is unset the attribute is not available and the next following attribute shall be processed in the stream.

EXAMPLE 4 Data structure with optional elements, signalled by a preceding bitarray as selector

<TimeInterval>:=	
<BitArray>(selector),	: DaySelector
if (bit 0 of selector is set)	
<IntUnTi>(years),	: Number of years between 0 and 100
if (bit 1 of selector is set)	
<IntUnTi>(months),	: Number of months between 0 and twelve
if (bit 2 of selector is set)	
<IntUnTi>(days),	: Number of days between 0 and 31
if (bit 3 of selector is set)	
<IntUnTi>(hours),	: Number of hours between 0 and 24
if (bit 4 of selector is set)	
<IntUnTi>(minutes),	: Number of minutes between 0 and 60
if (bit 5 of selector is set)	
<IntUnTi>(seconds);	: Number of seconds between 0 and 60

A.2.3.2 Using templates as interfaces

In addition to the possibility of coding the complete and static structural definition of a data structure, the syntax does foresee that parts of the structure are conditionally different; signalled by a well defined first part some other data types are different.

EXAMPLE

A tagged value (also known as TagLengthValue-Coding) starts with a type and length; afterwards the value follows. Let's assume the type is an enumeration of some possible values, one would first specify the interface having only the type defined. The different tagged value types would now inherit this interface, i.e. would have the type defined as first element amended with the definition of the tagged value data type. The decoder now reads the interface information (the type attribute) and knows how to proceed for reading the rest of the tagged value from the stream.

<DifferentDataList>:=	: A list of data
n * **<TaggedValue>**(value)**;**	: Different instances can have different types

<TaggedValue(x)>:=	: Template for tagged value
<tav001:ValueType>(type)**,**	: Type of this tagged value
<IntUnTi>(length)**;**	: Length in bytes in case that value type is unknown

Example table **tav001:ValueType**:

Code	Reference-English 'word'	Comment
001	Service name	
002	Price per month	

Then the resulting list of inherited tagged value data types would be:

<ServiceName<TaggedValue(1)>>:=	: Template for tagged value
<tav001:ValueType>(1)**,**	: Type of this tagged value
<IntUnTi>(length),	: Length in bytes in case that value type is unknown
<**ShortString**>(serviceName)**;**	: Service name

<ServiceName<TaggedValue(2)>>:=	: Template for tagged value
<tav001:ValueType>(2)**,**	: Type of this tagged value
<IntUnTi>(length),	: Length in bytes in case that value type is unknown
<**Float**>(pricePerMonth)**;**	: Price per month

This interface allows a subsequent list of data types which can easily be extended, by using the same interface.

A.2.3.3 Components

A component is understood as a declarative structure having an interface as described in the previous clause. A decoder of the data stream can identify the content of the structure with the help of the identifier which is unique in the scope of any one TPEG Application Standard. In addition to the identifier a length indicator allows the decoder to step over those components whose ids are unknown to it. This enables the possibility of introducing new components in the data stream although decoders in the market do not know their content. The old decoder does expect the content of the first version of a protocol and ignores simply unrecognized data with small performance loss. The new decoder expects the second version of the protocol and can fully decode that version of the protocol. Components should be used wherever *future extensions* are envisioned, and where 'future proofing' is a strong requirement.

NOTE With this method even non-backwards compatible changes can be introduced into the existing market by having a migration period being backward compatible and then later cutting of not longer supported devices, even though it is expected that the migration will take its time.

In Addition to the concept of declarative structuring a second step of improvement of size efficiency combined with the backward compatibility is specified. The first part following the header of a component in the data stream is defined as *attribute block*. The attribute block starts with the length of the block in bytes which again allows the decoder to step over attributes that are not specified in a first version of the protocol.

The decoder reads the attribute block length and decreases the count of bytes while reading the attributes in case that the last known attribute is read, and the attribute block count is not zero, the remaining bytes in the data stream are omitted to step over to the next well-known part of the data stream.

A.2.3.3.1 Definition of standard component interface

A component, including attributes, which is the general standard component, containing a unique "generic component id", a length indicating count of bytes following as data after the component length and an attribute length indicating the count of bytes in the attribute block (as first part of the component data). The structure is defined by:

<**Component(x)**>:=	: Component template used for standard components
<IntUnTi>(x),	: id is unique within the scope of the application.
<IntUnLoMB>(compLengthInByte),	: length of the component counted in bytes.
<IntUnLoMB>(attributeBlockLengthInByte);	: length of the attribute block in bytes.

A.2.3.3.2 Example for jumping over unknown content types

— let C1 be a component with an attribute a1 as ShortInt and a sub component C2;

— let C2 be a component with an attribute a2 as one IntUnTi and a second a3 as ShortString;

— let C3 be a component being the successor of C1.

<C1<Component(1)>>:=	
<IntUnTi>(1),	: id = 1.
<IntUnLoMB>(compLengthInByte),	: length of the component counted in bytes
<IntUnLoMB>(attributeBlockLengthInByte);	: length of the attribute block in bytes
<ShortInt>(a1),	: first attribute in C1
<C2>(c2);	: sub component from C1

<C2<Component(2)>>:=	
<IntUnTi>(2),	: id = 2.
<IntUnLoMB>(compLengthInByte),	: length of the component counted in bytes
<IntUnLoMB>(attributeBlockLengthInByte);	: length of the attribute block in bytes
<IntUnTi>(a2),	: first attribute in C2
<ShortString>(a3);	: second attribute in C2

<C3<Component(3)>>:=	
<IntUnTi>(3),	: id = 3.
<IntUnLoMB>(compLengthInByte),	: length of the component counted in bytes
<IntUnLoMB>(attributeBlockLengthInByte);	: length of the attribute block in bytes

For example to demonstrate the method some padding bytes with value CD hex could be added to the stream whereby a decoder could still read C1 – C3. In Figure A.1 one can see a first line with a position number, a second line with the abbreviated function of that byte and a third line with sample content. The arrows under the table show the possible jumps allowing the seeking over the different padding bytes.

Line function abbreviations mean:
CL : component (data) length in bytes AL : attribute block length in bytes
P : padding bytes A1, A2, A3 : attributes
C1, C2, C3 : component identifier, begin of the component

Pos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Func	C1	CL	AL	A1'	A1"	P	P	C2	CL	AL	A2	A3	A3	A3	A3	A3	P	C3	CL	AL
Val	1	15	4	42	12	CDh	CDh	2	8	7	3	4	'T'	'E'	'S'	'T'	CDh	3	1	0

Figure A.1 — Example for jumping over unknown content with component header information

A.2.4 Toolkits and external definition

Some functionality is shared between different TPEG Applications. This is for example the case for location referencing container and message management container. A TPEG Application therefore can refer to a data type definition not specified in the same Technical Specifcation.

Toolkits are designed, so that the root components usable as external reference are defined as templates. A TPEG Application using a toolkit template therefore needs to specify a unique generic component id for this instantiation of the interface.

All subsequent components in a toolkit are defined as out of scope of the TPEG Application; i.e. the toolkit on its own defines subcomponents beginning with 0. With that on one hand application decoder must be aware that component ids of the application may be repeated in sub components of a toolkit. On the other hand further development of application and toolkit can be done independently.

A.2.5 Application design principles

This clause describes design principles that will be helpful in building TPEG applications. A fundamental assumption is that applications will develop and new features will be added. If design principles are adopted properly then older decoders will still operate properly after extending features. Correct design should permit applications to be upgraded and extended over time, providing new features to new decoders, and yet permit existing decoders to continue to operate.

A.2.5.1 Variable data structures

Switches may be included within an application, which permit variations in the subsequent data structure. However, the switch fixes the values of variations. A new type cannot be introduced without breaking backward compatibility. This may be achieved by using components. When new features are likely to be incorporated, attention should be given to the fact that old decoders just 'skip over' new data fields and still expect the old components if they were mandatory.

A.2.5.2 Re-usable and extendable structures

Within an application there will be data structures, which are used repeatedly in a variety of places. There will also certainly be an ever-growing set of structures, as the application protocol develops and incorporates new features. Component templates may be used to minimize the number of occasions within the decoder's software in which the structure needs to be defined, and to permit an increasing variety of structures to be used in a given location.

A.2.5.3 Validity of declarative structures

The Identifier of a component is uniquely defined within each application. The same number may be used in different applications for completely different purposes. Within an application one identifier designates one definition of a component. The design of an application may use components to implement placeholders or to change the composition of elements in a fixed structure.

A.3 TPEG data stream description

A.3.1 Diagrammatic hierarchy representation of frame structure

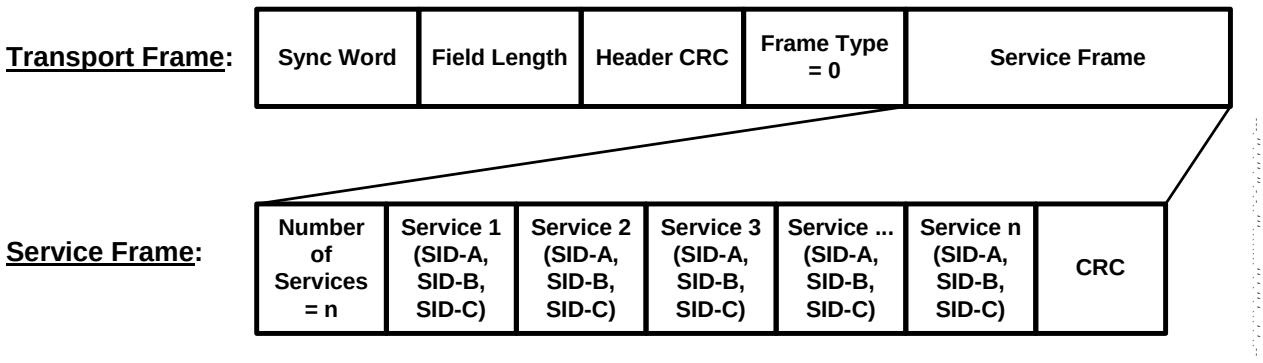

Figure A.2 — TPEG Frame Structure, Frame Type = 0 (i.e. stream directory)

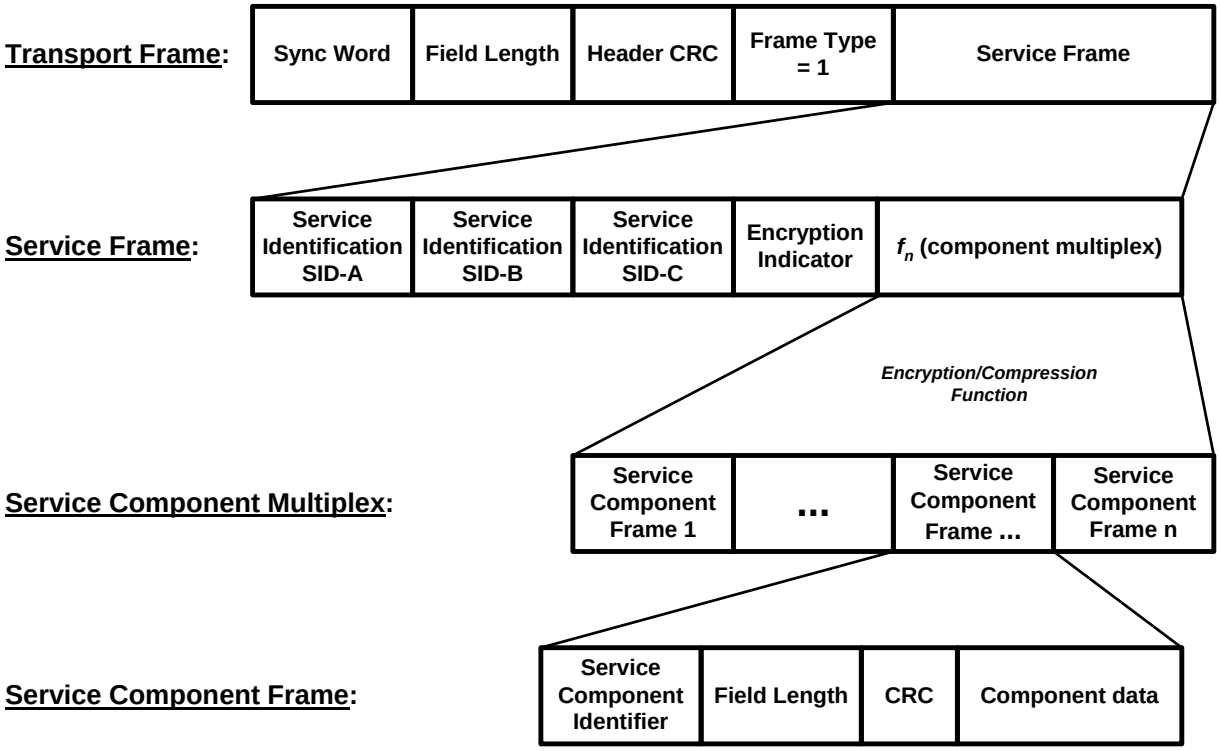

Figure A.3 — TPEG Frame Structure, Frame Type = 1 (i.e. conventional data)

A.3.2 Syntactical Representation of the TPEG Stream

A.3.2.1 TPEG transport frame structure

The following boxes are the syntactical representation of the TPEG frame structure shown in Clause A.3.1 The byte stream contains consecutive transport frames. Each frame includes:

The synchronization word (syncword)	2 bytes	(See Clause A.3.3.1)
The length of the service frame in bytes (field length)	2 bytes	(See Clause A.3.3.2)
The header CRC	2 bytes	(See Clause A.3.3.3)
The frame type indicator	1 byte	(See Clause A.3.3.4)
The service frame	n bytes (n = Field Length)	

The byte stream is built according to the above-mentioned repetitive structure of transport frames. Normally one transport frame should follow another directly, however if any spacing bytes are required these should be set to 0 hex (padding bytes).

```
<TpegStream>:=                          : The data stream.
    infinite {                          : Control element, (loop continues infinitely)
        n * <IntUnTi>(0),               : Any number of padding bytes (0 hex)
        <TransportFrame>                : Transport frames
    };
```

```
<TransportFrame>:=
    <IntUnLi>(FF0F hex),                : Sync word (FF0F hex)
    <IntUnLi>(m),                       : Number of bytes in Service Frame
    <CRC>(headCRC),                     : Header CRC, (See Clause A.3.3.4)
    <IntUnTi>(x),                       : Frame type of service frame
    <ServiceFrame(x)>;                  : Any service frame follows
```

A.3.2.2 TPEG service frame template structure

This service frame comprises:

<ServiceFrame(x)>:=	: Template for service frame
n * **<byte>;**	: Content of service frame

A.3.2.3 Service frame of frame type = 0

The service frame is solely used to transport the stream directory information.

Number of services (n)	1 byte
n *(SID-A, SID-B, SID-C)	n * (3 bytes)
CRC	2 bytes

<StreamDirectory<ServiceFrame(0)>>:=	: Stream directory
<IntUnTi>(n)**,**	: Number of services
n * **<ServiceIdentifier>,**	: Any number of Service IDs
<CRC>;	: CRC of Service IDs

A.3.2.4 Service frame of frame type = 1

Each service frame comprises:

SID-A, SID-B, SID-C	3 bytes	(See Clause A.3.4.2)
The encryption indicator	1 byte	(See Clause A.3.4.1)
The component data	m bytes	

The service level is defined by the service frame. Each transport frame carries one and only one service frame. The service frame includes a component multiplex comprising one or more component frames.

Each service frame may contain a different range and number of component frames as required by the service provider.

Each transport frame may be used by only one service provider and one dedicated service, which supports a mixture of applications. A multiplex of service providers or services is realized by concatenation of multiple transport frames. Each service frame includes service information that comprises the service identification elements and the encryption indicator.

<ConventionalData<ServiceFrame(1)>>:=	: Conventional data
<ServiceIdentifier>,	: Service identification
<IntUnTi>(encIdentifier)**,**	: Encryption indicator n. 0 = no encryption
f_n**(<ServCompMultiplex>);**	: Function f_n (…) is utilized according to the chosen encryption algorithm

A.3.2.5 TPEG service component frame multiplex

The component multiplex is a collection of one or more component frames, the type and order of which are freely determined by the service provider. The resultant multiplex is transformed according to the encryption method required (if the encryption indicator is not 0) or is left unchanged (if the encryption indicator = 0). The length of the resultant data must be less than or equal to 65531 bytes.

<ServCompMultiplex>:=	
n * **<ServCompFrame>**(data)**;**	: Any number of any component frames

A.3.2.6 Interface to application specific frames

The service component frame introduces the application specific code. This means further details of the data stream are specified by the application specification. In the history for different needs slightly different frames have been defined in the existing application specifications. To harmonize this kind of frames, especially for new developments of specifications, this clause specifies not only a basic frame, which is required for any application but also a selection of possible other frames, whereof an application can just choose one without the need to specify its own frame.

An application specification, however, can specify its own frame, which shall at minimum include the following base service component frame as first sub type.

A.3.2.6.1 TPEG base service component frame structure

In a TPEG data stream it shall be possible to have not only one content stream but more; even different from the same application. This is possible with the help of the Service and Network Information (SNI) Application, which is served like variable directory information in the data stream. Therein a table defines a unique number for any content stream being transmitted. This includes also the definition which application is expected in one specific frame. In other words the frame starts not with a typical interface template, but with a header, defining three first values being in common with all service component frames. Therefore, any service component frame is built as shown below:

<**ServCompFrame**>:=	: Service component frame
<**ServCompFrameHeader**>(header),	: Common service component header
<**ApplicationData**>(data);	: Component data

Where the service component header is specified as:

<**ServCompFrameHeader**>:=	: Common service component frame header
<**IntUnTi**>(scId),	: Service component identifier (scid is defined by SNI service component designating the application in this service component frame)
<**IntUnLi**>(lengthInByte),	: Length, n, of component data in bytes
<**CRC**>(headerCRC);	: Header CRC (See Clause A.3.5.3)

At the component level data is carried in component frames which have a limited length. If applications require greater capacity then the application must be designed to distribute data between component frames and to recombine this information in the decoder.

The inclusion of the field length enables the decoder to skip a component.

The maximum field length of the component data (assuming that there is no transformation, and only one component is included in the service frame) = 65526.

A.3.2.6.2 TPEG specialized service component data schemata

It is in interest of consistency to make sure that service component frames still become defined in as similar as possible in different applications. Specifically with three further attributes being of general nature. The following proposed specialized service component data schemata can be used to inform on this general level about following information:

b) The application data of a component frame with **dataCRC** is error-free.

Data CRC on this level makes it possible, that in case of errors only the service component frame (e.g. one relatively small package of data) would be lost. Other parts of the service multiplex may still be valid and could still be used. (See Clause A.3.5.4)

c) Count of messages the service component frame contains named **messageCount**.

Sometimes it is useful not only to know the opaque count of bytes, but also how many different message have to be expected by the decoder (e.g. for displaying purpose).

d) Prioritization can be made by assigning a **groupPriority**.

In some cases the different service components received shall not just be handled by a FIFO buffer but also with some qualification of priority of messages. In this case high priority message may take precedence over other messages in the decoder. These may be presented to the user even before low priority messages are decoded.

A.3.2.6.2.1 Service component data with dataCRC

Any application should at least specify a data CRC as defined in Clause A.3.5.4 at the end of application data ensuring that bit errors can be detected on service component frame level.

< **ServCompFrameProtected** >:=	: CRC protected service component frame
<**ServCompFrameHeader**>(header),	: Component frame header as defined in A.3.2.6.1
external <**ApplicationContent>**(content)**,**	: Content specified by the individual application
<**CRC**>(dataCRC);	: CRC starting with first byte after the header

A.3.2.6.2.2 Service component data with dataCRC and messageCount

This service frame is used for applications containing messages more or less directly presented to the user which indicate already on frame level how many messages are to be expected. Data CRC is contained as well.

< **ServCompFrameCountedProtected**>:=	: CRC protected service component frame with message count
<**ServCompFrameHeader**>(header),	: Component frame header as defined in A.3.2.6.1
<**IntUnTi**>(messageCount),	: count of messages in this ApplicationContent
external <**ApplicationContent**>(content),	: actual payload of the application
<**CRC**>(dataCRC);	: CRC starting with first byte after the header

A.3.2.6.2.3 Service component data with dataCRC and groupPriority

When messages need to be grouped by priority, this service component frame is used. If not all messages within the frame have the same priority, 'typ007_000: undefined' shall be used. Data CRC is contained as well.

< **ServCompFramePrioritisedProtected**>:=	: CRC protected service component frame with message count
<**ServCompFrameHeader**>(header),	: Component frame header as defined in A.3.2.6.1
<**typ007:Priority**>(groupPriority),	: group priority applicable to all messages in this ApplicationContent
external <**ApplicationContent**>(content),	: actual payload of the application
<**CRC**>(dataCRC);	: CRC starting with first byte of after the header

A.3.2.6.2.4 Service component frame with dataCRC, groupPriority, and messageCount

Additionally, an application can also make use of all features described in previous clauses.

< **ServCompFramePrioritisedCountedProtected**>:=	: CRC protected service component frame with group priority and message count
<**ServCompFrameHeader**>(header),	: Component frame header as defined in A.3.2.6.
<**typ007:Priority**>(groupPriority),	: group priority applicable to all messages in the ApplicationContent
<**IntUnTi**>(messageCount),	: count of messages in this ApplicationContent
external <**ApplicationContent**>(content),	: actual payload of the application
<**CRC**>(dataCRC);	: CRC starting with first byte after the header

A.3.2.6.3 Example of an application implementing a service component frame

An application specification is required to specify first the component frame just as a written sentence. It may for information repeat the definition of the frame, but in this case it shall add a note, that this definition can be superseded by a future release of this specification.

As second definition tree of application starts with:

<**ApplicationContent**>:=	: link provided by SSF
n * <**MyComponent**>(comp);	: n root components of the application

<MyComponent<Component(0)>>:=	
<IntUnTi>(0),	: id = 1
<IntUnLoMB>(compLengthInByte),	: length of the component in bytes
<IntUnLoMB>(attributeBlockLengthInByte),	: length of the attribute block in bytes
<ShortString>(myText),	: some first attribute of the application
<SubComp>(sub);	: some sub components of Component(0)

A.3.3 Description of data on Transport level

A.3.3.1 Syncword

The syncword is 2 bytes long, and has the value of FF0F hex.

The nibbles F hex and 0 hex have been chosen for simplicity of processing in decoders. The patterns 0000 hex and FFFF hex were deprecated to avoid the probability of false triggering in the cases of some commonly used transmission channels.

A.3.3.2 Field length

The field length consists of 2 bytes and represents the number of bytes in the service frame.

This derives from the need of variable length frames.

A.3.3.3 Header CRC

The Header CRC is two bytes long, and is based on the ITU-T polynomial $x^{16} + x^{12} + x^5 + 1$. The Header CRC is calculated on 16 bytes including the syncword, the field length, the frame type and the first 11 bytes of the

service frame. In the case that a service frame is shorter than 11 bytes, the sync word, the field length, the frame type and the *whole* service frame shall be taken into account.

In this case the Header CRC calculation does not run into the next transport frame.

The calculation of the CRC is described in CEN ISO/TS 18234-2, Annex C.

A.3.3.4 Frame type

The frame type (FTY) indicates the content of the service frame. Its length is 1 byte. The following table gives the meaning of the frame type:

FTY value (dec):	Content of service frame:	Kind of information in service frame:
0	Number of services, n * (SID-A, SID-B, SID-C)	Stream directory information
1	SID-A, SID-B, SID-C, Encryption ID, Component Multiplex	Conventional service frame data

If FTY = 0, an extra CRC calculation is done over the whole service frame, i.e. starting with n (number of services) and ending with the last SID-C of the last service.

The calculation of the CRC is described in CEN ISO/TS 18234-2, Annex C.

A.3.3.5 Synchronization method

A three-step synchronization algorithm can be implemented to synchronize the receiver:

a) search for an FF0F hex value;

b) calculate and check the header CRC, which follows;

c) check the two bytes, which follow the end of the service frame as defined by the field length.

The two bytes following the end of the service frame should either be a sync word or 00 hex, when spaces are inserted.

A.3.3.6 Error detection

The CRC header provides error detection and protection for the synchronization elements and not for the data within the service frame (except the first 11 bytes, when applicable).

A.3.4 Description of data on Service level

A.3.4.1 Encryption indicator

Length: 1 byte

The encryption indicator is defined as one byte according to TPEG primitive syntax. If the indicator has value 00 hex all data in the component multiplex are non-encrypted. Every other value of the encryption indicator indicates that one of several mechanisms for data encryption or compression has been utilized for all data in the following data multiplex. The encryption/compression technique and algorithms may be freely chosen by the service provider.

0 = no encryption/compression

1 to 127 = reserved for standardized methods

128 to 255 = may be freely used by each service provider, may indicate the use of proprietary methods

A.3.4.2 Service identification

The service IDs are structured in a similar way to Internet IP addresses as follows:

SID-A . SID-B . SID-C

The combination of these three SID elements must be uniquely allocated on a worldwide basis.

The following address allocation system applies:

— SID range for TPEG technical tests SIDs = 000.000.000 - 000.127.255

— SID range for TPEG public tests SIDs = 000.128.000 - 000.255.255

— SID range for TPEG regular public services SIDs = 001.000.000 - 100.255.255

— SID range: reserved for future use SIDs = 101.000.000 - 255.255.255

NOTE The above allocations and structure is significantly changed from that originally specified in CEN ISO/TS 18234-2.

A.3.5 Description of data on Service component level

A.3.5.1 Service component identifier

The service component identifier with the value 0 is reserved for the SNI Application. (See CEN ISO/TS 18234-3).

A.3.5.2 Field length

The field length consists of 2 bytes and represents the number of bytes of the component data.

A.3.5.3 Service component frame header CRC

The component header CRC is two bytes long, and based on the ITU-T polynomial $x^{16}+x^{12}+x^5+1$.

The component header CRC is calculated from the service component identifier, the field length and the first 13 bytes of the component data. In the case of component data shorter than 13 bytes, the component identifier, the field length and all component data shall be taken into account.

The calculation of the CRC is described in CEN ISO/TS 18234-2 Annex C.

A.3.5.4 Service component frame data CRC

The DataCRC is two bytes long, and is based on the ITU polynomial $x^{16}+x^{12}+x^5+1$. This CRC is calculated from all the bytes of the service component frame data after the service component frame header.

The calculation of the CRC is described in CEN ISO/TS 18234-2 Annex C.

A.4 General binary data types

This clause describes the primitive elements and compound elements that are used by TPEG applications.

A.4.1 Primitive data types

The fundamental data element in TPEG technology is the byte, which is represented by 8 bits. All other primitive data types are expressed in terms of bytes as follows:

A.4.1.1 Basic numbers:

The following data type represent the general notation of integral numbers either coded as signed or unsigned.

<IntUnTi>:=	: Integer Unsigned Tiny, range 0..255
<byte>;	: Primitive

<IntSiTi>:=	: Integer Signed Tiny, range -128..(+)127
<byte>;	: Two's complement

<IntUnLi>:=	: Integer Unsigned Little, range 0..65 535
<byte>,	: MSB, Most Significant Byte
<byte>;	: LSB, Least Significant Byte

<IntSiLi>:=	: Integer Signed Little, range -32 768..(+)32 767
<byte>,	: MSB, Two's complement
<byte>;	: LSB, Two's complement

<IntUnLo>:=	: Integer Unsigned Long, range 0..4 294 967 295
<byte>,	: MSB
<byte>,	
<byte>,	
<byte>;	: LSB

<IntSiLo>:=	: Integer Signed Long, range -2 147 483 648..(+)2 147 483 647
<byte>,	: MSB, Two's complement
<byte>,	
<byte>,	
<byte>;	: LSB, Two's complement

A.4.1.2 MultiByte

A.4.1.2.1 Unsigned Long MultiByte

A multi-byte integer consists of a series of bytes, where the most significant bit is the continuation flag and the remaining seven bits are a scalar value. The continuation flag indicates that a byte is not the end of the multi-byte sequence. A single integer value is encoded into a sequence of N bytes. The first N-1 bytes have the continuation flag set to a value of one (1). The final byte in the series has a continuation flag value of zero (0). This allows to know exactly the end of a series of bytes belonging to one multi-byte, being the one with MSB=0.

The bytes are encoded in “big-endian” order i.e. most significant byte first. The maximum number of concatenated bytes is 5, so that the maximum unsigned integer, which can be encoded is $2^{(40-5)}$ - 1. However, this specification defines the three most significant bits of the fifth, most significant byte as “reserved for future use”, to be set to 000. This leads into the maximum number $2^{(32)}$ - 1 which is the maximum value of a four byte unsigned integer.

<IntUnLoMB>:=	: Integer Unsigned Long, range 0..4 294 967 295
m***<byte>**[1..5];	: MS-Bit = 1 signals one more byte follows.

EXAMPLE

Positive number to be encoded

— in Decimal: 1093567633

— Binary (32bit): 0100 0001001 0111010 0001001 0010001

Multibyte encoded:

Byte [5] (MSB)	Byte [4]	Byte [3]	Byte [2]	Byte [1] (LSB)
(1)"000"0100	(1)0001001	(1)01110100	(1)0001001	(0)0010001

Bits in brackets are continuity flags, the ones in quotes are reserved and set to 0.

Multibyte encoded hex: 8489ba8911.

A.4.1.2.2 Signed Long MultiByte

The signed multi-byte is defined in the same way as IntUnLoMB except in case of signed value interpretation; the complement on two is used on the 7 bit wide byte series. The count of bytes is then defined by the magnitude of the positive value to be stored in multi-byte. The three reserved bits in byte #5 shall be set to 111 in case of negative numbers with 5 byte length and 000 otherwise, to be up-ward compatible in case of introduction of a 64-bit integer value in future. Signed values from 0 to -2^6 are stored in one byte, to -2^{13} in two bytes, to -2^{20} in three bytes, to -2^{27}in four bytes and to -2^{32} in five bytes.

For example a value 0x62 (0110 0010) would be encoded with the one byte 0x62. The integer value 0xA7 (1010 0111) would be encoded with a two-byte sequence 0x8127. The signed representation of -1 is 0x7F. And -2345 is represented in two bytes so that the complement on two is 0x36D7 = (110 1101.101 0111). A serialisation in multi-byte then results in 1110 1101.0101 0111 = 0xED57.

<IntSiLoMB>:=	: Integer Signed Long, range -2 147 483 648..(+)2 147 483 647
m***<byte>**[1..5];	: Two’s complement after elimination of continuation flags.
	MS-Bit = 1 signals one more byte follows.

EXAMPLE

Positive number to be encoded

— in Decimal: 1093567633

— Binary (32bit): 0100 0001001 0111010 0001001 0010001

Multibyte encoded:

Byte [5] (MSB)	Byte [4]	Byte [3]	Byte [2]	Byte [1] (LSB)
(1)"000"0100	(1)0001001	(1)01110100	(1)0001001	(0)0010001

Bits in brackets are continuity flags, the ones in quotes are reserved and set to 0.

Multibyte encoded hex: 8489ba8911

Negative number to be encoded

— in Decimal: -1093567633

— Binary (two's complement): 1011 1110110 1000101 1110110 1101111

Multibyte encoded:

Byte [5] (MSB)	Byte [4]	Byte [3]	Byte [2]	Byte [1] (LSB)
(1)"111"'1'011	(1)1110110	(1)1000101	(1)1110110	(0)1101111

Bits in brackets are continuity flags, the ones in quotes are reserved and set to 1.

Multibyte encoded hex: FBF6C5F66F.

A.4.1.3 BitArray

This is an encoding specific data type used for encoding an array of Booleans. The bits are encoded in a sequence of bytes, where the first bit of each byte is a continuation flag (shown as c in Figure A.4). If this bit is set (=1) there follows at least one more byte in this bit array. The last byte always has this bit cleared (=0). A BitArray represents a list of Boolean values which is implemented in the same way as for all lists. The first byte holds bits numbered from zero to six in that order. The second byte holds bits numbered seven to 13, again in that order, and so on.

The ordering is sequential from first to last bit. This use, to ensure consistency with other lists, differs from the encoding of numeric values which use a Big-endian bit and byte order.

Byte 0								**Byte 1**								**...**	
	Bit NR								**Bit NR**								
C	0	1	2	3	4	5	6	C	7	8	9	10	11	12	13		...

Figure A.4 — BitArray coding format

```
<BitArray>:=
    m * <byte>[1..*];          : byte of flags; MS-Bit = 1 signals one more byte follows
```

A.4.1.4 Boolean

A single true or false value. The Boolean is differently defined for the following cases:

A.4.1.4.1 Mandatory Boolean

Mandatory Booleans are defined directly in the selector bitarray used for signalling optional attributes. This saved additional bytes for Boolean values. If bit x of selector is set, the Boolean value is interpreted as being true, otherwise false.

A.4.1.4.2 Mandatory Multiple Booleans

For multiple Boolean values (Boolean value with multiplicity higher then 1) the coding requires as first a multibyte "n" as counter how many bit of the then following extra bitarray are in use. The bitarray then contains n valid bits, coding the same as single Booleans. If n = 0 the bitarray attribute is not existing.

A.4.1.5 Date and time information:

The number of seconds elapsed since the start 1970-01-01T00:00:00 Universal Coordinated Time (UTC). This gives values for 136 years until 2106, i. e. 2^{32} seconds from the year 1970.

The exact formula for the date and time calculation can be found in CEN ISO/TS 18234-2 Annex D.

<DateTime>:=	: Date and time
<IntUnLo>;	: Number of seconds since 1970-01-01T00:00:00 Universal Coordinated Time (UTC)

A.4.1.5.1 Day selection

This type gives the possibility to select one or more days of the week to indicate the repetition of an event.

A DaySelector attribute can be used to select one or more week days. The Boolean attributes indicate whether a particular day is included in the selection, if the attribute value is "true", the day is selected. These seven attributes are mandatory Booleans, encoded using a BitArray.

<DaySelector>:=	
<BitArray>(selector),	: DaySelector
if (bit **0** of selector is set)	
<Boolean>(saturday);	: every Saturday
if (bit **1** of selector is set)	
<Boolean>(friday),	: every Friday
if (bit **2** of selector is set)	
<Boolean>(thursday),	: every Thursday
if (bit **3** of selector is set)	
<Boolean>(wednesday),	: every Wednesday
if (bit **4** of selector is set)	
<Boolean>(tuesday),	: every Tuesday
if (bit **5** of selector is set)	
<Boolean>(monday),	: every Monday
if (bit **6** of selector is set)	
<Boolean>(sunday),	: every Sunday

EXAMPLE 1 **<DaySelector>** = 05 hex - Meaning: The event (e. g. service) is repeated every Sunday and Tuesday.

EXAMPLE 2 **<DaySelector>** = 7E hex - Meaning: The event (e. g. service) is repeated every day except Sunday.

A.4.1.5.2 Duration

Values of this type define temporal duration, expressed in a number of whole seconds. Values must be between 0 and 4294967295. Because it is expected that in many cases the amount of the value is low, variable length coding is used.

<Duration>:=	: Time duration
<IntUnLoMB>;	: Number of seconds

A.4.1.6 DistanceMetres

Distance in integer units of metres.

<DistanceMetres>:=	
<IntUnLoMB>;	: Distance in integer units of metres.

A.4.1.7 DistanceCentiMetres

Distance in integer units of centimetres.

<DistanceCentiMetres>:=	
<IntUnLoMB>;	: Distance in integer units of centimetres.

A.4.1.8 CRC-word data type

The Cyclic redundancy check (CRC) is a calculated hash value over a defined array of bytes. The definition of a CRC must include the definition of the array.

<CRC>:=	: Cyclic redundancy check
<IntUnLi>;	: According to ITU-T polynomial, over an indicated Range of elements. (See CEN ISO/TS 18234-2 Annex C)

A.4.1.9 FixedPercentage

FixedPercentage defines a fixed percentage value in integer units in the range 0 and 100.

A fixed percentage can *not* be used as an indication of a change, where both negative values and values larger than a 100% might be required.

<FixedPercentage>:=	
<IntUnTi>;	: valid values of percentage from 0 to 100

A.4.1.10 Probability

Probability defines a percentage value between zero and one with a precision of two decimals. Where zero denotes no probability and one hundred certainty.

<Probability>:=	
<FixedPercentage>;	: valid values from 0 to 100

A.4.1.11 Float

A float defines a number with decimal precision. It is encoded as an IEC 60559 single precision floating point number (32 bit).

<Float>:=	
<IntUnLo>;	: IEC 60559 single precision floating point number

A.4.1.12 Severity

Severity is application specific and defined in the range from 1 to 255 where higher values are expected to be more severe. Value 0 is predefined as undefined.

<Severity>:=	
<IntUnTi>;	: Application specific value of severity

A.4.1.13 Velocity

Velocity in integer units of metres per second in the range from 0 to 255.

<Velocity>:=	
<IntUnTi>;	: Speed in m/s

A.4.1.14 Weight

Weight in integer units of kilogram's. The value is in the range from 0 to 4294967295, encoded as IntUnLoMB.

<Weight>:=	
<IntUnLoMB>;	: Weight in kg

A.4.2 Compound data types

A.4.2.1 ServiceIdentifier

A service identifier is an data type that defines a single service identifier.

<ServiceIdentifier>:=	
<IntUnTi>(sidA),	: Service identification part A
<IntUnTi>(sidB),	: Service identification part B
<IntUnTi>(sidC);	: Service identification part C

A.4.2.2 FixedPointNumber

Defines a value from -2147483648.99 to 2147483647.99 with a fixed precision of 2 decimals.

<FixedPointNumber>:=	
<IntSiLoMB>(integralPart),	: integral part of the number
<IntUnTi>(decimalPart);	: fraction of 2 decimal digits values from 0 to 99

A.4.2.3 Strings

The string of characters is represented by a series of n bytes. These bytes need to be interpreted according to a character table, which will designate the byte width of each character. The encoding of the characters is defined in CEN ISO/TS 18234-3. Where multiple code tables are used, an application needs mechanisms to set the scope of applicability of each table.

<ShortString>:=	: Short string
<IntUnTi>(n),	: Number of bytes, n
n * **<byte>;**	: String of characters; count of characters depends on chosen coding

<LongString>:=	: Long string
<IntUnLi>(n)**,**	: Number of bytes, n
n * **<byte>;**	: String of characters; count of characters depends on chosen coding

A.4.2.4 Localised Strings

A string accompanied with a language code that identifies the language that the string is given in. The typ001:LanguageCode is derived from ISO 639:2002 - Codes for the representation of names of languages.

<LocalisedShortString>:=	
<typ001:LanguageCode>(languageCode),	: Specifies the language used for this string.
<ShortString>(string);	: Short string

<LocalisedLongString>:=	
<typ001:LanguageCode>(languageCode),	: Specifies the language used for this string.
<LongString>(string);	: Long string

A.4.2.5 Compound time information

A.4.2.5.1 TimeInterval

The TimeInterval data structure can be used when an interval in time must be specified with more flexibility than the simple Duration type allows.

Each TimeInterval attribute has a number of optional attributes. It is maximally 101 years long. Each attribute can be used as stand-alone attribute or in combination with other attributes. When an attribute is not given, the value zero is implied. Every TimeInterval must specify at least one attribute.

<TimeInterval>:=	
<BitArray>(selector),	: DaySelector
if (bit 0 of selector is set)	
<IntUnTi>(years),	: Number of years in the range from 0 to 100.
if (bit 1 of selector is set)	
<IntUnTi>(months),	: Number of months in the range from 0 to 12.
if (bit 2 of selector is set)	
<IntUnTi>(days),	: Number of days in the range from 0 to 31.
if (bit 3 of selector is set)	
<IntUnTi>(hours),	: Number of hours in the range from 0 to 24.
if (bit 4 of selector is set)	
<IntUnTi>(minutes),	: Number of minutes in the range from 0 to 60.
if (bit 5 of selector is set)	
<IntUnTi>(seconds);	: Number of seconds in the range from 0 to 60.

A.4.2.5.2 TimePoint

The TimePoint data structure can be used when a point in time must be specified with fewer granularities than the simple DateTime allows. Each TimePoint attribute has a number of optional attributes. Each attribute can be used as stand-alone attribute or in combination with other attributes. Every TimePoint must specify at least one attribute. In binary encoding, 1970 is subtracted from the year, mapping the range 1970-2100 to the values 0-130.

<TimePoint>:=	
<BitArray>(selector),	: DaySelector
if (bit 0 of selector is set)	
<IntUnTi>(year),	: The year in the range from 1970 to 2100.
if (bit 1 of selector is set)	
<IntUnTi>(month),	: Number of months in the range from 1 to 12.
if (bit 2 of selector is set)	
<IntUnTi>(day),	: Number of days in the range from 1 to 31.
if (bit 3 of selector is set)	
<IntUnTi>(hour),	: Number of hours in the range from 0 to 23.
if (bit 4 of selector is set)	
<IntUnTi>(minute),	: Number of minutes in the range from 0 to 59.
if (bit 5 of selector is set)	
<IntUnTi>(second);	: Number of seconds in the range from 0 to 59.

A.4.2.5.3 TimeToolkit

The time toolkit allows different date and time information to be described. For example where an event has a start but no known end-time. In such a case we should used only a start point but omit an end-time. Each TimeToolkit attribute has a number of optional attributes. Each attribute can be used as a stand-alone attribute or in combination with other attributes. Every TimeToolkit must specify at least one attribute. For a timestamp the DateTime type should be used.

<TimeToolkit>:=	
<BitArray>(selector),	: 1 byte containing 5 switches
if (bit 0 of *selector* is set)	
<TimePoint>(startTime),	: An event time point (e.g. flight departure) or an event starting time (e.g. open from)
if (bit 1 of *selector* is set)	
<TimePoint>(stopTime),	: An event stopping time (e.g. open to). The stop time can be used only together with a start time
if (bit 2 of *selector* is set)	
<TimeInterval>(duration),	: A time interval (e.g. free parking limit)
if (bit 3 of *selector* is set)	
<typ002:SpecialDay>(specialDay),	: Relevant days of a certain type (e.g. weekdays or holiday)
if (bit 4 of *selector* is set)	
<DaySelector>(daySelector);	: Gives the option to specify days of the week

A.4.3 Table definitions

A.4.3.1 Table entrye

In TPEG much information is based on tables. These tables represent clearly defined groupings of pre-defined concepts. The idea is to inform the device about the concept and let the device choose the best possible presentation of this concept in the context of the other parts of the TPEG message. This approach means that devices can present concepts e.g. in any language or even as graphical icons. This Table data type only serves as a basis for all tables used in the toolkits and applications.

A table can have up to 256 entries.

<Table>:=	
<IntUnTi>(entry)**;**	: The corresponding table defines valid entries of a table

A.4.3.2 Tables of general use

Some tables are of general use in different TPEG Applications, therefore this clause describes the content of those tables.

A.4.3.3 Typ001:LanguageCode

ISO 639:2002 - Codes for the representation of names of languages. See Clause A.4.4.1 for values.

<Typ001:LanguageCode>:=	
<Table>;	: Specifies the language

A.4.3.4 Typ002:SpecialDay

The SpecialDay table lists special types of days, such as “public holiday” or “weekdays” and similar. See Clause A.4.4.2 for values.

<Typ002:SpecialDay>:=	
<Table>;	: Identifies the special day

A.4.3.5 Typ003:CurrencyType

CurrencyType, based on the three-alpha codes of ISO 4217. See Clause A.4.4.3 for values.

<Typ003:CurrencyType>:=	
<Table>;	: Three-alpha codes of ISO 4217

A.4.3.6 Typ004:NumericalMagnitude

At a number of places within TPEG’s applications there is a need to use a number to describe a quantity of people, animals, objects, etc. The range of the number needs to be at least from 0 to a few million. At the bottom end of this range, numbers need to be in unit intervals, up to 50. Above 50, tens may be used up to 500, then hundreds up to 5000. This same principle is required for each decade. The table contains unsigned integer values in the range 0 to 3000000 with decreasing precision. See Clause A.4.4.4 for the translated values. For a formal mathematical definition of numerical magnitude values, refer to Annex B of ISO/TS 18234-2.

<Typ004:NumericalMagnitude>:=	
<Table>(n)**;**	: Numerical magnitude

A.4.3.7 Typ005:CountryCode

This table lists countries as defined by ISO 3166-1. See Clause A.4.4.5 for values.

NOTE "undecodable country" is to be used by a client device unable to read the typ005 code used by a service provider - no code value for this word is ever transmitted.

<Typ005:CountryCode>:=	
<Table>;	: Countries as defined by ISO 3166-1

A.4.3.8 Typ006:OrientationType

This is the table of values of the compass orientation like “north-west”. See Clause A.4.4.6 for values.

<Typ006:OrientationType>:=	
<Table>;	: Denotes a compass orientation

A.4.3.9 Typ007:Priority

This is the table of values of the priority of messages. See Clause a.4.4.7 for values.

<Typ007:Priority>:=	
<Table>;	: Denotes priority of a message

A.4.4 Tables

A.4.4.1 typ001:LanguageCode

ISO 639:2002 - Codes for the representation of names of languages. The Comment column lists the 2-alpha codes of ISO 639-1.

Code	Reference-English Language Name	Comment 2-alpha code
000	Unknown	
001	Afar	aa
002	Abkhazian	ab
003	Avestan	ae
004	Afrikaans	af
005	Akan	ak
006	Amharic	am
007	Aragonese	an
008	Arabic	ar
009	Assamese	as
010	Avaric	av
011	Aymara	ay
012	Azerbaijani	az
013	Bashkir	ba
014	Belarusian	be
015	Bulgarian	bg
016	Bihari	bh
017	Bislama	bi
018	Bambara	bm
019	Bengali	bn
020	Tibetan	bo

021	Breton	br
022	Bosnian	bs
023	Catalan	ca
024	Chechen	ce
025	Chamorro	ch
026	Corsican	co
027	Cree	cr
028	Czech	cs
029	Church Slavic	cu
030	Chuvash	cv
031	Welsh	cy
032	Danish	da
033	German	de
034	Divehi	dv
035	Dzongkha	dz
036	Ewe	ee
037	Greek	el
038	English	en
039	Esperanto	eo
040	Spanish	es
041	Estonian	et
042	Basque	eu
043	Persian	fa
044	Fulah	ff
045	Finnish	fi
046	Fijian	fj
047	Faroese	fo
048	French	fr
049	Western Frisian	fy
050	Irish	ga
051	Scottish Gaelic	gd
052	Galician	gl
053	Guaraní	gn
054	Gujarati	gu
055	Manx	gv
056	Hausa	ha
057	Hebrew	he
058	Hindi	hi
059	Hiri Motu	ho
060	Croatian	hr
061	Haitian	ht
062	Hungarian	hu
063	Armenian	hy
064	Herero	hz
065	Interlingua (International Auxiliary Language Association)	ia
066	Indonesian	id
067	Interlingue	ie
068	Igbo	ig
069	Sichuan Yi	ii
070	Inupiaq	ik

071	Ido	io
072	Icelandic	is
073	Italian	it
074	Inuktitut	iu
075	Japanese	ja
076	Javanese	jv
077	Georgian	ka
078	Kongo	kg
079	Kikuyu	ki
080	Kuanyama	kj
081	Kazakh	kk
082	Kalaallisut	kl
083	Khmer	km
084	Kannada	kn
085	Korean	ko
086	Kanuri	kr
087	Kashmiri	ks
088	Kurdish	ku
089	Komi	kv
090	Cornish	kw
091	Kirghiz	ky
092	Latin	la
093	Luxembourgish	lb
094	Ganda	lg
095	Limburgish	li
096	Lingala	ln
097	Lao	lo
098	Lithuanian	lt
099	Luba-Katanga	lu
100	Latvian	lv
101	Malagasy	mg
102	Marshallese	mh
103	Ma-ori	mi
104	Macedonian	mk /sl
105	Malayalam	ml
106	Mongolian	mn
107	Moldavian	mo
108	Marathi	mr
109	Malay	ms
110	Maltese	mt
111	Burmese	my
112	Nauru	na
113	Norwegian Bokmål	nb
114	North Ndebele	nd
115	Nepali	ne
116	Ndonga	ng
117	Dutch	nl
118	Norwegian Nynorsk	nn
119	Norwegian	no
120	South Ndebele	nr

121	Navajo	nv
122	Chichewa	ny
123	Occitan	oc
124	Ojibwa	oj
125	Oromo	om
126	Oriya	or
127	Ossetian	os
128	Panjabi	pa
129	Pa-li	pi
130	Polish	pl
131	Pashto	ps
132	Portuguese	pt
133	Quechua	qu
134	Raeto-Romance	rm
135	Kirundi	rn
136	Romanian	ro
137	Russian	ru
138	Kinyarwanda	rw
139	Sanskrit	sa
140	Sardinian	sc
141	Sindhi	sd
142	Northern Sami	se
143	Sango	sg
144	Serbo-Croatian	sh
145	Sinhalese	si
146	Slovak	sk
147	Slovenian	sl
148	Samoan	sm
149	Shona	sn
150	Somali	so
151	Albanian	sq
152	Serbian	sr
153	Swati	ss
154	Southern Sotho	st
155	Sundanese	su
156	Swedish	sv
157	Swahili	sw
158	Tamil	ta
159	Telugu	te
160	Tajik	tg
161	Thai	th
162	Tigrinya	ti
163	Turkmen	tk
164	Tagalog	tl
165	Tswana	tn
166	Tonga	to
167	Turkish	tr
168	Tsonga	ts
169	Tatar	tt
170	Twi	tw

171	Tahitian	ty
172	Uighur	ug
173	Ukrainian	uk
174	Urdu	ur
175	Uzbek	uz
176	Venda	ve
177	Vietnamese	vi
178	Volapük	vo
179	Walloon	wa
180	Wolof	wo
181	Xhosa	xh
182	Yiddish	yi
183	Yoruba	yo
184	Zhuang	za
185	Chinese	zh
186	Zulu	zu

A.4.4.2 typ002:SpecialDay

The SpecialDay table lists special types of days, such as public holiday and similar.

Code	Reference-English 'word'	Comment	Example
000	unknown		
001	weekdays	Monday to Friday	
002	weekends	Saturday and Sunday	
003	holiday		
004	public holiday		
005	religious holiday		e.g. Christmas Day
006	federal holiday		
007	regional holiday		
008	national holiday		e.g. In UK: Mayday
009	school days		
010	every day		

A.4.4.3 typ003:CurrencyType

CurrencyType, based on the three-alpha codes of ISO 4217.

Code	Reference-English 'word'	Comment
000	unknown	
001	AED	
002	AFA	
003	ALL	
004	AMD	
005	ANG	
006	AOA	
007	ARS	
008	AUD	
009	AWG	
010	AZM	

011	BAM	
012	BBD	
013	BDT	
014	BGN	
015	BHD	
016	BIF	
017	BMD	
018	BND	
019	BOB	
020	BRL	
021	BSD	
022	BTN	
023	BWP	
024	BYR	
025	BZD	
026	CAD	
027	CDF	
028	CHF	
029	CLP	
030	CNY	
031	COP	
032	CRC	
033	CSD	
034	CUP	
035	CVE	
036	CYP	
037	CZK	
038	DJF	
039	DKK	
040	DOP	
041	DZD	
042	EEK	
043	EGP	
044	ERN	
045	ETB	
046	EUR	
047	FJD	
048	FKP	
049	GBP	
050	GEL	
051	GGP	
052	GHC	
053	GIP	
054	GMD	
055	GNF	
056	GTQ	
057	GYD	
058	HKD	
059	HNL	
060	HRK	

061	HTG	
062	HUF	
063	IDR	
064	ILS	
065	IMP	
066	INR	
067	IQD	
068	IRR	
069	ISK	
070	JEP	
071	JMD	
072	JOD	
073	JPY	
074	KES	
075	KGS	
076	KHR	
077	KMF	
078	KPW	
079	KRW	
080	KWD	
081	KYD	
082	KZT	
083	LAK	
084	LBP	
085	LKR	
086	LRD	
087	LSL	
088	LTL	
089	LVL	
090	LYD	
091	MAD	
092	MDL	
093	MGA	
094	MKD	
095	MMK	
096	MNT	
097	MOP	
098	MRO	
099	MTL	
100	MUR	
101	MVR	
102	MWK	
103	MXN	
104	MYR	
105	MZM	
106	NAD	
107	NGN	
108	NIO	
109	NOK	
110	NPR	

111	NZD	
112	OMR	
113	PAB	
114	PEN	
115	PGK	
116	PHP	
117	PKR	
118	PLN	
119	PYG	
120	QAR	
121	ROL	
122	RUR	
123	RWF	
124	SAR	
125	SBD	
126	SCR	
127	SDD	
128	SEK	
129	SGD	
130	SHP	
131	SIT	
132	SKK	
133	SLL	
134	SOS	
135	SPL	
136	SRD	
137	STD	
138	SVC	
139	SYP	
140	SZL	
141	THB	
142	TJS	
143	TMM	
144	TND	
145	TOP	
146	TRL	
147	TTD	
148	TVD	
149	TWD	
150	TZS	
151	UAH	
152	UGX	
153	USD	
154	UYU	
155	UZS	
156	VEB	
157	VND	
158	VUV	
159	WST	
160	XAF	

161	XAG	
162	XAU	
163	XCD	
164	XDR	
165	XOF	
166	XPD	
167	XPF	
168	XPT	
169	YER	
170	ZAR	
171	ZMK	
172	ZWD	
255	undefined	

A.4.4.4 typ004:NumericalMagnitude

The numerical magnitude, or "numag", table contains unsigned integer values in the range 0 to 3000000 with decreasing precision. The relationship between the table entry number and the value is described in Clause A.4.3.6.

NOTE For clarity and to make this table easier to read it is shown on the next page as one complete table, not spread across multiple pages.

Code n:	r:
000	0
001	1
002	2
003	3
004	4
005	5
006	6
007	7
008	8
009	9
010	10
011	11
012	12
013	13
014	14
015	15
016	16
017	17
018	18
019	19
020	20
021	21
022	22
023	23
024	24
025	25
026	26
027	27
028	28
029	29
030	30
031	31
032	32
033	33
034	34
035	35
036	36
037	37
038	38
039	39
040	40
041	41
042	42
043	43
044	44
045	45
046	46
047	47
048	48
049	49
050	50
051	60

Code n:	r:
052	70
053	80
054	90
055	100
056	110
057	120
058	130
059	140
060	150
061	160
062	170
063	180
064	190
065	200
066	210
067	220
068	230
069	240
070	250
071	260
072	270
073	280
074	290
075	300
076	310
077	320
078	330
079	340
080	350
081	360
082	370
083	380
084	390
085	400
086	410
087	420
088	430
089	440
090	450
091	460
092	470
093	480
094	490
095	500
096	600
097	700
098	800
099	900
100	1000
101	1100
102	1200
103	1300

Code n:	r:
104	1400
105	1500
106	1600
107	1700
108	1800
109	1900
110	2000
111	2100
112	2200
113	2300
114	2400
115	2500
116	2600
117	2700
118	2800
119	2900
120	3000
121	3100
122	3200
123	3300
124	3400
125	3500
126	3600
127	3700
128	3800
129	3900
130	4000
131	4100
132	4200
133	4300
134	4400
135	4500
136	4600
137	4700
138	4800
139	4900
140	5000
141	6000
142	7000
143	8000
144	9000
145	10000
146	11000
147	12000
148	13000
149	14000
150	15000
151	16000
152	17000
153	18000
154	19000
155	20000

Code n:	r:
156	21000
157	22000
158	23000
159	24000
160	25000
161	26000
162	27000
163	28000
164	29000
165	30000
166	31000
167	32000
168	33000
169	34000
170	35000
171	36000
172	37000
173	38000
174	39000
175	40000
176	41000
177	42000
178	43000
179	44000
180	45000
181	46000
182	47000
183	48000
184	49000
185	50000
186	60000
187	70000
188	80000
189	90000
190	100000
191	110000
192	120000
193	130000
194	140000
195	150000
196	160000
197	170000
198	180000
199	190000
200	200000
200	200000
201	210000
202	220000
203	230000
204	240000
205	250000
206	260000

Code n:	r:
207	270000
208	280000
209	290000
210	300000
211	310000
212	320000
213	330000
214	340000
215	350000
216	360000
217	370000
218	380000
219	390000
220	400000
221	410000
222	420000
223	430000
224	440000
225	450000
226	460000
227	470000
228	480000
229	490000
230	500000
231	600000
232	700000
233	800000
234	900000
235	1000000
236	1100000
237	1200000
238	1300000
239	1400000
240	1500000
241	1600000
242	1700000
243	1800000
244	1900000
245	2000000
246	2100000
247	2200000
248	2300000
249	2400000
250	2500000
251	2600000
252	2700000
253	2800000
254	2900000
255	3000000

A.4.4.5 typ005:CountryCode

This table lists countries as defined by ISO 3166-1. The comment column shows the corresponding ISO 3166-1 2-alpha code elements.

NOTE "undecodable country" is to be used by a client device unable to read the typ005 code used by a service provider - no code value for this word is ever transmitted.

Code	Reference-English Country Name	Comment 2-alpha code
000	unknown	
001	Afghanistan	AF
002	Åland Islands	AX
003	Albania	AL
004	Algeria	DZ
005	American Samoa	AS
006	Andorra	AD
007	Angola	AO
008	Anguilla	AI
009	Antarctica	AQ
010	Antigua and Barbuda	AG
011	Argentina	AR
012	Armenia	AM
013	Aruba	AW
014	Australia	AU
015	Austria	AT
016	Azerbaijan	AZ
017	Bahamas	BS
018	Bahrain	BH
019	Bangladesh	BD
020	Barbados	BB
021	Belarus	BY
022	Belgium	BE
023	Belize	BZ
024	Benin	BJ
025	Bermuda	BM
026	Bhutan	BT
027	Bolivia	BO
028	Bosnia and Herzegovina	BA
029	Botswana	BW
030	Bouvet Island	BV
031	Brazil	BR
032	British Indian Ocean Territory	IO
033	Brunei Darussalam	BN
034	Bulgaria	BG
035	Burkina Faso	BF
036	Burundi	BI
037	Cambodia	KH
038	Cameroon	CM
039	Canada	CA
040	Cape Verde	CV

041	Cayman Islands	KY
042	Central African Republic	CF
043	Chad	TD
044	Chile	CL
045	China	CN
046	Christmas Island	CX
047	Cocos (Keeling) Islands	CC
048	Colombia	CO
049	Comoros	KM
050	Congo	CG
051	Congo, The Democratic Republic of the	CD
052	Cook Islands	CK
053	Costa Rica	CR
054	Côte D'ivoire	CI
055	Croatia	HR
056	Cuba	CU
057	Cyprus	CY
058	Czech Republic	CZ
059	Denmark	DK
060	Djibouti	DJ
061	Dominica	DM
062	Dominican Republic	DO
063	Ecuador	EC
064	Egypt	EG
065	El Salvador	SV
066	Equatorial Guinea	GQ
067	Eritrea	ER
068	Estonia	EE
069	Ethiopia	ET
070	Falkland Islands (Malvinas)	FK
071	Faroe Islands	FO
072	Fiji	FJ
073	Finland	FI
074	France	FR
075	French Guiana	GF
076	French Polynesia	PF
077	French Southern Territories	TF
078	Gabon	GA
079	Gambia	GM
080	Georgia	GE
081	Germany	DE
082	Ghana	GH
083	Gibraltar	GI
084	Greece	GR
085	Greenland	GL
086	Grenada	GD
087	Guadeloupe	GP
088	Guam	GU
089	Guatemala	GT
090	Guernsey	GG

091	Guinea	GN
092	Guinea-Bissau	GW
093	Guyana	GY
094	Haiti	HT
095	Heard Island and Mcdonald Islands	HM
096	Holy See (Vatican City State)	VA
097	Honduras	HN
098	Hong Kong	HK
099	Hungary	HU
100	Iceland	IS
101	India	IN
102	Indonesia	ID
103	Iran, Islamic Republic of	IR
104	Iraq	IQ
105	Ireland	IE
106	Isle of Man	IM
107	Israel	IL
108	Italy	IT
109	Jamaica	JM
110	Japan	JP
111	Jersey	JE
112	Jordan	JO
113	Kazakhstan	KZ
114	Kenya	KE
115	Kiribati	KI
116	Korea, Democratic People's Republic of	KP
117	Korea, Republic of	KR
118	Kuwait	KW
119	Kyrgyzstan	KG
120	Lao People's Democratic Republic	LA
121	Latvia	LV
122	Lebanon	LB
123	Lesotho	LS
124	Liberia	LR
125	Libyan Arab Jamahiriya	LY
126	Liechtenstein	LI
127	Lithuania	LT
128	Luxembourg	LU
129	Macao	MO
130	Macedonia, The Former Yugoslav Republic of	MK
131	Madagascar	MG
132	Malawi	MW
133	Malaysia	MY
134	Maldives	MV
135	Mali	ML
136	Malta	MT
137	Marshall Islands	MH
138	Martinique	MQ
139	Mauritania	MR
140	Mauritius	MU

141	Mayotte	YT
142	Mexico	MX
143	Micronesia, Federated States of	FM
144	Moldova, Republic of	MD
145	Monaco	MC
146	Mongolia	MN
147	Montenegro	ME
148	Montserrat	MS
149	Morocco	MA
150	Mozambique	MZ
151	Myanmar	MM
152	Namibia	NA
153	Nauru	NR
154	Nepal	NP
155	Netherlands	NL
156	Netherlands Antilles	AN
157	New Caledonia	NC
158	New Zealand	NZ
159	Nicaragua	NI
160	Niger	NE
161	Nigeria	NG
162	Niue	NU
163	Norfolk Island	NF
164	Northern Mariana Islands	MP
165	Norway	NO
166	Oman	OM
167	Pakistan	PK
168	Palau	PW
169	Palestinian Territory, Occupied	PS
170	Panama	PA
171	Papua New Guinea	PG
172	Paraguay	PY
173	Peru	PE
174	Philippines	PH
175	Pitcairn	PN
176	Poland	PL
177	Portugal	PT
178	Puerto Rico	PR
179	Qatar	QA
180	Réunion	RE
181	Romania	RO
182	Russian Federation	RU
183	Rwanda	RW
184	Saint Helena	SH
185	Saint Kitts and Nevis	KN
186	Saint Lucia	LC
187	Saint Pierre and Miquelon	PM
188	Saint Vincent and the Grenadines	VC
189	Samoa	WS
190	San Marino	SM

191	Sao Tome and Principe	ST
192	Saudi Arabia	SA
193	Senegal	SN
194	Serbia	RS
195	Seychelles	SC
196	Sierra Leone	SL
197	Singapore	SG
198	Slovakia	SK
199	Slovenia	SI
200	Solomon Islands	SB
201	Somalia	SO
202	South Africa	ZA
203	South Georgia and the South Sandwich Islands	GS
204	Spain	ES
205	Sri Lanka	LK
206	Sudan	SD
207	Suriname	SR
208	Svalbard and Jan Mayen	SJ
209	Swaziland	SZ
210	Sweden	SE
211	Switzerland	CH
212	Syrian Arab Republic	SY
213	Taiwan, Province of China	TW
214	Tajikistan	TJ
215	Tanzania, United Republic of	TZ
216	Thailand	TH
217	Timor-Leste	TL
218	Togo	TG
219	Tokelau	TK
220	Tonga	TO
221	Trinidad And Tobago	TT
222	Tunisia	TN
223	Turkey	TR
224	Turkmenistan	TM
225	Turks and Caicos Islands	TC
226	Tuvalu	TV
227	Uganda	UG
228	Ukraine	UA
229	United Arab Emirates	AE
230	United Kingdom	GB
231	United States	US
232	United States Minor Outlying Islands	UM
233	Uruguay	UY
234	Uzbekistan	UZ
235	Vanuatu	VU
236	Venezuela	VE
237	Viet Nam	VN
238	Virgin Islands, British	VG
239	Virgin Islands, U.S.	VI
240	Wallis And Futuna	WF

241	Western Sahara	EH
242	Yemen	YE
243	Zambia	ZM
244	Zimbabwe	ZW

A.4.4.6 typ006:OrientationType

This is the table of values of the compass orientation used in TPEG.

Code	Reference-English 'word'	Comment
000	unknown compass orientation	Service provider does not know at time of message generation
001	north	
002	north-east	
003	east	
004	south-east	
005	south	
006	south-west	
007	west	
008	north-west	

A.4.4.7 typ007:Priority

This is the table of values of the priority of messages.

Code	Reference-English 'word'	Comment
000	undefined	
001	low	Decoding of this message can be delayed if resources in the receiver are limited or overloaded.
002	medium	
003	high	This message should be decoded as soon as possible.

Annex B
(normative)

TPEG Message Management Container, MMC (Binary)

NOTE This Annex defines the message management that shall be used by this and all future TPEG applications when amended or newly published. Thus it will become a part of a future upgrade to CEN ISO/TS 18234-2:2006 (SSF) or a related Technical Specification. As soon as this happens this Annex will be superseded by that specification.

B.1 Terms and Definitions

For the purpose of this Annex, the following terms and definitions apply.

B.1.1 Message

Unit of information that is controlled under a message ID and contains a MessageManagementContainer, MMCMasterMessage or MMCMessagePart component.

B.1.2 Monolithic Message Management

All information for a message is transmitted within one packet controlled by the MessageManagementContainer component.

B.1.3 Multi-Part Message Management

Message management that allows parts of messages being transmitted in packets independently.

B.1.4 Top Level Container

Any component that is on the same level as the message management container.

B.2 Symbols (and abbreviated terms)

B.2.1 MMC

Message Management Container

B.2.2 PKI

Parking Information

B.3 Introduction

The message management container holds administrative information allowing a decoder to handle the message appropriately. This information is not aimed at the end user.

TPEG messages are received and managed based on the details held in the message management container. Each message has a unique identifier and a version number. These make it possible to track changes in already received messages.

There are two different mechanisms for managing messages, monolithic and multi-part management. Where the former replaces each complete message with the new version, the latter achieves a much more subtle mechanism by updating messages partially. The TPEG suite of applications has amongst it some which can simply replace the message version by version, others however have large parts that are static or quasi-static. Replacing messages of this type on a version by version basis would consume excessive bandwidth for such small content changes. To address this issue multi-part management has been introduced. Typical applications of this type would include TPEG-PKI, the parking information application, where the location rarely changes, while the fill state can fluctuate very rapidly at times.

Each TPEG application, that relies on the TPEG-MMC may state which features are or are not be used for that specific application. If nothing is stated, an application decoder has to support the complete TPEG-MMC functionality.

Applications that exclusively require monolithic message management should reference the MessageManagementContainer only, and not the MMCMasterMessage or the MMCMessagePart.

Application specific component IDs are assigned to the components MessageManagementContainer, MMCMasterMessage and MMCMessagePart within each application that references them.

B.4 Message Components

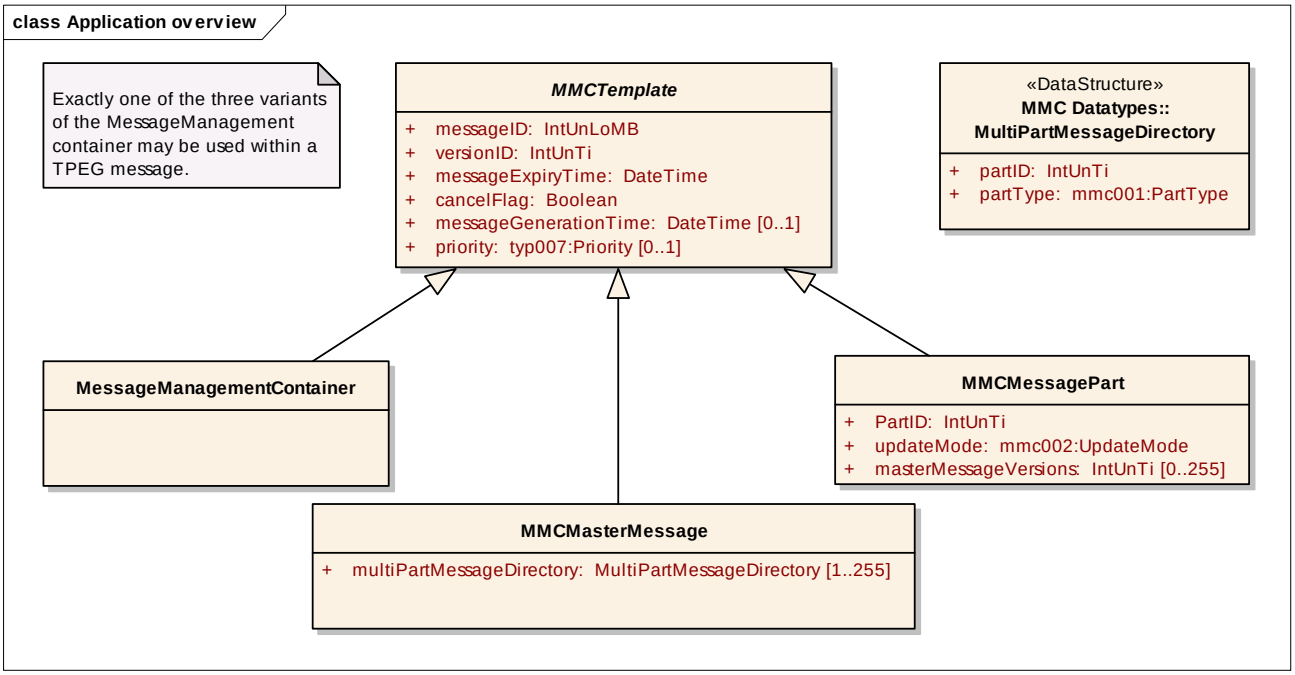

Figure B.1 — Structure of the Message Management Container

B.4.1 MMCTemplate

The MMCTemplate holds all information related to message management; it does not contain any event-related information.

The content producer, especially the incident reporting system providing the transmission data has to ensure that the message management information allows unambiguous interpretation over time. Moreover this should also be possible in scenarios with disturbed reception due to the transmission channel.

If monolithic message management is used for a message, only the MMessageManagementContainer component is instantiated.

In MMCMasterMessage components, an attribute multiPartMessageDirectory exists, allowing a master message gluing together two or more partial messages. The partial messages do contain an attribute partialMessageID and a list attribute called masterMessageVersions as well as an updateMode attribute.

The MessageManagementContainer, the MMCMasterMessage and the MMCMessagePart component inherit all the attributes from the MMCTemplate. The MMCTemplate itself is not instantiated, only the child classes can be instantiated.

<*MMCTemplate*(x)>:=	: Abstract class, no instantiation
<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),	: Unique identifier for a message relating to a particular event transmitted in a particular service component.
<IntUnTi>(versionID),	: Serial number that distinguish successive messages having a particular message identifier. Version numbers are used incrementally, allowing the progress of an event to be tracked from first notification, versionID = 0, through updates to cancellation. Wrap around is allowed. : Version numbers change every time the content of a message changes. Changes to the message management container shall not cause a change to 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 wrap around occurs, the following three 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 has to be set in an appropriate way. : If a message with the same message id, but a lower version number is received, a wrap around has occurred if the expiry time of the freshly received message is later.
<DateTime>(messageExpiryTime),	: Mandatory timestamp. If the current time is past the messageExpiryTime, the message has to be considered "invalid". The transmission system has to ensure that the message, or a newer version thereof, is sent before the message expires. Expiry times are channel specific and might differ from transmission channel to transmission channel.
<BitArray>(selector),	: 1 byte containing 3 switches.
If (bit 0 of selector is set)	
<Boolean>(cancelFlag),	: This flag has to be set (cancelFlag = 1), if a message, identified by its messageID, is no longer valid and has to be

	deleted in the client device. The message body of messages with the cancelFlag set shall be empty.
If (bit 1 of selector is set)	
<DateTime>(messageGenerationTime),	: Date and time when the message was inserted into the delivery channel. Where message carousels and delivery channel codec delays may exist, Message generation time provides an important system diagnostic tool used for testing. End-user devices should ignore this value.
If (bit 2 of *selector* is set)	
<typ007:Priority>(priority);	: Messages with higher priority should be decoded/processed prior to other messages if resources in the client device are limited or overloaded. 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.

B.4.2 MessageManagementContainer

The MessageManagementContainer component inherits all attributes from the abstract component MMCTemplate.

<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),	: Unique identifier for a message relating to a particular event transmitted in a particular service component.
<IntUnTi>(versionID),	: Serial number that distinguish successive messages having a particular message identifier. Version numbers are used incrementally, allowing the progress of an event to be tracked from first notification, versionID = 0, through updates to cancellation. Wrap around is allowed. : Version numbers change every time the content of a message changes. Changes to the message management container shall not cause a change to 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 wrap around 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 has to be set in an appropriate way. : If a message with the same message id, but a lower version number is received, a wrap around has occurred if the expiry time of the freshly received message is later.

<DateTime>(messageExpiryTime),	: Mandatory timestamp. If the current time is past the messageExpiryTime, the message has to be considered "invalid". The transmission system has to ensure that the message, or a newer version thereof, is sent before the message expires. Expiry times are channel specific and might differ from transmission channel to transmission channel.
<BitArray>(selector),	: 1 byte containing 3 switches.
If (bit 0 of selector is set)	
<Boolean>(cancelFlag),	: This flag has to be set (cancelFlag = 1), if a message, identified by its messageID, is no longer valid and has to be deleted in the client device. The message body of messages with the cancelFlag set shall be empty.
If (bit 1 of selector is set)	
<DateTime>(messageGenerationTime),	: Date and time when the message was inserted into the delivery channel. Where message carousels and delivery channel codec delays may exist, Message generation time provides an important system diagnostic tool used for testing. End-user devices should ignore this value.
If (bit 2 of *selector* is set)	
<typ007:Priority>(priority);	: Messages with higher priority should be decoded/processed prior to other messages if resources in the client device are limited or overloaded. 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.

B.4.3 MMCMasterMessage

Where parts of messages are updated dynamically, i.e. where parts are replaced or omitted without retransmission of complete messages (in contrast to monolithic message management) one MasterMessage has to be transmitted within a reasonable timeframe (at least within the message expiry time of the message parts), as no decoding of the message is possible until a MMCMasterMessage has been received.

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

The master message connects all MMCMessageParts thought the multiPartMessageDirectory listing.

For each entry in the multiPartMessageDirectory, a message with a MMCMessagePart container (with the identical messageID as the MMCMasterMessage, but otherwise independent attribute values) can be added to the transmission cycle.

<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),	: Unique identifier for a message relating to a particular event transmitted in a particular service component.

<IntUnTi>(versionID),	: Serial number that distinguish successive messages having a particular message identifier. Version numbers are used incrementally, allowing the progress of an event to be tracked from first notification, versionID = 0, through updates to cancellation. Wrap around is allowed. : Version numbers change every time the content of a message changes. Changes to the message management container shall not cause a change to 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 wrap around 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 has to be set in an appropriate way. : If a message with the same message id, but a lower version number is received, a wrap around has occurred if the expiry time of the freshly received message is later.
<DateTime>(messageExpiryTime),	: Mandatory timestamp. If the current time is past the messageExpiryTime, the message has to be considered "invalid". The transmission system has to ensure that the message, or a newer version thereof, is sent before the message expires. Expiry times are channel specific and might differ from transmission channel to transmission channel.
<BitArray>(selector),	: 1 byte containing 3 switches.
If (bit 0 of selector is set)	
<Boolean>(cancelFlag),	: This flag has to be set (cancelFlag = 1), if a message, identified by its messageID, is no longer valid and has to be deleted in the client device. The message body of messages with the cancelFlag set shall be empty.
If (bit 1 of selector is set)	
<DateTime>(messageGenerationTime),	: Date and time when the message was inserted into the delivery channel. Where message carousels and delivery channel codec delays may exist, Message generation time provides an important system diagnostic tool used for testing. End-user devices should ignore this value.
If (bit 2 of selector is set)	
<typ007:Priority>(priority),	: Messages with higher priority should be decoded/processed prior to other messages if resources in the client device are limited or overloaded. 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.
<IntUnLoMB>(n),	: Number of entries in array attribute, between 1 and 255.
n * <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.

B.4.4 MMCMessagePart

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

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

<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),	: Unique identifier for a message relating to a particular event transmitted in a particular service component.
<IntUnTi>(versionID),	: Serial number that distinguish successive messages having a particular message identifier. Version numbers are used incrementally, allowing the progress of an event to be tracked from first notification, versionID = 0, through updates to cancellation. Wrap around is allowed. : Version numbers change every time the content of a message changes. Changes to the message management container shall not cause a change to 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 wrap around 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 has to be set in an appropriate way. : If a message with the same message id, but a lower version number is received, a wrap around has occurred if the expiry time of the freshly received message is later.
<DateTime>(messageExpiryTime),	: Mandatory timestamp. If the current time is past the messageExpiryTime, the message has to be considered "invalid". The transmission system has to ensure that the message, or a newer version thereof, is sent before the message expires. Expiry times are channel specific and might differ from transmission channel to transmission channel.
<BitArray>(selector),	: 1 byte containing 4 switches.
If (bit 0 of selector is set)	
<Boolean>(cancelFlag),	: This flag has to be set (cancelFlag = 1), if a message, identified by its messageID, is no longer valid and has to be deleted in the client device. The message body of messages with the cancelFlag set shall be empty.

If (bit 1 of selector is set)	
<DateTime>(messageGenerationTime),	: Date and time when the message was inserted into the delivery channel. Where message carousels and delivery channel codec delays may exist, Message generation time provides an important system diagnostic tool used for testing. End-user devices should ignore this value.
If (bit 2 of selector is set)	
<typ007:Priority>(priority),	: Messages with higher priority should be decoded/processed prior to other messages if resources in the client device are limited or overloaded. 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.
<IntUnTi>(partID),	: Unique ID of the partial message among all messages with the same messageID.
<mmc002:UpdateMode>(updateMode),	: Defines how the content of the partial message has to be merged with the overall message content.
If (bit 3 of selector is set) {	
<IntUnLoMB>(n),	: Number of entries in array attribute, between 0 and 255.
n * <IntUnTi>\ \(masterMessageVersions) [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 may be applied to any, not expired, version of the master message.
};	

B.5 Datatypes

B.5.1 MultiPartMessageDirectory

Datastructure containing a partID and a partType attribute.

This pair is used to reference a partial message and specify if the referenced message needs to be received before the message may be presented to the user.

<MultiPartMessageDirectory>:=	
<IntUnTi>(partID),	: Unique ID of the partial message.
<mmc001:PartType>(partType);	: Defines the role of this partial message within the combined message.

B.6 Tables

B.6.1 Structure and semantics

TPEG tables provide a list of the so called Reference-English 'word' with associated code value, and additionally the tables provide comments, and where helpful, examples are given. The Reference-English 'word' describes a single entity as far as possible with a single word, but it is necessary to sometimes use a short phrase to describe the entity, e.g. north-east bound; nevertheless, TPEG tables are in essence tables of singular words. Where the coding allows multiplicity of the entity then the Reference-English 'word' shall be singular. In other cases there are a number of logical plurals, e.g. both ways, which are commented accordingly.

The key principle for the use of the Reference-English 'word' code value is that all client devices shall be designed to make their own assessment of the context and multiplicity, in order to deliver a semantically acceptable message in the chosen display language.

The encoding of each table is defined as follows:

```
<mmcxxx:yyyy>:=      : Where xxx is the table number and yyyy is the table name.
<Table>(entry);
```

In the case of TPEG-MMC it should be noted that Table content (e.g. the Reference-English 'word') is not intended for display (see Clause A3). The TPEG-MMC Table code values allow the client to interpret specific actions that it shall take.

B.6.2 Indexing

The TPEG tables title numbers and code values have no order-significance and have had number values randomly assigned during the development process. In order to aid navigation of these tables the following Table provides an "internal index" to the TPEG location reference tables, in name order.

Table B.1 — TPEG tables (mmc001 to mmc002) – ordered by names

Description	Table Nr
PartType	001
UpdateMode	002

B.6.3 Codes, Names and Comments

B.6.3.1 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.

B.6.3.2 mmc002:UpdateMode

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

Code	Name	Comment
001	replaceTopLevel	Replace all components in the combined message (on the same level as the message management container only) with the components that are transmitted via this MessagePart except the message management container. The message management information has to be stored for these components. If the combined message does not contain certain components the components of the partial message are added to the overall message. If a combined message contains more than one component with the same ID, all of these components are replaced by the ones contained in the MessagePart.
002	replaceAttributesWhile-KeepingStructure	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	addInformation	The information is independent of the information contained in the overall message. Decoders must 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.

Bibliography

[1] ISO/TS 14819-1:2003, *Traffic and Traveller Information (TTI) — TTI Messages via Traffic Message Coding — Part 1: Coding Protocol for Radio Data System — Traffic Message Channel (RDS-TMC) — RDS-TMC using ALERT-C*

[2] ISO/TS 14819-2:2003, *Traffic and Traveller Information (TTI) — TTI Messages via traffic message coding — Part 2: Event and information codes for Radio Data System — Traffic Message Channel (RDS-TMC) (ISO/FDIS 14819-2:2002)*

[3] ISO/TS 14819-3:2004, *Traffic and Traveller Information (TTI) — TTI Messages via traffic message coding — Part 3: Location Referencing for ALERT-C*

[4] ISO 17572-2, *Location Referencing for Geographic Databases — Part 2: Pre-coded Location References*

[5] ISO 17572-3, *Location Referencing for Geographic Databases — Part 3: Dynamic Location References*

[6] ISO/TS 18234-3:2006, *Intelligent transport systems — Traffic and travel information via transport protocol expert group, generation 1 (TPEG1) binary data format — Part 3: Service and network information (TPEG1-SNI)*

[7] ISO/TS 18234-6:2006,*Traffic and Travel Information (TTI) — TTI via Transport Protocol Expert Group (TPEG) data-streams — Binary representation of Location referencing applications*

[8] ISO/TS 19501:2005, *Information technology — Open Distributed Processing — Unified Modelling Language (UML) Version 1.4.2*

ICS 03.220.01; 35.240.60

Price based on 95 pages