TECHNICAL SPECIFICATION

ISO/TS 18234-10

First edition
2013-10-15

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

Part 10: Conditional access information (TPEG1-CAI)

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 10: Information d'accès conditionnel (TPEG1-CAI)

Reference number
ISO/TS 18234-10:2013(E)

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 18234-10 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 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.

TPEG applications are developed using UML modelling and a software tool is used to automatically select content which then populates this TS. 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 Conditional Access Information application, (TPEG-CAI) with the technical version number TPEG-CAI_1.0/001.

CAI application

The basic concept behind the CAI application is to transport CAI in separate TPEG service components of a dedicated application type and to define an SNI table that contains the link between scrambled content and related CAI.

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

Part 10: Conditional access information (TPEG-CAI)

1 Scope

This Technical Specification contains the definition of the TPEG Conditional Access Information (CAI) application. It enables dedicated conditional access data, such as management messages (e.g. Control Words and Entitlement Control Messages) to be delivered to recipient client devices. This TPEG application is designed for a service provider to: establish setup, prolongation or revocation of services to a specific client device, using a limited capacity unidirectional broadcast channel and without recourse to service-client handshaking.

This TPEG application defines:

— the logical channel, for the transmission of the additional CA information (CAI);

— how the CAI is linked and synchronized to the scrambled content.

This Technical Specification is related to conditional access applied at the service component level of a TPEG service. It is an open design for the integration of various different conditional access systems, externally specified, which are signalled by the TPEG service Encryption Indicator to allow client devices to operate correctly.

2 Normative References

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

ISO/TS 18234-1, *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)*

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 (TPEG1-SSF)*

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

3 Abbreviated terms

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

AID	Application Identification
CA	Conditional Access
CAI	Conditional Access Information
CRC	Cyclic redundancy check
ECM	Entitlement Control Message
EMM	Entitlement Management Message
TPEG	Transport Protocol Expert Group
SSF	Syntax, Semantics and Framing Structures
TTI	Traffic and Traveller Information

4 Application identification and version number signalling

4.1 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 ISO/TS 18234-1.

The application identification number is used within the TPEG-SNI application to indicate how to process TPEG content and facilitates the routing of information to the appropriate application decoder.

4.2 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 ISO/TS 18234-1.

Table 1 shows the current version numbers for signalling CAI within the SNI application:

Table 1 — Current version numbers for signalling of CAI

major version number	1
minor version number	0

5 Service Component Data

TPEG-CAI makes use of the "Service component data with dataCRC" according to Annex A, section A.3.2.6.2.1. For explanatory purposes, this is repeated here.

< **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

The main frame of CAI defines ApplicationContent as follows:

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

6 Conditional Access Methodology

Conditional access (CA) is specified within TPEG-SSF and TPEG-SNI as a function being applied on service frame or service component level. The method used is indicated via the Encryption Identifier (EncID) directly in the service frame or for components via the SNI Fast Tuning Table (Guide to the Services 1). This specification is related to conditional access applied on service component level.

Generally, a broadcast based CA-system requires encryption related data to be transmitted which is independent from the content, but necessary for decryption and subscriber management.

If a conditional access system is applied on the TPEG service component level, some service components may be encrypted using the same "encryption key", while others remain unencrypted or use different "encryption keys". Therefore, several service components can share the same conditional access information , if they are supposed to be offered as one bundle and hence are encrypted with the same keys.

Each of the aforementioned bundles may require CA-management-messages, which have to be transmitted separated from the (encrypted) content in the corresponding service components. The most appropriate way for the transport is the use of separate service components of a dedicated application type.

For each encrypted TPEG-Service component a link or reference to the service component carrying the relevant CA information is required. This is handled by TPEG-SNI GST-Table 6, Conditional Access Information Reference.

EXAMPLE

A TPEG Service may contain the following service components:

SCID	Application
0	SNI
2	TEC
5	TEC (encrypted)
7	TEC (encrypted)
8	PTI
10	PKI (encrypted)
20	CAI
21	CAI
30	CAI

The service components 5 and 7 are encrypted with key 1, while service component 10 is encrypted using key 2. Hence two components with CA-meta information for the corresponding component are required, in the example listed as SCID 20 and 21. A third CAI component, in the example number 30, contains CA-meta information that relates to all encrypted components independent which key is applied.

This specification describes the generic containers for the CAI application. The container content will be proprietary and specified individually for each CA-System indicated by the encryption indicator (EncID). The linking between encrypted service components and related CAI-Components is achieved via a reference table within the TPEG-SNI application.

7 Message Components

Unlike other TPEG applications, TPEG-CAI does not use a Message Management Container and does not use a Location Referencing Container; it only uses an Application Event Container.

Figure 1 visualises the logical structure of the Conditional Access Information (CAI) application.

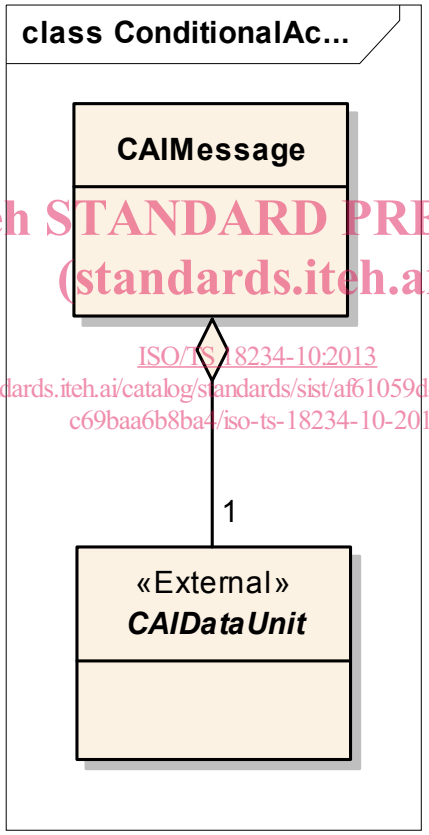

Figure 1 — Logical structure of CAI application

7.1 List of Generic Component Ids

Name	Id
CAIMessage	1

7.2 CAIMessage

A TPEG-CAI Message includes solely one single container for proprietary CA data.

The CAI Message Container is available to carry data, which is defined within the CA system specific specifications. The CAIDataUnit is directly following after the lengthAttr of the CAIMessage.

<CAIMessage(1)>:=	
<IntUnTi>(id),	: Identifier = 1
<IntUnLoMB>(lengthComp),	: Length of component in bytes, excluding : the id and length indicator
<IntUnLoMB>(lengthAttr),	: Length of attributes
<CAIDataUnit>(data);	: CAI data

7.3 CAIDataUnit

The CAIDataUnit carries the data that is specified by the corresponding conditional access specification.

<CAIDataUnit>:=	CAI data
m***<byte>;**	:proprietary CA data

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 Specification, 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 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:

[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.

1) “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;

2) “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;

3) “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);

4) “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