INTERNATIONAL STANDARD

ISO 12855

Third edition
2022-04

Electronic fee collection — Information exchange between service provision and toll charging

Perception du télépéage — Échange d'informations entre la prestation de service et la perception du péage

Reference number
ISO 12855:2022(E)

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

Published in Switzerland

Contents

Page

Foreword

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

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

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

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

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

This document was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems,* in collaboration with the European Committee for Standardization (CEN) Technical Committee CEN/TC 278, *Intelligent transport systems,* in accordance with the Agreement on technical cooperation between ISO and CEN (Vienna Agreement).

This third edition cancels and replaces the second edition (ISO 12855:2015), which has been technically revised.

The main changes are as follows:

— addition of new application data units (ADUs);

— alignment of the ASN.1 data definitions with the current edition of ISO 14906;

— removal of all dependencies on the ISO 17575 series ASN.1 data types and creation of corresponding definitions;

— re-classification of the electronic fee collection (EFC) context types by tolling and geographical characteristics and removal of the previous distinction based on tolling technology;

— splitting of the ASN.1 module into two modules: one containing ISO 12855-specific definitions, and another containing data-type definitions that are common to other standards in the EFC domain. This common data types module has then been moved to ISO/TS 17573-3;

— clarification of the semantics of all parameters in ADUs;

— alignment of the structure of all major clauses in a consistent manner to improve readability.

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

Introduction

The widespread use of road tolling requires provisions for users of vehicles that circulate through many different toll domains. Users should be offered a single contract for driving a vehicle through various toll domains. Where those vehicles require on-board equipment (OBE) this should be interoperable with the toll systems in the various toll domains. In Europe, for example, this need has been officially recognized and legislation on interoperability has already been adopted (see Directive 2019/520[8], related Commission delegated regulation 2020/2003[10] and Commission implementing regulation 2020/204[9]). There is both a commercial and economic justification regarding the OBE and the toll systems for International Standards supporting interoperability.

The system architecture defined in ISO 17573-1 is the basis for all International Standards that relate to tolling systems in the toll domain. With respect to ISO 17573-1, this document:

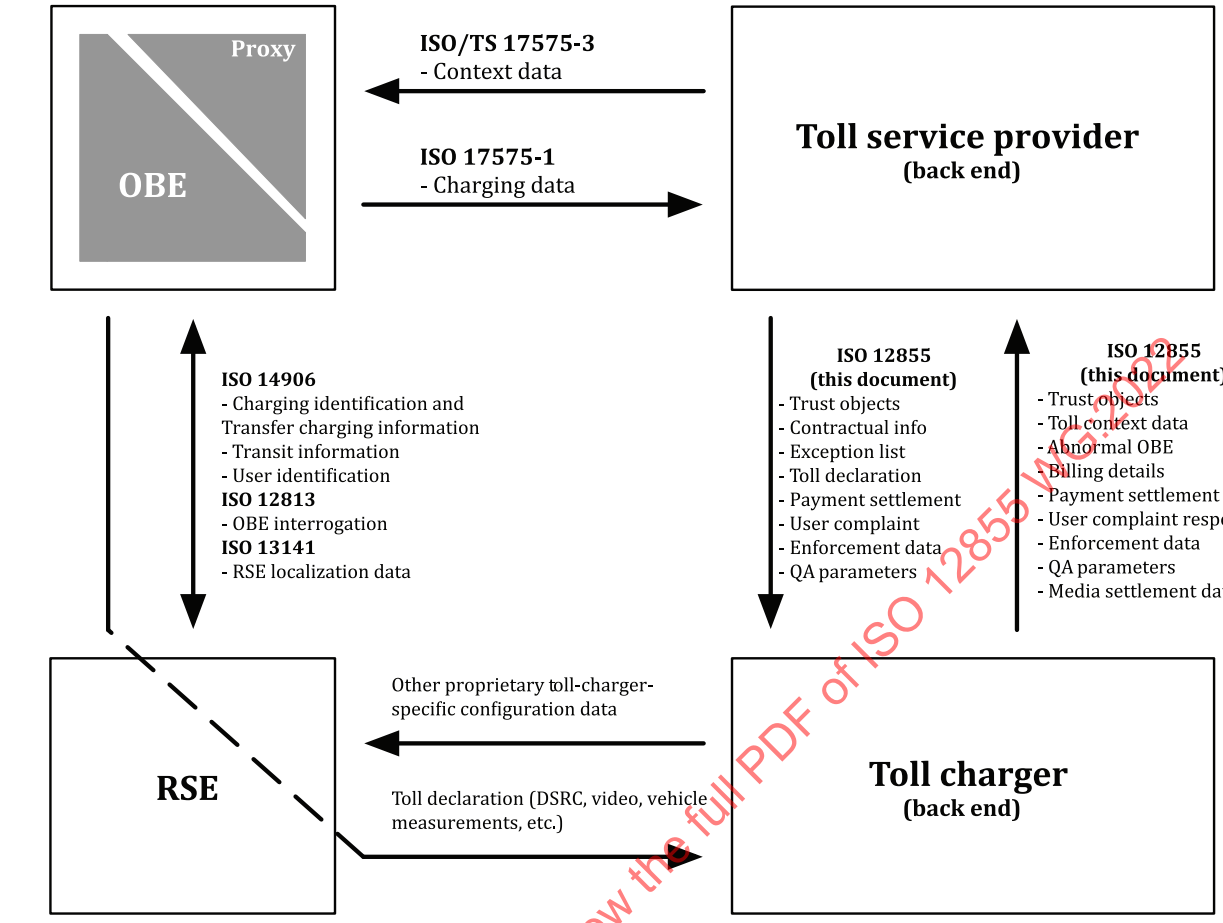
— adopts its definitions of terms and concepts and basic system functionalities and structure,

— uses its terminology, and

— specifies the interfaces identified therein.

ISO 17573-1 uses ISO/IEC 10746-3 for the description of the architecture.

Figure 1 shows the scope of the group of International Standards related to electronic fee collection (EFC) based upon the ISO 17573-1 system architecture.



Key

DSRC dedicated short-range communication
QA quality assurance
RSE roadside equipment

Figure 1 — Scope of EFC-related International Standards

A given transport service for a given vehicle is fully identified by one or several toll declarations made available to the toll charger (TC). It is necessary to make toll declarations available according to the rules of the toll regime of the toll domain.

The amount due for a given transport service used by a vehicle liable to toll is finalized by the TC with the use of toll declarations (as described above) and calculations are made according to the rules of the toll regime (formula, tariff tables, specific situations rules, traffic conditions, etc.). This means that the TC has the authority to decide on the amount due, even if it decides to assign the toll service provider (TSP) the task of calculating the amount due.

The information above, associated with a given transport service, is referred to as "billing details"; for a given transport service, the billing details refer to one or several toll declarations.

Depending on the toll regime, billing details are computed by means of the information collected by the TC and/or the relevant TSP; they are finalized by the TC or by the TSP if the TC has assigned this task to the TSP.

The TC settles the payment claims (or toll payment claims) and makes them available to each TSP, or requires the TSP to send payment announcements, according to the bilateral agreements it has with each TSP, referring to billing details. These payment claims include an amount due, taking into account any specific commercial conditions applicable to a vehicle, a fleet of vehicles or a given TSP.

This document defines a set of interactions in support of technical interoperability between back-office systems of TCs and TSPs. The EFC service and the EFC system model on which this document is based are defined in ISO 17573-1.

This document does not provide a full solution for interoperability and it does not define other parts of the EFC system, other services, other technologies and non-technical elements of interoperability. It is defined as a toolbox International Standard of application protocol data units (APDUs), which can be used for the assigned purpose. The detailed definitions of mandatory and optional elements in a real implementation are defined elsewhere. It does not define all communication sequences, communication stacks and timings.

The development of a common European Electronic Toll Service (EETS), as a part of the already cited European EFC Directive and related Regulation and Implementing acts, also calls for the definition of an interoperable EFC service. It should be noted that CEN/TS 16986 (to be revised and converted into a European Standard) specifies interoperable application profiles (IAP), applicable based on this document. These profiles define a specific coherent set of transactions, triggers, conditions, data elements, transfer mechanisms and supporting functions for an interoperable exchange of data between the back end system of TCs and TSPs. CEN/TS 16986 is consistent with and is intended to provide support for the technical specification of the EETS.

This document identifies and specifies the set of APDUs exchanged between two actors in the roles of TSP and TC as defined in ISO 17573-1. To specify these interfaces, this document uses the enterprise description of the toll environment, and the interactions defined between the named classes of roles, as defined in ISO 17573-1. This supports a complete specification of the data that is transferred between those identified entities. In addition, a number of computational interfaces are identified and interactions in terms of sequences of application protocol data units are defined.

Electronic fee collection — Information exchange between service provision and toll charging

1 Scope

This document specifies:

— the interfaces between electronic fee collection (EFC) back-office systems for vehicle-related transport services, e.g. road user charging, parking and access control;

— an exchange of information between the back end system of the two roles of service provision and toll charging, e.g.:

  — charging-related data (toll declarations, billing details),

  — administrative data, and

  — confirmation data;

— transfer mechanisms and supporting functions;

— information objects, data syntax and semantics.

This document is applicable for any vehicle-related toll service and any technology used for charging.

The data types and associated coding related to the data elements described in Clause 6 are defined in Annex A, using the abstract syntax notation one (ASN.1) according to ISO/IEC 8824-1.

This document specifies basic protocol mechanisms over which implementations can specify and perform complex transfers (transactions).

This document does not specify, amongst others:

— any communication between toll charger (TC) or toll service provider (TSP) with any other involved party;

— any communication between elements of the TC and the TSP that is not part of the back-office communication;

— interfaces for EFC systems for public transport;

— any complex transfers (transactions), i.e. sequences of inter-related application data units (ADUs) that can possibly involve several application protocol data unit (APDU) exchanges;

— processes regarding payments and exchanges of fiscal, commercial or legal accounting documents; and

— definitions of service communication channels, protocols and service primitives to transfer the APDUs.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 612, *Road vehicles — Dimensions of motor vehicles and towed vehicles — Terms and definitions*

ISO 639-1, *Codes for the representation of names of languages — Part 1: Alpha-2 code*

ISO 1176, *Road vehicles — Masses — Vocabulary and codes*

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions — Part 1: Country code*

ISO 4217, *Codes for the representation of currencies*

ISO 8583-1, *Financial transaction card originated messages — Interchange message specifications — Part 1: Messages, data elements and code values*

ISO/IEC 8824-1, *Information technology — Abstract Syntax Notation One (ASN.1) — Part 1: Specification of basic notation*

ISO/IEC 8825-4, *Information technology — ASN.1 encoding rules — Part 4: XML Encoding Rules (XER)*

ISO/IEC 9594-8, *Information technology — Open systems interconnection — Part 8: The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 9797-1:2011, *Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher*

ISO/IEC 10118-3, *IT Security techniques — Hash-functions — Part 3: Dedicated hash-functions*

ISO/IEC 11770-3, *Information security — Key management — Part 3: Mechanisms using asymmetric techniques*

ISO 13616-1, *Financial services — International bank account number (IBAN) — Part 1: Structure of the IBAN*

ISO/IEC 14888-2:2008, *Information technology — Security techniques — Digital signatures with appendix — Part 2: Integer factorization based mechanisms*

ISO 14906, *Electronic fee collection — Application interface definition for dedicated short-range communication*

ISO/TS 17444-1, *Electronic fee collection — Charging performance — Part 1: Metrics*

ISO/TS 17573-2, *Electronic fee collection — System architecture for vehicle related tolling — Part 2: Vocabulary*

ISO/IEC 18033-2, *Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers*

ISO 19299, *Electronic fee collection — Security framework*

ISO 20524-1:2020, *Intelligent transport systems — Geographic Data Files (GDF) GDF5.1 — Part 1: Application independent map data shared between multiple sources*

IETF RFC 4347, *Datagram Transport Layer Security, April 2006*

IETF RFC 5246, *The Transport Layer Security (TLS) Protocol, August 2008*

IETF RFC 5746, *Transport Layer Security (TLS) Renegotiation Indication Extension, February 2010*

IETF RFC 6040, *Tunnelling of Explicit Congestion Notification, February 2013*

W3C Recommendation *XML Signature Syntax and Processing Version 1.1*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/TS 17573-2 apply.

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

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

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

4 Symbols and abbreviated terms

ADU	application data unit
ANPR	automatic number plate recognition
APCI	application protocol control information
APDU	application protocol data unit
BIC	bank identifier code
CCC	compliance check communication
CRL	certificate revocation list
cXER	canonical XML encoding rules
DSRC	dedicated short-range communication
DST	daylight saving time
DTLS	datagram transport layer security
EFC	electronic fee collection
FTP	file transfer protocol
GDF	geographical data file
GNSS	global navigation satellite system
HOT	high occupancy tolling
HTTPS	hyper-text transfer protocol secure
IANA	internet assigned numbers authority
IBAN	international bank account number
ICC	integrated circuit card
IEC	International Electrotechnical Commission
ITU	International Telecommunication Union
LAC	localization augmentation communication
LPN	licence plate number
NMEA	National Marine Electronics Association
OBE	on-board equipment
OBU	on-board unit

OCSP	online certificate status protocol
OSI	open systems interconnection
PAN	personal account number
QA	quality assurance
RINEX	receiver independent exchange format
RSA	Rivest, Shamir and Adleman
RSE	roadside equipment
SLA	service level agreement
SMTP	simple mail transfer protocol
SU	service user
TC	toll charger
TLS	transport layer security
TSP	toll service provider
UTC	coordinated universal time
VAT	value added tax
VPN	virtual private network
VRM	vehicle registration mark
XER	XML encoding rules

NOTE RSA is an algorithm for public-key cryptography, also referred to as asymmetrical cryptographic technique.

5 Architectural concepts and information exchanges

5.1 Main roles in the toll charging environment

This document is built upon ISO 17573-1.

ISO 17573-1 specifies the four main roles shown in Figure 2.

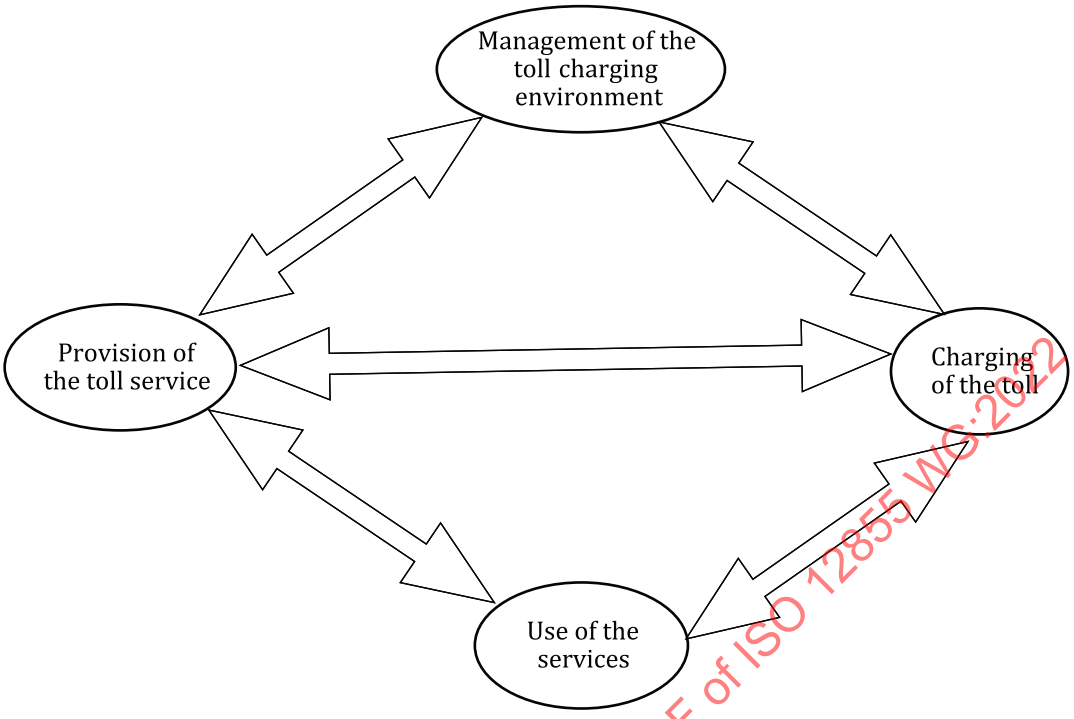


Figure 2 — Roles in the toll charging environment

Information exchanges are agreed upon between TC and TSP, taking into account privacy regulations. The information exchanges needed by the TC and the TSP to perform their roles are described in this clause.

5.2 Information exchange between toll charging and provision

5.2.1 General

The information exchange between the service provision and the toll charging roles supports the provision of functionalities that are based on the EFC system service specifications in ISO 17573-1. Figure 3 gives a general picture of the functionalities provided in this document.

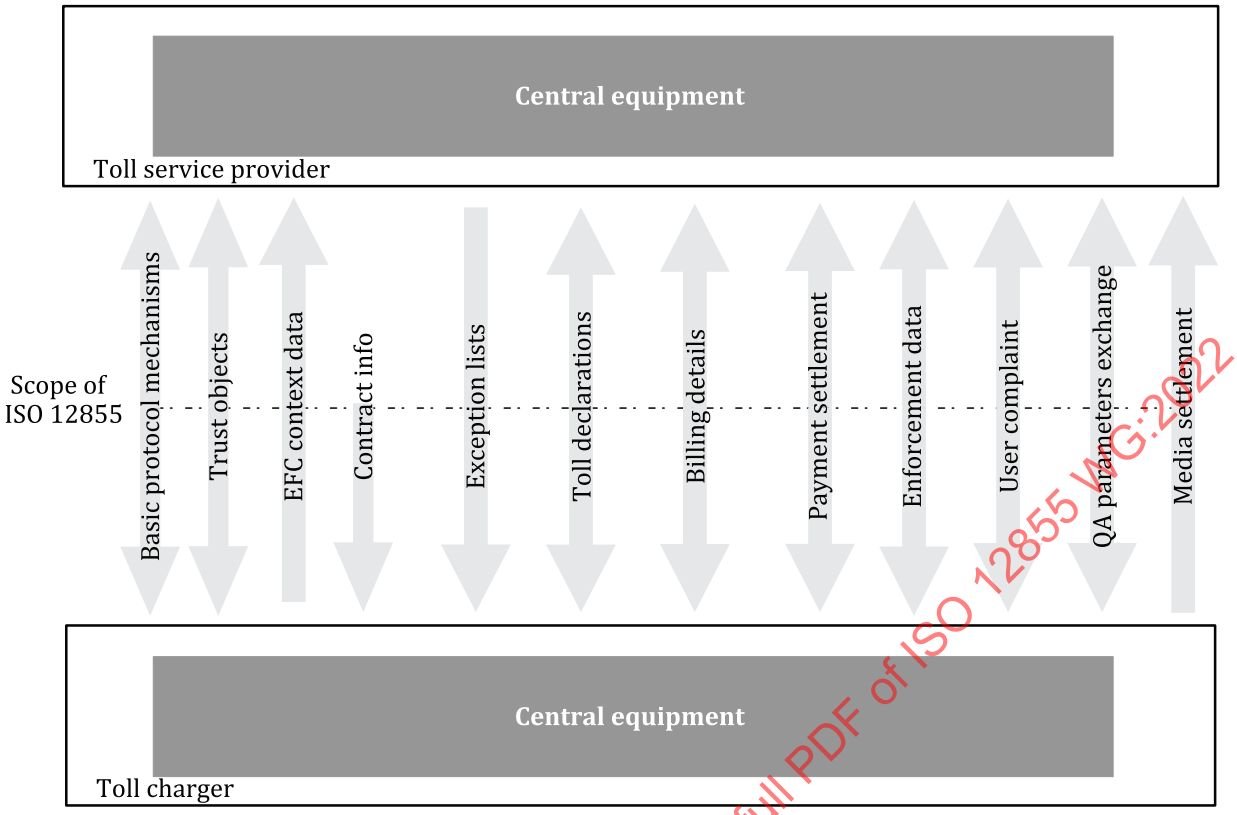


Figure 3 — Functionalities of this document (ISO 12855)

These functionalities are listed below, in the order in which they are given in Clause 5:

- basic protocol mechanisms;
- exchange trust objects;
- provide EFC context data;
- provide contract issuer list;
- manage exception list;
- report toll declarations;
- report billing details;
- payment settlement:
  - claim payment for service usage,
  - announce payments.
- exchange enforcement data:
  - exchange of user details,
- exchange of compliance check communication (CCC) events,
  - exchange of UserId lists,
- process user complaints:
  - provide a user complaint;

  - response to a user complaint.
- exchange quality assurance parameters.
- provide media settlement data.

This document leaves implementers the freedom of specifying suitable protocol procedures, i.e. for complex transactions. Therefore, it only specifies:

- a basic interaction protocol (request – response) for information exchange;
- basic protocol mechanisms, to be used to build more complex protocol procedures; and
- the semantics and the format of the APDUs that are exchanged.

These functionalities are described in 5.2.2 to 5.2.13.

5.2.2 Basic protocol mechanisms

5.2.2.1 General approach

Information exchanges are performed by means of APDU transfers.

Some APDU transfers need to be acknowledged. When this happens, related protocol procedures are specified. This document specifies no provisions for complex transfers (transactions), i.e. sequences of inter-related ADUs which may involve several APDU exchanges. Instead, this document specifies basic protocol mechanisms to be used by implementations that need to specify and perform transactions.

This document provides the following basic protocol mechanisms to exchange information between the TSP's and the TC's back end system. These basic protocol mechanisms consist of:

- an identification schema for the APDUs that are exchanged,
- a generic interaction (i.e. not related to any specific functionality) that supports requesting a specific information exchange from the counterpart. This interaction is provided by the "request" ADU,
- a generic acknowledge mechanism (i.e. not related to any specific functionality) that supports acknowledging a specific interaction. The "ack" ADU provides this mechanism, and
- an optional generic status mechanism (i.e. not related to any specific functionality) that supports providing status information for a specific interchange. This mechanism is provided by the "status" ADU.

By means of the above mechanisms, an implementation can build more complex protocol procedures, including rollback, recovery, checkpoint or restart.

This document does not specify timings and retry procedures for acknowledgements. Timeouts can be specified as agreements between TC and TSP to cover the case of missing acknowledgements.

5.2.2.2 Identification schema

Each APDU contains a unique identifier in the namespace of the originator of the APDU. The combination of originator identifier and APDU identifier ensures that all APDUs are uniquely identified.

5.2.2.3 Request functionality

The request functionality is used to:

- alert the counterpart that one is ready to accept any kind of information exchange,
- inform the counterpart that one is ready to accept a specific type of ADU, by indicating the type of ADU one is ready to accept,

— request the counterpart re-issue a specific APDU, by indicating the type and the identifier of the APDU, and

— request information identified by type and ADU content.

5.2.2.4 Acknowledgement functionality

The acknowledgement functionality is used to inform the counterpart that a specific ADU has been received correctly, or that errors have been detected.

5.2.2.5 Status functionality

The status functionality may be used to provide the counterpart with general status information about the interface or to inform it about a status on previously transferred information. It is used to:

— provide general information on the status of the interface,

— alert the counterpart that some previously provided information becomes invalid without any new information being currently available, and

— alert the counterpart that the previous information contained an error and should be recalled.

5.2.3 Exchange trust objects functionality

The exchange trust objects functionality is derived from the EFC system service "trust object exchange". The functionality is used whenever an entity, both TC and TSP, needs to send unsolicitedly new or updated trust objects to its counterpart or when asked by its counterpart to update or resend an already existing trust object.

NOTE Examples of trust objects are: asymmetric public keys, certificates, symmetric keys and certificate revocation lists.

Requiring the counterpart to send trust objects is performed by means of a "request" ADU. The "request" ADU supports requesting already issued trust objects to be resent, as well as requesting newly issued trust objects.

The exchange trust objects functionality provides the "trust objects" ADU to transfer the requested or newly generated trust objects to the counterpart, which has the possibility of confirming or rejecting the received trust objects by using the acknowledgement functionality.

The exchanged trust objects are valid from either the time of their acknowledgment or from the validity starting, if specified in the "trust object" ADU. Trust objects may also become valid based on bilateral agreements between TSP and TC (as defined in the implementation conformance statement).

5.2.4 Originating and providing EFC context data functionality

The originating and providing EFC context data functionality is derived from ISO 17573-1 as specified in the EFC system service "adding or modifying a toll regime".

The originating and providing EFC context data functionality gives a TC the possibility to communicate any change of a toll domain or toll regime, including the start of a new toll domain, by issuing an "EFC context data" ADU.

A TSP has the possibility of requiring a TC to provide information about the toll context data for a toll domain under its responsibility by using a "request" ADU. The TC provides the information requested by this "request" ADU by using an "EFC context data" ADU.

The TSP has the possibility to confirm reception of an "EFC context data" ADU by using the acknowledgement functionality.

5.2.5 Provide contract issuer information functionality

The contractual information functionality supports a TSP in delivering to the TC any type of user contract related information that is stored in the OBE, among which the personal account number (PAN), the supported security level and the key set to be used in the calculation of dedicated short-range communication (DSRC) authenticators. This functionality supports, among others, the TC in comparing data retrieved from the OBEs with information received by the TSP.

5.2.6 Manage exception list functionality

5.2.6.1 General

The manage exception list functionality originates from ISO 17573-1 and is specified in the EFC system service "exceptions handling".

NOTE 1 This document uses the term "exception list" to summarize all possibilities of limiting the availability of transport services to a user or giving information on the special handling of an OBE in a toll regime. Other standards use different terms, but they are all included in the term "exception list".

NOTE 2 The conditions and the periods of time that a user can be accepted within a toll regime are limited by putting it on the exception list or removing it. This is solely the responsibility of the related TSP. Any information sufficient for the identification of a specific vehicle or OBE by the TC, e.g. OBE ID, PAN, LPN (licence plate number), are included in the exception list as agreed between TC and TSP.

5.2.6.2 Exception list entry requested by a toll charger

The manage exception list functionality supports a TC in notifying a TSP about detected violations by a specific service user or wrong technical behaviour by a specific OBE by using a "report abnormal OBE" ADU. The "acknowledgement" functionality is used to confirm reception of a "report abnormal OBE" ADU. Subsequently, this OBE may be included in the exception list of the TSP.

5.2.6.3 Exception list entry decided by the toll service provider

The manage exception list functionality supports a TSP in notifying a TC about additions, modifications or deletions of items in its exception lists by using the "exception list" ADU.

NOTE 1 The ADU can include, among others, one of the following reasons:

— the TSP has terminated its support/responsibility for a vehicle/OBE;

— an OBE was lost or stolen;

— the TSP has started/accepted its support/responsibility for a vehicle/OBE;

— the TC is informed about the commercial conditions to apply to an OBE (e.g. discount for a group of vehicles).

NOTE 2 The exception list can be used to provide additional information on a vehicle/OBE for a toll regime (e.g. specific commercial conditions) and/or limit or restrict the acceptance of an OBE within a toll regime operated via the road infrastructure of a TC, where an exchange of data between TSP and TC is needed.

NOTE 3 The exception list can be used to provide information on the country of registration of a vehicle and a value added tax identifier (VAT ID) to support a TC in the correct application of VAT for ferry operations, where this information is needed for handling of reverse charge VAT in the EU.

The TC may dispute the received exception list by using the acknowledgment functionality and transmitting an error code. In this case, the last correct list remains active in the systems of the TC until a newly transmitted list has been positively acknowledged by using the acknowledgement functionality.

5.2.7 Report toll declarations functionality

The report toll declarations functionality originates from ISO 17573-1 and is specified in the EFC system service "collecting charging information (autonomous systems)".

NOTE 1 The charging data generated by an OBE is used to report a service user (SU) entering, moving around in or leaving a toll domain. A service usage statement with an amount due can be made either by a single tolled object or by a combination of several tolled objects. Any service usage is reported as charging data through an exchange of data between an OBE/proxy (front-end system) and the back end system managed by a TSP. This interface between front-end and TSP is specified in ISO 17575-1 and ISO 17575-3, and is not covered by this document.

The TSP has the possibility to enrich the gathered charging data that are stored in its back end system, before sending that data as a "toll declarations" ADU to the TC. This possibility supports the concept of shared user data, where only limited information may be included in the OBE, while the rest of it is held centrally at the issuing TSP.

The "toll declarations" ADU provides the TC with all information needed to calculate the amount due for the use of a toll domain or to verify the calculation done by the TSP, or it may contain only summary data, according to the optional data elements that are implemented.

NOTE 2 The toll declarations can be delivered periodically in an agreed frequency (e.g. weekly, daily, hourly, in real time, etc.) or upon triggering the delivery and with the quantity of information agreed for a toll regime.

The TC has the possibility to confirm a received "toll declaration" ADU by using the acknowledgement functionality.

NOTE 3 If the TC detects a contradiction between the toll declarations provided by the TSP and its own data (e.g. CCC data, LPN reading, etc.), it has the possibility to ask the TSP for additional information about the provided toll declarations for a specific vehicle or service user by sending a "request" ADU to the TSP to provide any detailed toll declarations for a specific service user and/or a specific period of time. This enables the TC to receive only summation records of the use of its toll domain from the TSP and to produce its billing details in normal operation without any detailed knowledge about each segment passed by an OBE. When the TC detects a contradiction in the provided high level toll declarations during comparison with the CCC data it recorded, it can ask the TSP for detailed toll declarations for this specific vehicle or Service User.

The procedure for handling a possible negative "ack" ADU as a response to a "toll declaration" ADU is outside the scope of this document.

5.2.8 Report billing details functionality

Depending on the characteristics of a toll system, a TC either acquires the toll declarations directly from the roadside equipment (RSE) it operates (e.g. DSRC-based systems) or receives it from the TSP's back end system (e.g. autonomous systems). That information is then used for the generation of billing details.

A single billing detail may refer to one or more toll declarations. The generation of billing details is based on the requirements specified for the toll regime. Thus, a billing detail may consist of:

— an elementary usage of a transport service (e.g. regarding the toll for a road section);

— several usages of a transport service within a given period of time (a day, a week, a month, etc.); or

— several elementary usages of a transport service within a given journey.

If some relevant information is missing to build a billing detail out of the available toll declarations, the TC has the possibility to request it from the back end system of the TSP (see 5.2.7).

The TC and TSP can agree to aggregate the generated billing details in order to reduce the number of lines to be processed in their bookkeeping systems. The TC and the TSP need to agree on common aggregation methods for billing details in order to avoid any rounding differences between VAT

declarations for the local tax authorities (when applicable). Consequently, rounding rules are not defined in this document.

NOTE 1 Aggregation methods can be, for example:

— after generating the billing details, the TC aggregates any billing details prior to claiming the payment, if this is bilaterally agreed. A unique identifier (reference number) for each aggregate is generated during aggregation and associated with all billing details in order to link them to the derived payment claim. By this method, the TSP is always able to check the consistency of the payment claim with the billing details from which they stem;

— the TC and TSP agree to use the same aggregation process and aggregate the billing details independently from each other for the representation on the invoice in their own back end system to avoid any rounding differences.

See informative Annex E for an example of one possible solution to deal with the aggregation of billing details.

The TC or the TSP, depending on the respective operational model, have the possibility of transferring billing details by means of a “billing detail” ADU.

NOTE 2 The operational model of the TC in a global navigation satellite system (GNSS)-based environment can require the generating of any billing details and the maintaining of complete control over the usage of its toll domain. This is typically the case for privately operated TCs or road concessionaires. If a government body conducts the role of TC, on the other hand, it can only want to carry out spot checks to avoid the operational workload of being a TC in a GNSS environment. In this case, the TC can require the TSP to generate the "billing details" ADU to be sent to it.

It can be necessary for the recipient of a “billing details” ADU to check the completeness and the conformity of the provided billing details for a given service usage. Therefore, the billing details can optionally reference previously exchanged toll declarations so as to provide further information.

The recipient of a “billing details” ADU can accept or deny it by using the acknowledgement functionality.

5.2.9 Payment settlement functionality

5.2.9.1 Claim payment for service usage

The claim payment for service usage functionality is derived from ISO 17573-1 and is specified in the EFC system service “payment settlement”.

The claim payment phase may, for example, start once the billing details have been agreed between the TC and the TSP (see 5.2.8). In this case, the payment claim is based upon these agreed and possibly aggregated billing details.

The TC uses the claim payment functionality by sending a “payment claim” ADU to the TSP.

The “payment claim” ADU also gives the possibility of including additional claims for other services that do not directly derive from the bilaterally agreed billing details (e.g. overdue payment, reimbursement, penalties, etc.). Discounts may be specified as a particular type of payment claim, which may, for example, carry the reference to the services for which a discount has been applied, or they may be included at a later time in a separate payment claim as a credit note (e.g. when billing details for a whole month specified the discount level of a scaled discount scheme).

The “payment claim” is intended to contain or to address all information required according to each toll regime to be used for invoicing between the TC, the TSP and the service user.

NOTE 1 The payment claim can be used by the TSP to generate invoices for its customers (service users) on behalf of the relevant TC or for its own account (if the TSP is buying the toll from the TC and reselling it to the service user).

The TSP can accept or refuse a received "payment claim" ADU by using the acknowledgement functionality.

NOTE 2 Possible reasons for dispute, among others, are inconsistency of the payment claim with received billing details or non-conformance of specific commercial conditions.

5.2.9.2 Payment announcement

The payment announcement functionality is derived from ISO 17573-1 and is specified in the EFC system service "payment settlement". This functionality supports a TSP in informing a TC of a payment that has been made consequently to the use of transport services. Unlike the payment claim functionality, the payment announcement functionality is initiated by the TSP either unsolicitedly or on request from the TC.

The TSP can use this functionality by sending a "payment announcement" ADU to the TC informing it about an upcoming payment. The information carried in the "payment announcement" ADU may include references to billing details, toll declarations or even single elements of a charge report.

The TC has the possibility of accepting or refusing a received "payment announcement" ADU by using the acknowledgement functionality.

5.2.10 Exchange enforcement data functionality

5.2.10.1 General

The exchange enforcement data functionality originates from ISO 17573-1 and is specified in the EFC system service "exceptions detection".

The exchange enforcement data functionality may be used each time a TC needs additional information for a compliance checking process. The compliance checking process consists of:

— verifying if the service user fulfils its obligation to co-operate; and

— gathering facts required for later performance monitoring and/or service level agreement (SLA) evaluation.

A subsequent offline process performed by the TC may:

— issue toll violation tickets for service users if the service user was responsible for the wrong toll declaration; and

— compare roadside observations with toll declarations received from the TSP and update the performance monitoring parameters with these results.

Even if the details on how to perform compliance checking/enforcement back end processes are left to the TCs, it can be necessary for some basic information exchange to be supported on both sides.

5.2.10.2 Retrieve user details functionality

The "request" ADU may be used by a TC to ask the TSP(s) whether they have a contractual relationship with a given service user identified in the ADU and/or to provide additional details on a service user by specifying the requested details.

This ADU can either be sent to a specific TSP or broadcasted to a group or all TSPs.

5.2.10.3 Provide user details

Provision of details of a given user is a functionality that supports a TSP in responding to a "request" ADU requiring user details. The functionality is performed by either sending a "provide user details"

ADU that contains details about a contractual relationship with the given user, or by using the acknowledgement functionality with a specific negative reason code.

When details on a user are provided, the minimum set of data contained in a "provide user details" ADU includes the userId.

Any additional details about the service user depend on what is specified in the "request" ADU and on the actual availability of the requested data. The actual delivery of user details can be subject to local data protection regulations.

The TC may acknowledge any received "provide user details" ADU by using the acknowledgement functionality.

5.2.10.4 Retrieve and report CCC event

The retrieve and report CCC event functionality supports a TC either unsolicitedly or after receiving a "request ADU" from the TSP in delivering the details of any CCC events for a specific service user and/or a specific period of time.

This service is realized by issuing a "report CCC event" ADU.

The TSP can acknowledge any received "report CCC event" ADU by using the acknowledgement functionality.

5.2.10.5 List of users

The list of users functionality provides support to the TC in its enforcement activities and is useful in particular for detecting potential fraud. The functionality supports a TC in requesting, by means of a specific "request" ADU, information about all user identifiers that belong to a given service user (e.g. haulier).

5.2.11 Process user complaints functionality

The process user complaints functionality is derived from the ISO 17573-1 service "Providing customer care". In ISO 17573-1, the providing customer care service belongs in the group of services that involve the user and the TSP. This functionality specifies an aspect of customer care that involves interaction between the TC and the TSP to handle user complaints, covering the case where a previously accepted billing detail, possibly included in an accepted payment claim, is requested to be revoked because of a complaint by the user.

The TSP, after having examined a user complaint and verified it is well-founded, can deliver the complaint to the TC, by means of a "user complaint" ADU.

The TC can acknowledge any received "user complaint" ADU by using the acknowledgement functionality, thereby indicating that the complaint will be processed.

When processing of the user complaint is complete, the TC may signal the completion of the evaluation process by issuing a "user complaint response" ADU, thereby indicating the acceptance or the refusal of the user complaint.

5.2.12 Exchange quality assurance parameters functionality

The report quality assurance parameters functionality is derived from ISO 17573-1 and is specified in the EFC system service "monitoring operations".

In ISO 17573-1, the report quality assurance parameters functionality is used to report and monitor the EFC activities of TCs and TSPs. In this document, it is used additionally to exchange any information necessary to monitor the quality of operations between the TC and the TSP.

EXAMPLE One of these quality assurance parameters is the detection rate (i.e. the percentage of detected OBE of a given TSP in a given toll regime of a TC). As it is important for both the TC and the TSP that the detection rate of the TSP's OBEs in a toll regime of a TC be as high as possible, this can need to be constantly monitored by an SLA level.

Types and ranges of quality assurance (QA) parameters are agreed upon between TCs and TSPs with bilateral agreements.

The functionality is used whenever an entity, both TC and TSP, needs to send unsolicitedly new or updated QA parameters to its counterpart or when asked by its counterpart to update or resend already sent QA parameters. It is performed by issuing a "report QA" ADU.

Requiring the counterpart to send QA parameters is performed by means of a "request" ADU. The "request" ADU supports requests for already issued QA parameters to be resent and also requests additional QA parameters.

The receiver of a "report QA" ADU can acknowledge the received QA parameters by using the "acknowledgement" functionality.

5.2.13 Provide media settlement data functionality

The provide media settlement data functionality supports a TC in responding to a "request" ADU from a TSP asking payment settlement data to be transmitted to the media provider. This functionality is performed by means of the "media settlement data" ADU. The information in the "media settlement data" ADU is generated from a medium connected with an OBE independently of the used tolling technology.

The settlement is initiated by an OBE and transparently performed via the OBE from the RSE and/or central server. Media settlement data is provided to an OBE when a vehicle is passing close to a physical charging station in a DSRC-based system or whenever settlement data need to be generated in a GNSS-based system.

NOTE 1 Media settlement data contain evidence of payment settlement using a medium for both road charging and invoicing to the service user and are transmitted to the media provider via the TC and the TSP without any modification since they contain a tamper-proof code.

The toll service can confirm the received media settlement data by using the acknowledgment functionality.

NOTE 2 If the media provider has doubts and/or suspicions about received media settlement data, another recovery processing is performed by the TSP and the TC under a bilateral agreement with the media provider.

6 Computational specification

6.1 Overview

This clause specifies the computational objects for the information exchanges between TCs and TSPs. Each computational object is identified by:

— a name;

— the function(s) it performs; and

— the ADUs it is able to generate or accept.

For the sake of simplicity, the functionalities identified in the previous clauses correspond to computational objects in this clause. This means that the functions performed by each object have

already been described in the previous clauses. A given computational object can be instantiated for the TSP or for the TC role, with the limitations and permissions stated as rules in the previous clauses. The fact that computational objects are modelled here as each having one logical interface for interactions does not put any limits on the number of real interfaces offered in an implementation of this document.

Table 1 lists the specified computational objects. For each object, the relevant interface interactions are listed, together with the limitations, permissions and obligations for each role derived by the rules specified in previous clauses.

An implementation conforming to this document shall declare which of the computational objects and related interactions listed in Table 1 are supported. For each implemented computational object and interaction, the implementer declares their support, and depending on the implementation's role (TC or TSP), the rules stated in Table 1 shall be obeyed, with the following meanings:

— may initiate: an entity is able and allowed to initiate an interaction;

— shall initiate: an entity is able and has to initiate an interaction;

— shall be able to receive: an entity shall be able to accept this kind of interaction. If a related ADU is not supported, the receiver shall respond with an ADU reason code "Not supported ADU".

Table 1 — Computational objects and interactions

Computational object	Interaction	TSP rules	TC rules
Exchange trust objects	Request	May initiate, shall be able to receive	May initiate, shall be able to receive
	Trust objects	May initiate, shall be able to receive	May initiate, shall be able to receive
	Acknowledge	Shall initiate, shall be able to receive	Shall initiate, shall be able to receive
	Status	May initiate, shall be able to receive	May initiate, shall be able to receive
Originating and providing EFC context data and contractual information	Request (EFC context data)	May initiate	Shall be able to receive
	Request (contractual information)	Shall be able to receive	May initiate
	EFC context data	Shall be able to receive	May initiate
	Contractual information	May initiate	Shall be able to receive
	Acknowledge (EFC context data)	Shall initiate	Shall be able to receive
	Acknowledge (contractual information)	Shall be able to receive	Shall initiate
	Status	May initiate, shall be able to receive	May initiate, shall be able to receive
Manage exception list	Request (exception list)	Shall be able to receive	May initiate
	Request (abnormal OBE)	May initiate	Shall be able to receive
	Exception list	May initiate	Shall be able to receive
	Report abnormal OBE	Shall be able to receive	May initiate
	Acknowledge (exception list)	Shall be able to receive	Shall initiate
	Acknowledge (report abnormal OBE)	Shall initiate	Shall be able to receive
	Status	May initiate, shall be able to receive	May initiate, shall be able to receive

Table 1 *(continued)*

Computational object	Interaction	TSP rules	TC rules
Toll declaration	Request	Shall be able to receive	May initiate
	Toll declaration	May initiate	Shall be able to receive
	Acknowledge	Shall be able to receive	Shall initiate
	Status	May initiate, shall be able to receive	May initiate, shall be able to receive
Report billing details	Request	May initiate, shall be able to receive	May initiate, shall be able to receive
	Billing details	May initiate, shall be able to receive	May initiate, shall be able to receive
	Acknowledge	Shall initiate, shall be able to receive	Shall initiate, shall be able to receive
	Status	May initiate, shall be able to receive	May initiate, shall be able to receive
Payment settlement	Request (payment claim)	May initiate	Shall be able to receive
	Request (payment announcement)	Shall be able to receive	May initiate
	Payment claim	Shall be able to receive	May initiate
	Payment announcement	May initiate	Shall be able to receive
	Acknowledge (payment claim)	Shall initiate	Shall be able to receive
	Acknowledge (payment announcement)	Shall be able to receive	Shall initiate
	Status	May initiate, shall be able to receive	May initiate, shall be able to receive
Exchange enforcement data	Request (user details)	Shall be able to receive	May initiate
	Request (CCC event)	May initiate	Shall be able to receive
	Request (list of users)	Shall be able to receive	May initiate
	Provide user details	May initiate	Shall be able to receive
	Report CCC event	Shall be able to receive	May initiate
	List of Users	May initiate	Shall be able to receive
	Acknowledge (user details)	Shall be able to receive	May initiate
	Acknowledge (report CCC event)	Shall initiate	Shall be able to receive
	Acknowledge (list of users)	Shall be able to receive	May initiate
	Status	May initiate, shall be able to receive	May initiate, shall be able to receive
Process user complaint	User complaint	May initiate	Shall be able to receive
	Acknowledge (user complaint)	Shall be able to receive	Shall initiate
	User complaint response	Shall be able to receive	May initiate
	Acknowledge (reception of user complaint response)	Shall initiate	Shall be able to receive

Table 1 *(continued)*

Computational object	Interaction	TSP rules	TC rules
Exchange quality assurance parameters	Request	May initiate, shall be able to receive	May initiate, shall be able to receive
	Report QA	May initiate, shall be able to receive	May initiate, shall be able to receive
	Acknowledge (report QA)	Shall initiate, shall be able to receive	Shall initiate, shall be able to receive
	Status	May initiate, shall be able to receive	May initiate, shall be able to receive
Provide Media Settlement Data	Request	May initiate	Shall be able to receive
	Media settelement data	Shall be able to receive	May initiate
	Acknowledge (media settlement data)	Shall initiate	Shall be able to receive

The offering of an interface by an actor may require different behaviours depending on the activities to be performed with the exchange of information.

The interface interactions summarized in Table 1 are performed by exchanging open systems interconnection (OSI) application layer APDUs. They are described in detail in the following subclauses and are formally defined in Annex A. In case of any conflict between this description and the definition in Annex A, the definition in Annex A takes precedence.

To facilitate reading, the following conventions are adopted in subclause 6.2:

a) the names and values of the fields in APDUs are in **boldface**;

b) the corresponding ASN.1 types are written using `Courier` font.

6.2 Application protocol data units

6.2.1 General

The APDU content is formally defined in Annex A by the ASN.1 type InfoExchange, which is made of the following elements:

a) a mandatory **infoExchangeContent** field;

b) an optional **infoExchangeAuthenticator** field.

Table 2 describes the fields in the infoExchange APDU.

Table 2 — InfoExchange APDU fields

Field name	Data type/data description	m/o
infoExchangeContent	`InfoExchangeContent`. Actual content of the APDU. See Table 3.	m
infoExchangeAuthenticator	`AuthenticatorEfc`. Record (ASN.1 SEQUENCE) made of two fields of types `authenticatorEfc`, and `ackAuthenticatorEfc`	o

Each **infoExchangeContent** field shall contain one **apci** field and one **adus** field as specified in the `InfoExchangeContent` data type (see Table 3).

Any APDU may optionally be signed by calculating an authenticator over the **infoExchangeContent** field. The calculated authenticator shall be included as **infoExchangeAuthenticator** in the resulting APDU. Semantics of the two fields of the **infoExchangeAuthenticator**, as well as the algorithms used to calculate them, are specified in 7.3.2.

The **infoExchangeAuthenticator** is a dynamic object and shall be generated each time a new APDU is issued. The **infoExchangeAuthenticator** can be calculated by either the APDU originator or the information sender (see 6.2.2).

Table 3 indicates the fields in the `InfoExchangeContent` data type, which is defined in Annex A.

Table 3 — InfoExchangeContent data type fields

Field name	Data type/data description	m/o
apci	`ApciFields`. Record (ASN.1 SEQUENCE) containing protocol control information	m
adus	`Adus`. Contains one or more (ASN.1 SEQUENCE OF) ADUs of the same type, as a selection (ASN.1 CHOICE) among the types of specified ADUs.	m

A top-level view of the structure of `InfoExchangeContent` data type and its fields is given in Figure 4.

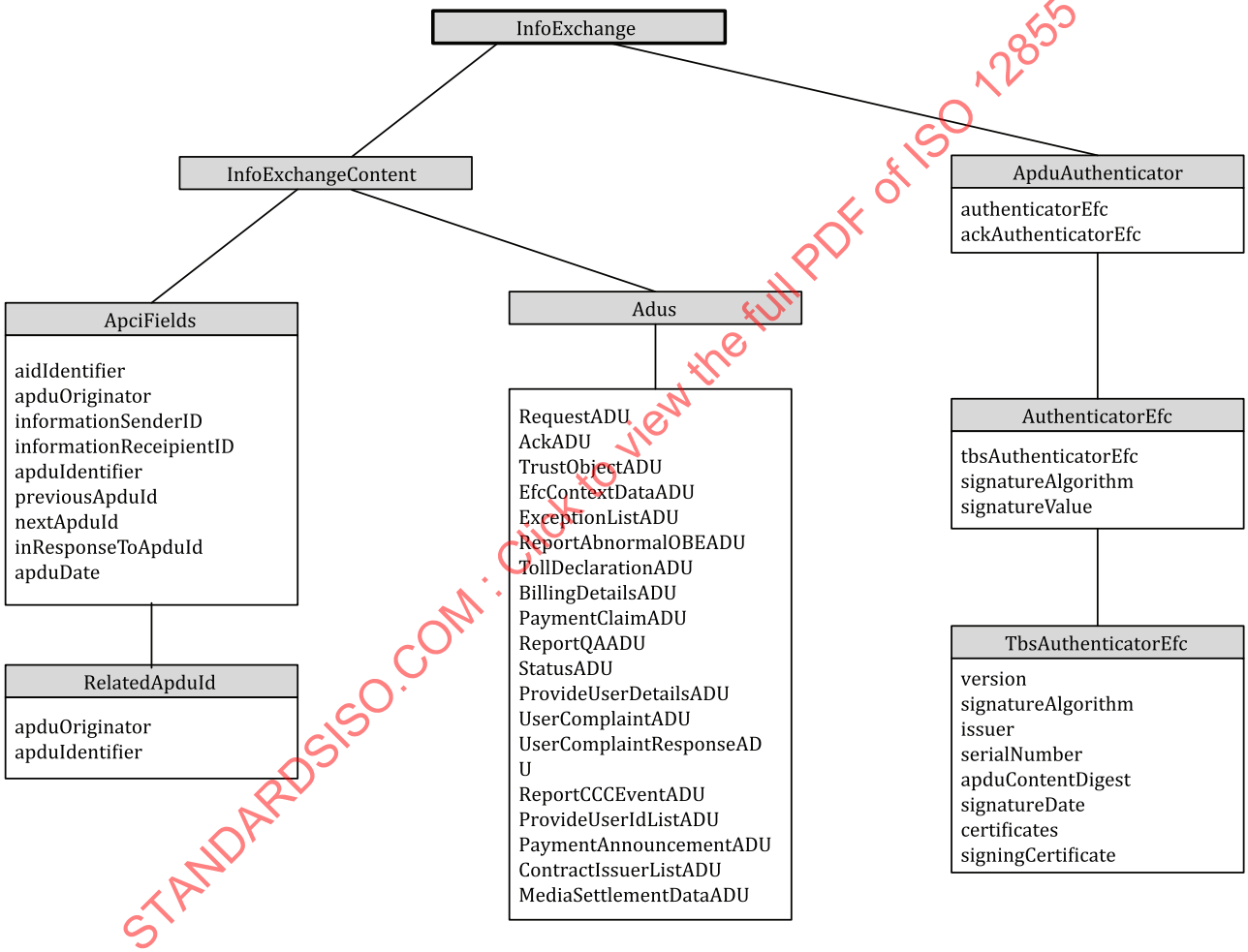


Figure 4 — InfoExchange data type structure

The following subclauses describe the APCI and the ADUs for the specified APDUs.

APDUs are described in the following terms:

a) tables that list the parameters that make up the APDUs. For each table, the parameter names are listed, together with an indication of their fields and formats and whether the parameter is mandatory (m) or optional (o). The parameter description is indicative and is formally specified in Annex A. In cases where there are discrepancies, the formal definition in Annex A takes precedence;

b) additional tables are inserted to better describe parameters with a complex structure;

c) additional semantics definition and usage of each parameter are given as text after each table.

6.2.2 Application protocol control information (APCI)

The APCI for all APDUs shall consist of the fields described in Table 4.

Table 4 — APCI fields

Field name	Data type/data description	m/o
aidIdentifier	`AidIdentifier`. Integer ranging from 0 to 255	m
apduOriginator	`Provider`	m
informationSenderId	`Provider`	m
informationRecipientId	`Provider`	m
apduIdentifier	`ApduIdentifier`. Integer ranging from 0 to $2^{63}-1$	m
previousApduId	`RelatedApduId`	o
nextApduId	`RelatedApduId`	o
inResponseToApduId	`RelatedApduId`	o
apduDate	`GeneralizedTime`	m

aidIdentifier shall identify the protocol version number. The following values are specified:

— **iso12855x2015,** which identifies the 2015 version of the IS document;

— **isoDIS12855x2020,** which identifies the 2020 version of the DIS document;

— **isoFDIS12855x2021,** which identifies the 2021 version of the FDIS document;

— **iso12855x2022,** which identifies the 2022 version of the IS document;

— **cen16986x2016,** which identifies the 2016 version of the CEN 16986 profile of the 2015 version of the IS 12855.

— **cen16986x2022,** which identifies the 2022 version of the CEN 16986 profile of the 2022 version of the IS document.

In addition to the values specified above and the ranges of ISO/CEN reserved values, a range of values is reserved for private usage. Private versions shall be identified by one of these values in association with the informationSenderId value.

apduOriginator shall be the identifier of the entity originating the contents of the APDU, which can be either a TC or a service provider. This may be, but is not necessarily, the actual sender of the information.

informationSenderId shall identify the entity actually sending the APDU. It may be used to identify a technical entity supporting the apdu originator in the task of transferring information. It may contain the same identifier as the **apduOriginator** field, when the originator and the sender are the same entity.

informationRecipientId shall identify the entity responsible for processing the content of the APDU and is an entity of type TC or service provider. In some APDUs a null value is admitted for this field, with the special meaning to indicate that the ADU is intended for more than one recipient. If the values of both the CountryCode and the IssuerIdentifier fields of informationRecipientId are zero, the PDU is meant to be sent to all connected entities. If only the value of the IssuerIdentifier is zero, the PDU is meant to be sent to all entities of the stated country.

apduIdentifier shall be a unique identifier for each given apdu originator that is assigned by the APDU originator themselves. It is used to either acknowledge or request retransmission of a previously issued APDU.

previousApduId may optionally be used to indicate that the APDU is related to a previously sent APDU from the apdu originator. If used, it shall identify the APDU in terms of the identifier of the originator

(of type `Provider`) and of the unique APDU identifier (of type `INTEGER` defined as `Int8Unsigned`) within the originator. It shall not be used if no APDU has been sent by the originator yet. This field, together with the **nextApduId** field, may be used to identify chains of APDUs. The `RelatedApduId` data type is a record (ASN.1 SEQUENCE) of:

— **apduOriginator**, of type `Provider` which shall identify the APDU originator;

— **apduIdentifier**, of type `ApduIdentifier` which shall identify the APDU within **apduOriginator**.

nextApduId may optionally be used to indicate that the APDU is related to a next to be sent APDU from the APDU originator. This field, together with the **previousApduId** field, may be used to identify chains of APDUs.

inResponseToApduId may optionally be used to indicate that the APDU is either a response to a request or an ACK. It may be needed to identify a chain of APDUs to be processed in a workflow.

apduDate shall specify a time stamp for the APDU.

6.2.3 Application data units

The ADUs field shall contain (ASN.1 SEQUENCE OF) one or more data structures of the same type.

The ADUs listed in Table 5 are specified in this document.

Table 5 — ADUs overview

Functionality	ADU type name	Description
Basic	`RequestAdu`	Generic request
	`AckAdu`	Generic acknowledge
	`StatusAdu`	Generic status
Exchange trust objects	`TrustObjectAdu`	Send trust objects
Provide EFC context data	`EfcContextDataAdu`	Send EFC context data
Provide contract issuer information	`ContractIssuerListAdu`	Provide information on contract issuer
Manage exception lists	`ExceptionListAdu`	Send exception list
	`ReportAbnormalObeAdu`	Report abnormal OBE
Report toll declarations	`TollDeclarationAdu`	Toll declaration
Report billing details	`BillingDetailsAdu`	Billing details
Payment settlement	`PaymentClaimAdu`	Payment claim
	`PaymentAnnouncementAdu`	Payment announcement
Exchange enforcement data	`ProvideUserDetailsAdu`	Provide user details
	`ReportCccEventAdu`	Report CCC event
	`ProvideUserIdListAdu`	Provide a user list
Exchange QA parameters	`ReportQaAdu`	Report QA
Process user complaint	`UserComplaintAdu`	Send a user complaint
	`UserComplaintResponseAdu`	Reply to a user complaint
Provide media settlement data	`MediaSettlementDataAdu`	Send media settlement data

These data structures are specified in 6.3 to 6.21.

6.2.4 ADU identification

Each ADU is identified by a mandatory field named **aduIdentifier** of type `AduIdentifier`, which is defined as an integer ranging from 0 to 2^{63}-1. The field **aduIdentifier** is specified by the ADU originator with the rule that the **aduIdentifier** shall be unique within the ADU type. With that rule, considering

that each ADU originator is univoquely identified by its **providerId**, each ADU may be univoquely identified by a triple of identifiers of types `ProviderId`, `AduType`, and `AduIdentifier`.

To ensure backwards compatibility, there is also another way to identify a toll declaration in the `TollDeclarationAdu` by the means of the value of its **tollDeclarationId** field (see Table 80).

To ensure backwards compatibility, there is also another way to identify a billing detail in the `BillingDetailsAdu` by the means of the value of its **billingDetailsInfo** field (see Table 84).

6.2.5 ADU action code

When sending or requesting an ADU, an action code may optionally be specified, to indicate to the receiver of the ADU which action is supposed to be done with that ADU. While the exact behaviour to be followed is implementation-specific and varies from ADU to ADU, the following generally available values are specified in this document:

— **send**, which specifies a normal (probably initial) sending of the ADU

— **revoke**, which specifies a revocation of a previously sent ADU

— **adjust**, which specifies an adjustment of a previously sent ADU

— **resend**, which specifies a resending of an ADU

6.2.6 User identification

A data type used in most ADUs to identify a user is the `UserId` data type. As a user can be identified in various ways, and by different methods, the `UserId` data type is represented as a series (ASN.1 SEQUENCE) of different types, all optional. However, when the data type `UserId` is used, at least one of the fields shall be specified. The structure of the `UserId` data type is shown in Table 6 and is formally specified as per Annex A.

Table 6 — UserId fields

Field name	Data type	m/o
pan	`PersonalAccountNumber`	o
contractSerialNumber	`ContractSerialNumber`	o
licencePlateNumber	`LPN`	o
obeId	Record (ASN.1 SEQUENCE) made of the following fields: **manufacturerId**, **equipmentObuId**	o
equipmentIccId	`EquipmentIccId`	o
customerId	`OCTET STRING`	o

pan may optionally be used to identify the user by their personal account number. It shall be used as specified in ISO/TS 17573-3.

contractSerialNumber may optionally be used to identify the user by their contract serial number. It shall be used as specified in ISO/TS 17573-3.

licencePlateNumber may optionally be used to identify the user by the licence plate number of the user's vehicle. It shall be used as specified in ISO/TS 17573-3.

obeId may optionally be used to identify the user by the OBE identifier of the user. The **obeId** field is made of the following data elements:

— **manufacturerId** which shall be used to uniquely identify the OBE manufacturer. It shall be used as specified in ISO 14906;

— **equipmentObuId** which shall be used to uniquely identify the OBE within the manufacturer. It shall be used as specified in ISO/TS 17573-3.

equipmentIccId may optionally be used to identify the integrated circuit card (ICC) which may be associated to a user of EFC equipment for the purposes of identification of payments. It shall be used as specified in ISO/TS 17573-3.

customerId may optionally be used to identify the user by means of their identifier as a TSP customer.

6.3 RequestAdu data structure

The `RequestAdu` data type is used to indicate a general readiness to receive, or to specifically request a data structure to be sent.

The `RequestAdu` type is a selection (ASN.1 CHOICE) among different substructures, each one specifying a different type of request. Table 7 describes the fields specified as CHOICE in the data structure, which is formally defined in Annex A.

Table 7 — RequestAdu fields

Field name	Data type/data description	m/o
genericRequest	Record (ASN.1 SEQUENCE) made of the following fields: **apduIdentifier**, **requestedAduType**, **aduIdentifier**, **actionCode**	N/A
exceptionListRequest	Record (ASN.1 SEQUENCE) made of the following fields: **requestedAduType**(exceptionListRequest), **aduIdentifier**, **period**, **exceptionListType**, **exceptionListVersion**, **actionCode**	N/A
trustObjectRequest	Record (ASN.1 SEQUENCE) made of the following fields: **requestedAduType**(trustObject), **aduIdentifier**, **trustObjectSpec**, **actionCode**	N/A
tollDeclarationRequest	Record (ASN.1 SEQUENCE) made of the following fields: **requestedAduType**(TollDeclaration), **aduIdentifier**, **userId**, **startTime**, **endTime**, **actionCode**	N/A
userDetailsRequest	Record (ASN.1 SEQUENCE) made of the following fields: **requestedAduType**(userDetails), **aduIdentifier**, **userId**, **listOfParametersRequested**, **userDetailsRequestReason**, **period**, **actionCode**	N/A
cccEventRequest	Record (ASN.1 SEQUENCE) made of the following fields: **requestedAduType**(reportCCCEventAdu), **aduIdentifier**, **userId**, **startTime**, **endTime**, **actionCode**	N/A
userListRequest	Record (ASN.1 SEQUENCE) made of the following fields: **requestedAduType**(provideUserIdListAdu), **aduIdentifier**, **userIdRequestType**, **userId**, **userIdRequestTime**, **actionCode**	N/A
mediaSettlementDataRequest	Record (ASN.1 SEQUENCE) made of the following fields: **requestedAduType**(mediaSettlementDataAdu), **aduIdentifier**, **period**, **equipmentIccId**, **mediaSettlementListVersion**, **actionCode**	N/A

If the **genericRequest** is selected, the following rules apply:

— If none of the fields is specified, the RequestAdu indicates a general readiness to receive;

— **apduIdentifier** may optionally be used to request that the APDU carrying the same identifier be (re) sent. If the **apduIdentifier** is specified, neither the **requestedAduType** nor the **aduIdentifier** may be specified. If the **apduIdentifier** is not specified, the **requestedAduType** or the **aduIdentifier** may be specified;

— if the **requestedAduType** field is specified, an ADU of the indicated data type is expected. It is used to indicate generic readiness to receive ADUs of the indicated type;

— **aduIdentifier** is a list (ASN.1 SEQUENCE OF) of identifiers of type `AduIdentifier` which may optionally be used to ask for retransmission of the specified ADU's. If **aduIdentifier** is specified, then **requestedAduType** shall be also specified, to indicate the type of the ADU's to be re-transmitted;

— **actionCode** may optionally be used to indicate specific actions to be pursued by the receiver of the requestAdu (see 6.2.5).

If the **exceptionListRequest** is selected, the following rules apply:

— **requestedAduType** field shall bear the value **exceptionListAdu**;

— if no optional fields are specified, this indicates generic readiness to receive ADUs of the ExceptionList type. In this case it is recommended to use a **genericRequest**, with the **requestedAduType** field bearing the value **exceptionListAdu**;

— **aduIdentifier** may optionally be used to identify the exceptionListAdu to be re-transmitted.

— **period** field may optionally be used to request to retransmit all ExceptionList ADUs that are related to the given period of time. If **period** is not specified, the request is not limited to a specific period of time;

— **exceptionListType** field may optionally be used to request to re-transmit all ExceptionList ADUs that are of the specified type;

— **exceptionListVersion** field may optionally be used to request to retransmit all ExceptionList ADUs that are of the specified version. If the field is not specified, the last available version is meant;

— **actionCode** may optionally be used to indicate specific actions to be pursued by the receiver of the requestAdu (see 6.2.5).

If the **trustObjectRequest** is selected, the following rules apply:

— **requestedAduType** field shall bear the value **trustObjectAdu**;

— if no optional fields are specified, this indicates generic readiness to receive ADUs of the `TrustObject` type. In this case it is recommended to use a **genericRequest**, with the **requestedAduType** field bearing the value **trustObjectAdu**;

— **aduIdentifier** may optionally be used to identify the requested TrustObjectAdu to be re-transmitted. If the **aduIdentifier** is specified, the **trustObjectSpec** field shall not be set;

— **trustObjectSpec** may optionally be used to specify the type and purpose of the requested trust object. The type of the trust object requested shall be specified in the field typeOfTrustObject, of type `RequestedTrustObjectType`, which is a selection (ASN.1 CHOICE) of:

  — certificates;

  — public keys;

  — DSRC master keys;

  — MAC master keys;

  — DSRC key references;

  — other trust objects.

  Detailed descriptions of trust objects types and purposes are in subclause 6.6.

  NOTE This document does not limit the use of individual combinations of trust object types and trust object purposes. In real systems, however, some of the theoretically possible combinations of these two parameters do not make sense and are not normally used in implementations. The correct use of these parameters remains under the responsibility of the implementor(s) and/or the entities operating the TC and TSP systems.

— **actionCode** may optionally be used to indicate specific actions to be pursued by the receiver of the requestAdu (see 6.2.5).

If the **tollDeclarationRequest** is selected, the following rules apply:

— **requestedAduType** field shall bear the value **tollDeclarationAdu**;

— if no optional fields are specified, this indicates generic readiness to receive ADUs of the TollDeclaration type. In this case it is recommended to use a **genericRequest**, with the **requestedAduType** field bearing the value **tollDeclarationAdu**;

— **aduIdentifier** may optionally be used to identify the requested TollDeclarationAdu to be re-transmitted. If the **aduIdentifier** is specified, the **userId**, **startTime,** and **endTime** fields shall not be specified;

— **userId** field may optionally be used to identify the user for which toll declarations are requested;

— **startTime** field may optionally be used to indicate a start of a period for which toll declarations are requested;

— **endTime** field may optionally be used to specify the end of a period for which toll declarations are requested. If **endTime** is specified, then the **startTime** shall be specified as well;

— **actionCode** may optionally be used to indicate specific actions to be pursued by the receiver of the requestAdu (see 6.2.5).

If the **userDetailsRequest** is selected, the following rules apply:

— **requestedAduType** field shall bear the value **provideUserDetailsAdu**;

— **aduIdentifier** may optionally be used to identify the requested UserDetailsAdu to be re-transmitted;

— **userId** field shall specify the identifier of the user for which additional details are requested;

— **listOfParametersRequested** may optionally be used to specify which additional parameters, if any, are required. If this field is not specified, then the response shall contain all available details;

— **userDetailsRequestReason** may optionally be used to specify the reason for the request;

— **period** may optionally be used to request user information related to the specified period in time, including cases where user information is varied during the period. If not specified, it is the currently available user information that is to be requested;

— **actionCode** may optionally be used to indicate specific actions to be pursued by the receiver of the requestAdu (see 6.2.5).

If the **cccEventRequest** is selected, the following rules apply:

— **requestedAduType** field shall bear the value “reportCCCEventAdu”;

— if no optional fields are specified, this indicates generic readiness to receive ADUs of the reportCccEvent type. In this case it is recommended to use a **genericRequest**, with the **requestedAduType** field bearing the value **reportCCCEventAdu**;

— **aduIdentifier** may optionally be used to identify the requested **reportCCCEventAdu** to be re-transmitted;

— **userId** field may optionally be used to specify the identifier of the user for which the CCC events are requested;

— **startTime** field may optionally be used to specify a start of a period for which CCC events are requested;

— **endTime** field may optionally be used to specify the end of a period for which CCC events are requested. If **endTime** is specified, the **startTime** shall be specified as well;

— **actionCode** may optionally be used to indicate specific actions to be pursued be the receiver of the requestAdu (see 6.2.5).

If the **userListRequest** is selected, the TC is meant to request a TSP to provide information on vehicles that belong to the same Service User. The following rules apply:

— **requestedAduType** field shall bear the value "**provideUserIdListAdu**";

— **aduIdentifier** may optionally be used to identify the requested `provideUserIdListAdu` to be re-transmitted;

— **userIdRequestType** field shall indicate the type of request. Currently, only one value is defined: **allUserIdsToGivenCustomer**;

— **userId** field may optionally be used to identify a vehicle or OBE that is known to the TC and where further vehicles or OBEs are requested that belong to the same service user. If the **userId** field is not specified, the request is intended to retrieve the identifiers of all service users of that TSP as filtered by the value of **userIdRequestType**;

— **userIdRequestTime** field may optionally be used to specify a point of time. If this attribute is specified, the TSP shall answer to the request based on data valid at the provided point of time and not at the time of the request. If this attribute is not specified, the latest information shall be provided;

— **actionCode** may optionally be used to indicate specific actions to be pursued be the receiver of the requestAdu (see 6.2.5).

If the **mediaSettlementDataRequest** is selected, the following rules apply:

— **requestedAduType** field shall bear the value "**mediaSettlementDataAdu**";

— if no optional fields are specified, this indicates generic readiness to receive ADUs of the `mediaSettlementData` type. In this case it is recommended to use a **genericRequest**, with the **requestedAduType** field bearing the value "**mediaSettlementDataAdu**";

— **aduIdentifier** may optionally be used to identify the requested mediaSettlementDataAdu to be re-transmitted;

— **period** may optionally be used to request to re-transmit all MediaSettlementDataAdus that are related to the given period of time;

— **equipmentIccId** may optionally be used to request to re-transmit all MediaSettlementDataAdus that contain the specified identifier;

— **mediaSettlementListVersion** may optionally be used to request to re-transmit MediaSettlementDataAdus that are of the specified version. If the field is not specified, the last available version is meant;

— **actionCode** may optionally be used to indicate specific actions to be pursued be the receiver of the requestAdu (see 6.2.5).

6.4 AckAdu data structure

The `AckAdu` data type is used to indicate that a specific data structure has been received. It can optionally indicate acceptance or rejection of the previously received data structure, or of the whole APDU that transported the data structure. When transferring this ADU, the value in the **informationRecipientID** field of the APCI shall identify a valid entity (a null value is not permitted).

Table 8 describes the fields in an `AckAdu` data type, which is formally defined in Annex A.

Table 8 — AckAdu fields

Field name	Data type/data description	m/o
apduIdentifier	`ApduIdentifier`. Integer ranging from 0 to 2^{63}-1	m
apduAckCode	`ApduReasonCode`. Integer ranging fro 0 to 255 (see Table 9)	m
apduAckText	`UTF8String`	o
issues	List of records (ASN.1 SEQUENCE OF SEQUENCE), each record defined as per Table 10	o
actionCode	`ActionCode`	o

apduIdentifier shall identify the APDU containing the data structure(s) being acknowledged.

apduAckCode shall contain the status of the APDU or a reason for accepting or rejecting the APDU. The **apduAckCode** accepts or rejects an APDU as a whole. If an APDU is accepted, there can be additional responses on individual elements of the APDU (ADUs). A single ADU could be rejected while the APDU itself is accepted. This is dealt with by using the optional **issues** field (see below). If the **apduAckCode** indicates rejection, the whole APDU is rejected, irrespective of the possible acceptance codes indicated in the **issues** field. The values defined in this document for **apduAckCode** are listed (in decimal notation) in Table 9.

Table 9 — ApduReasonCode values

Name	Semantics	Value
reserved	Code reserved for future ISO/CEN extensions.	0
obsoleteApdu	The APDU is no longer valid or to be used.	1
apduOk	The APDU was accepted.	2
apduNotOk	The APDU was rejected.	3
sequenceError	The APDU was rejected because the APDU was sent out of the logical sequence of a transaction.	4
otherReason	The APDU was rejected. No predefined APDU reason code could be used, or a bilateral agreement on the APDU reason code exists.	5
protocolVersionError	The APDU was rejected because the Protocol Version used is not supported by the recipient.	6
originatorRejected	The APDU was rejected because the Apdu Originator is not known, or no valid contract exists.	7
recipientUnknown	The APDU was rejected because the Information Recipient is not known, or no valid contract exists.	8
relatedApduUnknown	The APDU was rejected because the related APDU is unknown or the transaction was finished.	9
requestSentTooOften	The APDU was rejected because the request was sent too often or to early (processing of similar request still active).	10

The following rules apply for handling the reception of an AckAdu:

— if the **apduAckCode** field indicates rejection (value of apduNotOk), the whole referred APDU is rejected;

— if the **apduAckCode** field indicates acceptance, the referred APDU is accepted (syntactically and semantically correct;

apduAckText of type `UTF8String` may optionally be used to add a text to further explain the reason for accepting or rejecting the APDU.

EXAMPLE A number of ADUs of the type `BillingDetails` have been transferred in a previous APDU. The receiver accepts one specific BillingDetail and rejects all others. To do so, the receiver sends an AckAdu with the apduAckCode set to **apduOk** and indicates for each of the rejected ADUs the reason for rejection.

issues may optionally be used to list detailed error and/or warning information that respond to multiple instances of ADUs in the APDU being acknowledged. This supports the system in accepting an APDU as a whole, but to possibly individually reject single ADUs. The data structure of issues is specified in Table 10.

actionCode may optionally be used to indicate the action associated with the specific ADU (see 6.2.5).

Table 10 — issues fields

Field name	Data type/data description	m/o
issueAduIdentifier	`AduIdentifier`	o
issueLocation	`UTF8String`	o
issueContent	`UTF8String`	o
issueCode	`INTEGER`	m
issueText	`UTF8String`	o

issueAduIdentifier may optionally be used to specify the unique identifier of the ADU that contains an issue. It shall not be specified if the issue applies to all ADUs within the APDU.

issueLocation may optionally be used to address the location of the issue within the ADU by using XPath expression.

issueContent may optionally be used to provide further information on the content of the issue. The usage of this attribute is based on bilateral agreements between TC and TSP.

issueCode shall contain the reason code (see Table 11) of an issue.

issueText may optionally be used to provide a description of the issue. The usage of this attribute is based on bilateral agreements between TC and TSP.

Some values of the **AduReasonCode** field are reserved for CEN and ISO use. The values that are specified in this document are listed (in decimal notation) in Table 11 and are formally defined in Annex A. The value of the **AduReasonCode** provides feedback information to the sender or originator of an ADU about the acceptance or rejection of an ADU that was received and processed by the information recipient.

Table 11 — AduReasonCode values

Name	Semantics	Value
invalidAdu	The specified ADU is invalid.	0
notSupportedAdu	The specified ADU is not supported.	1
aduPartiallyRejected	Part of the specified ADU is rejected.	2
reserved	Codes reserved for general ADU-related errors.	3 .. 99
trustObjectExpired	The exchanged trust object is no longer valid.	100
trustObjectUnreadable	The received trust object is not readable.	101
reserved	Codes reserved for trust object data exchange errors.	102 .. 199
contextDataChargeObjectIdRejected	The charge object Id in context data has been rejected.	200
contextDataChargeObjectDescriptionRejected	The charge object description has been rejected.	201
contextDataSystemTypeRejected	The system type in context data has been rejected.	202
contextDataChargeObjectTypeRejected	The type of charge object in context data has been rejected.	203

Table 11 *(continued)*

Name	Semantics	Value
contextDataTollChargerRejected	The TC in context data has been rejected.	204
contextDataChargeObjectValidityStartRejected	The start of validity of the charge object in context data has been rejected.	205
contextDataChargeObjectValidityEndRejected	The end of validity of the charge object in context data has been rejected.	206
reserved	Codes reserved for context data exchange errors.	207 .. 299
reserved	Codes reserved for contract issuer exchange errors.	300 .. 399
exceptionListsVersionRejected	The exception list version has been rejected.	400
exceptionListsTypeRejected	The type of exception list has been rejected.	401
exceptionListsUserIdRejected	The user ID in exception list has been rejected.	402
exceptionListsStatusTypeRejected	The type of status in exception list has been rejected.	403
exceptionListsReasonTypeRejected	The reason code in exception list has been rejected.	404
exceptionListsDateAndTimeRejected	The date and time in exception list has been rejected.	405
reserved	Codes reserved for exception lists exchange errors.	406 .. 499
abnormalObeUserIdRejected	The user id in abnormal OBE has been rejected.	500
abnormalObeDateAndTimeRejected	The date and time in abnormal OBE has been rejected.	501
abnormalObeReasonCodeRejected	The reason code in abnormal OBE has been rejected.	502
reserved	Codes reserved for abnormal OBE exchange errors.	503 .. 599
tollDeclartionDuplicateTollDeclarationIdRejected	A duplicate of the tollDeclarationId has been identified.	600
tollDeclarationActionCodeRejected	The actionCode in a toll declaration ADU is unknown or not supported by the recipient of the toll declaration.	601
tollDeclarationChargeReportStructureRejected	One or more elements inside the charge report structure of the toll declaration ADU are not supported by the recipient of the toll declaration. NOTE For some data elements inside the charge report there are individual ADU reason codes defined below.	602
tollDeclarationActionCodeSendSentTooOftenRejected	The actionCode = send has been sent too often in toll declaration ADUs.	603
tollDeclarationInvalidChargeObjectRejected	One or more validations failed inside the elements chargeObjectId, or timeWhenUsed, or vehicleDescription.	604
tollDeclarationActionCodeAdjustSentTooOftenRejected	The actionCode = adjust has been sent too often in toll declaration ADUs.	605
tollDeclarationTariffClassIdRejected	The TariffClassId declared in the toll declaration ADU was not found by the recipient of the toll declaration.	606

Table 11 *(continued)*

Name	Semantics	Value
tollDeclarationTariffRejected	The tariff declared in the toll declaration ADU was not found by the recipient of the toll declaration.	607
tollDeclarationDuplicateVehiclePassageRejected	A duplicate vehicle passage has been identified by the recipient of the toll declaration. A duplicate vehicle passage is specified by the identical combination of: [chargeReport.obeId + chargeReport.vehicleLPNr + chargeReport.paymentMeans.personalAccountNumber + DetectedChargeObject.chargeObjectId + DetectedChargeObject.timeWhenUsed]	608
tollDeclarationInvalidDetectedChargeObjectManyRejected	One or more validation steps for a tollDeclaration failed in many DetectedChargeObjects.	609
tollDeclarationTollChargerIssueRejected	Toll declaration declined by TC due to error in TC system, further organizational handling required between TC and TSP. NOTE This can be used, for example, in case errors on the TC side prevent correct acceptance of a toll declaration.	610
tollDeclarationTollContextRejected	The tollContext declared in the toll declaration ADU is not known by the recipient of the toll declaration.	611
tollDeclarationTollFeeRejected	The toll fee decalared in the toll declaration ADU has been rejected by the recipient of the toll declaration. NOTE Depending on the use of optional data elements inside the toll declaration, toll fee information can be provided in different data elements.	612
reserved	Codes reserved for toll declarations exchange errors.	613 .. 699
billingDetailsIssuerIdRejected	The issuer ID in billing details has been rejected.	700
billingDetailsClaimIdRejected	The claim ID in billing details has been rejected.	701
billingDetailsTollChargerIdRejected	The TC ID in billing details has been rejected.	702
billingDetailsContextIdRejected	The context ID in billing details has been rejected.	703
billingDetailsUserIdRejected	The user ID in billing details has been rejected.	704
billingDetailsPeriodRejected	The period in billing details has been rejected.	705
billingDetailsAmountRejected	The amount in billing details has been rejected.	706
billingDetailsContextNameRejected	The context name in billing details has been rejected.	707
billingDetailsAppliedUserClassRejected	The applied user class in billing details has been rejected.	708
billingDetailsDeclaredVehicleClassRejected	The declared vehicle class in billing details has been rejected.	709

Table 11 *(continued)*

Name	Semantics	Value
billingDetailsAppliedTimeClassRejected	The applied time class in billing details has been rejected.	710
billingDetailsEntranceTimeRejected	The entrance time in billing details has been rejected.	711
billingDetailsEntranceChargeObjectRejected	The entrance charge object in billing details has been rejected.	712
billingDetailsIntermediateSectionRejected	The intermediate section in billing details has been rejected.	713
billingDetailsExitChargeObjectRejected	The exit charge object in billing details has been rejected.	714
billingDetailsExitTimeRejected	The exit time in billing details has been rejected.	715
billingDetailsReferenceRejected	The reference in billing details has been rejected.	716
reserved	Codes reserved for billing details exchange errors.	717 .. 799
claimRejectedByTSP	The related payment claim has been rejected by the TSP.	800
claimApprovedByTSP	The related payment claim has been approved and accepted by the TSP.	801
paymentClaimIdRejected	The ID in payment claim has been rejected.	802
paymentStartDateTimeRejected	The start date and time in payment claim has been rejected.	803
paymentEndDateTimeRejected	The end date and time in payment claim has been rejected.	804
paymentUserIdRejected	The user ID in payment claim has been rejected.	805
paymentAmountRejected	The amount in payment claim has been rejected.	806
paymentStatusRejected	The status in payment claim has been rejected.	807
paymentTypeOfFeeRejected	The type of fee in payment claim has been rejected.	808
paymentRelatedApduIdRejected	The related APDU ID in payment claim has been rejected.	809
reserved	Codes reserved for payment claim exchange errors.	810 .. 899
reserved	Codes reserved for payment announcement exchange errors.	900 .. 999
reserved	Codes reserved for provide user details exchange errors.	1 000 .. 1 099
reserved	Codes reserved for provide user lists exchange errors.	1 100 .. 1 199
reportQaAccepted	The quality assurance parameters have been accepted.	1 200
reportQaNotAccepted	The quality assurance parameters have been rejected.	1 201
paymentGuaranteeAccepted	The payment guarantee has been accepted.	1 202
smCCBackEndDataCheckingThresholdExceeded	Secure monitoring reason code.	1 203

Table 11 *(continued)*

Name	Semantics	Value
reserved	Codes reserved for quality assurance exchange errors.	1 204 .. 1 299
complaintIdMissing	Identifier missing in a UserComplaintAdu.	1 301
relatedBillingDetailsUnknown	The billing details ID referred to in the user complaint ADU is not known by the TC.	1 302
paymentClaimUnknown	The payment claim ID referred to in the user complaint ADU is not known by the TC.	1 303
dateOfComplaintIncorrect	The date of the complaint is incorrect.	1 304
dateOfComplaintMissing	The date of the complaint is missing.	1 305
complaintReasonMissing	The reason of the complaint is missing.	1 306
complaintReasonUnknown	The reason of the complaint is not known.	1 307
addComplaintReasonMissing	Additional complaint information is missing and the reason of the complaint is "other".	1 308
additionalInfoIdMissing	The ID in the additional complaint information is not specified.	1 309
infoDateTimeMissing	The field date in additional information is missing.	1 310
infoDateIncorrect	The field date in additional information is incorrect.	1 311
incorrectAuthenticator	The authenticator is wrong.	1 312
relatedBillingDetailsAlreadyComplained	The referred billing details is mentioned in another user complaint currently being processed.	1 313
reserved	Codes reserved for complaint-related errors.	1 314 .. 1 399
reserved	Codes reserved for future ISO/CEN extensions.	1 400 .. 2 999
semanticError	The semantics of the message received was not understood by the recipient. NOTE Every message (except AckAdu) can be responded to with a "semantic error" code when its semantics are not understood.	3 000
reserved	Codes reserved for future ISO/CEN extensions.	3 001 3 009
actionCodeNotSupported	The sent action code is not supported. NOTE 4 This AduReasonCode is used only in response to ADUs which contain action codes.	3 010
reserved	Codes reserved for future ISO/CEN extensions.	3 011 3 999
acceptedWithWarning	The ADU was accepted and will be processed by the recipient. Further information will be provided by means of warnings. NOTE Bilateral agreement on any other issue which semantics needs to be specified for the attributes issueLocation, issueContent and issueText.	4 000
reserved	Codes reserved for future ISO/CEN extensions.	4 001 ... 9 999

Table 11 *(continued)*

Name	Semantics	Value
reserved	Codes reserved for private extensions defined in bilateral agreements between TCs and TSPs.	1 000 0 ... 6 553 5

6.5 StatusAdu data structure

The `StatusAdu` data type can be used to indicate to the recipient that previously sent (and possibly acknowledged) APDUs are obsolete (e.g. to support rollback scenarios) or to indicate the inability to receive APDUs. When transferring this ADU, the value in the **informationRecipientID** field of the APCI shall identify a valid entity (a null value is not permitted).

Table 12 indicates the fields of the `StatusAdu` data type, which is formally defined in Annex A.

Table 12 — StatusAdu fields

Field name	Data type/data description	m/o
generalStatusCode	`GeneralStatusCode`	m
apduStatusCode	Record (ASN.1 SEQUENCE) made of the following fields: apdu, reasonOfAduStruct	o
actionCode	`ActionCode`	o

generalStatusCode shall indicate readiness to receive APDUs. The following values are allowed:

— **notReadyToReceive**: the issuing entity is not ready to receive;

— **readyToReceive**: the issuing entity is ready to receive.

apduStatusCode may optionally be used to provide APDU-specific reason codes as a sequence of one or more reason codes for each by ADU in a related APDU. The **apdu** field identifies the APDU that contains the ADUs. The **reasonOfAduStruct** is a list (ASN.1 SEQUENCE OF SEQUENCE) made of two fields: **aduIdentifier**, which identifies an ADU within the addressed APDU, and **reason,** which carries an `AduReasonCode` value, defined in Table 11.

actionCode may optionally be used to indicate specific actions to be pursued by the receiver of the statusAdu (see 6.2.5).

6.6 TrustObjectAdu data structure

The `TrustObjectAdu` data type can be used by either the TC or the TSP to provide trust objects. The ADU can be either sent as a response to a Request ADU or initiated as a push ADU if new versions of trust objects are available. When transferring this ADU, the value in the **informationRecipientID** field of the APCI may have a null value, to indicate that the ADU is intended for all possible recipients.

Each `TrustObjectsAdu` data structure shall contain one trust object and additional information and parameters for that trust object.

Table 13 indicates the fields in a `TrustObjectsAdu` data type, which is formally defined in Annex A.

Table 13 — TrustObjectAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
replacedTrustObject	`AduIdentifier`	o
purposesOfTrustObject	Record made of fields of the same type (ASN.1 SEQUENCE OF), each of type `TrustObjectPurpose`	m

Table 13 *(continued)*

Field name	Data type/data description	m/o
startValidity	`GeneralizedTime`	m
endValidity	`GeneralizedTime`	o
trustObjectStatus	`TrustObjectStatus`	m
trustObject	`TrustObjectCode`	o
actionCode	`ActionCode`	o

aduIdentifier shall indicate the unique identifier for the `TrustObjectAdu`. The uniqueness of the value of **aduIdentifier** shall be guaranteed by the initiator of the trust object (i.e. TC or TSP) inside the realm of the same provider ID.

replacedTrustObject may optionally be used to contain the trust object identifier of the trust object which shall be replaced by the new trust object that is transferred using the present `TrustObjectAdu` structure. This optional data element shall only be present and used in case an already existing and valid trust object shall be replaced by a new trust object. If **replacedTrustObject** is present, the new trust object shall have the **TrustObjectID** set to a new value.

purposesOfTrustObjects shall be used to indicate the possible usage(s) for which the trust object is intended. It contains a list of values which it supports in specifying the possible usage of the same trust object for one or more than one purposes. Allowed values shall be:

— **trustObjects**, unspecified purpose;

— **dSRCCharging**, to validate authenticators from DSRC charging applications;

— **dSRCAC**, to calculate access credentials for DSRC charging applications;

— **oBEInterrogation**, to validate authenticators received during a CCC transaction;

— **oBEInterrogationAC**, to calculate access credentials for CCC;

— **sIGExceptionList**, to authenticate received exception lists;

— **sIGContextData**, to authenticate received context data;

— **sIGBillingDetails**, to authenticate received billing details;

— **sIGFiscalObjects**, to authenticate received fiscal objects;

— **sIGCommunication**, to authenticate received ADUs by infoExchangeAuthenticator;

— **eNCCommunication**, to decrypt received ADUs;

— **dSRCKeyEncryption**, to encrypt DSRC keys;

— **secChannelEstablishment**, certificate to establish IPSec (virtual private network [VPN]);

— **certIssuing**, certificate to issue certificates;

— **sIGUserCommunication**, user certificate to verify communication via, e.g. email;

— **certRevocationListing**, certificate revocation list;

— **siGChargeReport**, certificate or certificate chain to validate charge report;

— **oBEInterrogationNonRep**, key reference for the non-repudiation authenticatication for CCC or localization augmentation communication (LAC).

NOTE This document does not limit the use of individual combinations of trust object types and trust object purposes. In real systems, however, some of the theoretically possible combinations of these two parameters do not make sense and can therefore not be used in implementations. The correct use of these parameters remains under the responsibility of the implementor(s) and/or the entities operating the TC and the TSP systems.

startValidity shall be used to indicate the time from which the trust object is valid. The actual initial time of usage of the trust object may be agreed by the TC and the TSP by separate arrangements that are outside the scope of this document.

endValidity may optionally be used to indicate the expiry date for a trust object. In case this optional element is not present, the specified trust object has no expiry date.

trustObjectStatus shall be used to indicate the status of the specified trust object. The following values shall be used:

— **valid**, for a valid trust object;

— **expired**, for an expired trust object;

— **revoked**, for a revoked trust object;

— **update**, for a trust object which replaces an already existing and valid trust object.

trustObject is of data type `TrustObjectCode` and may optionally be used to specify the value of the trust object. If **trustObject** is omitted, a new value of **trustObjectStatus** is sent for an already transferred trust object.

TrustObjectCode is supported in this document for the following trust object types:

— certificates;

— public keys;

— DSRC keys;

— MAC keys;

— DSRC key references;

— generic trust objects.

For each specified type of trust object, a different definition applies, as per the following.

The `CertificateObject` data type supports exchanging certificates between TC and TSP. The certificate structure and coding shall be according to ISO/IEC 9594-8. Table 14 explains the fields in the `CertificateObject` data type, which is formally defined in Annex A.

Table 14 — CertificateObject fields

Field name	Data type/data description	m/o
certificateType	`CertificateType`	m
certificate	`OCTET STRING` containing the certificate	m

The **certificateType** field specifies the type of certificate. The following values are specified:

— **certTcHTTPS**, indicating a TC certificate for HTTPS usage;

— **certTcMAIL**, indicating a TC certificate for email exchange;

— **certTcNSIG**, indicating a TC certificate for signatures;

— **certTcTA**, indicating the TCs' sub-certification authority certificate;

— **certTcpRoot**, indicating the TCs' root-certification authority certificate;

— **certTspHTTPS**, indicating a TSP certificate for HTTPS usage;

— **certTspMAIL**, indicating a TSP certificate for email exchange;

— **certTspNSIG**, indicating a TSP certificate for signatures;

— **certTspTA**, indicating the TSP sub-certification authority certificate;

— **certTspRoot**, indicating the TSP root-certification authority certificate;

— **certTspCR**, indicating the TSP charge report verification.

The **certificate** field shall contain the actual public-key certificate according to ISO/IEC 9594-8.

The `PublicKeyObject` data type is used to transfer public keys whose authenticity is proven by a fingerprint exchanged over a separate communication channel. Public keys without a certificate may be used to establish the initial trust relation between TC and TSP. Table 15 explains the fields in the `PublicKeyObject` data type, which is formally defined in Annex A.

Table 15 — PublicKeyObject fields

Field name	Data type/data description	m/o
publicKeyType	`PublicKeyType`	m
serialNumber	`INTEGER` ranging from 0 to 2^{63}-1	m
Issuer	`Provider`	m
algorithmIdentifier	Selection (ASN.1 CHOICE) between **rsa** and **ecc** fields	m
publicKey	`OCTET STRING`	m

publicKeyType shall specify the type of key. A key can be:

— **kpupTcSignature**, which is a signature key of the TC;

— **kpupTcEncrypt**, which is an encryption key for the TC;

— **kpupTspSignature**, which is a signature key of the TSP;

— **kpupTspEncrypt**, which is an encryption key for the TSP.

serialNumber shall indicate the key's serial number.

issuer shall identify the issuer of the key.

algorithmIdentifier shall identify the used public key algorithm, which can be either Rivest, Shamir and Adleman (RSA) algorithm, as imported from ISO/IEC 9594-8, or elliptic-curve cryptography algorithm.

publicKey field shall contain the key.

The `DSRCKeyObject` data type supports the exchange of CCC and LAC DSRC master keys. In addition, the structure supports sending a DSRC master key, which is encrypted by means of a public key without a certificate. Table 16 explains the fields in the `DSRCKeyObject` data type, which is formally defined in Annex A.

Table 16 — DsrcKeyObject fields

Field name	Data type/data description	m/o
encryptionKeyId	Selection (ASN.1 CHOICE) between **certificate** and **publicKey** defined as `EncryptionKeyId` type	m
timestamp	`UTCTime`	m
dsrcKeys	Series of records (ASN.1 SEQUENCE OF SEQUENCE), each record containing the following fields: **efcCm**, **key**	m

encryptionKeyId of type `EncryptionKeyId` is a selection (ASN.1 CHOICE) between two types of trust objects identified by the fields **certificate** and **publicKey** which shall be used to identify the encryption type used. Both fields **certificate** and **publicKey** are defined as a record (ASN.1 SEQUENCE) of two elements (**serialNumber**, **issuer**) identifying either a certificate or a public key by means of its originator and the originator's assigned serial number.

timestamp of type `UTCTime` shall contain the time of the encryption.

dsrcKeys is a list (ASN.1 SEQUENCE OF) of couples (ASN.1 SEQUENCE), each made of elements of types **EfcCm** (described in Table 17 and formally described in Annex A) and `EncrKey` (described in Table 18 and formally described in Annex A).

Table 17 — EfcCm fields

Field name	Data type/data description	m/o
efcContextMark	`EfcContextMark`	m
bitmask	`OCTET STRING`	m
efcContextMarkVersion	`EfcContextMarkVersion`	o

efcContextMark shall be used to specify the context mark to be used either in a DSRC tolling environment or for CCC.

bitmask shall be used to indicate which type of contract and context version the key applies to by setting the applicable bits corresponding to the position in the concatenation of the elements **typeOfContract** and **contextVersion** of **efcContextMark**. A 0 value means ignore the respective element, while a 1 value means consider the respective element.

NOTE If a service provider uses several values for type of contract, using a corresponding bitmask allows the transmission of only one key that applies to all EFC context marks with the various values of type of contract.

EXAMPLE The bitmask "0000 0000 0000 0000 1111 1111" allows the transfer of a key that is applicable to all EFC Contextmarks with the specified contextVersion. The values of the attribute typeOfContract are not considered. The key is valid for all values of typeOfContract.

efcContextMarkVersion may optionally be used to indicate the version of the EfcContextMark used. Its values reflect the contents of EN 15509:2020, Table A.1, and are defined as:

— **en15509Version0**, meaning the current version of EN 15509;

— **en15509Version2007**, meaning the 2007 version of EN 15509;

— **en15509Version2014**, meaning the 2014 version of EN 15509;

— **en15509Version2021**, meaning the 2021 version of EN 15509;

— **iso12813Version0**, meaning the current version of ISO 12813;

— **iso12813Version2009**, meaning the 2009 version of ISO 12813;

— **iso12813Version2015**, meaning the 2015 version of ISO 12813;

— **iso12813Version2015Amd12017**, meaning the 2015 version of ISO 12813, amended per Amendment 1 2017;

— **iso12813Version2019**, meaning the 2019 version of ISO 12813;

— **iso13141Version0**, meaning the current version of ISO 13141;

— **iso13141Version2010**, meaning the 2010 version of ISO 13141;

— **iso13141Version2015**, meaning the 2015 version of ISO 13141;

— **iso13141Version2015Amd12017**, meaning the 2015 version of ISO 13141, amended per Amendment 1 2017.

Table 18 — EncrKey fields

Field name	Data type/data description	m/o
keyType	`KeyType`	m
keyRef	`INTEGER` ranging from 0 to 255	m
encrKey	`OCTET STRING`	m
kVC	`OCTET STRING`	m
keyDescription	`PrintableString`	o

keyType is a record (ASN.1 SEQUENCE) defined as type `KeyType` which shall be used to identify the type of key. It is made of the the following fields:

— **normativeReference**: a selection (ASN.1 CHOICE) of either an `OBJECT IDENTIFIER` or a `UTF8String` which may optionally be used to identify the standard that identifies the key;

— **keyUsage**: shall be used to specify the way in which the key is to be used. The following values are defined for `KeyUsage`:

  — **authentication**, to compute authenticators;

  — **accessCredentials**, to compute access credentials;

  — **encryption**, to encrypt data;

  — **otherUsage**, for other usages.

keyRef shall be used to identify the key to be used in a DSRC tolling environment according to ISO 14906, or for CCC according to ISO 12813.

encrKey shall contain the key, encrypted using the certificate referred to by the **encryptionKeyId** field of `DsrcKeyObject`.

kVC shall be used to provide the Key Verification Code according to ISO 11568-2, calculated encrypting one block size of zeros with the plain key, then truncated to leftmost three bytes.

keyDescription may optionally be used to provide a textual description of the key.

Some data elements are possibly authenticated by a derived MAC. The `MacKeyObject` data type supports the exchange of MAC master keys. Table 19 explains the fields in the `MacKeyObject` data type, which is formally defined in Annex A.

Table 19 — MacKeyObject fields

Field name	Data type/data description	m/o
serialNumber	`CertificateSerialNumber` as imported from ISO/IEC 9594-8	m
issuer	`Name`, as imported from ISO/IEC 9594-8	m

Table 19 *(continued)*

Field name	Data type/data description	m/o
timestamp	`UTCTime`	m
algorithmIdentifier	Selection (ASN.1 CHOICE) between **des** and **aes** fields	m
masterKeyRef	`INTEGER`, ranging from 0 to 2^{32}-1	m
encrKey	`OCTET STRING`	m
kVC	`OCTET STRING`	m

serialNumber shall specify the serial number of the certificate used.

issuer shall specify the name of the certificate's receiver.

timestamp shall specify the time of the encryption.

algorithmIdentifier shall identify the used encryption algorithm, which can be either DES or AES.

masterKeyRef shall identify the master key used for encryption.

encrKey shall contain the key that has encrypted using the certificate referred to by serialNumber and issuer of key.

kVC shall contain the key verification code according to ISO 11568-2, calculated encrypting one block size of zeros with the plain key, then truncated to leftmost three bytes.

The `DSRCKeyRef` data type shall be used to exchange a non-repudiation key reference for, for example, CCC (ISO 12813) or LAC (ISO 13141). Table 20 explains the fields in the `DSRCKeyRef` data type, which is formally defined in Annex A.

Table 20 — DSRCKeyRef fields

Field name	Data type/data description	m/o
efcContextMark	`EfcContextMark`	m
keyRef	`INTEGER`, ranging from 0 to 255	m
referenceType	`INTEGER` defined as `ReferenceType`	m
efcContextMarkVersion	`EfcContextMarkVersion`	o

efcContextMark shall contain the EFC context mark as defined in ISO/TS 17573-3.

keyRef shall contain the reference to the key to be used in a DSRC tolling environment according to ISO 14906, or for CCC according to ISO 12813.

referenceType shall specify the usage of the key. The following values are supported:

— **cCCNonRepKeyRef**, used for non-repudiation in CCC (ISO 12813);

— **cCCAuthenticationKeyRef**, used for authentication in CCC (ISO 12813);

— **lACAuthenticationKeyRef**, used for authentication in LAC (ISO 13141).

efcContextMarkVersion may optionally be used to indicate the version of the EfcContextMark.

The `GenericTrustObject` data type supports exchanging a trust object that does not have a pre-defined structure. Table 21 explains the fields in the `GenericTrustObject` data type, which is formally defined in Annex A.

Table 21 — GenericTrustObject fields

Field name	Data type/data description	m/o
typeOfTrustObject	`INTEGER`, ranging from 0 to 255	m
genericTrustObject	`OCTET STRING`	m

The **typeOfTrustObject** field shall specify the type of trust object, defined as `TrustObjectType` data type. It may be a certificate, a symmetric key, a certificate revocation list, a certificate revocation list (CRL) distribution point, an online certificate status protocol (OCSP) address, an encrypted symmetric key, a public key, a key reference or an unspecified trust object (other trust object) as defined in Annex A. The following values are supported:

— **certificate**, used for certificates;

— **symmetricKey**, used for symmetric keys;

— **crl**, used for certificate revocation lists;

— **cdp**, used for cdps;

— **ocsp**, used for ocsp;

— **encryptedSymmetricKey**, used for encrypted symmetric keys;

— **publicKey**, used for public keys;

— **otherTrustObject**, used for other types of trust objects.

The **genericTrustObject** field shall contain the trust object.

6.7 EfcContextDataAdu data structure

6.7.1 General

The `EfcContextDataAdu` data type shall be used by the TC either as a response to the request ADU sent by the TSP or as a push ADU if a new version of EFC context data attributes is available.

The `EfcContextDataAdu` data type contains TC-related administrative information, defined by the `EntityOverview` data type, followed by data structures that specify tariffs and classes of users, vehicles, and time periods where tariffs apply for the toll domain, and specific information for either the `GeneralContextData` data type, or the `MeshedContextData` data type, which are two alternative specifications of geographical-tolling-related information of the toll domain. All of these structures are optional (with the exception of **aduIdentifier**) and contain elements with their own **version** fields, so they support not being obliged to transfer the whole EfcContextData each time only a part is updated.

The **`EfcContextDataAdu`** data type is described in Table 22 and is formally defined in Annex A.

Table 22 — EfcContextDataAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
entityOverview	List of (ASN.1 SEQUENCE OF) at least one set of records (ASN.1 SEQUENCE) of fields describing the toll context (see 6.7.4.1)	o
tollContextProperties	Record (ASN.1 SEQUENCE) made of fields specifying tariffs and toll related information (see 6.7.4).	o
domainType	Choice (ASN.1 CHOICE) among one of the following types: `GeneralContextData`, `MeshedContextData`	o
actionCode	`ActionCode`	o

aduIdentifier shall contain an unambiguous identifier for the `EfcContextDataAdu` (see 6.2.4).

entityOverview may optionally be used to provide administrative information about the TC responsible for the EFC context data. It is described in Table 55 (6.7.4), and is formally defined in Annex A.

tollContextProperties may optionally be used to specify toll specific information and rules and is a record (ASN.1 SEQUENCE) made of the following fields whose specifications are grouped in 6.7.4:

— **tollContextOverview**, which may optionally be used to name the toll context and provide the list of the identifiers of its partitions (see Table 57);

— **tariffTable**, which may optionally be used to specify the tariffs that apply to the toll domain (see Table 58);

— **currencyConversionTable**, which may be optionally used to specify the currency or currencies used and their conversion rates (see Table 62);

— **tariffClassDefinition**, which may optionally be used to specify which tariff classes are applied for the toll domain (see Table 63);

— **localVehicleClassDefinition**, which may optionally be used to specify the defined vehicle classes in the toll domain (see Table 65);

— **timeClassDefinition**, which may optionally be used to specify the time classes that determine toll in the toll domain (see Table 69);

— **userClassDefinition**, which may optionally be used to specify the recognized user classes in the toll domain (see Table 71);

— **locationClassDefinition**, which may optionally be used to specify the defined location classes in the toll domain (see Table 73).

domainType may optionally be used to specify the type of the toll context and shall contain the actual EFC supporting data which shall be applied in the toll context. Either `GeneralContextData`, or `MeshedContextData` data types may be selected. Subclauses 6.7.2 and 6.7.3 describe in detail the two types of EFC domains.

actionCode may optionally be used to indicate the action associated with the specific ADU (see 6.2.5).

6.7.2 GeneralContextData type

6.7.2.1 General

The `GeneralContextData` **data type** shall contain the EFC context data and supporting information for a generalized toll context with a simple (section-based, area-based or cordon-based) layout. It is described in Table 23 and is formally defined in Annex A.

Table 23 indicates the fields in the `GeneralContextData` data type, which is formally defined in Annex A.

Table 23 — GeneralContextData fields

Field name	Data type/data description	m/o
tollContextPartitionOverviews	`TollContextPartitionOverview` (see 6.7.2.2)	o
tollContextPartitionLayouts	List (ASN.1 SEQUENCE OF) elements of type `TollContextPartitionLayout` (see 6.7.2.3)	o
tollContextVersion	`VersionAndValidity`	o

tollContextVersion may optionally be used to indicate the version of the provided information. It is of type `VersionAndValidity`, which is described in Table 24 and formally defined in Annex A.

Table 24 — VersionAndValidity data structure

Field name	Data type/data description	m/o
version	OCTET STRING	m
validFrom	GeneralizedTime	m

version of type OCTET STRING shall contain the version information of the attribute to which it refers;

validFrom of type GeneralizedTime shall contain the starting date and time of the validity of the attribute to which it refers.

6.7.2.2 TollContextPartitionOverview

The TollContextPartitionOverview data type shall list all information about the identification and type of the toll context (e.g. section-based tolling, area charging, cordon charging), time zone information and information regarding the operational status of the respective toll context partition. The main purpose of the TollContextPartitionOverview data type is to give a minimum amount of basic information regarding a toll context partition. At least one field of the TollContextPartitionOverview data type shall be present, which represents a toll context that consists of one single partition.

Table 25 indicates the fields in the TollContextPartitionOverview data type, which is formally defined in Annex A.

Table 25 — TollContextPartitionOverview fields

Field name	Data type/data description	m/o
tollContextPartitionId	TollContextPartitionId	m
tollContextPartitionName	UTF8String	o
tollContextPartitionType	TollSchemeType	m
operationalStatus	Record (ASN.1 SEQUENCE) made of the following fields: **startsOperationAt** (of type GeneralizedTime), **stopsOperationAt** (of type GeneralizedTime)	m
timeZone	INTEGER ranging from -720 to 720	m
dstInformation	Record (ASN.1 SEQUENCE) made of the following fields: **dstOffset** (of type INTEGER ranging from -120 to 120), **dstStartDate** (of type GeneralizedTime), **dstEndDate** (of type GeneralizedTime)	o
tollContextPartitionBoundingPolygon	Polygon	o
sendChargeReportIfEntering	BOOLEAN	o
precedenceLevel	INTEGER ranging from 0 to 255	o
chargeReportFinalRecipient	Provider	o
tollContextPartitionOverviewVersion	VersionAndValidity (see Table 24)	m

tollContextPartitionId shall contain the identifier of the respective toll context partition for which the overview information is provided. The toll context partition ID provided in this field shall correspond to one element in the list of toll context partition IDs in the attribute tollContextOverview.

tollContextPartitionName may optionally contain a designation for the toll context partition.

tollContextPartitionType shall contain information regarding the type of the toll context partition (road section charging, distance-based area charging, time-based area charging, cordon charging). The following values are specified:

— **roadSectionPricing**, which indicates a road-section-based charging

— **areaPricingDistance**, which indicates a distance-based area charging

— **areaPricingTime**, which indicates a time-based area charging

— **cordonPricing**, which indicates a cordon charging

operationalStatus shall contain information regarding operational status and the period of operation of the toll context partition. It is made of the following fields:

— **startsOperationAt** of type `GeneralizedTime` shall contain date and time when the toll scheme starts or has started operation

— **stopsOperationAt** of type `GeneralizedTime` may optionally specify date and time when the toll scheme will stop operation

timeZone shall specifiy the offset in relation to the coordinated universal time (UTC) time for the toll context partition, expressed in minutes. When a toll domain geographically spans over more than one time zone, this toll domain shall be split into two separate toll context partitions and have two toll context partition data sets.

All absolute time information used in EFC attributes and data (especially in the time class specifications) shall be given in local real time. In cases where daylight saving time (DST) applies, the involved systems shall be capable of re-calculating time information taking into consideration the applicable legal rules.

dstInformation is a record (ASN.1 SEQUENCE) which may optionally contain information regarding the DST, in terms of a DST offset to the default local time, where positive values shall be used in cases where the DST is "ahead" of default local time, and negative values shall be used in cases where the DST is "behind" the default local time, plus the start and end dates of the daylight saving period.

tollContextPartitionBoundingPolygon of type `Polygon` may optionally contain a polygon representation that encloses the geographic area of the toll context partition. The Polygon data type shall be specified by a list of elements of type `Point`. A `Point` type is defined by choosing (ASN.1 CHOICE) one of the representations shown in Table 26 and is different from the type with the same name defined in ISO/TS 17573-3. The points shall be specified in an order that creates and closes the polygon in a clockwise direction. The segments of the polygon shall be created by connecting each of the points with its successor point. The polygon shall be closed by connecting the last point in the list with the first point. The points shall be specified in a way that means the connections between the points do not intersect. The inner side of the polygon (tolled area) is defined as the geographical area that is located at the right-hand side when moving on a polygon segment from its start point to its end point

Table 26 — Point data type

Field name	Data type/data description	m/o
absolutePointCoordinates	`AbsolutePosition3d`	n/a
relativePointCoordinates	`RelativePosition3d`	n/a

absolutePointCoordinates shall contain the absolute point coordinates by using the attributes **longitude**, **latitude** and optionally **altitude** as specified in ISO/TS 17573-3.

relativePointCoordinates shall contain the relative point coordinates by using the attributes **longitude**, **latitude** and optionally **altitude** relative to a reference point, as specified in ISO/TS 17573-3.

sendChargeReportIfEntering shall be applied in order to require the generation and transmission of a charge report in cases where the OBE is entering the geographical area of the respective toll context partition. The geographical area of the toll context partition is specified in the data element **tollContextBoundingPolygon**.

precedenceLevel may optionally be used to specify priorities for overlapping toll context partitions. In cases where two or more toll context partitions have got the same preceded level, charge reports shall be generated and transmitted for each individual toll context partition. In cases where the precedence levels of overlapping toll context partitions have different values, only charge reports for the toll context

partition having the higher precedence level shall be generated and transmitted. A precedence level value of 0 shall be used for the lowest priority and a value of 255 shall be used for the highest priority.

chargeReportFinalRecipient may optionally be used to provide supplementary information for the management of charge reports in cases where one TC operates the toll scheme of another TC. In such cases, a dedicated toll context partition shall be used. The data element shall hold the identification of the entity that is the final destination of the respective usage data generated by the TSP.

tollContextPartitionOverviewVersion shall contain version and validity information for the attribute tollContextPartitionOverview.

6.7.2.3 TollContextPartitionLayout data type

6.7.2.3.1 General

The data type `TollContextPartitionLayout` shall describe all partitions that apply in one toll context. It shall contain a list of single charge objects, including their identifiers, their geographic description and additional information such as applicable location classes. The main purpose of the data type `TollContextPartitionLayout` is to provide all information that supports the detection (and measurement) of toll liable usage of a tolled infrastructure.

Table 27 indicates the fields in the `TollContextPartitionLayout` data type, which is formally defined in Annex A.

Table 27 — TollContextPartitionLayout fields

Field name	Data type/data description	m/o
tollContextPartitionId	`TollContextPartitionId`	m
layoutDescription	`Layout`	m
geoRefPoint	`Point`	o
tollContextPartitionVersion	`VersionAndValidity` (see Table 24)	m

tollContextPartitionId of type `INTEGER` ranging from 0 to 255 defined as `TollContextPartitionId` shall contain the identifier of the toll context partition.

layoutDescription shall contain the description of the layout of the respective toll context partition. The `Layout` data type is a selection (ASN.1 CHOICE) among fields of different types, which are described in Table 28, and are formally defined in Annex A.

geoRefPoint may be used as the geographic reference point for the toll context partition by either using absolute point coordinates or relative point coordinates. This reference point shall be specified by its longitude and latitude values. In cases where this data element is present, all longitude and latitude values given in the layout description shall be relative to this point.

tollContextPartitionVersion shall contain version and validity information for the attribute **tollContextLayout**.

Table 28 — Layout selection fields

Field name	Data type/data description	m/o
sectionLayoutDescription	`SectionLayout` (see Table 29)	n/a
gdfLayoutDescription	`GdfLayout` (see Table 35)	n/a
areaPricingLayout	`AreaPricingLayout` (see Table 41)	n/a
cordonPricingLayout	`CordonPricingLayout` (see Table 44)	n/a

The following rules for selecting the appropriate type apply:

— **sectionLayoutDescription** shall be used for section or road-object-based charging schemes

— **gdfLayoutDescription** shall be used for section or road-object-based charging schemes using a geographic data file (GDF) definition

— **areaPricingLayout** shall be used for area charging schemes

— **cordonPricingLayout** shall be used for cordon charging schemes

6.7.2.3.2 SectionLayout data type

The data type `SectionLayout` is a record (ASN.1 type SEQUENCE) that describes the section-based layout in the toll context partition. The entire charged road network may consist of one or more tolled roads and shall contain at least one or more sections. The data type `SectionLayout` is described in Table 29 and is formally defined in Annex A.

Table 29 — SectionLayout fields

Field name	Data type/data description	m/o
sectionLayoutDescription	`SectionLayout` (see Table 29)	n/a
gdfLayoutDescription	`GdfLayout` (see Table 35)	n/a
areaPricingLayout	`AreaPricingLayout` (see Table 41)	n/a
cordonPricingLayout	`CordonPricingLayout` (see Table 44)	n/a

tolledRoads may optionally be used to specify a list of one or more elements of type `TolledRoad` that provide information about the tolled roads in the toll context partition. The data type `TolledRoad` is a record (ASN.1 SEQUENCE) of elements that is described in Table 30 and is formally defined in Annex A.

sections shall contain a list of elements of type **`Section`** that describe the sections in the toll context partition. It shall contain at least one element. The data type **`Section`** is a record (ASN.1 type SEQUENCE) of using elements that is described in that described in Table 31 and is formally defined in Annex A.

EXAMPLE 1 The Öresund bridge is a section-based toll regime, which is made of only one section.

EXAMPLE 2 Austrian GO Maut and German LKW-Maut systems are considered as section tolling schemes. Both tolling schemes consist of a certain number of road sections. In both EFC schemes the fee depends on predefined length information per section.

Table 30 — TolledRoad fields

Field name	Data type/data description	m/o
tollRoadId	`TolledRoadId`	m
tollRoadCountryCode	`UTF8String`	o
tollroadNetworkName	`UTF8String`	o
tollRoadName	`UTF8String`	o
tollroadDescription	`UTF8String`	o
tollroadDirection	Record (ASN.1 SEQUENCE) made of the following fields: **startSection**, **endSection**.	o

tollRoadId shall be used to uniquely identify the tolled road within the toll context partition by means of the data type `TollRoadId`, which is an INTEGER ranging from 0 to 65535.

tollRoadCountryCode may optionally be used to specify the country code of the tolled road according to ISO 3166-1, Alpha-2.

tollRoadNetworkName may optionally be used to specify the name of the network name of the tolled road.

EXAMPLE 3 A1, E59

tollRoadName may optionally be used to specify the commonly used road name of the tolled road.

EXAMPLE 4 Westautobahn

tollRoadDescription may optionally be used to textually describe the tolled road.

tollRoadDirection may optionally be used to identify the road direction by listing (ASN.1 SEQUENCE) the first and the last charge object of the tolled road:

— **startSection,** of type `ChargeObjectId`, shall identify the first section of the tolled road;

— **endSection,** of type `ChargeObjectId`, shall identify the last section of the tolled road.

Table 31 — Section fields

Field name	Data type/data description	m/o
chargeObjectId	`ChargeObjectId`	m
chargeObjectName	`UTF8String`	o
chargeObjectRefPoint	`AbsolutePosition3d`	o
networkPoints	List (ASN.1 SEQUENCE OF) of records of type `Point`	o
tollPath	`Link`	o
liabilityRules	`LiabilityRules`	o
pathStructureTowards	List (ASN.1 SEQUENCE OF) of records of type `Link`	o
supportingInformation	List (ASN.1 SEQUENCE OF) of records of type `SupportingPoint`	o
pathStructureOnwards	List (ASN.1 SEQUENCE OF) of records of type `Link`	o
chargeDistance	`Distance`	m
realDistance	`Distance`	o
locationClass	`LocationClassId`	m
applicableTimeClasses	List (ASN.1 SEQUENCE OF) of records of type `TimeClassId`	o
tollRoadId	`TollRoadId`	o
previousChargeObjects	List (ASN.1 SEQUENCE OF) of records of type `ChargeObjectId`	o
nextChargeObjects	List (ASN.1 SEQUENCE OF) of records of type `ChargeObjectId`	o
typeOfSection	`TypeOfSection`	o

chargeObjectId shall contain a unique identifier of the section as a charge object within the toll context partition. In cases where the section is part of a road network that is specified within an area charging scheme, the charge object identifier shall be unique within the area (within the given areaId). The data type `ChargeObjectId` is defined in ISO TS 17573-3, and is made of the following fields:

— **chargeReportOperator** of type `Provider` may optionally be used to identify the entity operating the charge object;

— **chargeObjectDesignation** of type `Int4Unsigned` shall contain the designation number of the detected charge object.

chargeObjectName may optionally contain the name or designation of the section.

EXAMPLE 5 “A8 Augsburg West – Augsburg Ost” or “M9 Exits 4a – 5”.

chargeObjectRefPoint may optionally be used to set a reference point of type `Point` (see Table 26) valid for this particular charge object. The **chargeObjectRefPoint** shall be specified by selecting absolute point coordinates.

In cases where this data element is present, all longitude and latitude values given within this charge object description are considered being relative to this point.

To set a reference point valid for the entire context layout the field **geoRefPoint** of the data type **TollContextPartitionLayout** (see Table 27) shall be used.

networkPoints is a list which may optionally be used to indicate relevant geographic points, defined as data elements of type `Point`.

tollPath may optionally be used to specify the geographic description of the respective road section or a certain piece of a road, respectively. Each section shall be specified by a start and an end point. Intermediate points may optionally be used to better reflect the actual road shape, e.g. curves. The data type `Link` is described in Table 32 and is formally defined in Annex A.

Table 32 — Link data type

Field name	Data type/data description	m/o
linkId	`LinkId`	m
startPoint	`Point`	m
intermediatePoints	List (ASN.1 SEQUENCE OF) made of elements of type `Point`	o
endpoint	`Point`	m

linkId shall contain the identifier of the link. It shall be unique within the toll context.

startPoint shall specify the start point of the link. The data element shall be used as indicated for the field **chargeObjectRefPoint**.

intermediatePoints may optionally be used to list one or more than one intermediate points that are on the section of road and may be used to more accurately model the real shape of the section.

endpoint shall specify the end point of the link. The data element shall be used as indicated for the field **chargeObjectRefPoint**.

A link shall always be one-directional. The direction is implicitly given by start point and end point.

Figure 5 explains the meaning and usage of the elements of a `Link` data type.

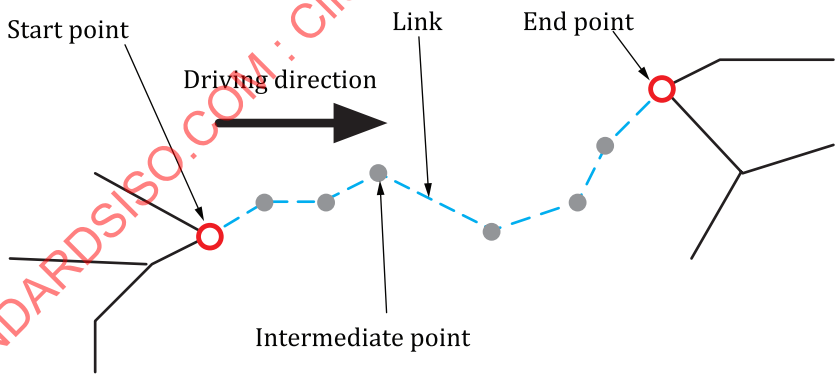


Figure 5 — Elements of a link

liabilityRules may optionally be used to specify under which circumstances a vehicle using the section is liable to pay for the use of the entire toll section (see examples in Table 33 with the related Figure 6, Figure 7, Figure 8 and Figure 9). The data type `LiabilityRules` is defined as a selection (ASN.1 CHOICE) of the following rules:

— passage through a specified number (one or two) of location points in the whole section implies payment of the toll for the entire section. The time of passage through the first point is used as the tolling event. In this case, the field **tollPoints1**, which is a list (ASN.1 SEQUENCE OF) of one or two toll point(s), shall be used. The data type `Point` shall be used to specify each toll point.

— passage through two location points on the whole section implies payment of the toll for the entire section. The time of passage through the second point is used as the tolling event. In this case the field **tollPoints2**, which is a list (ASN.1 SEQUENCE OF) of one or two toll point(s), shall be used. The data type `Point` shall be used to specify each toll point.

— passage of an absolute piece of the entire section implies payment of the toll for the entire section. The time of passage through an entire part of the section is used as the tolling event. In this case, the field **minTollPath** shall be used to specify a particular piece of the section which shall be passed through, using the data type `Link`.

— passage of a certain percentage of the section implies payment of the toll for the entire section. The time of passage through the given percentage is used as tolling event. The field **minimumUsage**, of type `INTEGER`, shall be used to specify the portion of the entire section (percentage value) that has to be passed as a minimum. The resolution shall be 0,1 %.

NOTE 1 Liability rules are typically specified in a legal framework belonging to each toll scheme. This field only gives the possibility to "convert" the legal rules into technical definitions.

The time of usage of the charge object or section, respectively, shall be reported in the corresponding data element (e.g. in the TollDeclarationAdu, using the **timeWhenUsed** field of the **chargeReport** element). This usage time may contain, among other things, the time when the liability rule applied, and the time when the entrance into the section or the exit from the section occcured. If liability rules are used, the time class specified in the data element **TariffClass** shall be derived from the usage time of the section and shall be applied to calculate the charge amount for the charge object.

The date and time which shall be reported regarding the usage of the charge object is up to bilateral agreements between the TC and the TSP.

Table 33 — Use of the attributes in data type LiabilityRules

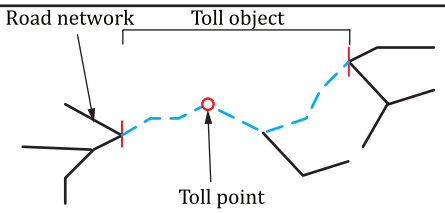
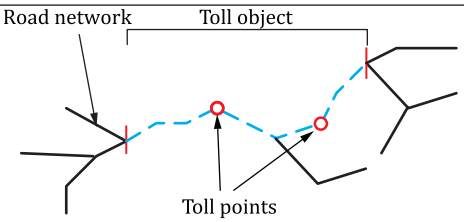
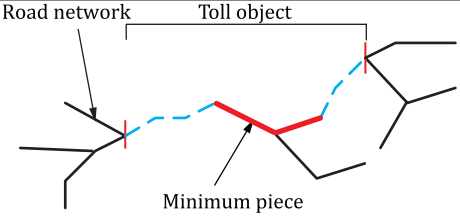
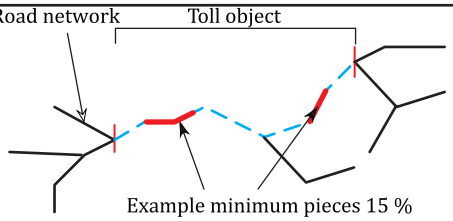
Rule	Description	Example	ASN.1 element in LiabilityRules
Toll Point(s)	Passage of a number of location points (one to two) makes the vehicle liable to pay for the use of the entire section at the time of passage of the first point. Figure 6 shows an example of one location point to be passed.	 **Figure 6 — One point for a section**	`tollPoints1`
Toll Points2	Passage of two location points makes the vehicle liable to pay for the use of the entire section at the time of passage of the second point. Figure 7 shows an example of two location points to be passed.	 **Figure 7 — Two points for a section**	`tollPoints2`
Absolute piece of section	Passage of a certain piece of the entire toll section makes the vehicle liable to pay for the use of the entire section at the time when the entire piece has been passed. Figure 8 shows an example of one absolute minimum piece to be passed.	 **Figure 8 — Absolute piece of section**	`minTollPath`

Table 33 *(continued)*

Rule	Description	Example	ASN.1 element in LiabilityRules
Percentage	Passage of a certain percentage of the entire toll section makes the vehicle liable to pay for the use of the entire section at the time when the entire specified percentage of the section has been passed. Figure 9 shows an example of two possible minimum relative pieces to be passed to reach an exemplary 15 % level.	 **Figure 9 — Percentage of a section**	`minimumUsage`

pathStructureTowards may optionally be used to list the layout description of the roads that are in front (relative to the driving direction) of the charged road section using the data type `Link`. The data type `Link` shall be used as specified in tollPath.

NOTE 2 This information is useful for supporting certain charge object detection algorithms in some implementations.

pathStructureOnwards may optionally be used to list the layout description of the roads that are ahead (relative to the driving direction) of the charged road section using the data type `Link`. The data type `Link` shall be used as specified in tollPath.

NOTE 3 This information is useful for supporting certain charge object detection algorithms in some implementations. Depending on their particular implementations, systems can use this additional information for performance improvements with respect to the detection of the toll liable use of a charge object. It gives more details about the road network that leads to the Link and the road network that connects to the Link. Thus, such additional data about the context of the Link can help to prevent the false detection of the charge object or the missing detection of a charge object.

In cases where **pathStructureTowards** and **pathStructureOnwards** are used to describe more complex road structures, the following rule shall apply:

— Links that physically connect to each other shall be identified by using the same point as end point of the first link and the start point of the second link. Depending on the referencing method chosen in the data type Point, this shall be achieved by one of the following methods:

  1) applying the same longitude, latitude and optional altitude values in data elements absolutePointCoordinates and relativePointCoordinates for both the end point of the first link and the start point of the connected link;

  2) applying the same point identifier in the data element pointIdentifier for both the end point of the first link and the start point of the connected link.

supportingInformation may optionally be used to list (intermediate) points on the charged road section. For each element in the list, a distance to other (non-tolled) road infrastructure may be given. The data type `SupportingPoint` is described in Table 34 and formally defined in Annex A.

Table 34 — SupportingPoint fields

Field name	Data type/data description	m/o
SupportPoint	`Point`	m
supportPointDescription	`UTF8String`	o
distanceToNextRoad	`Distance`	o

supportPoint shall be used to give information of the point that is located on the charged section. The data element shall be used as specified in **chargeObjectRefPoint**;

supportPointDescription may optionally be used to provide a description of the support point;

distanceToNextRoad may optionally be used to specify the minimum distance to the adjacent road infrastructure from the point specified in **supportPoint**.

— **chargeDistance** shall contain length information for the toll section. This information shall be used as the basis for calculation of the fee to be paid for the use of the section. The data type `Distance` is defined in ISO/TS 17573-3.

realDistance may optionally be used to provide the real measured length of the toll section. This length information may be different from the one given in data element **chargeDistance**.

locationClass shall specify the identification of the location class information for this toll section as type `LocationClassId`, as specified in **tariffClasses**.

applicableTimeClasses may optionally be used to list all time classes, identified by data type `TimeClassId`, that apply to this specific toll section, as specified in **tariffClasses**.

NOTE 4 This data element enables single road sections to be charged at certain times only.

previousChargeObjects may optionally be used to list (ASN.1 SEQUENCE OF) sections, identified by means of elements of data type `ChargeObjectId`, which precede this toll section in driving direction. If not specified, or empty, the section is at the beginning of the tolled road.

nextChargeObjects may optionally be used to list (ASN.1 SEQUENCE OF) sections, identified by means of elements of data type `ChargeObjectId`, which follow this toll section in driving direction. If not specified, or empty, the section is at the end of the tolled road.

typeOfSection is an INTEGER ranging from 0 to 255 defined as `TypeOfSection` which may optionally be used to specify the type of a tolled road section that is described in **invoicingRelatedData**. The following types of road section are specified in this document:

— **manual**, which shall be used if a section is tolled by a manual toll booth.

— **closedEntry**, which shall be used to describe the entry toll station of a closed network.

— **closedExit**, which shall be used to describe the exit toll station of a closed network.

— **checkpoint**, which shall be used to describe a checkpoint (intermediate point) in a closed network to distinguish between different paths between entry and exit toll station.

— **openRoadTolling**, which shall be used to describe an open road tolling section.

— **freeFlowPhysicalTollPoint**, which shall be used to describe a physical toll equipment on a free flow toll section.

— **freeFlowVirtualTollPoint**, which shall be used to describe a virtual toll equipment on a free flow toll section. There is no RSE installed. The usage of this toll section is specified by the previous and next physical toll equipment on neighbouring toll sections in a free flow environment.

NOTE 5 Such solutions are typically used if the installation of a physical toll equipment is not cost effective.

6.7.2.3.3 GdfLayout data type

The data type `GdfLayout` provides the possibility to describe a section-based layout by means of references to digital maps that use a GDF (geographic data files)-format type. The `GdfLayout` data type is a list (ASN.1 SEQUENCE OF) of fields of type `SectionGdf`, which described in Table 35 and is formally defined in Annex A.

NOTE 1 The GDF-based method is not always the best and most suitable choice for some technologies of GNSS front-end implementations. Alternative methods and formats can provide better performance and benefits from an operational perspective. In contrast, GDF-based methods are a good choice for the interface between the TC and the service provider.

Table 35 — SectionGdf fields

Field name	Data type/data description	m/o
efcLayerId	INTEGER ranging from 0 to 127	m
tollContextName	`UTF8String`	o
chargeObjects	Record (ASN.1 SEQUENCE) of elements of type `GdfChargeObject`	m
referenceGdfSource	`GdfSource`	m

efcLayerId shall contain an identifier of the respective EFC layer description. The efcLayerId shall be unique within the toll context partition.

tollContextName may optionally be used to textually describe the toll context.

chargeObjects shall contain the list of charge objects which are specified in this toll context partition. Each list entry shall be of type `GdfChargeObject`. The data structure of the data type `GdfChargeObject` is specified in Table 36.

referenceGdfSource of type `GdfSource` shall contain the reference to the particular source from which the GDF data describing the geographical objects in the toll context description have been taken. Usually this shall be a digital map of a specific vendor, brand, type and version. The data structure of the data type `GdfSource` is specified in Table 40.

Table 36 — GdfChargeObject fields

Field name	Data type/data description	m/o
chargeObjectId	`ChargeObjectId`	m
locationClass	`LocationClassId`	o
applicableTimeClasses	Record (ASN.1 SEQUENCE) of `TimeClassId`	o
tollRelevantLength	`Distance`	m
equivalentMeasuredLength	`Distance`	o
tollRoad	`TollRoad`	m
liabilityRules	Selection (ASN.1 CHOICE) of liability rules	m

chargeObjectId shall contain the identifier of the respective charge object. This identifier shall be unique within the toll context partition. The uniqueness shall be guaranteed by the respective TC which is identified in the structure of the data type `ChargeObjectId`.

locationClass shall contain the location class information for this charge object as specified in **tariffClasses**.

applicableTimeClasses may optionally be used to specify a list of those time classes that apply to this specific toll section. Each list entry shall be of data type `TimeClassId`.

tollRelevantLength shall contain length information for the toll section. This length information shall be used as the basis for calculation of the fee to be paid for the use of this road section. However, this length information can or can not represent the real length of the section.

equivalentMeasuredLength may optionally be used to provide the real measured length of the toll section. This length information may be different from the length information provided in data element tollRelevantLength.

tollRoad shall specify the geographical layout of the toll section by means of nodes and links. The data structure of the data type `TollRoad` is described in Table 37 and formally defined in Annex A.

liabilityRules may optionally contain rules (see Table 33) which specify under which circumstances a vehicle using this section is liable to pay the tolls for the use of the entire toll section.

Table 37 — TollRoad fields

Field name	Data type/data description	m/o
efcNodeFrom	`EfcNodeId`	m
efcLink	Record (ASN.1 SEQUENCE) of the following fields: **roadElementsTowardChargePoint** and **junctionId**	m
efcNnodeTo	`EfcNodeId`	m

A toll section, in terms of its geographical location and layout, can be specified as a set of two nodes, one start node and one end node, plus one or more links connecting the nodes to each other. Such description is also one of the basic principles applied in the lower layers of geographical digital maps.

efcNodeFrom shall specifiy the reference of a node description of a start node to a node element in a GDF-based digital map. The field **efcNodeFrom** shall be unique within the toll context. The data structure of the data type `EfcNodeId` is described in Table 38.

efcLink shall contain a record of two elements which connect the start node defined in **efcNodeFrom** to the end node. The data structure of **efcLink** is composed of:

— **roadElementsTowardsChargePoint** shall contain a list (ASN.1 SEQUENCE OF) of one or more elements of the type `RoadElementsTowardsChargePoint`, which is made of:

  — **junctionIdFrom** of type `GdfReference`, which shall contain the identifier of the junction that specifies the starting point segment of the entire path. The data type `GfdReference` is described in Table 39.

  — **gdfRoadElement** of type `GdfReference`, which shall contain the identifier of the road segment which connects to the junction (identified in element junctionIdFrom) and forms one segment of the entire path. The data type `GfdReference` is described in Table 39.

— **junctionId** of type `GdfReference`, which shall contain the identifier of the junction that specifies the end point of the chain of segments which form the entire path. The data type `GdfReference` is described in Table 39.

efcNodeTo shall be used to specify the reference of a node description of an end node to a node element in a GDF-based digital map. **efcNodeFrom** shall be a unique identifier within the toll context. The data structure of the data type `EfcNodeId` is described in Table 38.

Table 38 — EfcNodeId fields

Field name	Data type/data description	m/o
efcContextSpecificId	`ChargeObjectId`	m
sectionName	`UTF8String`	o
gdfSpecificId	`GdfReference`	m

efcContextSpecificId shall contain the identifier of the respective charge object. This identifier shall be unique within the toll context partition. The uniqueness shall be guaranteed by the respective TC which is identified in the structure of the data type `ChargeObjectId`.

sectionName may optionally be used to provide the name or designation of the section.

EXAMPLE “A8 Augsburg West – Augsburg Ost” or “M9 Exits 4a – 5”.

gdfSpecificId shall contain the detailed referencing structure to a node geographical element in a GDF-based digital map. The data structure is of type `GdfReference` and is described in Table 39.

Table 39 — GdfReference fields

Field name	Data type/data description	m/o
datasetId	`INTEGER` ranging from 0 to 2^{32}-1	m
layerId	`INTEGER` ranging from 0 to 2^{32}-1	m
sectionId	`INTEGER` ranging from 0 to 2^{32}-1	m
objectId	`INTEGER` ranging from 0 to 2^{32}-1	m

datasetId shall contain the identifier of the dataset within the digital map. For detailed semantic description and use of this data element refer to ISO 20524-1.

layerId shall contain the identifier of the specific data description layer in which the sectionId and objectId are specified. For detailed semantic description and use of this data element refer to ISO 20524-1.

sectionId shall contain the identifier of the section to which the node description makes a reference. For detailed semantic description and use of this data element refer to ISO 20524-1.

objectId shall contain the identifier of the particular node object in the digital map. For detailed semantic description and use of this data element refer to ISO 20524-1.

Table 40 — GdfSource fields

Field name	Data type/data description	m/o
dataProvider	`UTF8String`	m
albumId	`INTEGER` ranging from 0 to 2^{32}-1	m
versionNumber	`UTF8String`	m

dataProvider shall contain a textual designation (e.g. a commonly used brand name) of the vendor of the digital map used for referencing the geographic descriptions.

albumId shall contain an identifier of the digital map. This can be a product code, a serial number, an article number an EAN code or similar, for example.

versionNumber shall contain an identifier of the particular version of the digital map. The version number shall be unique within the context of the dataProvider and albumId.

NOTE 2 According to ISO 20524-1:2020, 10.2.2.6 or 11.2.4, this has a value of "5,0" or "5,1".

6.7.2.3.4 AreaPricingLayout data type

The data type `AreaPricingLayout` provides the full description of the context layout of an area charging scheme. It shall contain a list of the descriptions of one or more geographic areas forming one toll context and belonging to one toll scheme (applying one single tariff table, tariff class definitions, etc.). In cases where more than one area layout is present, the areas shall not overlap.

EXAMPLE An area charging scheme is split into two non-overlapping areas. One area covers the southern part of a metropolitan area, the second area covers the northern part, whereas the city centre is not covered.

The data type `AreaPricingLayout` is a list (ASN.1 SEQUENCE OF) of elements of type `AreaLayout` which is described in Table 41 and formally defined in Annex A.

Table 41 — AreaLayout fields

Field name	Data type/data description	m/o
areaId	`AreaId`	m
areaBorder	`Polygon`	m
locationClass	`LocationClassId`	o

Table 41 *(continued)*

Field name	Data type/data description	m/o
applicableTimeClasses	List of (ASN.1 SEQUENCE OF) elements of type `TimeClassId`	o
roadNetworks	List of records (ASN.1 SEQUENCE OF) made of fields of type `RoadNetwork`	o

areaId shall contain a unique identifier for the area within the toll context.

areaBorder shall contain the description of the border of the area. A `Polygon` is a list (ASN.1 SEQUENCE OF) of elements of type `Point`, which is formally defined in Annex A. Points shall be specified in such an order as to begin and end the polygon in a clockwise direction. The segments of the polygon shall be created by connecting each point with its successor point. The polygon shall be closed by connecting the last point with the first point in the list. The points shall be specified in such a way as to prevent the connections between the points from intersecting. The inner side of the polygon (tolled area) is specified as the geographical area that is located at the right-hand side when moving on a polygon segment from its start point to its end point.

locationClass may optionally be used to specify the location class for this area as specified in **locationClasses**.

applicableTimeClasses may optionally be used to list all time classes that apply to this area as specified in **tariffClasses**.

NOTE This data element enables area charging schemes to be active at certain time periods only (e.g. 08:00 – 12:00).

roadNetworks may optionally be used to list descriptions of road networks of type `RoadNetwork`. Each of these networks may have a different location class. This data element may be used for area charging schemes that charge according to the measured distance on a specified road network. Moreover, the area may contain more than one network supporting different tariffs. The structure of the data tpe `RoadNetwork` is described in Table 42 and formally defined in Annex A.

Table 42 — RoadNetwork fields

Field name	Data type/data description	m/o
networkId	`INTEGER` ranging from 0 to 127	m
locationClass	`LocationClassId`	m
roadNetworkObjects	List of (ASN.1 SEQUENCE OF) elements of type `RoadNetworkObject`	m

networkId shall contain a unique identifier for the network within the toll context.

locationClass may optionally be used to specify the location class for this network as specified in **locationClasses**.

roadNetworkObjects shall be used to specify all charge objects belonging to the road network within the specified area. The structure of the `RoadNetworkObject` data type is described in Table 43 and formally defined in Annex A.

Table 43 — RoadNetworkObject fields

Field name	Data type/data description	m/o
chargeObjectId	`ChargeObjectId`	m
chargeObjectName	`UTF8String`	o
chargeObjectRefPoint	`AbsolutePosition3d`	o
networkPoints	List (ASN.1 SEQUENCE OF CHOICE) elements of type `Point`	o
tollPath	`Link`	m
supportingInformation	List (ASN.1 SEQUENCE OF) of elements of type `SupportingPoint`	o

Table 43 *(continued)*

Field name	Data type/data description	m/o
applicableTimeClasses	List (ASN.1 SEQUENCE OF) of elements of type `TimeClassId`	o

chargeObjectId shall contain a unique identifier of the respective charge object within the area (within the given areaId).

chargeObjectName may optionally contain the name or designation of the charge object.

chargeObjectRefPoint may optionally be used to set a reference point valid for this particular charge object. In cases where this data element is present, all longitude and latitude values given within this charge object description are considered to be relative to this point.

networkPoints may optionally be used to list all geographic points used for the various location specifications as given in the description of the charge object. Both when **absolutePointCoordinates** and when **relativePointCoordinates** are selected for the **networkPoints** field, their usage and definition shall be as specified in ISO/TS 17573-3.

tollPath contain the geographic description of the respective charge object or a certain piece of a road, respectively.

supportingInformation may optionally be used to list (intermediate) points on the charge object defined as type `SupportingPoint` which is described in Table 34 and formally defined in Annex A.

applicableTimeClasses may optionally be used to list the identifiers of all the time classes that apply to this specific charge object.

6.7.2.3.5 CordonPricingLayout data type

The data type `CordonPricingLayout` provides the full description of the context layout of a cordon charging scheme as a list (ASN.1 SEQUENCE OF) of elements of type `CordonLayout`, each describing one geographic cordon among those that form one toll context and belong to one toll domain. In cases where more than one cordon layout is present, the cordons shall not overlap. The data structure of `CordonLayout` is described in Table 44 and formally defined in Annex A.

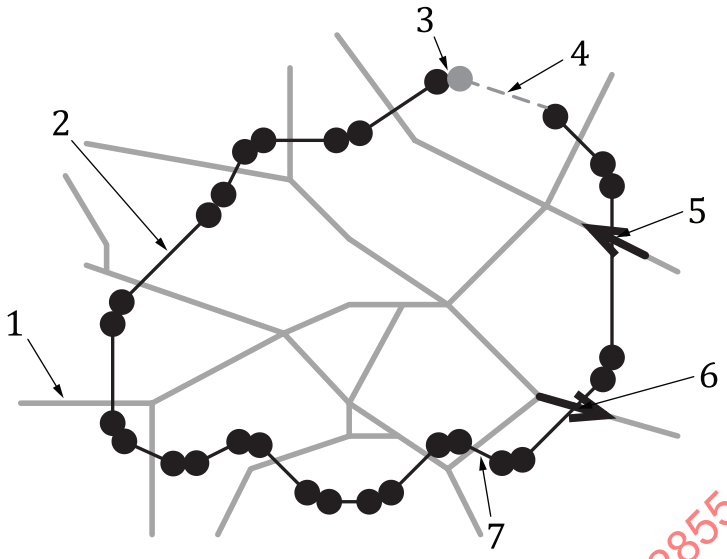
Table 44 — CordonLayout fields

Field name	Data type/data description	m/o
cordonId	`CordonId`	m
cordonBorderPolygon	List of (ASN.1 SEQUENCE OF) elements of type `CordonBorderSegment`	m

cordonId of type INTEGER ranging from 0 to 255 shall contain a unique identifier for the cordon within the toll context.

cordonBorderPolygon of type `CordonBorderSegment` shall list the full description of the segments of border of tolled cordon. The segments of the cordon border shall be specified in clockwise order. The segments of the cordon border polygon shall be specified by connecting the start point of a segment with the start point of the succeeding segment. The polygon shall be closed by connecting the end point of the last segment with the start point of the first segment. End and start points shall be specified in such a way as to prevent segments from intersecting (see Figure 10).

The inner side of the cordon border polygon is defined as the geographical area that is located at the right-hand side when moving on a polygon segment from its start point to its end point.



Key to figure

1 road network
2 cordon border
3 start point of a cordon border segment
4 cordon border segment
5 cordon border segment with entry point
6 cordon border segment with exit point
7 cordon border segment with entry and exit point

Figure 10 — Elements of a cordon layout

An entry and/or exit point is created where cordon segments and road networks intersect. A cordon segment shall be specified in such a manner that either one entry point or one exit point or one combined entry/exit point is created. Cordon segments shall not carry more than one entry point or one exit point or one combined entry/exit point.

The data structure of `CordonBorderSegment` type is described in Table 45 and is formally defined in Annex A.

Table 45 — CordonBorderSegment

Field name	Data type/data description	m/o
cordonSegmentId	`CordonSegmentId`	m
startPoint	`AbsolutePosition3D`	m
cordonEntryLocation	`CordonEntryLocation`	o
cordonExitLocation	`CordonEntryLocation`	o

cordonSegmentId shall contain a unique identifier for the cordon segment within the cordon layout.

startPoint shall contain the description of the start point of the border segment. It shall be specified by using an absolute position specified by the type `AbsolutePosition3D` which is defined in ISO/TS 17573-3.

cordonEntryLocation may optionally be used in cases where the respective cordon border segment constructs an entry location of the tolled cordon (intersection of the cordon border and the road network). The field **cordonEntryLocation** is of type `CordonEntryLocation` which is described in Table 46 and formally defined in Annex A.

Table 46 — CordonEntryLocation fields

Field name	Data type/data description	m/o
entryLocationId	`ChargeObjectId`	m
entryLocationName	`UTF8String`	o
entryLocationClass	`LocationClassId`	m
applicableTimeClasses	List of (ASN.1 SEQUENCE OF) elements of type `TimeClassId`	o

entryLocationId shall contain a unique identifier of the entry charge object within the toll context.

entryLocationName may optionally be used to specify a name for the entry location.

entryLocationClass shall contain the location class which is applicable to this particular entry point.

applicableTimeClasses may optionally list the identifiers of all time classes applicable to this particular entry point.

NOTE 1 This option supports the specification of cordon charging schemes with different time classes at different entry points.

cordonExitLocation may optionally be used in cases where the respective cordon border segment constructs an exit location of the tolled cordon (intersection of cordon border and road network). The type `CordonExitLocation` is described in Table 47 and is formally defined in Annex A.

Table 47 — CordonExitLocation fields

Field name	Data type/data description	m/o
exitLocationId	`ChargeObjectId`	m
exitLocationName	`UTF8String`	o
exitLocationClass	List of records (ASN.1 SEQUENCE OF SEQUENCE) of `LocationClassId` and entry locations as records of `ChargeObjectId`	m
applicableTimeClasses	List of (ASN.1 SEQUENCE OF) elements of type `TimeClassId`	o

exitLocationId shall contain a unique identifier of the exit charge object within the toll context.

exitLocationName may optionally be used to specify a name for the exit location.

exitLocationClass shall list all location classes which may also depend on the entry location, each as a record made of the following fields:

— **locationClass** shall identify the location class which is applicable to this particular exit point.

— **entryLocation** is a list (ASN.1 SEQUENCE OF) which may optionally be used to indicate all entry locations that have an identical location class as the exit location.

NOTE 2 This option supports the specification of individual tariffs that depend on the entry-exit-combination when using the cordon.

NOTE 3 Such tariffs can be applied to make the use of a tolled cordon when passing particular entry and/or exit locations more expensive compared to alternative entry and/or exit locations.

NOTE 4 Such tariffs can also be applied to make the use of a tolled cordon when passing particular entry and/or exit locations free of charge, e.g. for inhabitants of particular residential areas located inside or outside cordon area.

applicableTimeClasses may optionally be used to list the identifiers of all time classes applicable to this particular exit point.

6.7.3 MeshedContextData type

6.7.3.1 General

A general layout of a meshed complex context is depicted in Figure 11. In the figure, different TCs are indicated by hexagons in different shades of gray, while lines represent paths that can be traversed by vehicles when in the system.

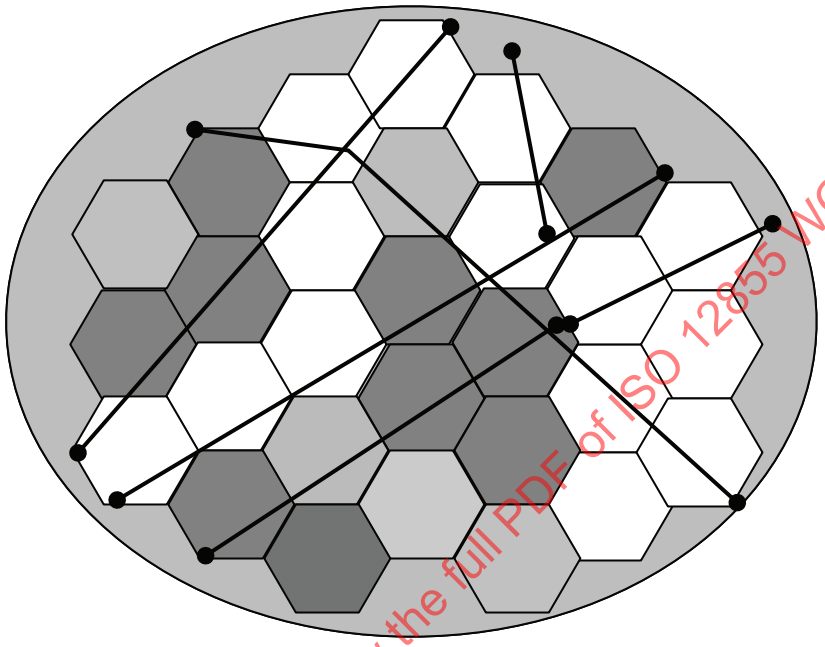


Figure 11 — Meshed context

Context information related to a complex meshed tolling system take into consideration the various possible configurations of such tolling systems. For example, to describe a DSRC closed tolling system, it is necessary to identify a boundary (points where vehicles enter/exit the system) in terms of entry and exit stations. This is the minimum required description of the closed layout. However, it is also necessary to specify additional information in the toll domain when multiple TCs are operating within it. In addition to that, as a tolling system is generally a meshed network, if the toll is determined not just on entry/exit stations (boundaries) but on the actual path in the network, additional detection points located inside the network can be needed to determine the actual path of a vehicle. Such detection points are named **intermediate points**. The most complex case is an interconnected scheme with multiple TCs (no intermediate barriers), with the possibility of varying tariffs by sections, TCs and types of roads, and, of course, types of vehicle. The data structures that follow specify this general case, from which each specific toll domain characteristics can be mapped by appropriate usage of optional fields.

The **meshedContext** field shall contain the context data and supporting information for a toll context that requires a more complex specification than that needed for the simplified context specified in 6.7.2. The **meshedContext** field is of type `MeshedContextData`, which is a record (ASN.1 SEQUENCE) of all optional fields, of which at least one shall be specified. The `MeshedContextData` structure is described in Table 48 and is formally defined in Annex A.

Table 48 — MeshedContextData type

Field name	Data type/ata description	m/o
highWaysList	List (ASN.1 SEQUENCE OF) of elements of type `HighWay` (see 6.7.3.2)	o
tollChargersList	List (ASN.1 SEQUENCE OF) of elements of type `Tollcharger`	o
intermediateCheckList	List (ASN.1 SEQUENCE OF) of elements of type `IntermediateCheck`	o

Table 48 *(continued)*

Field name	Data type/ata description	m/o
intermediateDoubleChecksList	List (ASN.1 SEQUENCE OF) of elements of type `IntermediateDoubleCheck`	o
tollDef	Record (ASN.1 SEQUENCE) of the following fields: **tollComponents-Def**, **tollTable**,	o

highWaysList is a record (ASN.1 SEQUENCE) of fields of type `HighWay`, that describe the higways that form the toll domain. The `Highway` data type is described in Table 51 and formally defined in Annex A.

tollChargersList is a list (ASN.1 SEQUENCE OF) of the TCs operating in the toll domain. Each TC is specified by the type `TollCharger` that contains the unique identifier of the TC (ASN.1 type `Provider`), and an optional description (ASN.1 `UTF8String`).

intermediateCheckList is a list (ASN.1 SEQUENCE OF) of the checkpoints distributed within the toll domain. Each intermediate checkpoint is a record (ASN.1 SEQUENCE) of the following fields:

— **ckId,** an identifier of type IntermediateCheckId which shall be used to univoquely identify the checkpoint in the tolling system**;**

— **ckDescription,** a textual field (ASN.1 UTF8String) which may be optionally used to describe the checkpoint;

— **ckCoordinates,** a field of type `AbsolutePosition3d` which may optionally be used to specify the position of the checkpoint;

— **tollsPerDirection,** a record (ASN.1 SEQUENCE) which shall be used to specify the tolls applied for each direction the checkpoint is traversed;

— **version,** which shall contain the version and validity of the information.

intermediateDoubleChecksList: it is a list (ASN.1 SEQUENCE OF) of the double checkpoints distributed within the toll domain. Each double intermediate checkpoint is a record (ASN.1 SEQUENCE) of the following fields:

— **dckOrderedCouple**, a record (ASN.1 SEQUENCE) which shall be used to specify the two intermediate checkpoints that constitute the double checkpoint;

— **dckDescription**, a textual field (ASN.1 `UTF8String`) which may be optionally used to describe the double checkpoint;

— **tollsPerDirection**, a record (ASN.1 SEQUENCE) which shall be used to specify the tolls applied for each direction the checkpoint is traversed;

— **version**, which shall contain the version and validity of the information.

The double checkpoint concept is introduced to cover the business rule depicted in Figure 12, which represents a tolling system where toll is incrementally calculated by checking vehicles passing at checkpoints. In Figure 12, **B** and **E** are two intermediate checkpoints, **d** and **f** are two possible destinations of a vehicle, **a** and **c** are two intersection points of the highway. If for the segment (**c**, **d**) the due toll is different from that due for segment (**c**, **f**), then a vehicle coming from **a** and exiting in **d** should pay more than if it exits in **f.** The association of **B** and **E** as a double checkpoint ensures that the correct toll is calculated.

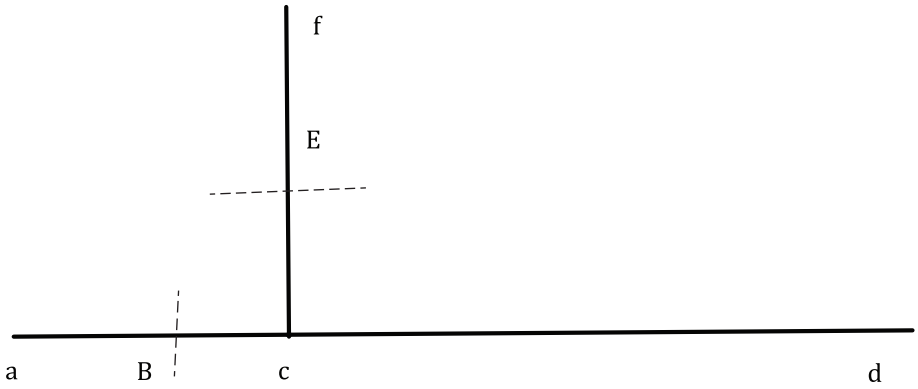


Figure 12 — Differentiation of tolls as a function of path

EXAMPLE Considering Figure 12, the following toll and checkpoints definitions help in understanding the double checkpoint concept:

Path	Related toll	Checkpoint description
path aBcd	toll=10	B described in intermediateCheck
path fEcd	toll=11	E described in intermediateCheck
path fEcBa	toll=5	(E,B) described in intermediateDoubleCheck
path aBcEf	toll=6	(B,E) described in intermediateDoubleCheck

tollDef is a record (ASN.1 SEQUENCE) of two optional fields, of which at least one shall be specified:

— **tollComponentsDefinition**, which lists the elements that determine the toll (see Table 49). It is of type `TollComponentsDefinition` and is a record (ASN.1 SEQUENCE) of a series of optional data fields, of which at least one shall be specified, and one mandatory data field (version). The fact that all significant components of `TollComponentsDefinition` are optional gives the possibility of updating one or more components and specifying the related version, while leaving all others unaltered, should the need arise.

— **tollTable**, of type `TollTable` which lists the identifiers of predefined paths in the road network, and the associated tariffs per vehicle classes.

Table 49 — TollComponentsDefinition data structure

Field name	Data type/data description	m/o
stationsList	Record (ASN.1 SEQUENCE) of the following fields: **stationId**, **stationDescription**, **absolutePointCoordinates**, **stationType**	o
intermediatePointList	List (ASN.1 SEQUENCE OF) of fields of type `IntermediatePoint`	o
predefinedPathsList	List (ASN.1 SEQUENCE OF) of fields of type `PredefinedPath`	o
paymentFeeUnit	`PayUnit`	o
vatRate	`INTEGER`, ranging from 0 to 10000	o
Version	`VersionAndValidity` (see Table 24)	m

stationsList may optionally be used to indicate the tolling stations used to compute the toll. Each element of type `Station` is a record (ASN.1 SEQUENCE) of the following fields:

— **stationId**, an `INTEGER` ranging from 0 to 2^{32}-1 defined as `StationId` data type which shall be used to univoquely identify the toll station in the toll domain;

— **stationDescription**, a textual field (ASN.1 `UTF8String`) which may be used to describe the tolling station;

— **absolutePointCoordinates**, a field of type `AbsolutePosition3d` which may be used to express the geographical coordinates of the tolling station;

— **stationType**, an `INTEGER` ranging from 0 to 255 which shall be used to indicate whether the station is:

  — **entryOnly**, only for entry into the toll domain;

  — **exitOnly**, only for exit from the toll domain;

  — **entryAndExit**, for entry and exit to/from the toll domain;

  — **virtualEntryStationUTurn**, virtual station used in the event that the entry station is the same as the exit station and a U-turn is authorized;

  — **virtualEntryStationNoUTurn**, virtual station used in the event that the entry station is the same as the exit station and a U-turn is not authorized;

  — **virtualEntryStationFarthest**, virtual station used in the event that the entry station has not been set (due for example to a failure of entry EFC transaction) and toll has to be computed considering the entry station as the farthest station in the closed system;

  — **virtualEntryStationNearest**, virtual station used in the event that the entry station has not been set (due for example to a failure of entry EFC transaction) and toll has to be computed considering the entry station as the nearest station in the closed system;

  — **barrier**, intermediate barrier where tolling is applied;

  — **interchange**, interchange station between different TCs;

  — **restArea**, rest area with mixed flows;

  — **parking**, parking area.

intermediatePointList may optionally be used to indicate the intermediate points used to compute the toll. The `IntermediatePoint` data type is a record (ASN.1 SEQUENCE) of the following fields:

— **intermediatePointId**, which is an `INTEGER` ranging from 0 to 2^{32}-1 which shall be used to identify the intermediate point in the closed system;

— **intermediatePointDescription**, which is a character string (ASN.1 `UTF8String`) which may optionally be used to associate a textual description to the intermediate point.

predefinedPathsList may optionally be used to indicate the paths used to compute the toll. The `PredefinedPath` data type is a record (ASN.1 SEQUENCE) of the following fields:

— **predefinedPathId**, which is of type `INTEGER` ranging from 0 to 2^{32}-1 defined as Int4, which shall be used to identify the predefined path in the toll domain network;

— **pathComposition**, which shall be used to describe the predefined path and is a record (ASN.1 SEQUENCE) of the following fields:

  — **entryStationId**, which shall identify of the entry station in a closed network. Station identifiers for stations of type **virtualEntryStationUTurn**, **virtualEntryStationNoUTurn**, and **virtualEntryStationFarthest** shall always be present in this sequence;

  — **operatedBy**, which is of type `Provider` and may optionally be used to identify the TC operating the entry station;

  — **exitStationId**, which shall be used to identify the exit station in a closed network. The TC that issues the ADU containing this field is the operator of the exit station.

  - **relativePathId**, which is an `INTEGER` ranging from 0 to 255 which shall be used to identify this path in the couple {entry station, exit station};

— **intermediateChecks**, which is a SEQUENCE OF fields of type IntermediatePointId which may optionally be used to identify the intermediate checkpoints of the path;

— **description**, which is a character string (ASN.1 UTF8String) which may optionally be used to associate a textual description to the predefined path.

paymentFeeUnit may optionally be used to indicate the currency to be used;

vatRate may optionally be used to indicate the VAT rate in units of 0,01 %;

version shall be used to indicate the version of the information being transferred. The type `VersionAndValidity` is described in Table 24 and formally defined in Annex A.

The `TollTable` data type is a multi-dimensional structure which may be used to associate predefined paths in the toll domain's road network with the tariffs to be applied for the recognized classes of vehicles. The `TollTable` data type is built as a record (ASN.1 SEQUENCE) made of of a list (ASN.1 SEQUENCE OF) of **paths** followed by the TollTable's associated validity. The **paths** fields are described in Table 50.

NOTE This type of tolling is currently in use in the Italian tolling system.

Table 50 — paths data structure

Field name	Data type/data description	m/o
pathId	`PredefinedPathId`	m
tollsPerTariffClass	`TollsPerTariffClass`	m

pathId shall be used to identify the predefined path in the set of paths defined in **tollComponentsDefinition**.

tollsPerTariffClass shall be used to associate a tariff to a vehicle class. The `TollsPerTariffClass` data type is a list (ASN.1 SEQUENCE OF) of records (ASN.1 SEQUENCE) with the following fields:

— **tariffClassId**, which shall identify the tariff class;

— **paymentFeeAmount**, which shall specify the amount of the toll as type `PaymentAmount`.

6.7.3.2 Description of a meshed toll domain layout

Referring to the diagram in Figure 11, a general multi-operator, meshed toll domain is represented by a graph, i.e. a set of vertices and edges. Each portion of the whole toll domain is thus a subgraph. The union of all subgraphs in a toll domain describes completely the toll domain. This subdivision is necessary for taking into account different tolling rules that can be applied to different portions of the graph, as well as the possible presence of multiple TCs in the same domain. For the scope of this document, a subgraph is named highway, which is defined by the `Highway` data type, which is described in Table 51 and is formally defined in Annex A.

Table 51 — Highway data structure

Field name	Data type/data description	m/o
hwId	`TollContextPartitionId`	m
hwCountryCode	`UTF8String`	o
hwNetworkName	`UTF8String`	o
hwRoadName	`UTF8String`	o
hwDescription	`UTF8String`	o

Table 51 *(continued)*

Field name	Data type/data description	m/o
hwLayout	`HighWayLayout` (see Table 52)	m
roadConventionalDirection	Record (ASN.1 SEQUENCE) of two vertex identifiers	m
hwType	`INTEGER`, ranging from 0 to 255	m
operatedBy	`Provider`	m
version	`VersionAndValidity` (see Table 24)	m

hwId is an `INTEGER` ranging from 0 to 255 defined as `TollContextPartitionId` which shall univoquely identify the highway within the toll domain.

NOTE This implies that in the case of a toll domain with multiple TCs, an agreement has to be reached among all interested parties to univoquely assign identifiers to the highways in the toll domain.

hwCountryCode may optionally be used to specify the country code of the highway according to ISO 3166-1, Alpha-2.

hwNetworkName may optionally be used to specify the name of the network of which the highway is part.

hwName may optionally be used to specify the commonly used name of the highway.

hwDescription may optionally be used to add a textual description to the highway.

hwLayout shall contain the formal description of the layout of the highway in terms of vertices and edges. See Table 52 for details.

roadConventionalDirection shall specify the direction associated with the highway in terms of starting vertex and ending vertex. This supports the association of two different highways to the same physical road, if the toll depends on the direction the road is traversed.

hwType shall be used to indicate the type of highway. It can assume one of the following three values:

1) **closedSystemStandAlone**, for a highway that is a stand-alone closed tolling system;

2) **closedSystemInterconnected**, for a highway that is part of a network of connected closed tolling systems;

3) **openSystem**, for a highway that is (part of) an open tolling system.

operatedBy may be optionally used to specify the TC that operates the highway.

version shall specify the version of the highway data structure.

Two possible highway layout description techniques are available:

— physical description, where each road fork or confluence point and each elementary segment is described by means of its “physical” characteristics (the type for each point and the length and the *location class* for each segment). Here, "elementary segment" means the portion of a highway that starts and ends with a fork or with a confluence;

— logical description, where relevant points are described by their “logical” characteristics, i.e. entry/exit station, intermediate control point, border point between two TCs, intersection points between two highways, and segments that connect the above.

The technique to be used to represent a specific highway can be either physical or logical, depending on the details needed in both the description of the toll domain and in the **usageList** field of the `BillingDetailsAdu` (see 6.12).

The `HighwayLayout` data type is the formal description of the graph representing the highway. The HyghwayLayout data structure is described in Table 52 and formally specified in Annex A.

Table 52 — HighwayLayout data structure

Field name	Data type/data description	m/o
parentHighWayId	`TollContextPartitionId`	m
layoutDefinition	Selection (ASN.1 CHOICE) of either type `PhysicalRoadDescription` or type `LogicalRoadDescription`	m
version	`VersionAndValidity` (see Table 24)	m

parentHighWayId shall be used to identify the toll context partition the highway belongs to.

layoutDefinition shall be used to describe the layout of the highway. It is a selection between physical and logical description. See Table 53 and Table 54 for details.

version shall specify the version of the highway layout data structure.

A physical description of a highway network is an oriented graph made by vertices and oriented edges, which are described by the `PhysicalVertex` and `PhysicalEdge` data types, as shown in Table 53 and formally defined in Annex A.

Table 53 — Physical road description data structure

Field name	Data type/data description	m/o
vertices	List (ASN.1 SEQUENCE OF) of `PhysicalVertex`	m
edges	List (ASN.1 SEQUENCE OF) of `PhysicalEdge`	m

A `PhysicalVertex` is a record (ASN.1 SEQUENCE) of the following fields:

— **phId**, an `INTEGER` ranging from 0 to 2^{32}-1 to which shall be used to univoquely identify the vertex in the toll domain;

— **description**, a character string (ASN.1 `UTF8String`) which may optionally be used to add a textual description to the vertex;

— **vertexType**, an `INTEGER` data type ranging from 1 to 255 which shall be used to specify the type of the vertex. It can take one of the following values (see also Figure 13 for a pictorial representation of the vertex types):

  — **beforeEntry**: entry station: start-point of the segment before the station;

  — **entry**: entry station: end-point of the segment before the station and start-point of the bifurcation segments;

  — **entryFork1:** entry station: end-point of a bifurcation segment to one road direction;

  — **entryFork2:** entry station: end-point of a bifurcation segment to the other road direction;

  — **afterExit:** exit station: end-point of the segment after the station;

  — **exit:** exit station: start-point of the segment after the station and end-point of the junction segments;

  — **exitJoint1:** exit station: start-point of a junction segment from one road direction;

  — **exitJoint2:** exit station: start-point of a junction segment from the other road direction;

  — **road1Fork1:** start-point of the bifurcation segments to one road, two directions;

  — **road1Fork2:** start-point of the bifurcation segments to the other road, two directions;

  — **road1Joint1:** end-point of a junction segment from one road, two directions;

  — **road1Joint2:** end-point of a junction segment from the other road, two directions;

— **road2Fork1**: start-point of the bifurcation segments to one road, one direction;

— **road2Fork2**: start-point of the bifurcation segments to the other road, one direction;

— **road2Joint1**: end-point of a junction segments from one road, one direction;

— **road2Joint2**: end-point of a junction segments from the other road, one direction;

— **changeLocationClassPoint**: point of location class change;

— **intermediateCheckPoint**: intermediate check point;

— **boundaryPoint**: boundary point between two TC domains or two roads of the same TC where some characteristics change (for example, highwayId, tariff, etc.).

The types of physical vertices listed above are depicted in Figure 13:

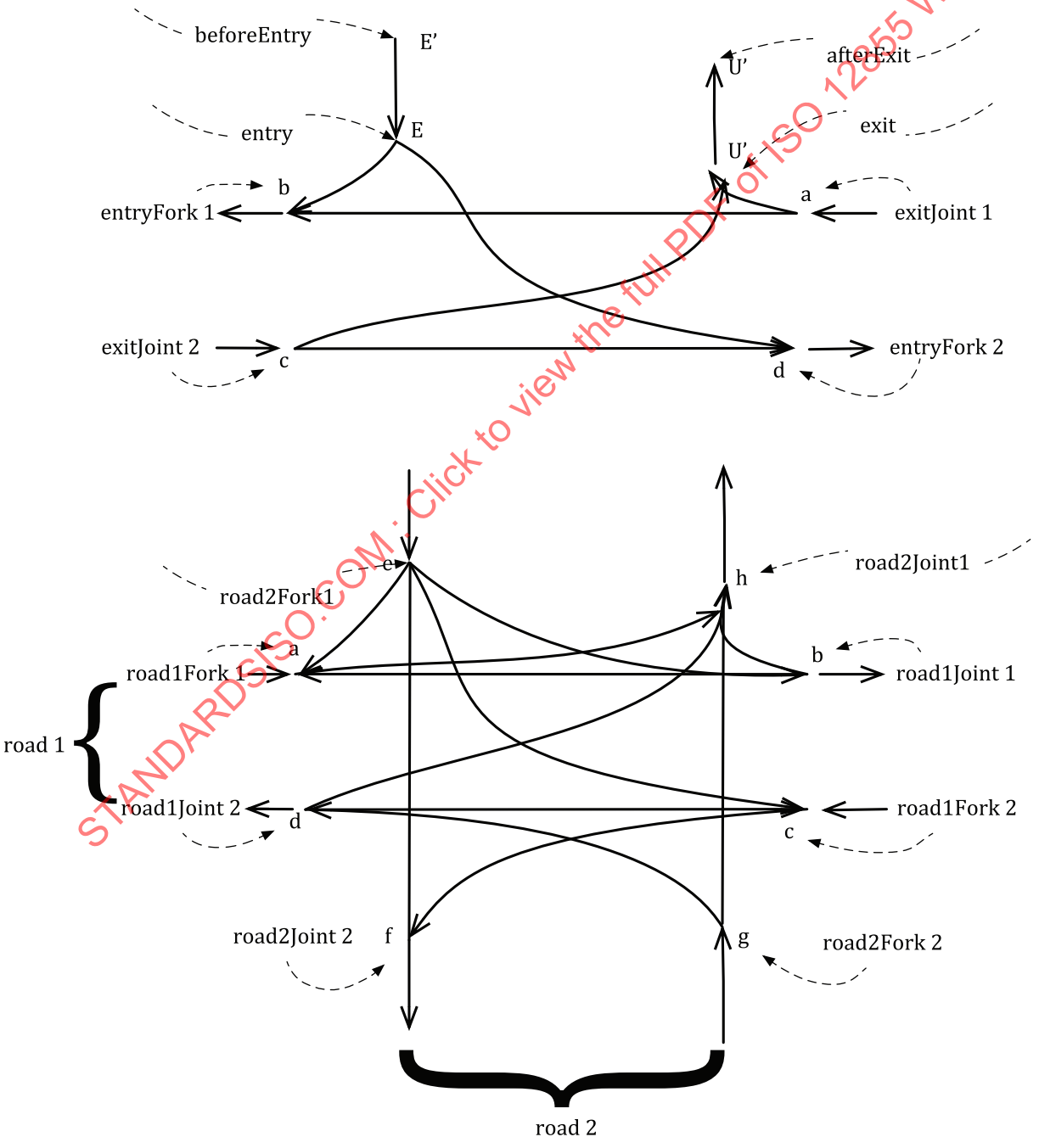


Figure 13 — Types of physical vertices

A `PhysicalEdge` is a record (ASN.1 SEQUENCE) of the following fields:

— **phId**, an identifier of type `PhysicalEdgeId` which shall be used to univoquely identify the edge in the toll domain. The `PhysicalEdgeId` data type is a record (ASN.1 SEQUENCE) of two ordered vertex identifiers, where the order shall be intended as the edge's direction (from first vertex to second vertex);

— **description,** a character string (ASN.1 `UTF8String`) which may optionally be used to add a textual description to the edge;

— **locationClass,** a field of type `LocationClassId` which may optionally be used to identify the location class associated with the edge;

— **chargeDistance,** a field of type `Distance` which shall bear the distance over which the toll is calculated;

— **realDistance**, a field of type `Distance` which may optionally be used to specify the real length of the physical edge.

When there is no need or obligation for the TC to specify the detailed physical layout of the highway, a logical description may be used. The logical description of a highway, as described in Table 54, is a not-oriented graph made of vertices and edges described by the types `LogicalVertex` and `LogicalEdge` which are formally defined in Annex A. As the graph is not oriented, an edge (*a*, *b*) can be travelled in both directions, i.e. from *a* towards *b* and from *b* towards *a*.

Table 54 — Logical Road Description data structure

Field name	Data type/data description	m/o
vertices	List (ASN.1 SEQUENCE OF) of `LogicalVertex`	m
edges	List (ASN.1 SEQUENCE OF) of `LogicalEdge`	m

A `LogicalVertex` is a record (ASN.1 SEQUENCE) of the following fields:

— **lVertexId**, an `INTEGER` ranging from 0 to 2^{32}-1 defined as `VertexId` data type which shall be used to univoquely identify the logical vertex in the toll domain;

— **lVertexDescription**, a character string (ASN.1 `UTF8String`) which may optionally be used to add a textual description to the vertex;

— **absolutePointCoordinates**, a field of type `AbsolutePosition3d` which may optionally be used to indicate the coordinates of the logical vertex;

— **lVertexType**, an INTEGER ranging from 0 to 255 which shall be used to specify the type of the logical vertex. It may take one of the following values:

  — **entryOnlyStation**: entry station (no exit);

  — **exitOnlyStation**: exit station (no entry);

  — **entryAndExitStation**: entry and exit station;

  — **intermediateCheckPoint**: intermediate checkpoint;

  — **boundaryPoint:** boundary point between two TCs domains or two roads of the same TC where some characteristics change (for example, highwayId, tariff, etc.);

  — **intersectionPoint4Directions**: intersection point between two highways with four possible directions;

  — **intersectionPoint3Directions**: intersection point between two highways with three possible directions;

  - **intersectionPoint2Directions**: intersection point between two highways with two possible directions;

- **boundaryAlsoOperatedBy**, a field of type `Provider` which may optionally be used to specify another TC if the boundary point is between two different TC domains;

- **additionalDistance**, a field of type `EdgeSegment` which may be optionally used to add a virtual distance to the vertex in order to cover for, for example, the length of the tolling station junctions or the length of the junctions between different running directions in intersections. The **EdgeSegment** is a record (ASN.1 SEQUENCE) made of two fields of type `LocationClassId` and `Distance`.

In order to describe the situation when a vertex can be of two types, a couple of vertices can be used, one of a type and the other one of the other type, connected by an edge of zero length (distance = 0).

An edge connects two vertices if a segment of highway connects the two vertices. A vertex has as many incoming and outgoing edges as the number of vertices that are connected to it by a highway segment.

A `LogicalEdge` is a record (ASN.1 SEQUENCE) of the following fields:

- **leId**, an identifier of type `LogicalEdgeId` which shall be used to univoquely identify the edge in the toll domain. The **LogicalEdgeId** is a record (ASN.1 SEQUENCE) of two unordered vertex identifiers, because the logical edge has no assigned direction;

- **leDescription**, a character string (ASN.1 `UTF8String`) which may optionally be used to add a textual description to the edge;

- **leCharacteristics**, a selection (ASN.1 CHOICE) between two types of edge characteristics which may optionally be added to the edge's specification to indicate how the toll is computed. The two possible choices are:

  - **forClosedSystems:** two series of couples, expressed as ASN.1 SET OF, one for the running direction from **StartOfEdge** to **EndOfEdge**, the other one for the opposite running direction to compute the toll, each couple being made of a `Distance` and a `LocationClassId`. The series of couples is necessary because the edge can be made of more highway segments, each with different value for the **locationClass**. Note that the order is not significant because the edge is completely traversed by the vehicle. If the edge can be traversed in one direction only (e.g. from **StartOfEdge** to **EndOfEdge**), the series of couples corresponding to the opposite direction (from **EndOfEdge** to **StartOfEdge**) shall be missing;

  - **forOpenSystems**: a field of type `Distance`.

6.7.4 Common data structures

6.7.4.1 The EntityOverview data structure

Each entry in the **entityOverview** field describes one toll context. At least the `EntityOverview` of the issuer of the `EfcContextDataAdu` shall be specified. If **relatedEntityIds** are specified, an **entityOverview** for each related entity shall be provided. Table 55 describes the fields of the `EntityOverview` data structure, which is formally defined in Annex A.

Table 55 — EntityOverview fields

Field name	Data type/data description	m/o
entityId	`Provider`	m
entityType	`EntityType`	m
relatedEntityId	List (ASN.1 SEQUENCE OF) of elements of type `RelatedEntityId`	o
entityClass	`UTF8String`	o
nameLine1	`UTF8String`	m

Table 55 *(continued)*

Field name	Data type/data description	m/o
nameLine2	`UTF8String`	o
addressLine1	ASN.1 `UTF8String`	m
addressLine2	`UTF8String`	o
poBox	`UTF8String`	o
zip	`UTF8String`	m
city	`UTF8String`	m
country	`UTF8String`	m
countryCode	`UTF8String` according to ISO 3166-1, Alpha-2 code elements	m
description	`UTF8String`	o
mainContact	`EntityContact` (see Table 56)	m
customerServiceContact	`EntityContact` (see Table 56)	o
itContact	`EntityContact` (see Table 56)	m
operationalContact	`EntityContact` (see Table 56)	m
commercialContact	`EntityContact` (see Table 56)	m
webSite	`UTF8String`	o
companyRegistrationNumber	`UTF8String`	o
companyRegistrationRegistrar	`UTF8String`	o
vatId	List (ASN.1 SEQUENCE OF) of elements of type `UTF8String`	o
established	`GeneralizedTime`	o
bankDetails	Record (ASN.1 SEQUENCE), containing the information to identify the bank for each bank account of the entity.	m
entityOverviewVersion	`VersionAndValidity` (see Table 24)	o

entityId shall identify the entity responsible for the EFC context data of a toll domain.

entityType shall indicate the type of entity. The following values are specified:

— **tc**, which specifies that the entity is a TC;

— **tsp**, which specifies that the entity is a TSP;

— **centralHUB**, which specifies that the entity is an international service entity that acts on behalf of one or more TCs;

— **localHUB**, which specifies that the entity is a national service entity that acts on behalf of one or more TCs;

— **groupOfTc**, which specifies that the entity is representing a group of TCs.

relatedEntityId may optionally contain a list of the identifiers of the related entities (used in case of service entities to indicate their relation). It is made of the following fields:

— **entityId** of type `Provider` shall uniquely identify the related entity. A separate **EntityOverview** shall be provided for each related entity;

— **relationDescription** of ASN.1 type `UTF8String` shall describe the relation to the entity responsible for the EFC context data.

entityClass may optionally describe the class of the entity as text for grouping of entities to be handled the same way.

nameLine1 shall contain the name of the entity.

nameLine2 may optionally contain an extension of the name of the entity, if one line is not sufficient.

addressLine1 shall contain the address of the entity.

addressLine2 may optionally contain an extension of the address of the entity, if one line is not sufficient.

poBox may optionally contain the post box address of the entity.

zip shall contain the zip code of the entity.

city shall contain the city of the entity.

country shall contain the country of the entity as a character string.

countryCode shall contain the country code of the entity according to ISO 3166-1, Alpha-2.

description may optionally contain further descriptive information about the entity.

mainContact shall contain information regarding the main contact for the entity.

customerServiceContact may optionally contain a name acting as a customer service contact for the entity.

itContact shall contain a name acting as an information technology contact for the entity.

operationalContact shall contain a name acting as an operational contact for the entity.

commercialContact shall contain a name acting as a commercial contact for the entity.

webSite may optionally contain the url of a website of the entity.

companyRegistrationNumber may optionally contain the registration number of the entity.

companyRegistrationRegistrar may optionally contain information on the registrar in which the entity is registered.

vatId may optionally list all VAT numbers/identifiers according to local country rules where the entity is VAT registered; at least the VAT number of the county in which the entity is situated should be stated.

established may optionally give the registration date of the entity.

bankDetails shall list the bank accounts of the entity. At least one bank account shall be specified. It is made of the following fields:

— **bankAccount** is of type `UTF8String` and optionally contains the national bank account number;

— **sortCode** is of type `UTF8String` and optionally contains the national sort code of the bank;

— **bic** is of type UTF8String and optionally contains the (bank identifier code) BIC of the bank;

— **iban** is of type `UTF8String` and optionally contains the international bank account number (IBAN) according to ISO 13616-1.

— **currencyCode** is of type `UTF8String` and shall contain the currency code according to ISO 4217.

The `EntityContact` data type supports providing contact information related to the entity. It is described in Table 56 and is formally defined in Annex A.

Table 56 — EntityContact fields

Field name	Data type/data description	m/o
contact	`UTF8String`	m

Table 56 *(continued)*

Field name	Data type/data description	m/o
telWorkingTime	`UTF8String`	m
telOutsideWorkingTime	`UTF8String`	m
email	`UTF8String`	m
mobile	`UTF8String`	o
telFax	`UTF8String`	o
workingTime	List of records (ASN.1 SEQUENCE OF) each made of the following fields: **wtStart**, **wtEnd**, **timeZone**	m

contact shall contain the name of a single person or a group (preferred) of the contact.

telWorkingTime shall contain the telephone number of the contact for the regular working time (see **workingTime** below).

telOutsideWorkingTime shall contain the telephone number of the contact for the timeframe outside of the regular working time (see **workingTime** below).

email shall contain the email address of the contact.

mobile may optionally be used to specify the mobile number of the contact.

telFax may optionally be used to specify the fax number of the contact.

workingTime is a list of the regular working times of the contact and shall contain at least one entry. It is made of the following fields:

— **wtStart** is of type `GeneralizedTime` and shall specifiy the starting point of time of the regular working time;

— **wtEnd** is of type `GeneralizedTime` and shall specifiy the ending point of time of the regular working time;

— **timeZone** is of type `INTEGER` ranging from -720 to 720 and shall specifiy the offset to the UTC time in minutes.

entityOverviewVersion may optionally be used to specify the version and validity of the EntityOverview structure.

6.7.4.2 TollContextOverview

The `TollContextOverview` data type shall contain information about toll context identification, number of present toll context partitions and additional supporting information for the toll context. The main purpose of the attribute `TollContextOverview` data type is to give a minimum amount of basic information regarding a toll scheme.

Table 57 indicates the fields in the `TollContextOverview` data type data structure, which is formally defined in Annex A.

Table 57 — TollContextOverview fields

Field name	Data type/data description	m/o
tollContext	`Provider`	m
tollContextPartitions	List of (ASN.1 SEQUENCE OF) `TollContextPartitionId`	m
tollContextName	`UTF8String`	o
tollContextOverviewVersion	`VersionAndValidity` (see Table 24)	m

tollContext shall identify the TC that operates the toll scheme for which the context description is valid.

tollContextPartitions shall contain a list of identifiers of the toll context partitions belonging to the single toll context. At least one toll context partition identifier shall be present in the list. The toll context partition identifiers shall be unique within a toll context.

tollContextName may optionally contain a designation for the toll context.

tollContextOverviewVersion shall contain version and validity information for the attribute tollContextOverview.

6.7.4.3 TariffTable

The `TariffTable` data type shall contain all the tariff-related information.

Table 58 indicates the fields in the `TariffTable` data type, which is formally defined in Annex A.

Table 58 — Tarifftable fields

Field name	Data type/data description	m/o
applicablePartitions	List of records (ASN.1 SEQUENCE OF), each of type `TollContextPartitionId`	m
tariffs	List of records (ASN.1 SEQUENCE OF) of type `Tariff`	m
defaultCurrency	`PayUnit`	m
typeOfTariff	`TypeOfTariff`	o
tariffTableVersion	`VersionAndValidity` (see Table 24)	m

applicablePartitions shall contain a list of the identifiers of the toll context partitions for which the tariff table is applicable. Each list entry shall be of type `TollContextPartitionId`.

tariffs shall contain a list of tariff related data for each tariff class. The fields of the data type `Tariff` are described in Table 59.

defaultCurrency shall contain the currency which shall be used. This currency shall be applicable for all data containing any sort of fee information in the fields of type `Tariffs` (**basicFeePerChargeUnit**, **offsetFee**, **minFee**, **thresholdFee** and **maxFee**). In cases where more than one currency is allowed or required in one toll domain (e.g. for particular user groups or because of legal requirements), the alternative currencies are specified in the respective tariff (data type `Tariff`).

typeOfTariff may optionally provide information about the nature of the fee (e.g. private fee, tax, duty or custom). The following values are specified:

— **fee**, which indicates that the type of the tariff is a fee;

— **tax**, which indicates that the type of the tariff is a tax;

— **custom**, which indicates that the type of the tariff is a custom.

tariffTableVersion shall contain version and validity information for the attribute **tariffTable**.

Table 59 indicates the fields in the `Tariff` data type, which is formally defined in Annex A.

Table 59 — Tariff fields

Field name	Data typed/data description	m/o
tariffClass	`TariffClassId`	m
chargeUnit	`ChargeUnit`	m
roundingRuleForChargeUnit	`RoundingRule`	m
basicFeePerChargeUnit	`INTEGER` ranging from 0 to 2^{32}-1	m
offsetFee	`INTEGER` ranging from 0 to 2^{32}-1	o
minFee	`INTEGER` ranging from 0 to 2^{32}-1	o

Table 59 *(continued)*

Field name	Data typed/data description	m/o
thresholdFee	`INTEGER` ranging from 0 to 2^{32}-1	o
maxFee	`MaxFee`	o
roundingRuleForFee	`RoundingRule`	m
vatRate	`INTEGER`	o
roundingRuleForVat	`RoundingRule`	o
intervalScaleParameters	List of records (ASN.1 SEQUENCE OF), each made of the following fields: **zeroOffset**, **resolution**, **max**	o
alternativeCurrency	`Currency`, according to ISO 4217	o
infrastructureFeePerChargeUnit	`INTEGER` ranging from 0 to 2^{32}-1	o
externalCostsPerChargeUnit	`ExternalCostsPerChargeUnit`	o
externalCostsPerChargeUnit	Record (ASN.1 SEQUENCE) made of the following fields: **airPollutionFeePerChargeUnit**, **noisePollutionFeePerChargeUnit**, **accidentFeePerChargeUnit**, **CO2FeePerChargeUnit**, **crossFinancingPerChargeUnit.**	o
availableDiscounts	Selection (ASN.1 CHOICE) between **volumeDiscounts** and **userDiscount**	o

tariffClass of type `TariffClassId` (common data type among several EFC standards) shall contain an unambiguous identifier of the tariff class within the toll context description.

chargeUnit of type `ChargeUnit` shall specify the charge unit valid for the toll context by using either (ASN.1 CHOICE) distance, time or event:

— **distance**, of data type `Distance`, shall contain the smallest interval of the distance that is of relevance for the fee calculation by specifying the value of the distance (**dist**) and the unit of the distance (**disUnit**) (e.g. kilometres, miles, metres, etc.);

— **time**, of data type `Duration`, shall contain the smallest interval of the time period that is of relevance for the fee calculation by specifying the value of the duration (**dur**) and the unit of the duration (**durUnit**) (e.g. seconds, minutes, hours, etc.);

— **event** of type `INTEGER` ranging from 0 to 255 shall contain the number of events that is of relevance for the fee calculation (e.g. 1 passage);

The use of charge units depends on the type of charging scheme. The following combinations of elements of `ChargeUnit` type and of `TollContextPartitionType` type are specified in this document:

— **distance** (e.g. kilometres, miles, metres), which shall only be used in section-based (e.g. roadSectionPricing) or are- based charging schemes (e.g. areaPricingDistance, areaPricingTime);

— **time** (e.g. seconds, minutes, hours), which shall only be used in area-based charging schemes (areaPricingDistance, areaPricingTime);

— **event** (e.g. number of passages), which shall only be used in cordon-based (cordonPricing) or section-based charging schemes (roadSectionPricing).

roundingRuleForChargeUnit shall be used to specify if and how rounding of distance driven or time in the toll liable network shall be applied when determining the charge unit used. The following values are specified:

— **no**, which specifies that no rounding is applied;

— **up**, which specifies that it shall always be rounded up to the next full larger value of the charge unit specified in `ChargeUnit`;

— **down**, which specifies that it shall always be rounded down to the next full lower value of the charge unit specified in `ChargeUnit`;

— **accounting**, which specifies that it shall always be rounded according to the methods used in accounting (e.g. as specified in DIN 1333[7] or other national specifications);

basicFeePerChargeUnit shall contain the basic fee per charge unit excluding VAT (see example of calculation in Annex E). The value shall be given in the currency and unit as defined in the attribute **defaultCurrency** within the tariff table. The given value shall be equal to the sum of **infrastructureFeePerChargeUnit** and any included **externalCostsPerChargeUnit,** if they are specified (see related item below in this list).

offsetFee may optionally contain an offset value that should be added to the basic fee (see example of calculation in Annex E). The value shall be given in the currency and currency units that are specified by the data element **defaultCurrency** in the tariff table.

minFee may optionally contain a minimum value for the fee valid for the particular tariff class (see example of calculation in Annex E). This minimum fee may be higher than the fee that would be applied according to the real usage of the charged network. The value shall be given in the currency and currency units that are specified by the data element defaultCurrency in the tariff table.

EXAMPLE 1 A minimum fee of EUR 1,00 is specified in a section-based charging scheme. However, there are very short sections for which, based on the applicable toll section length and the tariff, a fee of less than EUR 1,00 would apply (e.g. 1 km × EUR 0,10/km = EUR 0,10). The minimum fee is used in this case.

thresholdFee of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify a threshold fee (see example of calculation in Annex E). In cases where the fee resulting from the real usage of the charged network is lower than the threshold value, no fee shall apply. The value shall be given in the currency and currency units that are specified by the data element defaultCurrency in the data element tariffTable

EXAMPLE 2 A vehicle uses a time-based area charging scheme for 10 minutes. The fee valid for the applicable tariff class is specified by EUR 0,02 per minute. The threshold fee is specified as EUR 0,30. As the calculated fee does not reach the threshold value, the resulting fee is EUR 0,00.

maxFee may optionally contain a maximum value for the fee valid for the particular tariff class. This maximum fee may be lower than the fee that is computed according to the real usage of the charged network. The value shall be given in the currency and currency units that are specified by the data element **defaultCurrency** in the data element **tariffTable**.

NOTE 1 The rules of how a given user is identified (e.g. by pan, obeId, or other elements or combinations thereof of `UserId` data type) in order to determine whether the fee to be paid is greater than the value of **maxFee** are outside of the scope of this document.

maxFee is a record (ASN.1 SEQUENCE) made of the following fields:

— **perDay** of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the maximum value for the fee per day;

— **perWeek** of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the maximum value for the fee per week;

— **perMonth** of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the maximum value for the fee per month;

— **perYear** of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the maximum value for the fee per year.

The values of the above fields shall be specified in the currency and currency units according to the value of the data element **defaultCurrency** in the tariff table.

NOTE 2 The parameters can be used in cordon charging schemes to cap the maximum fee due per day in cases where there are multiple passages across the cordon border, for example.

EXAMPLE 3 A maxFee value of "1 165 0" and currency value of "EUR" specifies EUR 116,50 maximum fee, if **defaultCurrency** is defined as "2978".

roundingRuleForFee shall be used to specify if and how rounding of the resulting toll fee shall be applied when calculating the fee per used charge object. The following values are defined:

— **no**, which specifies that no rounding is applied;

— **up**, which specifies that it shall always be rounded up to the next full larger value of the charge unit specified in `ChargeUnit`;

— **down**, which specifies that it shall always be rounded down to the next full lower value of the charge unit specified in `ChargeUnit`;

— **accounting**, which specifies that it shall always be rounded according to the methods used in accounting (e.g. as specified in DIN 1333 or other national specifications);

vatRate of type `INTEGER` ranging from 0 to 10000 may optionally contain the VAT per charge unit. The value shall be given in 0,01 %.

roundingRuleForVat of type `INTEGER` ranging from 0 to 255 may optionally be used to specify if and how rounding of the resulting VAT for the toll fee shall be applied when calculating the VAT per used charge object. The following values are specified:

— **no**, which specifies that no rounding is applied;

— **up**, which specifies that it shall always be rounded up to the next full larger value of the charge unit specified in `ChargeUnit`;

— **down**, which specifies that it shall always be rounded down to the next full lower value of the charge unit specified in `ChargeUnit`;

— **accounting**, which specifies that it shall be always rounded according to the methods used in accounting (e.g. as specified in DIN 1333);

If the attribute **vatRate** is present but no information regarding the rounding rules for VAT (**roundingRuleForVat**) was provided, no rounding shall be applied for VAT.

intervalScaleParameters may optionally provide the possibility to scale the basic fee which is specified per charge unit (**basicFeePerChargeUnit**) depending on the specified vehicle weight range and the steps between the lower and upper limits of the vehicle weight range. If the field is not specified, its default value is 1 (see example of calculation in Annex E). The field **intervalScaleParameters** is a list (ASN.1 SEQUENCE OF) of records of type `IntervalScaleParameter`, which is a record (ASN.1 SEQUENCE) of the following fields:

— **vehicleMaxLadenWeightIntervals**, which is a record (ASN.1 SEQUENCE) made of:

  — **zeroOffset** of type `VehicleMaxLadenWeight` shall specify the lower limit of the vehicle weight interval by defining the offset from the zero point. VehicleMaxLadenWeight shall be used as specified in ISO/TS 17573-3 in VehicleWeightLimits;

  — **resolution** of type `VehicleMaxLadenWeight` shall specify the steps in 10 kg units that enable movement between the lower and upper vehicle weight limits. `VehicleMaxLadenWeight` shall be used as specified in ISO/TS 17573-3 in `VehicleWeightLimits`;

  — **max** of type `VehicleMaxLadenWeight` shall specify the upper limit of the vehicle weight interval. `VehicleMaxLadenWeight`. It shall be used as specified in ISO/TS 17573-3 in VehicleWeightLimits.

— **factorAtZeroOffset** of type `INTEGER` shall specify the factor for scaling the basic fee for vehicles at the lower end of the specified vehicle weight range (the value of **zeroOffset**). The value is expressed in hundredths, meaning that the specified value shall be divided by 100 to calculate the factor to be used.

NOTE 3 A **factorAtZeroOffset** of value zero (0) means that the basic fee for vehicles with a laden weight below or equal to the value of **zeroOffset** is zero (0).

— **factorAtMax** of type `INTEGER` shall specify the factor for scaling the basic fee for vehicles at the upper end of the specified vehicle weight range (the value of **max**). The value is expressed in hundredths, meaning that the specified value shall be divided by 100 to calculate the factor to be used.

EXAMPLE 4 A **factorAtMax** of value 200 means that the basic fee for vehicles with a laden weight equal to or above the value of **max** will be doubled.

alternativeCurrency of type `INTEGER` ranging from 1 to 999, specified according to ISO 4217, may optionally be used to specify an alternative currency in cases where the toll context uses more than one currency (e.g. for particular user groups or because of legal reasons). If this data element is present for this particular tariff class, the alternative currency shall be valid instead of the default currency that is specified in the tariff table. However, the data provided in the elements **basicFeePerChargeUnit**, **offsetFee**, **thresholdFee**, **minFee** and **maxFee** shall still be given in the currency and units specified in the tariff table. In these cases, a conversion rate shall be applied between the two currencies. The conversion rates for alternative currencies are provided by the data type `CurrencyConversionTable` (see Table 62).

infrastructureFeePerChargeUnit of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the part of **basicFeePerChargeUnit** which is related to infrastructure costs. The value shall be given in the currency and unit as defined in the attribute **defaultCurrency** within **tariffTable**. The result may be multiplied by a valid conversion rate if an **alternativeCurrency** is specified.

externalCostsPerChargeUnit may optionally be used to specify the part of **basicFeePerChargeUnit** which is related to external costs. It is made of the following optional fields, of which at least one shall be specified if the **externalCostsPerChargeUnit** is used:

— **airPollutionFeePerChargeUnit** of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the air pollution (e.g. NOX emission) fee per charge unit excluding VAT. External cost information regarding CO_2 emissions shall be specified in **CO2FeePerChargeUnit**. The value shall be given in the currency and unit as defined in the attribute **defaultCurrency** within the tariff table;

— **noisePollutionFeePerChargeUnit** of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the noise pollution fee per charge unit excluding VAT. The value shall be given in the currency and unit as defined in the attribute **defaultCurrency** within the tariff table;

— **accidentFeePerChargeUnit** of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the accident fee per charge unit exluding VAT. This attribute may be used to specify the economic impact of potential accidents that can occur on the described charge unit. The value shall be given in the currency and unit as defined in the attribute **defaultCurrency** within the tariff table;

— **CO2FeePerChargeUnit** of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the CO_2 fee per charge unit excluding VAT. The value shall be given in the currency and unit as defined in the attribute **defaultCurrency** within the tariff table;

— **crossFinancingPerChargeUnit** of type `INTEGER` ranging from 0 to 2^{32}-1 may optionally be used to specify the cross-financing costs fee per charge unit excluding VAT. The value shall be given in the currency and unit as defined in the attribute **defaultCurrency** within the tariff table.

If used, the sum of the fees provided in the attributes **externalCostsPerChargeUnit** and **infrastructureFeePerChargeUnit** shall correspond to the fee specified in **basicFeePerChargeUnit**.

availableDiscounts may optionally be used to specify the available discounts in the toll domain. It is a selection between:

— **volumeDiscounts**, which is a list (ASN.1 SEQUENCE OF) of elements of type `VolumeDiscount` (see Table 60);

— **userDiscounts**, which is a list (ASN.1 SEQUENCE OF) of elements of type `UserDiscount` (see Table 61);

Table 60 — VolumeDiscount fields

Field name	Data type/data description	m/o
discountId	`DiscountId`	m
discountDescription	`UTF8String`	o
activationFee	`PaymentAmount`	o
timeInterval	`TimeInterval`	o
relevantChargeObjects	List (ASN.1 SEQUENCE OF) of elements of type `ChargeObjectId`	o
amountOfFee	List (ASN.1 SEQUENCE OF) of records (ASN.1 SEQUENCE) made of the following fields: **minAmountOfFee**, **discountLevel**.	o
numberOfTrips	List (ASN.1 SEQUENCE OF) of records (ASN.1 SEQUENCE) made of the following fields: **minNumberOfTrips**, **discountLevel**.	o

discountId is an integer (ASN.1 Int2Unsigned) which shall be used to identify the discount.

discountDescription is a text field which may optionally be used to describe the discount.

activationFee, of type `PaymentAmount`, may optionally be used to specify the fee required to activate the discount.

timeInterval, of type `TimeInterval`, may optionally be used to indicate the period of time when the discount applies.

relevantChargeObjects is a list (ASN.1 SEQUENCE OF) of charge object identifiers which may optionally be used to indicate the charge objects which shall be passed by for a billing detail to be considered as eligible for the discount.

amountOfFee may optionally be used to indicate the minimal fee over which the discount is applicable and the amount of the discount. It is made of a record (ASN.1 SEQUENCE) of the following fields:

— **minAmountOfFee,** of type `PaymentAmount`, which may optionally be used to specify the minimal amount to be paid in order to activate the discount;

— **discountLevel**, of type `DiscountLevel`, which shall be used to indicate the amount of discount. It is a record (ASN.1 SEQUENCE) made of the following fields:

  — **discountPercentage**, of type `INTEGER`, which shall specify the percentage of the tariff to be discounted, expressed as a number ranging from 0 to 1 000;

  — **discountAggregation**, which is a `BOOLEAN` which may be used to indicate whether the discount can be aggregated with other discounts.

numberOfTrips may optionally be used to indicate the minimal number of trips over which the discount is applicable and the amount of the discount. It is made of a record (ASN.1 SEQUENCE) of the following fields:

— **minNumberOfTrips,** of type `INTEGER`, which shall be used to specify the minimal number of trips to be done in order to activate the discount;

— **discountLevel**, of type `DiscountLevel`, which shall be used to indicate the amount of discount.

Table 61 — UserDiscount fields

Field name	Data type/data description	m/o
discountId	`DiscountId`	m
discountDescription	`UTF8String`	o
userCategory	`UserCategory`	m
discountLevel	`DiscountLevel`	m

discountId is an integer (ASN.1 Int2Unsigned) which shall be used to identify the discount.

discountDescription is a text field which may optionally be used to describe the discount.

userCategory, of type `UserCategory`, shall be used to specify the category of users for which the discount applies. The following categories are defined:

— **genericDiscounted,**

— **serviceContractor,**

— **enforcementAgent,**

— **sanitaryOperator,**

— **disabledUser,**

— **officialAuthority,**

— **fireBrigade.**

discountLevel, of type `DiscountLevel`, shall be used to indicate the amount of discount.

6.7.4.4 CurrencyConversionTable

The data type `CurrencyConversionTable` shall contain a list of conversion rates that are applicable in cases where the toll context applies more than one currency for the calculation of the fees due. The default currency is specified in the tariff table.

NOTE Additional currencies can apply for particular user groups or for legal reasons.

Alternative currencies shall be specified individually per tariff class for the instances to which they apply. In cases where an alternative currency is specified in data element **tariff**, an attribute of type `CurrencyConversionTable` shall be present.

Table 62 indicates the fields in the `CurrencyConversionTable` data type, which is formally defined in Annex A.

Table 62 — CurrencyConversionTable fields

Field name	Data type/data description	m/o
conversions	List of records (ASN.1 SEQUENCE OF SEQUENCE) made of the following fields: **alternativeCurrency**, **conversionRate**	m
currencyConversionTableVersion	`VersionAndValidity` (see Table 24)	m

conversions shall contain a list of the applicable alternative currencies (of type `AlternativeCurrency`) and the corresponding conversion rate (of type `ConversionRate`) to the default currency (of type `DefaultCurrency`) specified in the tariff table. The attribute conversions is made of the following fields:

— **alternativeCurrency** shall contain the currency code of the individual currency, according to ISO 4217.

— **conversionRate** of type INTEGER ranging from 0 to 2^{32}-1 shall provide the conversion rate between the respective alternative currency and the default currency in the tariff table. The value shall be given in 0,000 1 units of the alternative currency compared to one major unit of the default currency.

The following formula shall apply:

$$V_{ac} = V_{sc} \times r_c$$

where:

V_{ac} is the value in alternative currency;

V_{sc} is the value in default currency;

r_c is the conversion rate.

EXAMPLE 1 The default currency is specified as EUR. The alternative currency is specified as NOK. The exchange rate assumed is NOK 7,677 1 = EUR 1. In this example, the value in the data element conversionRate is 76771.

EXAMPLE 2 The default currency is specified as EUR. The alternative currency is specified as HUF. A basic fee per charge unit for the particular tariff class is specified as EUR 0,11. The value in the element conversion rate is set to 2 948 500. The calculated basic fee per charge unit in the alternative currency is HUF 32,433 5 (EUR 0,11 × 2 948 500 × 0,000 1).

currencyConversionTableVersion of type `VersionandValidity` shall contain version and validity information for the attribute currencyConversionTable.

6.7.4.5 TariffClassDefinition

The data type `TariffClassDefinition` shall contain all valid combinations of the tariff determinants. It may optionally also contain information on the validity period of the tariff determinants.

Table 63 indicates the fields in the `TariffClassDefinition` data type, which is formally defined in Annex A.

Table 63 — TariffClassDefinition fields

Field name	Data type/data description	m/o
tariffClasses	List (ASN.1 SEQUENCE OF) of elements of type `TariffClass`	m
tariffClassDefinitionVersion	`VersionAndValidity`	m

tariffClasses shall contain a list of all valid combinations of the tariff determinants local vehicle class, time class, location class and user class for a tariff class. Tariff classes and the specification of the tariff determinants are applicable for the entire toll context including all toll context partitions. The data fields of the type `TariffClass` are specified in Table 64.

tariffClassDefinitionVersion shall contain version and validity information.

Changes in tariffs and/or class specifications that apply in the operational lifetime of toll contexts can, for example, be applied the following manner (the following is a non-exhaustive list):

— applying the same tariff after a change in tariffs or class specifications, but using the version and validity data elements in order to identify and specify respective changes in tariffs and classes;

— applying a new tariff that was not used before the change in tariffs or class specifications and specifying the tariff classes in a way that they include only those vehicle/location/time/user classes effective after the change in tariffs or class specifications.

Table 64 — TariffClass fields

Field name	Data type/data description	m/o
tariffClassId	`TariffClassId`	m
tariffClassDescription	UTF8String	o
localVehicleClasses	List (ASN.1 SEQUENCE) of `LocalVehicleClassId`	m
timeClasses	List (ASN.1 SEQUENCE) of `TimeClassId`	o
locationClasses	List (ASN.1 SEQUENCE) of `LocationClassId`	o
userClasses	List (ASN.1 SEQUENCE) of `UserClassId`	o

tariffClassId shall contain an unambiguous identifier of the tariff class within the toll context description.

tariffClassDescription may optionally be used to provide a textual description of the tariff class.

localVehicleClasses is a list (ASN.1 type SEQUENCE OF) of type `LocalVehicleClassId` that references the vehicle's local parameters for tolling.

timeClasses is a list (ASN.1 type SEQUENCE OF) of type `TimeClassId` that references the time parameters for tolling.

locationClasses is a list (ASN.1 type SEQUENCE OF) of type `LocationClassId` that references the location parameters for tolling.

userClasses is a list (ASN.1 type SEQUENCE OF) of type `UserClassId` that references user parameters for tolling.

In cases where one of the optional elements timeClasses, locationClasses or userClasses is not present, the respective tariff class shall be valid for all possible values of this element.

EXAMPLE 1 Tariff Class 56 is defined as the combination of Local Vehicle Class 23 and User Class 2.

EXAMPLE 2 Tariff Class 106 is defined as the combination of Local Vehicle Classes 23, 78 and 98 Time Classes 32 and 206.

EXAMPLE 3 Tariff Class 87 is defined as User Class 1.

EXAMPLE 4 Tariff Class 202 is defined as the combination of Local Vehicle Classes 156, 223 and 224 and Time Class 45 and Location Classes 2 and 5 and User Class 2.

6.7.4.6 LocalVehicleClassDefinition

The data type `LocalVehicleClassDefinition` shall contain all necessary data for specifying vehicle classes that locally apply to a toll context.

Table 65 indicates the fields in the `LocalVehicleClassDefinition` data type, which is formally defined in Annex A.

Table 65 — LocalVehicleClassDefinition fields

Field name	Data type/data description	m/o
localVehicleClasses	List of (ASN.1 SEQUENCE OF) of elements of type `LocalVehicleClass`	m
localVehicleClassVersion	`VersionAndValidity` (see Table 24)	o

localVehicleClasses shall contain a list of fields of type `LocalVehicleClass`. Each field shall contain the specification of one vehicle class. The data fields of `LocalVehicleClass` are specified in Table 66.

localVehicleClassVersion of type `VersionandValidity` may optionally contain version and validity information.

Table 66 describes the `LocalVehicleClass` type, which is formally defined in Annex A.

Table 66 — LocalVehicleClass fields

Field name	Data type/data description	m/o
localVehicleClassId	`LocalVehicleClassId`	m
localVehicleClassDescription	`UTF8String`	o
nominalVehicleParameters	`NominalVehicleParameters`	m
ordinalVehicleParameters	`OrdinalVehicleParameters`	o
priorityValue	`INTEGER`	o

localVehicleClassId shall contain a unique identifier for each local vehicle class.

EXAMPLE 1 Vehicle class 21 is defined as all trucks between 3,5 and 12 tons and having 3 axles.

EXAMPLE 2 Vehicle class 107 is defined as passenger cars having 3 or 4 axles and euro classes better than 5.

EXAMPLE 3 To exempt vehicles with very low emissions from charges (e.g. in a city charging scheme) these vehicles can be grouped into one vehicle class (e.g. 67), which is defined by all vehicles with euro class 6.

In the context of tariffs, vehicle specific parameters may have a different nature. Vehicle specific parameters may be used as nominal scale parameters (**nominalVehicleParameters**) or ordinal scale parameters (**ordinalScaleParameters**). These attributes shall contain the description of vehicle parameters for each local vehicle class.

NOTE Depending on the toll context, some of the vehicle parameters can be used in a different manner, e.g. as ordinal scale parameter in one toll context and as nominal scale parameter in a second toll context.

localVehicleClassDescription may optionally contain an additional description of the local vehicle class

nominalVehicleParameters of type `NominalVehicleParameters` shall contain the applicable nominal vehicle parameters for each local vehicle class. The structure of the `NominalVehicleParameters` data type is shown in Table 67 and is formally defined in Annex A.

ordinalVehicleParameters may optionally be used to list applicable ordinal vehicle parameters (value ranges) for each local vehicle class. The structure of the type `OrdinalVehicleParameters` is shown in Table 68 and is formally defined in Annex A.

priorityValue may optionally be used to specify priority information to determine which vehicle class is valid in cases where more than one local vehicle class is applicable (e.g. due to overlapping class specifications). A value of 0 shall be used for the lowest priority level and a value of 255 shall be used for the highest priority level. In cases where local vehicle classes overlap (one single vehicle falling into more than one vehicle class), the attribute **priorityValue** shall be present and the priority values of (partially) overlapping local vehicle classes shall have different values.

EXAMPLE 4 Vehicle class 45 is defined as all trucks having an overall length of 6,50 m to 12,50 m. Priority level for this class is set to 8. Vehicle class 46 is defined as all trucks having an overall length of 8,00 m to 15,00 m. Priority level for this class is set to 5. In cases where a vehicle would fall into both classes, as it has a length of 10,65 m, it is grouped into vehicle class 45 as the priority level of class 45 is higher.

Table 67 — NominalVehicleParameters fields

Field name	Data type/data description	m/o
vehicleClasses	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleClass`	m
vehicleTrainAxles	`List` (ASN.1 SEQUENCE OF) of elements of type `INTEGER` ranging from 0 to 15	o
euroValues	`List` (ASN.1 SEQUENCE OF) of elements of type `EuroValue`	o
copValues	`List` (ASN.1 SEQUENCE OF) of elements of type `CopValue`	o
engineCharacteristics	`List` (ASN.1 SEQUENCE OF) of elements of type `EngineCharacteristics`	o

Table 67 *(continued)*

Field name	Data type/data description	m/o
vehicleCategoryType	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleCategoryType`	o

vehicleClasses shall list service provider specific information pertaining to the vehicle**.**

NOTE `VehicleClass` can be structured as specified in EN 15509 if bilaterally agreed between the TC and the TSP.

vehicleTrainAxles may optionally be used to list information regarding the number of axles of the trailer and the tractor. The attribute shall be used as specified in ISO 14906.

euroValues may optionally be used to list all valid euro emission classes for each local vehicle class.

copValues may optionally be used to list all valid information regarding CO_2 emissions for each local vehicle class;

engineCharacteristics may optionally be used to list all valid engine characteristics (type of fuel or power source) for each local vehicle class.

vehicleCategoryType may optionally be used to specify the types of vehicles. The allowed values are:

undefined categorytype,

handicappedPeople,

military,

police,

roadMaintenance,

circusTruck,

mobileShopTruck,

truckCarryingMilk,

truckCarryingTimber,

publicTransport

enforcementAgent,

ambulance,

fireBrigade,

officialAuthority,

agriculturalVehicle,

bus,

mobileHome,

mobileCrane,

exceptionalTransport,

emperor

Table 68 — OrdinalVehicleParameters fields

Field name	Data type/data description	m/o
vehicleLengthOverall	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleLengthOverallRange`	m
vehicleHeightOverall	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleHeightOverallRange`	o
vehicleWidthOverall	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleWidthOverallRange`	o
vehicleFirstAxleHeight	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleFirstAxleHeightRange`	o
vehicleTractorAxlesNumber	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleTractorAxlesNumberRange`	o
vehicleTrailerAxlesNumber	`List` (ASN.1 SEQUENCE OF) of elements of type VehicleTrailerAxlesNumberRange	o
vehicleMaxLadenWeight	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleMaxLadenWeightRange`	o
vehicleTrainMaximum-Weight	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleTrainMaximumWeightRange`	o
vehicleWeightUnladen	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleWeightUnladenRange`	o
vehicleWeightLaden	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleWeightLadenRange`	o
euroValue	`List` (ASN.1 SEQUENCE OF) of elements of type `EuroValueRange`	o
copValue	`List` (ASN.1 SEQUENCE OF) of elements of type `CopValueRange`	o
vehicleClass	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleClass`	o
co2EmissionValue	`List` (ASN.1 SEQUENCE OF) of elements of type `Co2EmissionValueRange`	o
dieselEmissionValue	`List` (ASN.1 SEQUENCE OF) of elements of type `DieselEmissionValueRange`	o
exhaustEmissionValue	`List` (ASN.1 SEQUENCE OF) of elements of type `ExhaustEmissionValueRange`	o
engineCharacteristics	`List` (ASN.1 SEQUENCE OF) of elements of type `EngineCharacteristicsRange`	o
vehicleCategoryType	`List` (ASN.1 SEQUENCE OF) of elements of type `VehicleCategoryType`	o

vehicleLengthOverall may optionally list the valid overall vehicle length ranges. The attribute shall be used as specified in ISO/TS 17573-3. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the overall vehicle length, in accordance with ISO 612. The value shall be expressed in dm, rounded to the next dm.

— **upperLimit** shall specify the upper limit of the overall vehicle length, in accordance with ISO 612. The value shall be expressed in dm, rounded to the next dm.

vehicleHeightOverall may optionally be used to list the valid overall vehicle height ranges. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the overall vehicle heigth, in accordance with ISO 612. The value shall be expressed in dm, rounded to the next dm.

— **upperLimit** shall specify the upper limit of the overall vehicle heigth, in accordance with ISO 612. The value shall be expressed in dm, rounded to the next dm.

vehicleWidthOverall may optionally be used to list the valid overall vehicle width ranges. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the overall vehicle width, in accordance with ISO 612. The value shall be expressed in dm, rounded to the next dm.

— **upperLimit** shall specify the upper limit of the overall vehicle width, in accordance with ISO 612. The value shall be expressed in dm, rounded to the next dm.

vehicleFirstAxleHeight may optionally be used to list the valid vehicle first axles height ranges. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the vehicle first axles height, in accordance with ISO 612. The value shall be expressed in dm, rounded to the next dm.

— **upperLimit** shall specify the upper limit of the vehicle first axles height, in accordance with ISO 612. The value shall be expressed in dm, rounded to the next dm.

vehicleTractorAxlesNumber may optionally be used to list the valid tractor axles ranges. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the tractor axles.

— **upperLimit** shall specify the upper limit of the tractor axles.

vehicleTrailerAxlesNumber may optionallybe used to list the valid trailer axles ranges. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the trailer axles.

— **upperLimit** shall specify the upper limit of the trailer axles.

vehicleMaxLadenWeight may optionally be used to list the valid maximum permissible total vehicle weight ranges including payload. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the vehicle weight, in accordance with ISO 1176. The value shall be expressed in 10 kg, rounded down to the next 10 kg.

— **upperLimit** shall specify the upper limit of the vehicle weight, in accordance with ISO 1176. The value shall be expressed in 10 kg, rounded down to the next 10 kg.

vehicleTrainMaximumWeight may optionally be used to list the valid maximum permissible weight ranges of the complete vehicle train. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the weight of the complete vehicle train, in accordance with ISO 1176. The value shall be expressed in 10 kg, rounded down to the next 10 kg.

— **upperLimit** shall specify the upper limit of the weight of the complete vehicle train, in accordance with ISO 1176. The value shall be expressed in 10 kg, rounded down to the next 10 kg.

vehicleWeightUnladen may optionally be used to list the valid nominal unladen weight ranges. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the vehicle unladen weight, in accordance with ISO 1176. The value shall be expressed in 10 kg, rounded down to the next 10 kg.

— **upperLimit** shall specify the upper limit of the vehicle unladen weight, in accordance with ISO 1176. The value shall be expressed in 10 kg, rounded down to the next 10 kg.

vehicleWeightLaden may optionally be used to list the valid actual weight ranges of a vehicle. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the actual vehicle weight, in accordance with ISO 1176. The value shall be expressed in 10 kg, rounded down to the next 10 kg.

— **upperLimit** shall specify the upper limit of the actual vehicle weight, in accordance with ISO 1176. The value shall be expressed in 10 kg, rounded down to the next 10 kg.

euroValue may optionally be used to list the valid EURO emission class ranges of vehicles. The attribute shall be used as specified in ISO 14906. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the EURO emission class range.

— **upperLimit** shall specify the upper limit of the EURO emission class range.

copValue may optionally be used to list the valid information regarding CO_2 emission ranges of vehicles. It is made of the following fields:

— **lowerLimit** shall specify the lower limit of the CO_2 emission range.

— **upperLimit** shall specify the upper limit of the CO_2 emission range.

vehicleClass may optionally be used to list service-provider-specific information ranges pertaining to the vehicle by specifying values for the following elements of the `VehicleClass` type:

— **lowerLimit** shall specify the lower limit of the service-provider-specific information pertaining to the vehicle.

— **upperLimit** shall specify the upper limit of the service-provider-specific information pertaining to the vehicle.

co2EmissionValue may optionally be used to list the CO_2 emission value ranges of vehicles. In European toll contexts this attribute shall not be used and the attribute **copValue** may be used instead (see above). The elements of **co2EmissionValue** are of type `CO2EmissionValueRange`, which is specified in ISO/TS 17573-3.

dieselEmissionValue may optionally be used to list information of diesel emission value ranges of vehicles. This is of type `DieselEmissionValueRange`, which is specified in ISO/TS 17573-3.

exhaustEmissionValue may optionally be used to specify information regarding different exhaust emission parameters, e.g. CO, HC, NOX, etc. This is of type `ExhaustEmissionValueRange`, which is a record (ASN.1 SEQUENCE) made of the following fields:

— **emissionUnit**, of type `EmissionUnit`, which shall be used to indicate the unit used to measure the diesel emitted particulates. It is defined in ISO/TS 17573-3.

— **emissionCORange** may optionally be used to specify information regarding CO exhaust emission value ranges. It is a record (ASN.1 SEQUENCE) made of the following fields:

  — **lowerLimit** shall specify the lower limit of the exhaust emission value range.

  — **upperLimit** shall specify the upper limit of the exhaust emission value range.

— **emissionHCRange** may optionally be used to specify information regarding HC exhaust emission value ranges. It is a record (ASN.1 SEQUENCE) made of the following fields:

  — **lowerLimit** shall specify the lower limit of the exhaust emission value range.

  — **upperLimit** shall specify the upper limit of the exhaust emission value range.

— **emissionNOXRange** may optionally be used to specify information regarding NOX exhaust emission value ranges. It is a record (ASN.1 SEQUENCE) made of the following fields:

  — **lowerLimit** shall specify the lower limit of the exhaust emission value range.

— **upperLimit** shall specify the upper limit of the exhaust emission value range.

— **emissionHCNOXRange** may optionally be used to specify information regarding HC and NOX exhaust emission value ranges. The range shall be expressed in g/km in the attribute unitType. It is a record (ASN.1 SEQUENCE) made of the following fields:

  — **lowerLimit** shall specify the lower limit of the exhaust emission value range.

  — **upperLimit** shall specify the upper limit of the exhaust emission value range.

engineCharacteristics may optionally be used to list all valid ranges of engine characteristics (type of fuel or power source) for each localVehicleClass. It is of type `EngineCharacteristics`, defined in ISO/TS 17573-3.

EXAMPLE 1 A range defined with lowerRange = 3,5 t and upperRange = 12 t includes vehicles having 7,5 t in the respective vehicle class.

EXAMPLE 2 A range defined with lowerRange = 3,5 t and upperRange = 12 t includes vehicles having exactly 3,5 t in the respective vehicle class. It excludes vehicles having exactly 12 t from the respective vehicle class.

vehicleCategoryType may optionally be used to specify the types of vehicles.

6.7.4.7 TimeClassDefinition

The data type `TimeClassDefinition` shall contain necessary definitions for all time classes which apply in the respective toll context.

Table 69 describes the fields in the `TimeClassDefinition` type, which is formally defined in Annex A.

Table 69 — TimeClassDefinition fields

Field name	Data type/data description	m/o
timeClasses	List of records (ASN.1 SEQUENCE OF) made of the following fields: **timeClassId**, **timeClassDescription**, **nominalTimeParameters**, **ordinalTimeParameters**, **priorityValue**	m
timeClassDefinitionVersion	`VersionAndValidity` (see Table 24)	o

timeClasses shall contain a list (ASN.1 SEQUENCE OF) of the definitions of all the time classes that apply in the respective toll context.

timeClassDefinitionVersion of type `VersionandValidity` may optionally contain version and validity information for the attribute timeClassDefinition.

Table 70 indicates the fields in the `TimeClass` data type, which is formally defined in Annex A.

Table 70 — TimeClasses fields

Field name	Data type/data description	m/o
timeClassId	`TimeClassId`	m
timeClassDescription	`UTF8String`	o
nominalTimeParameters	Record (ASN.1 SEQUENCE) made of the following fields: **weekdays**, **dates**, **classesSetExternally**	o
ordinalTimeParameters	List of records (ASN.1 SEQUENCE OF) made of the following fields: **weekdays**, **absoluteTimeOfDay**, **relativeTimePeriods**, **periodsInYear**	o
priorityValue	`INTEGER` ranging from 0 to 255	o

timeClassId shall contain a unique identifier for each time class specified in the attribute timeClassDefinition.

Each time class specification may be based on nominal scale parameters and/or ordinal scale parameters. For each specified time class, at least one these attributes shall be specified and all absolute time information in the attributes **nominalTimeParameters** and **ordinalTimeParameters** shall be given in local real time. In cases where both attributes are present in the specification of an individual time class, a Boolean or combination shall apply.

If, at a certain point in time, the conditions given in nominal parameters, the definitions given in ordinal parameters or the definitions in both parameters are met, then the respective time class shall be active.

timeClassDescription may optionally contain an additional textual description of the time class.

nominalTimeParameters may optionally be used to list all necessary information of applicable nominal time parameters for each time class. The type `NominalTimeParameters` is made of:

— **weekdays** of type `INTEGER` ranging from 0 to 7 may optionally list one or more days of the week on which the time class shall be applied (e.g. Monday and Friday). The value 0 shall not be used.

— **dates** may optionally list (ASN.1 SEQUENCE OF) one or more particular dates of type `DateCompact`, on which the time class shall be applied (e.g. 2022-10-03 and 2022-06-18).

— **classesSetExternally** is a list (ASN.1 SEQUENCE OF) of elements of type `INTEGER` ranging from 0 to 255 which may optionally be used to indicate one or more predefined time classes by external input sources independently from the effective time class. If used, these special congestion tariff classes shall be applied and shall overrule the time classes which would apply based on other time class determinants and the local time. The following values are specified:

  — **congestionChargeLevel1**, which defines the special congestion tariff level 1. The functional implementation of this value is up to bilateral agreements;

  — **congestionChargeLevel2**, which defines the special congestion tariff level 2. The functional implementation of this value is up to bilateral agreements.

In cases where one instance of the attribute **nominalTimeParameters** contains more than one entry (in one or more of the attributes **weekdays**, **dates** and **externalSetClasses**), this shall be understood as a Boolean OR combination.

NOTE This function can be used to overrule the time class (and tariff), which would apply based on other time class determinants and the local time. This function can be used in cases where there is a fully dynamic congestion charging scheme. In such schemes, time classes (and tariffs) can be selected based on the real time traffic situation. The external source can, for example, be the back end or a DSRC device installed at the roadside.

EXAMPLE 1 A TC wants to run a toll scheme in which a particular expensive tariff is valid during periods of congestion. However, the TC does not want to predefine the period of congestion as a fixed period, but to apply this tariff only when real congestion is present. In periods of normal traffic, the predefined tariff (based on the predefined time class) is valid. If the traffic amount suddenly exceeds a certain limit, then the TC increases the tariff to make use of the roads less attractive and keep the traffic demand at a lower level. In these cases, the special congestion tariff is applied by activating a new time class (classesSetExternally) that has higher priority compared to the predefined time class.

ordinalTimeParameters may optionally be used to list all necessary information of applicable ordinal time parameters for each time class. The data type `OrdinalTimeParameters` is composed of the following fields:

— **weekdays** may optionally list (ASN.1 SEQUENCE OF) one or more periods, each indicated by a couple (ASN.1 SEQUENCE) of days of type `Weekday`, during a week. The first day of the period shall be given in the attribute **startDay** and the last day of the period shall be given in the attribute **endDay**. The definition of this attribute is as for the **nominalTimeParameters** (see above);

— **absoluteTimeOfDay** may optionally list (ASN.1 SEQUENCE OF) one or more periods in time during a day, each indicated by a couple (ASN.1 SEQUENCE) of days of type `Time`. Start times shall be given

in the attribute **startTime** and end times shall be given in the attribute **endTime**. The `Time` data type is made of the following fields:

- **hours** of type `INTEGER` ranging from 0 to 23 shall contain the hours of the point in time;
- **mins** of type `INTEGER` ranging from 0 to 59 shall contain the minutes of the point in time;
- **secs** of type `INTEGER` ranging from 0 to 59 shall contain the seconds of the point in time.

— **relativeTimePeriods** may optionally be used to list (ASN.1 SEQUENCE OF) one or more time periods. The lower limit of the time period shall be defined in the attribute **minPeriod** of type `INTEGER` ranging from 0 to 2^{16}-1. The upper limit of the time period shall be defined in the attribute **maxPeriod** of type `INTEGER` ranging from 0 to 2^{16}-1. The values in both attributes shall be given in minutes;

— **periodsInYear** may optionally list (ASN.1 SEQUENCE OF) one or more periods of days within a year, each period indicated by a couple (ASN.1 SEQUENCE) of days of type `DateCompact`. The first day of the respective periods shall be given in in the attribute **startDay**. The last day of each period shall be given in the attribute **endDay**.

In cases where one instance of the attribute **ordinalTimeParameters** contains more than one entry (in one or more of the attributes **weekdays**, **absoluteTimeOfDay**, **relativeTimePeriods** and **periodsInYear**), this shall be understood as Boolean AND combination.

EXAMPLE 2 Time class 23 is defined as valid on Monday to Friday between 08:00–10:00 and 16:00-18:00. Priority level of this time class is set to 18.

EXAMPLE 3 Time class 178 is defined valid at 2022-12-27 and 2022-12-29. Priority level of this time class is set to 245.

EXAMPLE 3 In cases where a toll context used time classes 23 and 178, as defined in the examples above, and 2022-12-27 is a Tuesday, time class 178 is valid as it has the higher priority.

EXAMPLE 4 Time class 18 is defined valid on Monday and Friday between 16:00 and 20:15.

EXAMPLE 5 Time class 221 is defined valid on Saturday between 06:00 and 21:00 and only in cases where the vehicle uses the toll service for less than 2 hours.

priorityValue may optionally be used to specify priority information to determine which time class is valid in cases where more than one time class is active (e.g. due to overlapping class specifications). A value of 0 shall be used for the lowest priority level and a value of 255 shall be used for the highest priority level. In cases where time classes overlap (one specific point in time falling into more than one time class), the attribute **priorityValue** shall be present and the priority values of (partially) overlapping time classes shall have different values.

6.7.4.8 UserClassDefinition

The data type `UserClassDefinition` shall contain necessary definitions for all user classes that apply in the respective toll context.

User classes may be used to specify different tariffs depending on particular characteristics of the user of toll contexts. This data type supports differentiation of users according to their contracts and to the number of passengers in the vehicles (high occupancy tolling – HOT).

Table 71 describes the fields in the `UserClassDefinition` data type, which is formally defined in Annex A.

Table 71 — UserClassDefinition fields

Field name	Data type/data description	m/o
userClasses	List of (ASN.1 SEQUENCE OF) `UserClass`	m

Table 71 *(continued)*

Field name	Data type/data description	m/o
userClassDefinitionVersion	`VersionAndValidity` (see Table 24)	o

userClasses shall contain a list (ASN.1 SEQUENCE OF) of the definitions of all user classes of type `UserClass` (see Table 72) that apply in the respective toll context.

userClassDefinitionVersion is of type `VersionAndValidity` and may optionally contain version and validity information.

Table 72 indicates the fields in the `UserClass` data type, which is formally defined in Annex A.

Table 72 — UserClass fields

Field name	Data type/data description	m/o
userClassId	`UserClassId`	m
userClassDescription	`UTF8String`	o
contractTypes	List of records (ASN.1 SEQUENCE OF) of type `ContractTypes`	o
minimumNumberOfPassengers	`ActualNumberOfPassengers`	o
priorityValue	`Int1Unsigned`	o

userClassId shall contain a unique identifier for each user class.

userClassDescription may optionally be used to contain an additional textual description of the user class.

contractTypes is a list of types of contract, defined by the type `ContractTypes`, which may optionally be used to list contracts that fall into the particular user class. The `ContractTypes` data type is made of:

— **contractProvider**, of type `Provider` shall identify the organization that issued the service rights given in the contract. The use and the specification of this attribute shall be according to ISO/TS 17573-3.

— **typeOfContract**, of type `OCTET STRING`, shall contain specific contract provider information regarding the rules that apply to the contract. The use and the specification of this attribute shall be according to ISO/TS 17573-3.

NOTE 1 Contract types are allocated by the service provider. It is up to the service provider to use particular contract types to support certain TC-dependent user classes.

EXAMPLE 1 Element **contractTypes** can support fee exceptions for fire brigades and police cars.

minimumNumberOfPassengers may optionally be used to indicate the threshold number of passengers in the vehicle, including the driver, for which the specified user class applies.

EXAMPLE 2 A value of **minimumNumberOfPassengers** can state the minimum occupancy of a vehicle for which a discounted user class is applicable (HOT type EFC applications).

NOTE 2 The attribute **actualNumberOfPassengers** can be generally used for passenger vehicles. It is not intended to be used for vehicle types that are designed for the transportation of a larger number of passengers, e.g. buses.

priorityValue may optionally be used to specify priority information to determine which user class is valid in cases where more than one user class is active at a time. A value of 0 shall be used for the lowest priority level and a value of 255 shall be used for the highest priority level. In cases where user classes overlap (more than one user class being active at a point in time), the attribute **priorityValue** shall be present and the priority values of (partially) overlapping time classes shall have different values.

6.7.4.9 LocationClassDefinition

The data type `LocationClassDefinition` shall list all location classes that are applicable in the specified toll context.

Table 73 indicates the fields in the `LocationClassDefinition` data type, which is formally defined in Annex A.

Table 73 — LocationClassDefinition fields

Field name	Data type/data description	m/o
locationClasses	List (ASN.1 SEQUENCE OF) of `LocationClass`	m
locationClassDefinitionVersion	`VersionAndValidity` (see Table 24)	o

locationClasses is a list (ASN.1 SEQUENCE OF) of all applicable location classes defined as `LocationClass` data type which shall provide information on the characteristics of the specified location classes. The `LocationClass` data type is made of:

— **locationClassId** of type `LocationClassId` shall identify the location class.

— **locationClassDescription** of type `UTF8String` may optionally contain a description of the specified location class.

— **locationType** is an `INTEGER` ranging from 0 to 255 defined as `LocationType` data type which may optionally provide additional information on the type of the location (road) that is associated with the location class. The following values are specified:

  — **notDefined**, which specifies that the type of the location is not specified.

  — **normalTollRoad**, which specifies that the type of the location is a normal toll road, where the normal toll fee rules apply.

  — **specialTollRoad**, which specifies that the type of the location is a special toll road, where special toll fee rules apply.

    EXAMPLE Mountain roads, where the maintenance cost is higher due to tunnels and bridges and a higher toll fee per km is levied.

  — **nonTollRoad**, which specifies that the type of the location is not a toll road and hence, no tolling is applied.

locationClassDefinitionVersion may optionally contain version and validity information.

6.8 ContractIssuerListAdu data structure

The `ContractIssuerListAdu` data type is used by a TSP to provide a TC with contractual information about their issued OBE.

The `ContractIssuerListAdu` contains the information about one type of OBE issued by a TSP for a given toll domain of a TC.

NOTE 1 Not all types of OBE issued by a TSP can be accepted in all toll domains or by all TCs.

NOTE 2 The inclusion of an entry in this list requires bilateral agreement between the TSP and the TC and can require a prior test of the OBE.

When transferring this ADU, the value in the **informationRecipientID** field of the APCI shall identify a valid entity (a null value is not permitted).

Table 74 indicates the fields in the `ContractIssuerListAdu` data type, which is formally defined in Annex A.

Table 74 — ContractIssuerListAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
contractIssuerList	List (ASN.1 SEQUENCE OF) `ContractIssuerListEntry` (see Table 75)	m
versionAndValidity	`VersionAndValidity` (see Table 24)	m
actionCode	`ActionCode`	o

aduIdentifier shall contain an unambiguous identifier for the `ContractIssuerListAdu` (see 6.2.4).

contractIssuerList shall contain a list of entries of data type `ContractIssuerListEntry`, containing information about contractual information of issued OBEs. Table 75 indicates the fields of data type `ContractIssuerListEntry`, which is formally defined in Annex A.

versionAndValidity shall specify the version number and the starting date and time when the whole ContractIssuerList is considered valid.

actionCode may optionally be used to indicate the action associated with the specific ADU (see 6.2.5).

The TSP shall provide one structure for each OBE type to be included. This means, for example, that if a TSP is using OBEs from different manufacturers for the same PAN, the TSP has to provide a series of ContractIssuerListAdus, one for each manufacturer identifier.

Table 75 — ContractIssuerListEntry fields

Field name	Data type/data description	m/o
efcContextMark	`EFCContextMark`	m
efcContextMarkVersion	INTEGER ranging from 0 to 255	o
equipmentClass	INTEGER ranging from 0 to 2^{15}-1	m
manufacturerId	`INTEGER` ranging from 0 to 2^{16}-1	m
uniquePartOfPan	`UTF8String`	m
typeOfEfcApplication	`UTF8String`	m
securityLevel	`SecurityLevel`	m
acCrKeyReference	`Int1Unsigned`	m
authKeyReference	`Int1Unsigned`	m
validFrom	`VersionAndValidity` (see Table 24)	m

efcContextMark shall contain the EFC context mark stored in the OBE issued by the TSP, which shall be accepted by the TC.

efcContextMarkVersion may optionally be used to indicate the version of the EfcContextMark.

equipmentClass shall contain the equipment class as stored in the OBE and specified in ISO 14906.

manufacturerId shall contain the manufacturer ID as stored in the OBE as specified in ISO 14906.

uniquePartOfPan shall contain the significant digits of the PAN to identify the issuer. This may vary based on the issuer.

EXAMPLE One issuer's PANs are identified based on the first four characters, while other issuers six require six or even more digits to uniquely identify their PANs.

typeOfEfcApplication shall specify the EFC application to use for the specific context mark (e.g. EN 15509, GNSS, PISTA, UNI, etc.).

securityLevel shall specify the security level to use for the specified EFC application according to the definition in Table 76 based upon EN 15509.

Table 76 — SecurityLevel

Field name	Data type/data description
0	No use of access credentials
1	Security level 1
2	Security level 2
3 - 127	Reserved for future CEN and ISO use
128 - 255	Reserved for private use

acCrKeyReference shall specify the number of the key in the keyset to use for access credentials.

authKeyReference shall specify the number of the key in the keyset to use for TSP authenticator.

validFrom shall specify the version number and the starting date and time from which the information is considered valid.

6.9 ExceptionListAdu data structure

The `ExceptionListAdu` data type can be used by the TSP to provide either a response to a request for a specific version of an exception list by a TC or as a push ADU if an updated exception list is available.

When transferring this ADU, the value 0 in the **informationRecipientId** field of the APCI has the special meaning of indicating that the ADU is intended for more than one recipient. If the value of both the CountryCode and the IssuerIdentifier fields of informationRecipientId are zero, the PDU is meant to be sent to all connected entities. If only the value of the IssuerIdentifier is zero, the PDU is meant to be sent to all entities of the stated country.

Table 77 indicates the fields in the `ExceptionListAdu` data type, which is formally defined in Annex A.

Table 77 — ExceptionListAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
exceptionListVersion	`ExceptionListVersion`	m
exceptionListType	`ExceptionListType`	m
exceptionValidityStart	`GeneralizedTime`	o
exceptionValidityEnd	`GeneralizedTime`	o
exceptionListEntries	List (ASN.1 SEQUENCE OF) of fields of type `ExceptionListEntry`	m
actionCode	`ActionCode`	o

aduIdentifier shall contain a unique identifier for the `ExceptionListAdu` (see 6.2.4).

exceptionListVersion shall contain a version number, which will be incremented with each version of the exception list.

exceptionListType shall be used to indicate the type of exception list. Possible values of the `ExceptionListType` data type are just means to differentiate lists. Bilateral agreements or further specifications are needed to specify the exact semantics for each list type, e.g. for EETS. However, the following values of the exceptionListType are specified:

— **blackListFull**, which lists all users for which the TSP temporarily or permanently rejects responsibility;

— **whiteListFull**, which lists all users for which the TSP accepts responsibility;

— **discountedListFull,** which lists all users that are entitled to discounts. The means to apply these discounts and their amounts are subject to bilateral agreements and are out of the scope of this document;

— **blackListIncremental**, which provides an incremental update of list of users for which TSP temporarily or permanently rejects responsibility;

— **whiteListIncremental**, which provides an incremental update of list of users for which TSP accepts responsibility;

— **discountedListIncremental**, which provides an incremental update of list of users that are entitled to discounts.

exceptionValidityStart may optionally be used to specify a point in time in the future from which the exception list is valid.

exceptionValidityEnd can optionally be used to specify an expiry date and time for the exception list.

exceptionListEntries shall be used to specify the TSP's exceptions. Table 78 explains the fields of `ExceptionListEntry` data type, which is formally defined in Annex A.

actionCode may optionally indicate the action associated with the specific ADU (see 6.2.5), e.g. normal sending, revocation, adjustment, etc. The following values are specified for the exceptionListAdu:

— **send**, which specifies a normal (typically, initial) sending of the exception list data.

— **revoke**, which specifies that the previously sent exception list data has been revoked.

— **adjust**, which specifies an adjustment of the previously sent exception list data.

Table 78 — ExceptionListEntry fields

Field name	Data type	m/o
userId	`UserId`	m
statusType	`ExceptionListStatusType`	o
reasonCode	List (ASN.1 SEQUENCE OF) of fields of type `ExceptionListReasonType`	m
entryValidityStart	`GeneralizedTime`	o
entryValidityEnd	`GeneralizedTime`	o
nominalVehicleParameters	`NominalVehicleParameters`	o
ordinalVehicleParameters	`OrdinalVehicleParameters`	o
actionRequested	`ExceptionListActionType`	o
efcContextMark	`EfcContextMark`	o
mediaProviderId	`Provider`	o
applicableDiscounts	List (ASN.1 SEQUENCE OF) of records (ASN.1 SEQUENCE) of the following fields: **discountId**, **proofDocuments**	o

userId shall contain the identifier of the user related to the entry. The data structure of the type `UserId` is specified in Table 6.

statusType may optionally be used to indicate the limitation to which the user is subject in the related toll domain. The following values are specified for data type `ExceptionListStatusType`:

— **allApplications**, which means that the user is blocked for all possible applications;

— **locallyBlocked**, which means that the user is blocked for the TC receiving the information;

— **blockedForSchemesRequiringOdometer**, which means that the user is blocked for all tolling schemata requiring an odometer;

— **noLimits**, which means that the user is not blocked for any application.

reasonCode shall specify a list of the reasons for the insertion of the user in the specified list in the related toll domain. The following values are specified for data type `ExceptionListReasonType`:

— **notToBeDisclosed**, which indicates that the reason is not disclosed.

— **obeDeactivated**, which indicates that the OBE has been deactivated.

— **obeIsStolen**, which indicates that the OBE has been stolen.

— **temporaryTechnicalProblem**, which indicates that the OBE has temporary technical problems.

— **suspicionOnTechnicalManipulation**, which indicates suspected technical manipulation related to the user/OBE.

— **latePayment**, which indicates that the payments of the user are late.

— **noPayment**, which indicates that the payments of the user were not made.

— **contractHolderInsolvent**, which indicates that the contract holder is insolvent.

— **whiteListedUser**, which indicates that the user is white listed. This value shall not be used if the exception list type is **blackListFull** or **blackListIncremental**;

— **suOptOut**, which indicates that the user optioned out of this toll domain.

— **suTemporalSuspension**, which indicates that the user requested temporal suspension.

— **contractClosedByTSP**, which indicates that the contract of the user is closed by the TSP and the OBE has not yet been returned.

— **contractClosedBySU**, which indicates that the contract has been closed by the user and the OBE has not yet been returned.

— **obeNotValid**, which indicates that the OBE is not valid.

— **obeIsLost**, which indicates that the OBE is lost.

— **obeNonExistent**, which indicates that the OBE was not provided by the TSP to the user.

— **obeOnStock**, which indicates that the OBE is in stock and has not yet been assigned by the TSP.

— **obeReturnedEndOfContract**, which indicates that the OBE has been returned after the end of a contract.

— **obeReturnedMalfunction**, which indicates that the OBE has been returned due to malfunction.

— **suspicionOnUseOfJammingDevice**, which indicates that a suspicious usage of a jamming device had been detected;

— **iccWhiteListedUser**, which indicates that the user is whitelisted for using the associated ICC. This value shall not be used if the exception list type is **blackListFull** or **blackListIncremental**;

— **iccNotValid**, which indicates that the associated ICC is (no longer) valid;

— **iccIsStolen**, which indicates that the associated ICC has been stolen;

— **iccIsLost**, which indicates that the associated ICC has been lost;

— **iccNonExistent**, which indicates that the associated ICC has not been produced by the media provider;

— **iccOnStock**, which indicates that the associated ICC has not been issued yet by media provider;

— **iccReturnedEndOfContract**, which indicates that the associated ICC has been returned due to end of contract;

— **iccReturnedMalfunction**, which indicates that the associated ICC has been returned due to malfunction;

— **iccNoPayment**, which indicates that there is no payment associated with the associated ICC;

— **discountedUser**, which indicates that the user is eligible to discounts.

entryValidityStart may optionally be used to indicate a point in time from which the entry is valid.

entryValidityEnd may optionally be used for **ExceptionListType** discountedList to indicate an expiry date and time for the entry. **entryValidityEnd** shall not be used for any other **ExceptionListType**. Possible deviations or conflicts with the similar fields on the ExceptionListAdu level (**exceptionValidityStart** or **exceptionValidityEnd**, respectively) shall be clarified by bilateral agreements between TC and TSP.

nominalVehicleParameters may optionally be used to specify the nominal vehicle parameter of the vehicle that is related to the user referenced by the data element **userId**. The data structure of `NominalVehicleParameters` is shown in Table 67 and is formally defined in Annex A.

ordinalVehicleParameters may optionally be used to specify the ordinal vehicle parameters (value ranges) of the vehicle that is related to the user referenced by the data element **userId**. The data structure of `OrdinalVehicleParameters` is shown in Table 68 and is formally defined in Annex A.

actionRequested may optionally be used to indicate an action the TSP requests to be carried out by the TC. The following values are specified:

— **rejectObe**, which indicates that the OBE shall be rejected. This value shall not be used if the exception list is of type whiteList or discountedList;

— **invalidateObe**, which indicates that the black list bit shall be set by the TC. The usage of this value is allowed, if supported by the TC. It shall not be used if the exception list is of type whiteList or discountedList;

— **acceptObe**, which indicates that the OBE shall be accepted. This value shall not be used if the exception list is of type blackList;

— **removeObe**, which indicates that the OBE shall be removed. This value shall not be used if the exception list is of type whiteList or discountedList;

— **removeUserId**, which indicates that the user has to be removed from the specified list.

efcContextMark may optionally specify the EFC context in the OBE that is related to the exception. It shall be used as specified in ISO/TS 17573-3.

mediaProviderId may optionally identify the provider of the media when the exceptionListType has the value “ICC”. It shall not be used otherwise.

applicableDiscounts may optionally be used to associate the user with a list of discounts. It is a list of records, each made of the following fields:

— **discountId**, of type `DiscountId`, which shall indicate the discount;

— **proofDocuments**, of type `OCTET STRING`, which may optionally be used to attach documents (e.g. PDF files, pictures, etc.) in support of the eligibility of the user to the identified discount.

6.10 ReportAbnormalObeAdu data structure

The `ReportAbnormalObeAdu` data type can be used by the TC to report an abnormal detected OBE behaviour either in a response to a request from the TSP or as a push ADU when abnormal OBE behaviour is detected.

When transferring this ADU, the value in the **informationRecipientID** field of the APCI shall identify a valid entity (a null value is not permitted).

Table 79 indicates the fields of the `ReportAbnormalObeAdu` data type, which is formally defined in Annex A.

Table 79 — ReportAbnormalObeAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
userId	`UserId`	m
dateandTime	`GeneralizedTime`	m
efcContextMark	`EfcContextMark`	o
abnormalOBEReasonCode	List of records (ASN.1 SEQUENCE OF SEQUENCE) of the following fields: **abnormalObeReasonCode**, **additionalInformation**.	m
tollEventId	`TollEventId`	o
actionCode	`ActionCode`	o

aduIdentifier shall contain an unambiguous identifier for the `ReportedAbnormalObeAdu` (see 6.2.4).

userId shall contain the user identification as recorded by the TC's equipment for which an abnormal behaviour is reported. See Table 6 for the definition of `UserId`.

dateandTime shall contain the date and time of the detection of the abnormal behaviour.

efcContextMark may optionally be used to specify the EFC context mark related to the on-board unit (OBU) as read by the TC equipment.

abnormalObeReasonCode shall contain a list of reasons, each defined as a record (ASN.1 SEQUENCE) of a code of type `AbnormalObeReasonCode` and an optional text description of type `UTF8String`. The `AbnormalObeReasonCode` can take the following values:

— **reasonNotToBeDisclosed**, when the TC does not specify the reason;

— **obeIsDefect**, when the TC detected a defect in the OBE;

— **obeIsNotWorkingProperly**, when the TC detected an OBE which shows incorrect functioning (for example, bad transaction behaviour);

— **userShowsFraudBehaviour**, when the TC detected a fraudulent behaviour of the user specified in **userId**.

— **userShowsViolatingBehaviour**, when the TC detected a behaviour violating local rules of the user specified in **userId**.

— **licensePlateDifferenceBetweenObeAndVehicle**, when the TC detected a difference between the declared licence plate in the OBE and the actual licence plate as externally shown.

— **numberOfAxlesMismatch**, when the TC detected a difference between the declared number of axles in the OBE and the actual number of axles as externally shown.

— **euroEmissionCategoryProofMissing**, when the TC cannot validate the declared emission category.

— **euroEmissionCategoryMismatch**, when the TC detected a difference between the declared emission category in the OBE and the recognized emission category.

— **engineCharacteristicsProofMissing**, when the TC cannot validate the declared engine characteristics.

— **engineCharacteristicsMismatch**, when the TC detected a difference between the declared engine characteristics in the OBE and the recognized engine characteristics.

— **co2EmissionValueProofMissing**, when the TC cannot validate the declared CO_2 emissions.

— **co2EmissionValueMismatch**, when the TC detected a difference between the declared CO_2 emissions in the OBE and the recognized CO_2 emissions.

— **obeMissingOnWhiteList**, when the OBE is not included in a white list.

— **obeOnBlacklistButTolling**, when the OBE is blacklisted, but it is tolling.

— **vehicleWeightMismatch**, when the TC detected a difference between the declared vehicle weight in the OBE and the recognized vehicle weight.

— **obeAtExitStationWithoutEntryStation**, when the TC recognized an OBE at an exit station with no history of entry in a closed system.

— **transactionsFromMultipleObe**, when the TC recognized an OBE participating in a DSRC transaction at the same time with another OBE. For example, this can happen when a user who has multiple contracts and multiple OBEs on board of a vehicle passes through a tolling point with more than one OBE active.

— **obeDeclaredVehicleClassNotAllowed**, when the TC recognized a declared vehicle class (from the OBE) not valid or not allowed, for example, a truck travelling in a day when trucks are not allowed to circulate in the road network.

— **missingUserIdInUserlist**, when the TC received a charge report of an OBE or UserId, respectively, that was not communicated in the exceptionListAdu to the TC.

— **chargeReportDeliveredTooLate**, when the TC received a charge report of an OBE or UserId, respectively, that was not provided in due time. The maturity of charge reports is based on bilateral agreements between the TC and the TSP.

— **velocityAccuracy**, when the TC detected by means of OBE locations recognition a possible abnormality in the vehicle's speed. For example, an impossibly short time or a suspiciously long time between an entry and an exit in a closed system.

actionCode may optionally be used to indicate an action associated with the specific ADU (see 6.2.5).

6.11 TollDeclarationAdu data structure

The `tollDeclarationAdu` data type can be used by both the TSP and the TC either in response to a request to receive toll declarations or as a push ADU when toll declarations are available to be transferred. When transferring this ADU, the value in the **informationRecipientID** field of the APCI shall identify a valid entity (a null value is not permitted).

Table 80 indicates the fields of the `tollDeclarationAdu` data type, which is formally defined in Annex A.

Table 80 — TollDeclarationAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
tollDeclarationId	`TollDeclarationId`	o

Table 80 *(continued)*

Field name	Data type/data description	m/o
chargeReport	List (ASN.1 SEQUENCE OF) of records of type `ChargeReport`	m
actionCode	`ActionCode`	o

aduIdentifier shall contain a unique identifier for the `TollDeclarationAdu` (see 6.2.4).

tollDeclarationId may optionally be used to uniquely identify the toll declaration within the generating entity. It is made of the following fields:

— **issuerId** of type `Provider` (imported from ISO/TS 17573-3) shall contain the identification of the entity that generates the toll declaration;

— **declarationId** of type `INTEGER` ranging from 0 to 2^{63}-1 shall contain the unique within the entity identified by **issuerId**.

The following rules shall apply for **tollDeclarationId**:

— If **tollDeclarationID** is used, the field **declarationId** shall contain the same value as in **aduIdentifier**.

— If **issuerId** differs from the **apduOriginator**, the **tollDeclarationId** shall be specified.

chargeReport shall list charge data records of type `ChargeReport`, which is described in Table 81 and is formally defined in Annex A.

actionCode may optionally be used to indicate an action associated with reception of the specific ADU, (see 6.2.4).

Table 81 — ChargeReport fields

Field name	Data type/data description	m/o
obeId	`ObeId`	o
vehicleLPNr	`VehicleLicencePlateNumber`	o
paymentMeans	`PaymentMeans`	o
serviceProviderContract	`EfcContextMark`	m
tollContext	`Provider`	o
chargeReportFinalRecipient	`Provider`	o
timeOfReport	`GeneralizedTime`	o
reportPeriod	Record (ASN.1 SEQUENCE) of the following fields: **beginOfPeriod**, **endOfPeriod**	o
appliedTollContextVersion	`VersionId`	o
usageStatementList	List (ASN.1 SEQUENCE OF) of records of type `UsageStatement`	m
sumVatForThisSession	`PaymentFee`	o
chargeReportCounter	`INTEGER` ranging from 0 to 2^{63}-1	o
mileage	`Distance`	o
listOfCccContainers	List (ASN.1 SEQUENCE OF) of attributes of type `DsrcData`	o
frontEndVersion	`FrontEndVersion`	o

obeId may optionally be used to identify the OBE (see 6.2.6).

vehicleLPNr may optionally be used to specify the claimed licence plate of the vehicle.

paymentMeans may optionally be used to specify the personal account number and the validity of the payment means of the user.

serviceProviderContract shall contain provider-specific information regarding the contract and the context version of the contract to which the charge report data pertain.

tollContext may optionally be used to indicate the toll context to which the charge report relates.

chargeReportFinalRecipient may optionally be used to identify the entity which is the final recipient of the charge data. This field can be used within **chargeReport** and within **usageStatement**, which is part of the **chargeReport**. To avoid inconsistencies, it is recommended to use it once either within either **chargeReport** or within **usageStatement**.

timeOfReport may optionally be used to indicate the date and time when the charge report was compiled.

reportPeriod may optionally be used to indicate the time period covered by the charge report. The `Period` data type is defined by the date and time of its beginning and of its end, both of type `GeneralizedTime`.

appliedTollContextVersion may optionally be used to specify the version of the charge data.

usageStatementList is a list (ASN.1 SEQUENCE OF) of records that contain all usage statements of type `UsageStatement`. The `UsageStatement` data type is shown in Table 82 and is formally defined in Annex A.

sumVatForThisSession may optionally be used to specify the aggregated VAT for all the fees communicated in the respective charge report.

chargeReportCounter may optionally be used to enumerate the charge report. This counter shall be incremented after compilation of a charge report, facilitating distinction between different charge reports. In the case of overflow, the counter shall restart at 0. The values of **chargeReportCounter** are determined for each OBU separately; there is no correlation (nor uniqueness) between different OBUs.

mileage may optionally be used to contain the reading of an internal mileage counter. The counter shall start at 0 for a new OBE and be continuously incremented while the OBE is active; the counter shall restart from zero in case of overflow (i.e. when reaching the maximum value).

listOfCccContainers may optionally be used to provide a list of OBU-related data of type `DsrcData`. The data type `DsrcData` is shown in Table 103.

NOTE 1 Data contained in **listOfCccContainers** can also be transferred using the element **listOfDSRCUsageData** within the type `UsageStatement`.

frontEndVersion may optionally be used to specify GNSS Front-End information of the OBE that was in operation in the context of the generated charge report. It is made of the following fields:

— **frontEndSWVersion** of type `UTF8String` which shall contain software version information;

— **frontEndHWVersion** of type `UTF8String` which shall contain hardware version information.

Table 82 — UsageStatement fields

Field name	Data type/data description	m/o
usageStatementId	`Int2Unsigned`	o
tollContext	`Provider`	o
chargeReportFinalRecipient	`Provider`	o
aggregatedFee	`AggregatedFee`	o
sumVat	`PaymentFee`	o
aggregatedSingleTariffClassSession	`AggregatedSingleTariffClassSession`	o
listOfChargeObjects	List (ASN.1 SEQUENCE OF) of fields of type `DetectedChargeObject`	o
listOfDsrcUsageData	List (ASN.1 SEQUENCE of) of fields of type `DsrcUsageData`	o

Table 82 *(continued)*

Field name	Data type/data description	m/o
listOfRawUsageData	`ListOfRawUsageData`	o
noUsage	`BOOLEAN`	o
additionalUsageInformation	`OCTET STRING`	o
costCentre	`UTF8String`	o

usageStatementId may optionally be used to uniquely identify the usage statement.

tollContext may optionally be used to identify the toll context the usage statement relates.

chargeReportFinalRecipient may optionally be used to identify the entity which is the final recipient of the usage data. This field can be used within **chargeReport** and within **usageStatement**, which is part of the **chargeReport**. To avoid inconsistencies, it is recommended to use it once either within either **chargeReport** or within **usageStatement**.

aggregatedFee may optionally be used to specify the aggregated fee for the usage data. The data type `AggregatedFee` is made of the following fields:

— **timePeriodCovered** of type `Period` shall contain the time period covered by the usage statement.

— **feeExclVat** of type `PaymentFee` (imported from ISO/TS 17573-3) shall contain the total amount of fee excluding VAT that is covered by the usage statement within the time period indicated by **timePeriodCovered**.

— **sumVat** of type `PaymentFee` may optionally be used to specify the aggregated VAT covered by the usage statement within the time period indicated in **timePeriodCovered**.

sumVat may optionally be used to specify the aggregated VAT for the fees communicated in the usage statement.

aggregatedSingleTariffClassSession may optionally be used to indicate a single tariff class session, which may be related to a part of a trip or to a whole trip in case of no changes of charge relevant parameters. Session changes may be due to changes in vehicle category, road category, time of day, etc. The **aggregatedSingleTariffClassSession** field is of `AggregatedSingleTariffClassSession` data type, which is defined in ISO/TS 17573-3.

listOfChargeObjects may optionally be used to provide information about the detected charge objects covered by the usage statement. The **listOfChargeObjects** field is of `DetectedChargeObject` data type, which is defined in ISO/TS 17573-3.

listOfDsrcUsageData is a list (ASN.1 SEQUENCE OF) of attributes obtained in a selection (ASN.1 CHOICE) of different DSRC transaction types, specified by data types of either `EfcContainer` (imported from ISO 14906), `CccContainer` (imported from ISO 12813) or `LacContainer` (imported from ISO 13141). Usage of this attribute is based on bilateral agreements between the TC and TSP.

listOfRawUsageData may optionally be used to list (ASN.1 SEQUENCE) measured position data by an OBU which may be used for the calculation of the tariff covered by the usage statement. The data type is shown in Table 83 and is formally defined in Annex A.

noUsage may optionally be used for the transmission of empty usage statements.

NOTE 2 The field **noUsage** can be useful in situations where, for example, an agreement between the TC and the service provider requests that **UsageStatement** be delivered at given times, even when there is no usage data to communicate.

additionalUsageInformation may optionally be used to provide additional information regarding the usage statement. Usage of this attribute is based on bilateral agreements between the TC and TSP.

costCentre may optionally be used to provide textual information useful for splitting the costs of a single journey into more cost centres.

Table 83 — ListOfRawUsageData fields

Field name	Data type/data description	m/o
rawDataList	List (ASN.1 SEQUENCE OF) of records of type `MeasuredRawData`	m
currentTariffClass	`TariffClassDescription`	o
vehicleDescription	`VehicleDescription`	o

rawDataList shall be used to provide the data as measured by the positioning capabilities of the OBE. The data type MeasuredRawData is a record (ASN.1 SEQUENCE) of the following optional fields, of which at least one shall be specified:

— **measuredPosition** of type `AbsolutePosition3d` may optionally be used to specify the measured position data of the OBE. The position data shall be specified by using the fields **longitude**, **latitude** and, optionally, **altitude**. If the field **altitude** is used, its default unit is metre.

— **timeWhenMeasured** of type `GeneralizedTime` may optionally be used to indicate the date and time at which the position was measured.

— **additionalGnssData** of type `OCTET STRING` may optionally be used to provide additional information delivered by the GNSS modules or chipsets. Possible formats include but are not restricted to receiver independent exchange format (RINEX[13]), additional data in the format defined by the National Marine Electronics Association (NMEA) 0183[14], or data in proprietary binary formats provided by several kinds of satellite positioning modules or chipsets.

currentTariffClass may optionally be used to provide all necessary information for determining the tariff class to compute the tariff based on the measured position data. The **currentTariffClass** field shall be used as specified for the data type `AggregatedSingleTariffClassSession`.

vehicleDescription may optionally be used to provide all vehicle specific information for the calculation of the tariff. The **vehicleDescription** field structure shall be used as specified for the data type `AggregatedSingleTariffClassSession`.

6.12 BillingDetailsAdu data structure

6.12.1 General

The `BillingDetailsAdu` data type can be used by both the TSP and the TC either in response to a request to receive unsent billing details or as a push ADU when billing details are available.

When transferring this ADU, the value in the **informationRecipientId** field of the APCI shall identify a valid entity (a null value is not permitted).

Table 84 indicates the fields of the `BillingDetailsAdu` data structure, which is formally defined in Annex A.

Table 84 — BillingDetailsAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
billingDetailsInfo	`BillingDetailsInfo`	o
tollContext	`Provider`	m
userId	`UserId`	o
paymentMeans	`PaymentMeans`	o

Table 84 *(continued)*

Field name	Data type/data description	m/o
relatedBillingDetails	Selection (ASN.1 CHOICE) between an `AduIdentifier` and BillingDetailsInfo to reference to another `BillingDetailsAdu`	o
period	`Period`	o
billingDetailsAmount	`PaymentAmount`	m
usage	Record (ASN.1 SEQUENCE) of the following fields: **tollcontextName**, **costCenter**, **usageList**	m
refTollDeclaration	Record made of selections (ASN.1 SEQUENCE OF CHOICE) between `AduIdentifier` and `TollDeclarationId` to reference to one or more TollDeclarationAdus	o
paymentReference	`UTF8String`	o
usageInformation	`BillingDetailsUsageInformation`	o
actionCode	`ActionCode`	o

aduIdentifier shall contain a unique identifier for the `BillingDetailsAdu` (see 6.2.4).

billingDetailsInfo may optionally be used to identify the billing details. The `BillingDetailsInfo` type is made of the following fields:

— **issuerId,** of type `Provider`, which shall identify the issuer of the billing details;

EXAMPLE In tolling systems where multiple TCs are present, it is not uncommon that smaller TCs consociate in clusters in order to reduce live expenses. The cluster organization acts as originator for all ADUs to be sent by their associates. If the cluster organization uses an external service centre for running the back-office system, a BillingDetailsAdu sent by that centre will carry as **issuerId** the identifier of the actual TC, as **originator** the cluster organization identifier, and as **sender** the service centre identifier.

— **billingDetailsNum**, of type `INTEGER` ranging from 0 to 2^{63}-1, which shall uniquely identify the billing details within the **issuerId**.

— **dateOfService**, of type `GeneralizedTime`, which may optionally be used to indicate the date and time at which the billing details were issued.

— **billingDetailsReference**, of type `UTF8String`, which may optionally be used to transfer textual information in a field of a maximum length of 32 octets.

tollContext shall contain the unique identifier of the toll context to which the billing details are related. The semantics of this field are further explained in the description of **usage** field.

userId may optionally be used to identify the service user to whom the billing details shall be charged. See 6.2.6 for details.

paymentMeans may optionally be used to specify information related to the user's account.

relatedBillingDetails may optionally be used to identify previously issued billing details by means of either its **aduIdentifier** or its **BillingDetailsInfo**, to which the present billing details are associated. The meaning of this association is not specified in this document.

period may optionally be used to specify the period of time to which the billing details are related.

billingDetailsAmount shall contain the sum of all fee-related elements in the **usage** field, their associated VAT rate and currency. This requires that the field **usage** only refers to records with the same VAT rate and the same currency. Separate billing details ADUs shall be used if any VAT rate or currency differs. The `PaymentAmount` data type is made of a record (ASN.1 SEQUENCE) with the following fields:

— **paymentFeeAmount**, of tpe `INTEGER` ranging from -2^{63} to 2^{63}-1, which indicates the amount.

— **paymentFeeUnit**, of type `PayUnit`, which indicates the currency and the unit of that currency to be used.

— **vatRate**, of type `INTEGER` ranging from 0 to 10000, which may optionally be used to indicate the VAT rate in 0,01 %.

usage shall be specified to provide extended information about the usage of tolling objects by the user. The **usage** field is a record (ASN.1 SEQUENCE) of the following fields:

— **tollContextName**, of type `UTF8String`, which may optionally contain the name of the toll context.

— **costCenter**, of type `UTF8String`, which may optionally contain the name of the cost centre for this billing details ADU.

— **usageList**, of type `UsageList`, which may optionally contain detailed information on how the billing details have been generated. As the UsageList data type varies according to the different types of toll context, its structure is explained separately in 6.12.2.

refTollDeclaration may optionally be used to identify a previously issued toll declaration ADU by means of either (ASN.1 CHOICE) its **aduIdentifier** or its **TollDeclarationId**, to which the present billing details ADU is associated. The meaning of this association is not specified in this document.

paymentReference may optionally contain a text to relate the billing details to a payment;

actionCode may optionally be used to indicate specific actions to be pursued by the receiver of the billingDetailsAdu (see 6.2.5).

6.12.2 UsageList data type

The `UsageList` data type is a record (ASN.1 SEQUENCE) of fields which may be of different types, so covering all possible types of toll domains a user will be billed according to the specified billing details.

The `UsageList` data type is described in Table 85 and formally specified in Annex A.

Table 85 — UsageList fields

Field name	Data type/data description	m/o
usageListEntry	Selection (ASN.1 CHOICE) among the following fields **forSectionedRoads, forTravellingInArea, forStayingInArea, forCordonCrossings, freeTextDetail, forMeshedSystem, noUsageDetails**	m
includedDiscounts	List (ASN.1 SEQUENCE OF) of records (ASN.1 SEQUENCE) of fields of the following types: `LanguageID, UTF8String, DiscountId, PaymentAmount, DiscountQualifier`	o
usageInformation	List (ASN.1 SEQUENCE OF) of fields of type `BillingDetailsUsageInformation`	o

usageListEntry shall be used to specify the type of tolling characteristics related to the specific entry and is a selection (ASN.1 type CHOICE) among:

— **forSectionedRoads**, where information on the entry, intermediate traversed and exit sections are given. For each section, the charge objects used to detect the usage may be further qualified as physical objects (e.g. gantries) or computed values. The **forSectionedRoads** is a record (ASN.1 SEQUENCE) of fields that are detailed in Table 86.

— **forTravellingInArea,** where the distance travelled in areas and optionally the time spent in areas are specified. The **forTravellingInArea** is a record (ASN.1 SEQUENCE) of fields that are detailed in Table 89.

— **forStayingInArea**, where time of staying in given areas are specified. The **forStayingInArea** field is a record (ASN.1 SEQUENCE) of fields that are detailed in Table 90.

— **forCordonCrossings,** where indication of the entrance time, exit time and tolling objects indicating a cordon crossing are specified. The **forCordonCrossing** field is a record (ASN.1 SEQUENCE) of fields that are detailed in Table 91.

— **freeTextDetail**, where a free text is given, expressed as a character string (UTF8String), to describe the reasons for tolling. The **freeTextDetail** field is a record (ASN.1 SEQUENCE) of fields that are detailed in Table 92.

— **forMeshedSystem**, where indication of travels, or of traversed check points are given to describe the reason for tolling. The **forMeshedSystem** is a record (ASN.1 SEQUENCE) of fields that are detailed in Table 93.

— **noUsageDetails**, which may optionally indicate a non-usage during the reporting period. This information may be needed when, for example, billing details have to be sent by contract at given times, even if the user has not travelled in that toll context during the related period of time.

includedDiscounts is a list of records (ASN.1 SEQUENCE OF SEQUENCE) which may optionally be used to specify applied discounts. Each record in the list is made of the following fields:

— **textLanguage**, of type `LanguageID`, which may optionally be used to specify the language of the text in the **textDetail** field;

— **textDetail**, of type `UTF8String`, which may optionally be used to add a textual description of the applied discount;

— **appliedDiscountId**, of type `DiscountId`, which shall be used to identify the type of discount;

— **discountAmount**, of type `PaymentAmount`, which shall be used to specify the applied discount;

— **discountQualifier**, of type `INTEGER` ranging from 0 to 255 defined as DiscountQualifier, which shall be used to indicate the reason of the discount. The following values are specified for `DiscountQualifier`:

  — **noDiscount**, to indicate no discount applied.

  — **timeDiscount**, to indicate discount related to time of usage.

  — **vehicleDiscount**, to indicate discount related to type of vehicle.

  — **tripDiscount**, to indicate discount related to type of trip.

  — **userDiscount**, to indicate discount related to type of user.

  — **dayOfWeekDiscount**, to indicate discount related to working day usage.

  — **holidayDiscount**, to indicate discount related to festivity usage.

  — **routeDiscount**, to indicate discount related to selection of alternative route usage.

  — **companyDiscount**, to indicate discount related to contract agreement (mass usage).

  — **eligibleForDiscount**, to indicate if the billing details are eligible for a discount.

usageInformation is a list of items of type BillingDetailsUsageInformation which may optionally be used to indicate warnings or errors while processing and generating the billing details. The following values are defined for elements of type BillingDetailsUsageInformation:

— **noUsageInformation**, which indicates that no additional information is available;

— **userIdUnknown**, which indicates that the related userId was not known to the entity that generated the billing detail (for example, not on the white list).

Table 86 — forSectionedRoads fields

Field name	Data type/data description	m/o
invoiceAggregationNumber	`UTF8String`	o

Table 86 *(continued)*

Field name	Data type/data description	m/o
howManyTimes	`INTEGER` ranging from 0 to 2^{63}-1	m
usedSections	List (ASN.1 SEQUENCE OF) fields of type `UsedSection`	o
appliedTariffTableVersion	`VersionId`	o
appliedLocalVehicleClass	`AppliedLocalVehicleClass`	m
appliedTimeClass	`AppliedTimeClass`	m
appliedUserClass	`AppliedUserClass`	m
vehicleDescription	`VehicleDescription`	o
usageDistance	`Distance`	o
usageFee	`PaymentAmount`	o
usageFeeQualifier	`FeeQualifier`	o
usageExternalCosts	`ExternalCosts`	o

invoiceAggregationNumber may optionally be used to associate the billing details to an invoice aggregation.

howManyTimes shall be used to indicate how many times the same trip has been carried out. This can be used to apply special tariffs to commuters, for example.

usedSections may optionally be used to specify the sections traversed for these billing details. The data type `UsedSection` is specified in Table 87.

Table 87 — UsedSection fields

Field name	Data type/data description	m/o
tollContext	`Provider`	o
usedChargeObjectId	`ChargeObjectId`	o
directUsageInformation	Record (ASN.1 SEQUENCE) of the following fields: **usedChargeObjectName, usedSectionDistance**.	o
appliedLocationClass	`AppliedLocationClass`	o
tollEventId	`TollEventId`	o
tollEventTime	`GeneralizedTime`	o
modeOfOperation	`INTEGER (0..255)`	o
associatedEventData	Record (ASN.1 SEQUENCE OF) of fields of type `AssociatedEventData`	o
usedSectionFee	`PaymentAmount`	o
usedSectionFeeQualifier	`FeeQualifier`	o
usedSectionExternalCosts	`ExternalCosts`	o
usageInformation	Record (ASN.1 SEQUENCE OF) of fields of type `BillingDetailsUsageInformation`	o

tollContext may optionally be used to indicate the toll context the section belongs to.

usedChargeObjectId may optionally be used to reference the ChargeObject as defined in the **SectionLayout**.

directUsageInformation may optionally be used to specify information about the used section. It is made of the following fields:

— **usedChargeObjectName**, which is a UTF8String which shall be used to specify the name of the section when usedChargeObjectId is not specified.

— **usedSectionDistance**, which is of type Distance and shall be used to indicate length of the used section when usedChargeObjectId is not specified.

appliedLocationClass may optionally be used to reference the **LocationClass** of the section (which can differ from section to section).

tollEventId may optionally be used to reference a tolling event related to the billing details. The `TollEventId` data type is a selection (ASN.1 type CHOICE) from among the following fields:

— **tollDeclarationEventId**, which is a record (ASN.1 type SEQUENCE) of the following fields:

  — **chargeReportCounter**, which identifies a charge report;

  — **usageStatementID**, which identifies a usage statement in a TollDeclarationAdu.

— **tollTransactionEventId,** which identifies a toll transaction in a charge report.

— **transactionCounter,** which identifies a toll transaction, e.g. for a DSRC system.

tollEventTime may optionally be used to indicate the time of the tolling event.

modeOfOperation may optionally be used to indicate the mode by which the information about the used section has been obtained. The following values are allowed for the `ModeOfOperation` data type:

— **normal,** to indicate a normal mode of operation.

— **degraded,** to indicate that the information has been obtained in a degraded mode. For example, in a DSRC system where the DSRC transaction incurred errors, the information on traversed sections was obtained by means of cameras.

— **virtual**, to indicate that the information has not been obtained by physical means. For example, in a DSRC system where the DSRC road units in a section were not functioning, the traversal of that section by a user was inferred algorithmically from information gathered from adjacent sections.

associatedEventData may optionally be used to associate a series of events to the used section. The `AssociatedEventData` data type is described in subclause 6.12.3.

usedSectionFee may optionally be used to indicate the fee for the used section. The sum of these fees shall correspond to usageFee in UsageList (if present therein).

usedSectionFeeQualifier may optionally qualify the fee amount as a default charge, a toll substitute, a belated payment or a reimbursement.

usedSectionExternalCosts may optionally indicate the external costs for the used section. The sum of these fees shall correspond to **usageExternalCosts** in UsageList (if present there). If **usageExternalCosts** is specified in UsageList, its value takes precedence over the sum of all specified **usedSectionExternalCosts** fields in **usedSection**.

usageInformation may optionally be used to further qualify the provided information. It is a list of elements of type `INTEGER` defined as `BillingDetailsUsageInformation`. The following values may be used:

— **noUsageInformation**, to indicate no additional information is available;

— **userIdUnknown**, to indicate that the userId was not known to the entity that generated the billing detail (for example, user not in the white list).

appliedTariffTableVersion may optionally be used to indicate the version of the tariff table used to compute the fee.

appliedLocalVehicleClass may optionally be used to indicate the local vehicle class used to compute the fee.

appliedTimeClass may optionally be used to indicate the time class used to compute the fee.

appliedUserClass may optionally be used to indicate the user class used to compute the fee.

vehicleDescription may optionally be used to specify the vehicle's parameters.

usageDistance may optionally be used to indicate the total distance to be billed. If specified, its value overrides the sum of all **usedSectionDistance** fields that can possibly be specified in the **directUsageInformation** field of **usedSections**.

usageFee may optionally be used to indicate the total fee for this usage of sections.

usageFeeQualifier may optionally be used to qualify the fee amount as a default charge, a toll substitute, a belated payment or a reimbursement.

usageExternalCosts may optionally be used to indicate the external costs for this usage of sections. If specified, its value takes precedence over the sum of all specified **usedSectionExternalCosts** fields. It is of type ExternalCosts, which is specified in Table 88.

Table 88 — ExternalCosts fields

Field name	Data type/data description	m/o
externalCostsAir	`PaymentAmount`	o
externalCostsNoise	`PaymentAmount`	o
externalCostsAccident	`PaymentAmount`	m
externalCostsCo2	`PaymentAmount`	o
externalCostsCrossFinancing	`PaymentAmount`	o
externalCostsTotal	`PaymentAmount`	o

When the usageExternalCosts field is used, at least one of its optional components shall be specified:

— **externalCostsAir** may optionally indicate external costs for air pollution;

— **externalCostsNoise** may optionally indicate external costs for noise;

— **externalCostsAccident** may optionally indicate external costs for accidents;

— **externalCostsCo2** may optionally indicate external costs for CO_2 effects;

— **externalCostsCrossFinancing** may optionally indicate external costs (mark-up) for cross-financing of other projects not related to the road being tolled;

— **externalCostsTotal** may optionally indicate the sum of all external costs and applying the rounding rules.

When the usageExternalCosts field is used, at least one of its optional components shall be specified.

If externalCostsTotal component is specified, its value takes precedence over the sum of all other specified components, if any.

NOTE This rule and the usage of externalCostsTotal prevents misinterpretations in rounding when calculating sums.

Table 89 — forTravellingInArea fields

Field name	Data type/data description	m/o
areaDisplayName	`UTF8String`	m
usedArea	Selection (ASN.1 CHOICE) between **usedAreaId** and **usedAreaObject**	m
accumulatedDistance	`Distance`	m

Table 89 *(continued)*

Field name	Data type/data description	m/o
beginOfAccumulation	`GeneralizedTime`	o
endOfAccumulation	`GeneralizedTime`	o
associatedEventData	Record (ASN.1 SEQUENCE OF) of fields of type `AssociatedEventData`	o
appliedTariffTableVersion	`VersionId`	o
appliedLocalVehicleClass	`AppliedLocalVehicleClass`	m
appliedTimeClass	`AppliedTimeClass`	m
appliedLocationClass	`AppliedLocationClass`	m
appliedUserClass	`AppliedUserClass`	m
vehicleDescription	`VehicleDescription`	o
usageFee	`PaymentAmount`	o
usageFeeQualifier	`FeeQualifier`	o
usageExternalCosts	`ExternalCosts`	o

areaDisplayName shall be used to indicate textually the name of the area.

usedArea shall be used to identify the area as specified in the **EfcContextDataAdu** by selecting either:

a) **usedAreaId**, of type AreaId, to reference the **AreaPricingLayout** defined in the `TollContextPartitionLayout`;

b) **usedAreaObject**, of type `AreaLayout`, to directly reference the used area.

accumulatedDistance, of type `Distance`, shall indicate the distance, in terms of quantity and unit used for charging. In this case, the **distance** branch shall be selected.

beginOfAccumulation and **endOfAccumulation** may optionally be used to specify the starting time and the ending time of the travel in the area.

associatedEventData may optionally be used to associate a series of events to the used section. The `AssociatedEventData` data type is described in 6.12.3.

appliedTariffTableVersion may optionally be used to indicate the version of the tariff table used to compute the fee.

appliedLocalVehicleClass shall be used to indicate the local vehicle class used to compute the fee.

appliedTimeClass shall be used to indicate the time class used to compute the fee.

appliedLocationClass shall be used to indicate the time class used to compute the fee.

appliedUserClass may optionally be used to indicate the user class used to compute the fee.

VehicleDescription may optionally be used to specify the vehicle's parameters.

usageFee, **usageExternalCosts** and **usageFeeQualifier,** may optionally be used to indicate the amount to be paid and to qualify that amount, as described in **forSectionedRoads**.

Table 90 — forStayingInArea fields

Field name	Data type/data description	m/o
areaDisplayName	`UTF8String`	m
usedArea	Selection (ASN.1 CHOICE) among the following fields **usedAreaId**, **usedAreaObject**	m
entranceTime	`GeneralizedTime`	o

Table 90 *(continued)*

Field name	Data type/data description	m/o
stayedDuration	`Duration`	m
associatedEventData	Record (ASN.1 SEQUENCE OF) of fields of type `AssociatedEventData`	o
appliedTariffTableVersion	`Identifier defined as VersionId`	o
appliedLocalVehicleClass	`AppliedLocalVehicleClass`	m
appliedTimeClass	`AppliedTimeClass`	m
appliedUserClass	`AppliedUserClass`	m
vehicleDescription	`VehicleDescription`	o
usageFee	`PaymentAmount`	o
usageFeeQualifier	`FeeQualifier`	o
usageExternalCosts	`ExternalCosts`	o

areaDisplayName shall be used to indicate textually the name of the area.

usedArea shall be used to identify the used area. The area may be either identified with a field of type `AreaId`, which is specified in the field **layoutDescription** of `TollContextPartitionLayout` data type, or as a direct specification, by using a field of type `AreaLayout`.

entranceTime may optionally be used to specify the time of entrance into the area.

stayedDuration shall indicate the time of staying in the area in terms of quantity and unit, used for charging, as a `Duration` type.

associatedEventData may optionally be used to associate a series of events to the used section. The `AssociatedEventData` data type is described in 6.12.3.

appliedTariffTableVersion may optionally be used to indicate the version of the tariff table used to compute the fee.

appliedLocalVehicleClass shall be used to indicate the local vehicle class used to compute the fee.

appliedTimeClass shall be used to indicate the time class used to compute the fee.

appliedLocationClass shall be used to indicate the time class used to compute the fee.

appliedUserClass shall be used to indicate the user class used to compute the fee.

VehicleDescription may be optionally used to specify the vehicle's parameters.

usageFee, **usageExternalCosts** and **usageFeeQualifier,** may optionally be used to indicate the amount to be paid and to qualify that amount, as described in **forSectionedRoads.**

Table 91 — forCordonCrossing fields

Field name	Data type/data description	m/o
chargeObject	Selection (ASN.1 CHOICE) between **entryLocation** and **exitLocation**	m
entryTime	`GeneralizedTime`	o
exitTime	`GeneralizedTime`	m
associatedEventData	Record (ASN.1 SEQUENCE OF) fields of type `AssociatedEventData`	o
appliedTariffTableVersion	`VersionId`	o
appliedLocalVehicleClass	`AppliedLocalVehicleClass`	m
appliedTimeClass	`AppliedTimeClass`	m
appliedEntranceLocationClass	`AppliedLocationClass`	m

Table 91 *(continued)*

Field name	Data type/data description	m/o
appliedExitLocationClass	`AppliedLocationClass`	m
appliedUserClass	`AppliedUserClass`	m
vehicleDescription	`VehicleDescription`	o
usageFee	`PaymentAmount`	o
usageFeeQualifier	`FeeQualifier`	o
usageExternalCosts	`ExternalCosts`	o

chargeObject shall be used to specify the criteria by which the cordon crossing has been detected. It is a selection between:

— **entryLocation,** which is a selection between:

  — **entryChargeObject**, defined as `ChargeObjectId`, to reference the CordonPricingLayout defined in the `TollContextPartitionLayout`;

  — **entryLocationObject**, defined as `CordonEntryLocation`, to directly reference the traversed cordon.

— **exitLocation,** which is a selection between:

  — **exitChargeObject**, defined as `ChargeObjectId`, to reference the CordonPricingLayout defined in the TollContextPartitionLayout;

  — **exitLocationObject**, defined as `CordonEntryLocation`, to directly reference the traversed cordon.

entryTime may optionally be used to specify the time of entrance into the area.

exitTime shall be used to indicate the time of exit from the cordon.

associatedEventData may optionally be used to associate a series of events to the used section. The `AssociatedEventData` data type is described in 6.12.3.

appliedTariffTableVersion may optionally be used to indicate the version of the tariff table used to compute the fee.

appliedLocalVehicleClass shall be used to indicate the local vehicle class used to compute the fee.

appliedTimeClass shall be used to indicate the time class used to compute the fee.

appliedEntranceLocationClass shall be used to indicate the entrance location class used to compute the fee.

appliedExitLocationClass shall be used to indicate the exit location class used to compute the fee.

appliedUserClass shall be used to indicate the user class used to compute the fee.

vehicleDescription may optionally be used to specify the vehicle's parameters.

usageFee, **usageExternalCosts** and **usageFeeQualifier,** may optionally be used to indicate the amount to be paid and to qualify that amount, as described in **forSectionedRoads**.

Table 92 — freeTextDetail fields

Field name	Data type/data description	m/o
textLanguage	`UTF8String`	m
textDetail	`UTF8String`	m

Table 92 *(continued)*

Field name	Data type/data description	m/o
associatedEventData	Record (ASN.1 SEQUENCE OF) of fields of type `AssociatedEventData`	o
usageFee	`PaymentAmount`	o
usageFeeQualifier	`FeeQualifier`	o
usageExternalCosts	`ExternalCosts`	o

textLanguage shall indicate the language used for the text as two letter language code according to ISO 639-1.

textDetail shall specify the reason for tolling as a character string.

associatedEventData may optionally be used to associate a series of events to the used section. The `AssociatedEventData` data type is described in 6.12.3.

usageFee, **usageExternalCosts** and **usageFeeQualifier**, may optionally be used to indicate the amount to be paid and to qualify that amount, as described in **forSectionedRoads**.

Table 93 — forMeshedSystem fields

Field name	Data type/data description	m/o
travelsList	List (ASN.1 SEQUENCE OF) of records of type `Travel`	o
checkPointsCrossingList	List (ASN.1 SEQUENCE OF) of records of type `CheckPointCrossing`	o
invoiceAggregationNumber	`UTF8String`	o

The fields of forMeshedSystem are all optional, but at least one of **travelsList** or **checkPointsCrossingList** shall be specified.

travelsList may optionally list the trips carried out by the user. The `Travel` data type is described in Table 94 and is formally defined in Annex A.

checkPointsCrossingList may optionally list the checkpoints crossed by the user. The `CheckPointsCrossing` data type is described in Table 95 and is formally defined in Annex A.

invoiceAggregationNumber may optionally be used to qualify the usage entry for later aggregation.

Table 94 — Travel fields

Field name	Data type/data description	m/o
travelDescription	Choice (ASN.1 CHOICE) among different styles of description of the travel, which shall be one of: **actualPathByPhysicalEdges**, **actualPathByPhysicalVertices**, **actualPathByLogicalEdges**, **actualPathByLogicalVertices**, and **predefinedPath**	m
fee	`PaymentAmount`	o
feeQualifier	`FeeQualifier`	o

travelDescription shall be used to describe the user trip to be billed. It is a selection of one of the following:

— **actualPathByPhysicalEdges**, which is a record of (ASN.1 SEQUENCE OF) of records (ASN.1 SEQUENCE), each made of the following fields:

  — **edgeID**, which shall identify one physical edge traversed in the trip. Physical edges are described in 6.7.3.2.

- **event**, which may optionally be used to associate an event, defined as `EventIdentification` data type, to the traversal of the physical edge. The `EventIdentification` data type is defined as a record (ASN.1 SEQUENCE) made of two optional fields, of which at least one shall be specified:
  - **time,** of type `GeneralizedTime`, that indicates the time the event occurred.
  - **associatedEventData,** of type `AssociatedEventData`, described in 6.12.3.

- **actualPathByPhysicalVertices**, which is a record of (ASN.1 SEQUENCE OF) records (ASN.1 SEQUENCE), each made of the following fields:
  - **vertexID**, which shall identify one physical vertex traversed in the trip. Physical vertices are described in 6.7.3.2.
  - **event**, which may optionally be used to associate an event, defined as `EventIdentification` data type, to the traversal of the physical edge. The `EventIdentification` data type is defined as a record (ASN.1 SEQUENCE) made of two optional fields, of which at least one shall be specified:
    - **time,** of type `GeneralizedTime`, that indicates the time the event occurred.
    - **associatedEventData,** of type `AssociatedEventData`, described in 6.12.3.

- **actualPathByLogicalEdges**, which is a record of (ASN.1 SEQUENCE OF) records (ASN.1 SEQUENCE), each made of the following fields:
  - **edgeID**, which shall identify one logical edge traversed in the trip. Logical edges are described in 6.7.3.2.
  - **event**, which may optionally be used to associate an event, defined as `EventIdentification` data type, to the traversal of the physical edge. The `EventIdentification` data type is defined as a record (ASN.1 SEQUENCE) made of two optional fields, of which at least one shall be specified:
    - **time,** of type `GeneralizedTime`, that indicates the time at which the event occurred.
    - **associatedEventData,** of type `AssociatedEventData`, described in 6.12.3.

- **actualPathByLogicalVertices**, which is a record of (ASN.1 SEQUENCE OF) records (ASN.1 SEQUENCE), each made of the following fields:
  - **vertexID**, which shall identify one logical vertex traversed in the trip. Logical vertices are described in 6.7.3.2.
  - **event**, which may optionally be used to associate an event, defined as `EventIdentification` data type, to the traversal of the physical edge. The `EventIdentification` data type is defined as a record (ASN.1 SEQUENCE) made of two optional fields, of which at least one shall be specified:
    - **time,** of type `GeneralizedTime`, that indicates the time the event occurred.
    - **associatedEventData,** of type `AssociatedEventData`, described in 6.12.3.

- **predefinedPath**, which is a record of (ASN.1 SEQUENCE OF) of the following fields:
  - **predefinedPathId**, which shall identify one predefined path in the toll domain. Predefined paths are described in 6.7.3.2.
  - **entryEvent**, of type `EventIdentification`, which may optionally be used to associate an event description to the beginning of the predefined path.
  - **intermediateEvents**, which may optionally identify other events in the course of the trip. It is a record (ASN.1 SEQUENCE OF) of records (ASN.1 SEQUENCE) made of:
    - **intermediatePointId**, which is of type `INTEGER` defined as `IntermediatePointId`, which shall identify one intermediate checkpoint.

— **eventIdentification**, of type `EventIdentification`.

— **exitEvent,** of type `EventIdentification`, which may optionally be used to associate an event description to the end of the predefined path.

fee and **feeQualifier** may optionally be used to indicate the amount to be paid and to qualify that amount, as described for the fields **usageFee** and **usageFeeQualifier** in **forSectionedRoads**.

Table 95 — CheckPointCrossing fields

Field name	Data type/data description	m/o
pathByCheckPoints	List (ASN.1 SEQUENCE OF) checkpoints, defined as records (ASN.1 SEQUENCE) made of the following fields: **direction**, **checkpoint**, **eventIdentification**, and **fee**	m
aggregatedFee	`PaymentAmount`	o
feeQualifier	`FeeQualifier`	o

pathByCheckPoints is a list of checkpoints which shall be used to identify the path to be paid for. Each checkpoint is defined as a record (ASN.1 SEQUENCE) made of:

— **direction**, of type `INTEGER`, which shall be used to specify the direction in the highway. Start and end points of the highway are specified by means of the **roadConventionalDirection** field of the `HighWay` data type. Two values may be used for the field direction:

  — **997**, indicating from start to end of the highway.

  — **998**, indicating from end to start of the highway.

— **checkpoint**, of type `INTEGER` ranging from 0 to 2^{63}-1 defined as `IntermediatePointId`, which shall be used to identify the checkpoint.

— **eventIdentification**, of type `EventIdentification`, which may optionally be used to identify a tolling event associated with the traversal of the checkpoint.

— **fee**, of type `PaymentAmount`, which may optionally be used to associate a fee to the checkpoint.

Aggregatedfee and **feeQualifier** may optionally be used to indicate the amount to be paid and to qualify that amount, as described for the fields **usageFee** and **usageFeeQualifier** in **forSectionedRoads**.

6.12.3 AssociatedEventData data type

The data type `AssociatedEventData` can optionally be used to include raw event data that originated the billing details and may optionally include:

— **cccRecord**, when event data come from a CCC transaction (ISO 12813),

— **imageRecord**, when event data are shown in an image,

— **anprRecord**, when event data contain results from an automatic number plate recognition,

— **classificationRecord**, when event data contain a classification record for the vehicle,

— **operatorRecord**, when event data are generated by a human intervention, and

— **dsrcData**, when event data come from a generic DSRC interaction.

If the data type is used, at least one of these subfields shall not be null.

The structure of the `AssociatedEventData` type is shown in Table 96 and is formally defined in Annex A.

Table 96 — AssociatedEventData fields

Field name	Data type/data description	m/o
cccRecord	`CccEvent`. See Table 97 for details	o
imageRecord	`ImageRecord`. See Table 98 for details	o
anprRecord	`AnprRecord`. See Table 99 for details	o
classificationRecord	`ClassificationRecord`. See Table 100 for details	o
operatorRecord	`OperatorRecord`. See Table 101 for details	o
dsrcData	`DsrcData`. See Table 103 for details	o

`CccEvent` is a record (ASN.1 SEQUENCE) whose fields are indicated in Table 97. The `CccEvent` data type is formally defined in Annex A.

Table 97 — CccEvent fields

Field name	Data type/data description	m/o
userId	`UserId`	o
timeOfEvent	`GeneralizedTime`	o
locationOfEvent	Record (ASN.1 SEQUENCE) of the following fields: **positionOfLocation**, **location**	o
cccMessages	Record of fields of the same type (ASN.1 SEQUENCE OF), each field containing DSRC transaction data as `DsrcData{CccContainer}` type	m
initiatedActions	Record of fields of the same type (ASN.1 SEQUENCE OF), each field of type INTEGER defined as `InitiatedActions` type.	m

timeOfEvent can optionally be used to indicate the date and time of the **cccRecord**.

locationOfEvent can optionally be used to indicate the TC coding of the location of where the **cccRecord** was generated.

cccMessages shall contain the raw DSRC data for the **cccRecord**.

initiatedActions shall contain a record (ASN.1 SEQUENCE) of actions that have been initiated by the TC as a result of the **cccRecord**. The following values are allowed:

— **vehicleWasStopped**, vehicle stopped following the CCC result.

— **violationCaseIndicated**, CCC result indicated that a violation may have occurred.

— **evidenceDataGathered**, CCC data has been retained as evidence of violation.

— **putOnTSPExceptionList**, request to add UserID to the TSP exception list made.

— **putOnTCExceptionList**, UserId added to the TC exception list.

`ImageRecord` is a record (ASN.1 SEQUENCE) of the fields **imageToBeSigned** and **subRecordAuthenticator**, where **subRecordAuthenticator**, of type `AuthenticatorEfc`, may optionally be used to provide an authenticator calculated over the content of the **imageToBeSigned** field. The **imageToBeSigned** is a record (ASN.1 SEQUENCE), whose fields are indicated in Table 98. The `ImageRecord` data type is formally defined in Annex A.

Table 98 — imageToBeSigned fields

Field name	Data type/data description	m/o
imageRecordId	`RecordId` (see Table 102)	m
imageDateTime	GeneralizedTime type	m
imageCameraId	`UTF8String`	m
imageReference	`UTF8String`	o

Table 98 *(continued)*

Field name	Data type/data description	m/o
imageData	`OCTET STRING`	o

imageRecordID shall contain the unique TC identifier for the ImageRecord.

imageDateTime shall contain the date and time when the image record was generated.

imageCameraId shall contain the TC identifier of the camera which generated the ImageRecord.

imageReference may optionally be used to specify the TC unique identifier for the image.

imageData may optionally be used to provide the JPEG binary encoded image.

`AnprRecord` data type is a record (ASN.1 SEQUENCE) of the fields **anprToBeSigned** and **subRecordAuthenticator**, where **subRecordAuthenticator**, of type `AuthenticatorEfc`, may optionally be used to provide an authenticator calculated over the content of the **anprToBeSigned** field. The **anprToBeSigned** field is a record (ASN.1 SEQUENCE), whose fields are indicated in Table 99. The `AnprRecord` data structure is formally defined in Annex A.

Table 99 — anprToBeSigned fields

Field name	Data type/data description	m/o
anprRecordId	`RecordId` (see Table 102)	m
associatedImages	Series of records (ASN.1 SEQUENCE OF SEQUENCE), each record made of the following fields: **anprImage**, and **contextImage**	m
imageDateTime	`GeneralizedTime`	m
imageCameraId	`UTF8String`	m
determinedVRM	Record (ASN.1 SEQUENCE) made of the following fields: **anprResult**, **anprConfidence**, **secondaryAnprResult**, **manualResult**, **operatorId**	m
vehicleDetails	Record (ASN.1 SEQUENCE) made of the following fields: **vehicleMake**, **vehicleModel**, **vehicleColour**	m

anprRecordId shall contain the unique TC identifier for the anprRecord.

associatedImages shall contain the references to the associated images from which the **anprRecord** was generated.

imageDateTime shall contain the date and time at which the image record was generated.

imageCameraId shall contain the TC identifier of the camera which generated the ImageRecord.

determinedVRM shall contain the ANPR result(s).

vehicleDetails shall contain the determined vehicle details if available.

`ClassificationRecord` is a record (ASN.1 SEQUENCE) of the fields **classificationToBeSigned** and **subRecordAuthenticator**, where **subRecordAuthenticator**, of type `AuthenticatorEfc`, may optionally be used to provide an authenticator calculated over the content of the **classificationToBeSigned** field. The **classificationToBeSigned** field is a record (ASN.1 SEQUENCE), whose fields are indicated in Table 100. The `ClassificationRecord data` type is formally defined in Annex A.

Table 100 — ClassificationRecord fields

Field name	Data type/data description	m/o
classificationRecordId	`RecordId` (see Table 102)	m
derivedLocalVehicleClass	VehicleClass	o
nominalVehicleParameters	`NominalVehicleParameters`	o

Table 100 *(continued)*

Field name	Data type/data description	m/o
ordinalVehicleParameters	`OrdinalVehicleParameters`	o

classificationRecordID shall contain the unique TC identifier for the classificationRecord.

derivedLocalVehicleClass may optionally be used to indicate the TC derived vehicle class.

nominalVehicleParameters may optionally be used to indicate the nominal vehicle parameters.

ordinalVehicleParameters may optionally be used to indicate the ordinal vehicle parameters.

`OperatorRecord` is a record (ASN.1 SEQUENCE) of the fields **operatorToBeSigned** and **subRecordAuthenticator**, where **subRecordAuthenticator**, of type `AuthenticatorEfc`, may optionally be used to provide an authenticator calculated over the content of the **operatorToBeSigned** field. The **operatorToBeSigned** field is a record (ASN.1 SEQUENCE), whose fields are indicated in Table 101. The `OperatorRecord` data type is formally defined in Annex A.

Table 101 — operatorRecord fields

Field name	Data type	m/o
operatorRecordId	`RecordId` (see Table 102)	m
operatorData	Record (ASN.1 SEQUENCE) made of the following fields: **operatorTime**, **vehicleClass**, **operatorId**, **operatorVrm**, **operatorPan**, **operatorObuId**, **machineReadPan**, **machineReadObuId**, **operatorIccId**	m

operatorRecordID shall provide the unique TC identifier for the operatorRecord.

operatorData shall provide the information collected and recorded by the human operator. It is a record made of the following fields:

— **operatorTime**, of type `GeneralizedTime`, which contains a time stamp;

— **vehicleClass**, of type `vehicleClass`, which may optionally be used to specify the class of the vehicle according to the operator;

— **operatorId**, of type `INTEGER` ranging from 0 to 2^{16}-1, which may optionally be used to specify the identifier of the operator;

— **operatorVrm**, of type `OCTET STRING`, which may optionally be used to specify the vehicle registration mark (VRM), as collected by the operator;

— **operatorPan**, of type `UTF8String`, which may optionally be used to specify the PAN, as collected by the operator;

— **operatorObuId**, of type `UTF8String`, which may optionally be used to specify the OBU identifier, as read by the operator;

— **machineReadPan**, of type `UTF8String`, which may optionally be used to specify the PAN, as read automatically;

— **machineReadObuId**, of type `UTF8String`, which may optionally be used to specify the OBU identifier, as read automatically.

— **operatorIccId**, of type `EquipmentIccId`, which may optionally be used to specify the ICC identifier, as read by the operator.

Table 102 — RecordId fields

Field name	Data type	m/o
providerId	`Provider`	o
recordType	`RecordType`	o
uniqueId	`INTEGER` ranging from 0 to 2^{63}-1	m

providerID may optionally be used to identify the TC for the operatorRecord. The TC is identified by the **apduOriginator** if this field is not specified.

recordType may optionally be used to specify the type of the record. The data type RecordType may have the following values:

— **cccRecord**;

— **imageRecord**;

— **anprRecord**;

— **classificationRecord**;

— **operatorRecord**;

— **dsrcData**.

uniqueId shall specifiy the unique identifier of a record.

`DsrcData` is a record (ASN.1 SEQUENCE) whose fields are indicated in Table 103. The `DsrcData` data type is formally defined in Annex A. If used, at least one field shall be not null.

Table 103 — DsrcData fields

Field name	Data type/data description	m/o
dsrcRseData	`DsrcRseData`	o
dsrcAttributesRead	Record made of fields of the same type (ASN.1 SEQUENCE OF), that are of the `AttributeList{Container}` type	o
dsrcAttributesWritten	Record made of fields of the same type (ASN.1 SEQUENCE OF), that are of the `AttributeList{Container}` type	o
dsrcAttrAuth	Series of records (ASN.1 SEQUENCE OF SEQUENCE), each record of type `DsrcAttrAuth`	o

dsrcRseData field may optionally be used to specify the data recorded by the RSE during the DSRC transaction. It can also optionally link to an exceptionListEntry related to the user. The DsrcRseData data type is defined as a record (ASN.1 SEQUENCE) of the following fields:

— **transactionId**, of type `INTEGER` ranging from 0 to 2^{63}-1, which shall identify the DSRC transaction;

— **transactionDescription**, of type `UTF8String`, which may optionally be used to textually describe the transaction;

— **rseDateTime**, of type `GeneralizedTime`, which shall specify the RSE time of the transaction;

— **transactionResult**, of type `ResultOp`, which shall specify the result of the transaction;

— **transactionStatus**, of type `TransactionStatus`, which shall specify the status of the transaction. TransactionStatus may take the following values:

  — **completed**, when the transaction was correctly completed;

  — **abortedAfterFirstPhase**, when the transaction was aborted after its first phase;

  - **abortedAfterSecondPhase**, when the transaction was aborted after its second phase.

- **exceptionListMatch**, which is a list of records (ASN.1 SEQUENCE OF SEQUENCE) which may optionally be used to couple the event with list exception list items. The fields of the exceptionListMatch are:
  - **originTSP**, of type `Provider`, which shall be used to specify the TSP from which the exception list entry originates;
  - **matchVersion**, of type `ExceptionListVersion`, which shall be used to specify the version of the referenced exception list;
  - **matchType**, of type `ExceptionListType`, which shall be used to indicate the type of exception list being referred to;
  - **matchEntry**, of type `ExceptionListEntry`, which may optionally be used to bring the whole exception list entry being referred to.
- **tariffId**, of type `TariffClassId`, which may optionally be used to specify the tariff class;
- **fee** and **feeQualifier** may optionally be used to indicate the fee and fee qualifier;
- **tollStationId**, of type `ChargeObjectId`, may optionally be used to identify the RSE as a charge object;
- **sessionLocation**, of type `SessionLocation`, may optionally be used to indicate where the DSRC transaction happened;
- **typeOfTransaction** of type `StationType`, may optionally be used to specify the type of tolling station;
- **vstData** of type `ObeConfiguration`, may optionally be used to report the OBE data as received by the RSE in the DSRC transaction.

dsrcAttributesRead may optionally be used to specify the details of the attributes read from the OBE during the DSRC transaction.

dsrcAttributesWritten may optionally be used to specify the details of the attributes written to the OBE during the DSRC transaction.

dsrcAttrAuth may optionally be used to specify the information that is required to verify the authenticators received during the DSRC transaction. It is a series of records, one for each DSRC transaction. Each record is made of the following fields:

- **attrOrigEncoding** of type `BIT STRING` shall be used to provide the encoding of the AttributeList;
- **rndRSE** of type `OCTET STRING` shall be used to provide the random number generated by the RSE for encoding data;
- **keyRef** of type INTEGER ranging from 0 to 255 shall be used to provide the reference to the key used for encoding data;
- **authCode** of type `OCTET STRING` shall be used to provide the authentication code;
- **result** of type `AuthCheckResult` shall be used to provide the results of the authentication. The `AuthCheckResult` data type may take the following values:
  - **notChecked**, authentication was not checked;
  - **ok**, the authentication was positively verified;
  - **nok**, the authentication was not verified.

6.13 PaymentClaimAdu data structure

The `PaymentClaimAdu` data type can be used by the TC either in response to the request ADU sent by the TSP or as a push ADU when new payment claims are available.

When transferring this ADU, the value in the informationRecipientID field of the APCI shall identify a valid entity (a null value is not permitted).

Table 104 indicates the fields of the PaymentClaimAdu data structure, which is formally defined in Annex A.

Table 104 — PaymentClaimAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
startDateTime	`GeneralizedTime`	m
endDateTime	`GeneralizedTime`	o
userId	`UserId`	o
paymentClaimAmount	`PaymentAmount`	m
paymentClaimStatus	`PaymentClaimStatus`	m
typeOfFee	`TypeOfFee`	o
referenceDetailsList	List (ASN.1 SEQUENCE OF) of records of type `ReferenceDetail`	o
paymentReference	`UTF8String`	o
paymentClaimDetails	List (ASN.1 SEQUENCE OF) of records of type `PaymentClaimDetail`	o
paymentClaimReference	`UTF8String`	o
discountId	`DiscountId`	o
discountReference	`UTF8String`	o
actionCode	`ActionCode`	o

aduIdentifier shall contain the unique identifier for the `PaymentClaimAdu` (see 6.2.4).

startDateTime shall specify the start of the period which the payment claim relates to.

endDateTime may optionally be used to specify the end of the period for which the payment claim relates. If not used, then the end time shall be assumed as being the time when the ADU was generated.

userId may optionally be used to identify a specific user for whom the payment is claimed. See 6.2.6 for details.

paymentClaimAmount shall contain the details of the net amount, currency and VAT for the payment claim.

paymentClaimStatus shall indicate the version and status of the payment claim. The `PaymentClaimStatus` data type may take the following values:

— **firstVersion**, which indicates the first version of the payment claim;

— **amendedVersion**, which indicates an amended version of an already issued payment claim.

typeOfFee may optionally be used to indicate the type of the fee. The `TypeOfFee` data type may take the following values:

— **toll**, to indicate a toll;

— **discount,** to indicate a discount;

— **creditnote,** to indicate a credit note;

— **penalty**, to indicate a penalty;

— **processingfee,** to indicate processing fee.

referenceDetailsList may optionally be used to associate previously exchanged toll declarations, billing details or toll events to the payment claim by providing a list (ASN.1 SEQUENCE) of selected (ASN.1 CHOICE) identifiers, which may be one of:

— **referencedbillingDetailsInfo**, of type `BillingDetailsInfo`, to identify related billing details (see 6.12 for description of the type);

— **referencedTollDeclarationId**, of type `TollDeclarationId`, to identify a toll declaration (see 6.11 for description of the type);

— **referencedTollEventId**, of type `TollEventId`, to identify a tolling event (see 6.12.2 for description of the type);

— **referencedAduIdentifier**, which is a record (ASN.1 SEQUENCE) made of two fields of types `AduIdentifier` and `AduType`, to identify any ADU.

paymentReference may optionally be used to specify a reference to this payment claim in the form of a character string.

paymentClaimDetails is a list (ASN.1 SEQUENCE OF) of fields of type PaymentClaimDetail which may optionally be used to specify detailed information about the payment claim (see Table 105).

paymentClaimReference may optionally be used to uniquely identify a payment claim by using a text string of maximum 16 characters.

discountId, of type `DiscountId`, may optionally be used to identify the applied discount.

discountReference may optionally be used to reference the applied commercial discount by means of a text string.

actionCode may optionally be used to indicate the action associated with the specific ADU (see 6.2.5).

Table 105 — PaymentClaimDetail fields

Field name	Data type/data description	m/o
paymentDetailCode	`Int2Unsigned`	m
paymentDetailNumber	`Int2Unsigned`	o
paymentDetailType	`UTF8String`	o
paymentDetailText	`UTF8String`	o
paymentDetailQuantity	`Int8Signed`	o
paymentDetailRate	`Int8Signed`	o
paymentDetailBasicAmount	`PaymentAmount`	o
paymentDetailAmountVATExcl	`PaymentAmount`	o
paymentDetailAmountTotal	`PaymentAmount`	o

paymentDetailCode shall be used to identify the usage and edition parameters of the detail in the invoice.

paymentDetailNumber may optionally be used to identify the number of the line containing the detail in the invoice.

paymentDetailType may optionally be used to identify the content of the detail in the invoice.

paymentDetailText may optionally be used to specify the text to be printed on the invoice for this detail.

paymentDetailQuantity may optionally be used to specify the quantity associated to the detail in the invoice.

paymentDetailRate may optionally be used to specify the discount or surcharge rate applied to the detail.

paymentDetailBasicAmount may optionally be used to specify the basic amount without VAT before discount or surcharge is applied.

paymentDetailAmountVATExcl may optionally be used to specify the amount applied after discount or surcharge without the corresponding VAT rate.

paymentDetailAmountTotal may optionally be used to specify the total amount of the detail including VAT.

6.14 PaymentAnnouncement Adu data structure

The `PaymentAnnouncementAdu` data type can be used by the TSP either in response to the request ADU sent by the TC or as a push ADU when new payment announcements are available.

When transferring this ADU, the value in the informationRecipientID field of the APCI shall identify a valid entity (a null value is not permitted).

Table 106 indicates the fields of the `PaymentAnnouncementAdu` data type, which is formally defined in Annex A.

Table 106 — PaymentAnnouncementAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
dueDate	`GeneralizedTime`	m
totalamount	`PaymentAmount`	o
paymentStatus	`PaymentStatus`	m
numberOfItems	`INTEGER` ranging from 0 to 2^{63}-1	o
referenceDetailsList	List of records (ASN.1 SEQUENCE OF SEQUENCE), each record being made of the following fields: **referenceDetailList**, **amount**, **paymentMeansType**, **valueDate**, **interestAmount**	o
attachment	`OCTET STRING`	o
paymentReference	`UTF8String`	o
actionCode	`ActionCode`	o

aduIdentifier shall contain a unique identifier for the `PaymentAnnouncementAdu` (see 6.2.4).

dueDate shall provide the date and time when the amount has been actually paid to the TC.

totalamount may optionally be used to indicate the amount paid.

paymentStatus may optionally be used to provide detailed information regarding the payment. The following values are specified for the `PaymentStatus` data type:

— **paid:** the communicated **totalamount** has been paid on or before the due date by the TSP without any interest or after the due date including all applicable interest;

— **newOverdue:** the communicated **totalamount** was not yet paid by the TSP and interests are accumulating until payment;

— **notYetDue**: the communicated **totalamount** is not yet due and has not yet been paid;

— **due**: the communicated **totalamount** is due on this date but has not been paid yet.

numberOfItems may optionally be used to indicate the number of referenced items in the referenceDetailsList.

referenceDetailsList may optionally be used to specify references to billing details, toll declarations and individual usage statements with additional information about the payment announcement data. The structure of **referenceDetailsList** is specified in Table 107.

attachment may optionally be used to attach supporting documents to the payment announcement data, e.g. a PDF version of a payment report.

paymentReference may optionally be used to specify a reference for this payment announcement.

actionCode may optionally be used to indicate the action associated with the reception of this specific ADU (see 6.2.5).

Table 107 — referenceDetailsList fields

Field name	Data type/data description	m/o
referenceDetail	List (ASN.1 SEQUENCE) of records of type `ReferenceDetail`	m
amount	`PaymentAmount`	m
paymentMeansType	`PaymentMeansType`	o
valueDate	`GeneralizedTime`	o
interestAmount	`PaymentAmount`	o

referenceDetail shall be used to specify additional information for the payment announcement by referring to previously exchanged ADU's or tolling events. The `ReferenceDetail` data type is a selection (ASN.1 CHOICE) of various structures, which are described in Table 108 and formally defined in Annex A.

amount shall contain the amount paid for the announced referenced details.

paymentMeansType may optionally be used to indicate the means of payment used. The `PaymentMeansType` data type may take one of the following values:

— **cash**: The payment was made with physical cash (e.g. Euro in a country of the Euro-area);

— **prepaid**: The payment was made through a prepaid account, which can be loaded by various payment means;

— **credit card**: The payment was made with a credit card;

— **fleet card**: The payment was made with a fleet card;

— **advanced payment provider**: The payment was made using a valid subsitute of physical cash by electronic money (e.g. Bitcoin). The acceptance of substitutes of cash money is specified by the corresponding TSP;

— **debit card**: The payment was made using a debit card;

— **bank transfer**: The payment was done through a bank transfer;

— **sepa direct debit**: The payment was done by using the payment method SEPA Direct Debit. The prerequisite of this method is a valid mandate signed by the debtor as the amount of the payment is pulled by the creditor from the bank account of the debtor;

— **unknown**: The used payment means is unknown (e.g. payment service provider provides no information to the TSP regarding the payment means used).

valueDate may optionally be used to specify the date when the amount of the specific reference detail was due according to contractual agreements.

interestAmount may optionally be used to indicate the interest that has to be paid because the specific reference detail was overdue. It shall be present if the **paymenStatus** value is **newOverdue**.

Table 108 — ReferenceDetail fields

Field name	Data type/data description	m/o
referencedBillingDetailsInfo	`BillingDetailsInfo`	n/a
referencedTollDeclarationId	`TollDeclarationId`	n/a
referencedTollEventId	`TollEventId`	n/a
referencedAduIdentifier	Identifier of an ADU as a record (ASN.1 SEQUENCE) of two fields of type `AduIdentifier` and `AduType`	n/a

referencedBillingDetailsInfo shall be used to reference one billing detail ADU to which the payment announcement ADU is related. See 6.12 for the explanation of the `BillingDetailsInfo` data type.

referencedTollDeclarationId shall be used to reference one toll declaration to which the payment announcement ADU is related. See 6.11 for explanation of the `TollDeclarationId` data type.

tollEventId shall be used to uniquely identify the usage statement which the toll event is related to. See 6.12.2 for description of the `TollEventId` data type.

referencedAduIdentifier shall be used to uniquely identify the ADU which the payment announcement ADU is related to. The **referencedAduIdentifier** is made of:

— **referencedAduIdentifier** of type `AduIdentifier` that identifies the ADU within its type;

— **referencedADUType** of type `AduType`, that identifies the ADU type. It shall be restricted to either **tollDeclarationADU** or **billingDetailsADU**.

6.15 ProvideUserDetailsAdu data structure

The `ProvideUserDetailsAdu` data type shall only be used by a TSP in response to a specific request ADU from a TC.

When transferring this ADU, the value in the informationRecipientID field of the APCI shall identify a valid entity (a null value is not permitted). Table 109 indicates the fields of the `ProvideUserDetailsAdu` data type, which is formally defined in Annex A.

Table 109 — ProvideUserDetailsAdu fields

Field name	Data type/data description	m/o
aduIdentifier	`AduIdentifier`	m
originalUserIdRequest	`UserId`	m
userId	`UserId`	m
statusFlag	`UserStatus`	o
listOfUserParameters	List (ASN.1 SEQUENCE OF) of records of type `UserParameterResponse`	o
actionCode	`ActionCode`	o

aduIdentifier shall contain a unique identifier for the `ProvideUserDetailsAdu` (see 6.2.4).

originalUserIdRequest shall contain the user identifier specified in the originating request ADU with the same fields and values.

userId shall specify the actual user identifier for which the user details are provided. It shall contain all data elements present in the **originalUserIdRequest** field and may be enhanced with additional missing elements not present in **originalUserIdRequest**.