INTERNATIONAL STANDARD

ISO 14906

Second edition
2011-10-15

Electronic fee collection — Application interface definition for dedicated short-range communication

Perception du télépéage — Définition de l'interface d'application relative aux communications dédiées à courte portée

Reference number
ISO 14906:2011(E)

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

Published in Switzerland

Contents

Page

Foreword

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

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

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 14906 was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*, in collaboration with Technical Committee CEN/TC 278, *Road transport and traffic telematics*.

This second edition cancels and replaces the first edition (ISO 14906:2004), which has been technically revised.

Introduction

This International Standard specifies an application interface for electronic fee collection (EFC) systems, which are based on dedicated short-range communication (DSRC). It supports interoperability between EFC systems on an EFC-DSRC application interface level. This International Standard is intended for DSRC charging applications, but specifically the definition of EFC data elements is valid beyond the use of a DSRC charging interface and might be used for other DSRC applications (e.g. compliance checking communication) and/or on other interfaces (e.g. the application interface of autonomous systems).

This International Standard provides specifications for the EFC transaction model, EFC data elements (referred to as attributes) and functions, from which an EFC transaction can be built. The EFC transaction model provides a mechanism that allows handling of different versions of EFC transactions and associated contracts. A certain EFC transaction supports a certain set of EFC attributes and EFC functions as defined in this International Standard. It is not envisaged that the complete set of EFC attributes and functions be present in each piece of EFC equipment, on-board equipment (OBE) or roadside equipment (RSE).

This International Standard provides the basis for agreements between operators, which are needed to achieve interoperability. Based on the tools specified in this International Standard, interoperability can be reached by operators recognising each others' EFC transactions (including the exchange of security algorithms and keys) and implementing the EFC transactions in each others' RSEs, or they can reach an agreement to define a new transaction (and contract) that is common to both. Considerations should also be made by each operator so that the RSE has sufficient resources to implement such additional EFC transactions.

In order to achieve interoperability, operators should agree on issues such as

— which optional features are actually being implemented and used,

— access rights and ownership of EFC application data in the OBE,

— security policy (including encryption algorithms and key management, if applicable),

— operational issues, such as how many receipts may be stored for privacy reasons, how many receipts are necessary for operational reasons (for example as entry tickets or as proof of payment),

— the agreements needed between operators in order to regulate the handling of different EFC transactions.

In this revision, users are faced with issues related to backward compatibility. This issue can be managed by using the following:

— EfcModule ASN.1 module, including a version number;

— Efc-ContextMark (incl. the ContextVersion), denoting the implementation version, provides a means to ensure co-existence of different implementation versions by means of a look-up table and associated appropriate transaction processing. This will enable the software of the RSE to determine the version of the OBE and his capabilty to accept the new features of this version of this International Standard.

Annex A provides the normative ASN.1 specifications of the used data types (EFC action parameters and attributes).

Annex B presents an informative example of a transaction based on the CARDME specification, including bit-level specification.

Annex C presents informative examples of EFC transaction types, using the specified EFC functions and attributes.

Annex D presents an informative listing of functional requirements, which can be satisfied by using the tools provided by this International Standard.

Annex E presents an informative mapping table from LatinAlphabetNo2 & 5 to LatinAlphabetNo1 to ease for a Service Provider the use of LatinAlphabetNo1 to encode an OBE for data available wiitten with non-Latin1 characters.

Annex F presents an informative mapping table between EFC vehicle data attributes and European registration certificates to ease the task of a service provider when he needs to personalise an OBE by obtaining vehicle data.

This application interface definition can also be used with other DSRC media which do not use a layer 7 according to ISO 15628/EN 12834. Any DSRC medium which provides services to read and write data, to initialise communication and to perform actions is suitable to be used as a basis for this application interface. Adaptations are medium specific and are not further covered here. As Annex B describes in detail a transaction for central account systems, this International Standard can also be used for onboard account systems, in conjunction with ISO/TS 25110, which provides examples of systems based on onboard accounts.

Electronic fee collection — Application interface definition for dedicated short-range communication

1 Scope

This International Standard specifies the application interface in the context of electronic fee collection (EFC) systems using the dedicated short-range communication (DSRC).

The EFC application interface is the EFC application process interface to the DSRC application layer, as can be seen in Figure 1 below. This International Standard comprises specifications of

— EFC attributes (i.e. EFC application information) that can also be used for other applications and/or interfaces,

— the addressing procedures of EFC attributes and (hardware) components (e.g. ICC and MMI),

— EFC application functions, i.e. further qualification of actions by definitions of the concerned services, assignment of associated ActionType values and content and meaning of action parameters,

— the EFC transaction model, which defines the common elements and steps of any EFC transaction,

— the behaviour of the interface so as to ensure interoperability on an EFC-DSRC application interface level.

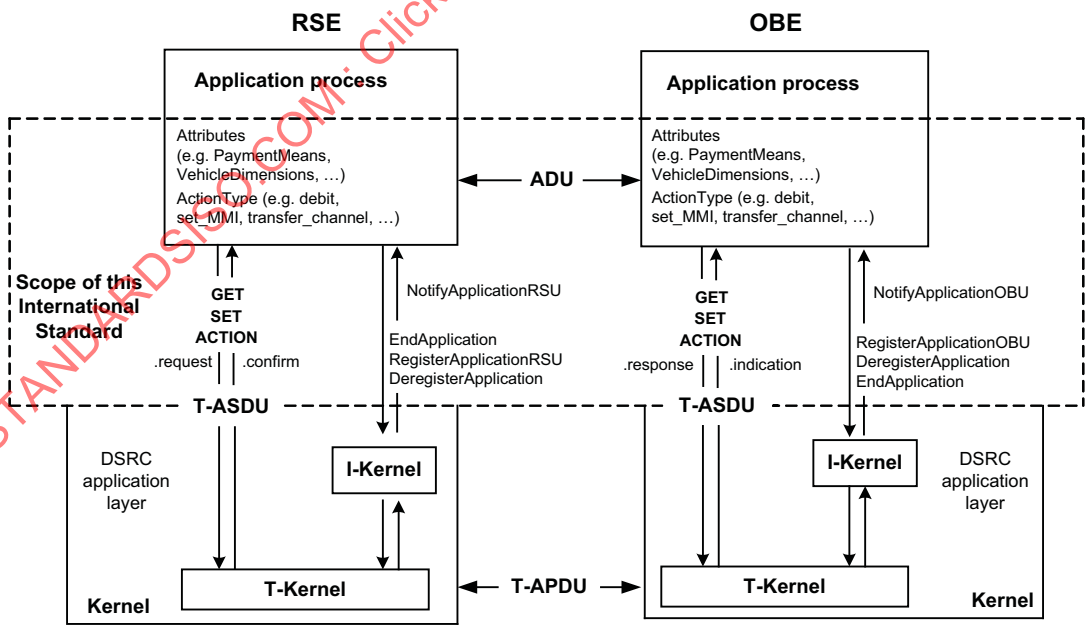

Figure 1 — The EFC application interface

This is an interface standard, adhering to the open systems interconnection (OSI) philosophy (see ISO/IEC 7498-1), and it is as such not concerned with the implementation choices to be realised at either side of the interface.

This International Standard provides security-specific functionality as place holders (data and functions) to enable the implementation of secure EFC transactions. Yet the specification of the security policy (including specific security algorithms and key management) remains at the discretion and under the control of the EFC operator, and hence is outside the scope of this International Standard.

2 Normative references

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

ISO 612, *Road vehicles — Dimensions of motor vehicles and towed vehicles — Terms and definitions*

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 codes*

ISO 3779, *Road vehicles — Vehicle identification number (VIN) — Content and structure*

ISO 4217, *Codes for the representation of currencies and funds*

ISO 7812-1, *Identification cards — Identification of issuers — Part 1: Numbering system*

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

ISO/IEC 8825-2, *Information technology — ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*

ISO 14816:2005, *Road transport and traffic telematics — Automatic vehicle and equipment identification — Numbering and data structure*

ISO 15628:2007, *Road transport and traffic telematics — Dedicated short range communication (DSRC) — DSRC application layer*

EN 12834:2003, *Road transport and traffic telematics — Dedicated Short Range Communication (DSRC) — DSRC application layer*

3 Terms and definitions

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

3.1
access credentials
data that is transferred to on-board equipment (OBE), in order to establish the claimed identity of a roadside equipment (RSE) application process entity

NOTE The access credentials carry information needed to fulfil access conditions in order to perform the operation on the addressed element in the OBE. The access credentials can carry passwords as well as cryptographic based information such as authenticators.

3.2
action
function that an application process resident at the roadside equipment can invoke in order to make the on-board equipment (OBE) execute a specific operation during the transaction

3.3
attribute
application information formed by one or by a sequence of data elements, and that is managed by different actions used for implementation of a transaction

3.4
authenticator
data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and/or the integrity of the data unit and protect against forgery

3.5
channel
information transfer path

[ISO 7498-2:1989, definition 3.3.13]

3.6
component
logical and physical entity composing an on-board equipment, supporting a specific functionality

3.7
contract
expression of an agreement between two or more parties concerning the use of the road infrastructure

3.8
cryptography
discipline which embodies principles, means, and methods for the transformation of data in order to hide its information content, prevent its undetected modification and/or prevent its unauthorised use

[ISO 7498-2:1989, definition 3.3.20]

3.9
data group
collection of closely related EFC data attributes which together describe a distinct part of an EFC transaction

3.10
data integrity
property that data has not been altered or destroyed in an unauthorised manner

[ISO 7498-2:1989, definition 3.3.21]

3.11
element
⟨DSRC⟩ directory containing application information in the form of attributes

3.12
empty list
container for attributeValues (OCTET STRING) with the length equal to zero

3.13
on-board equipment
equipment fitted within or on the outside of a vehicle and used for toll purposes

NOTE The OBE does not need to include payment means.

3.14
on-board unit
minimum component of an on-board equipment, whose functionality always includes at least the support of the DSRC interface

3.15
operator
entity involved in the process outside the user

NOTE An operator is a generic term which can be used for a toll service provider or a toll charger.

3.16
roadside equipment
equipment located along the road transport network, for the purpose of communication and data exchanges with on-board equipment

3.17
service
⟨EFC⟩ road transport related facility provided by a service provider, normally a type of infrastructure, the use of which is offered to the user, for which the user may be requested to pay

3.18
service primitive
⟨communication⟩ elementary communication service provided by the application layer protocol to the application processes

NOTE The invocation of a service primitive by an application process implicitly calls upon and uses services offered by the lower protocol layers.

3.19
session
exchange of information and interaction occurring at a specific EFC station between the roadside equipment and the user/vehicle

3.20
toll charger
legal entity charging toll for vehicles in a toll domain

[ISO/TS 17574:2009, definition 3.27]

3.21
toll domain
area or part of a road network where a toll regime is applied

[ISO 17573:2010, definition 3.18]

3.22
toll service
⟨EFC⟩ service enabling users having only one contract and one set of on-board equipment (OBE) to use a vehicle in one or more toll domains

NOTE Adapted from ISO/TS 12813:2009.

3.23
toll service provider
⟨EFC⟩ legal entity providing to his customers toll services on one or more toll domains for one or more classes of vehicle

NOTE 1 In other documents the terms issuer or contract issuer may be used.

NOTE 2 The toll service provider may provide the OBE or may provide only a magnetic card or a smart card to be used with OBE provided by a third party (like a mobile telephone and a SIM card can be obtained from different parties).

NOTE 3 The toll service provider is responsible for the operation (functioning) of the OBE.

[ISO/TS 17574:2009, definition 3.28]

3.24
transaction
whole of the exchange of information between the roadside equipment and the on-board equipment necessary for the completion of an EFC operation over the DSRC

3.25
transaction model
functional model describing the general structure of electronic payment fee collection transactions

3.26
user
customer of a toll service provider, one liable for toll, the owner of the vehicle, a fleet operator, a driver, etc., depending on the context

4 Abbreviated terms

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

4.1
APDU
Application Protocol Data Unit

4.2
AP
Application Process

4.3
ASN.1
Abstract Syntax Notation One (ISO/IEC 8824-1)

4.4
BST
Beacon Service Table

4.5
CCC
Compliance check communication

4.6
cf
Confirm

4.7
DSRC
Dedicated Short-Range communication

4.8
EID
Element Identifier

4.9
EFC
Electronic Fee Collection

4.10
GPS
Global Positioning System

4.11
ICC
Integrated Circuit(s) Card

4.12
I-Kernel
Initialisation Kernel

4.13
IID
Invoker Identifier

4.14
ind
Indication

4.15
LAC
Localisation Augmentation Communication

4.16
L1
Layer 1 of DSRC (Physical Layer)

4.17
L2
Layer 2 of DSRC (Data Link Layer)

4.18
L7
Application Layer Core of DSRC

4.19
LID
Logical Link Control Identifier

4.20
LLC
Logical Link Control

4.21
LPDU
LLC Protocol Data Unit

4.22
MAC
Medium Access Control

4.23
MMI
Man-Machine Interface

4.24
n.a.
Not applicable

4.25
OBE
On-Board Equipment

4.26
PDU
Protocol Data Unit

4.27
PER
Packed Encoding Rules (ISO/IEC 8825-2)

4.28
req
Request

4.29
rs
Response

4.30
RSE
Roadside Equipment

4.31
RTTT
Road Transport and Traffic Telematics

4.32
SAM
Secure Application Module

4.33
T-APDU
Transfer-Application Protocol Data Unit

4.34
T-ASDU
Transfer-Application Service Data Unit

4.35
T-Kernel
Transfer Kernel

4.36
VST
Vehicle Service Table

5 EFC application interface architecture

5.1 Relation to the DSRC communication architecture

The DSRC services are provided to an application process by means of the DSRC Application Layer service primitives, which are abstract implementation interactions between a communication service user and a provider. The services are offered by the DSRC communication entities by means of its DSRC Application Layer (EN 12834/ISO 15628).

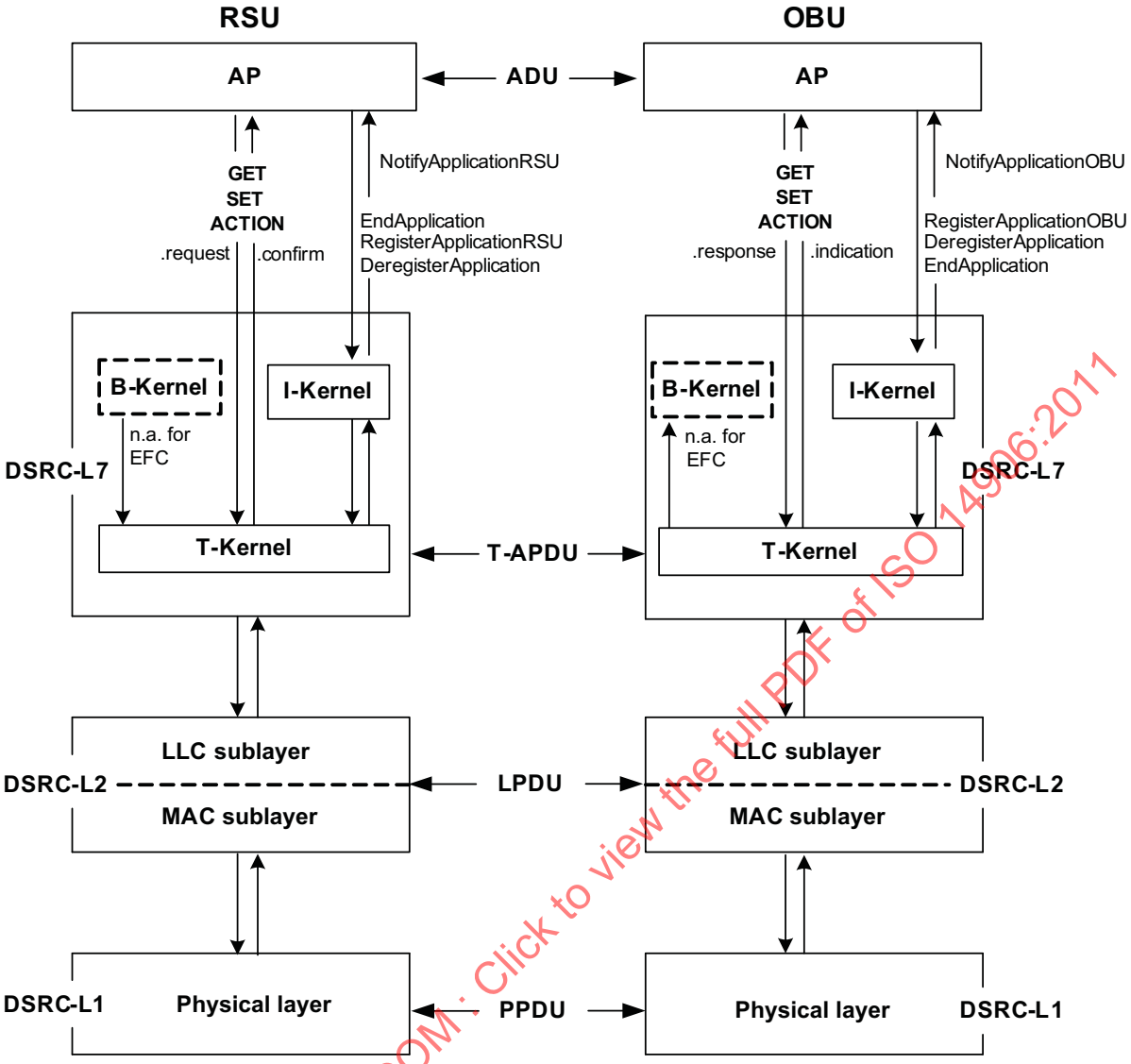

Figure 2 — The EFC application process on top of the DSRC communication stack

NOTE The abbreviations used in Figure 2 are defined in Clause 4.

The Transfer Kernel of DSRC Application Layer offers the following services to application processes (see also Figure 2 above):

— GET: The invocation of a GET service request results in retrieval (i.e. reading) of application information (i.e. Attributes) from the peer service user (i.e. the OBE application process), a reply is always expected.

— SET: The invocation of a SET service request results in modification (i.e. writing) of application information (i.e. Attributes) of the peer service user (i.e. the OBE application process). This service may be requested in confirmed or non-confirmed mode, a reply is only expected in the former case.

— ACTION: The invocation of an ACTION service request results in a performance of an action by the peer service user (i.e. the OBE application process). An action is further qualified by the value of the ActionType. This service may be requested in confirmed or non-confirmed mode, a reply is only expected in the former case.

— EVENT-REPORT: The invocation of an EVENT-REPORT service request forwards a notification of an event to the peer service user.

— INITIALISATION: The invocation of an initialisation service request by RSE results in an attempt to initialise communication between a RSE and each OBE that has not yet established communication with the concerned RSE. The Initialisation service is only used by the Initialisation Kernel as defined in EN 12834/ISO 15628.

5.2 Usage of DSRC application layer by the EFC application interface

EFC uses the following services offered by DSRC Application Layer (as defined in EN 12834/ISO 15628):

— The INITIALISATION services:

  — Notify Application RSU (at RSE);

  — End Application (at RSE);

  — Register Application RSU (at RSE);

  — Deregister Application (at RSE and OBE);

  — Notify Application OBU (at OBE);

  — Register Application OBU (at OBE)

  are used to realise the EFC-specific initialisation mechanism (see Clause 6);

— The GET service is used to retrieve EFC attributes (For attribute specifications see Clause 8);

— The SET service is used to set EFC attributes;

— The ACTION services are applied to realise additional EFC specific functionality needed to support EFC application processes, such as TRANSFER_CHANNEL, SET_MMI and ECHO (see 7.2).

In the following, the EFC-specific usage of the DSRC Layer 7 services is specified in detail.

NOTE The EVENT-REPORT-service can be implicitly used by EFC application processes. It is e.g. used indirectly as part of an already defined command to release an application process (see EN 12834/ISO 15628, Ready Application). However as the EVENT-REPORT-service is not explicitly used by EFC application processes, this service is not further referred to in this International Standard.

5.3 Addressing of EFC attributes

5.3.1 Basic mechanism

EFC Attributes are used to transfer the EFC application-specific information.

EFC Attributes are composed of one or more data elements of specified ASN.1 types. Each data element is associated with, within the context of this International Standard, an unambiguous name.

To each EFC Attribute, an AttributeID is associated. The AttributeId enables to unambiguously identify and address an EFC Attribute.

EXAMPLE Figure 3 illustrates the basic addressing mechanism.

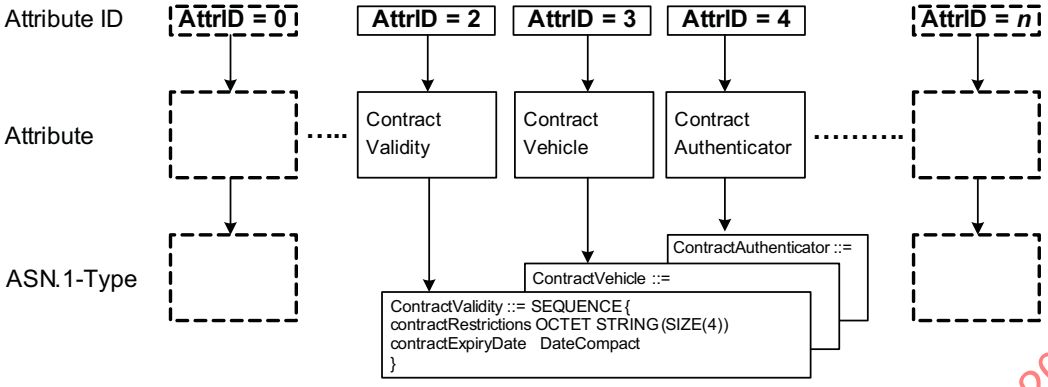

Figure 3 — Basic addressing mechanism

5.3.2 Role of the EID

In a given OBE, the DSRC-EID (different from 0) is used to address an EFC context, identified by the EFC-ContextMark (see 6.2.3), in which Attributes can be addressed unambiguously by AttributeIDs inside an Element of the OBE. In the VST, the OBE specifies one or several of these EFC contexts, each corresponding to an EFC ContextMark and the EID to be used for addressing the attributes and using the EFC functions supported by it.

EXAMPLE

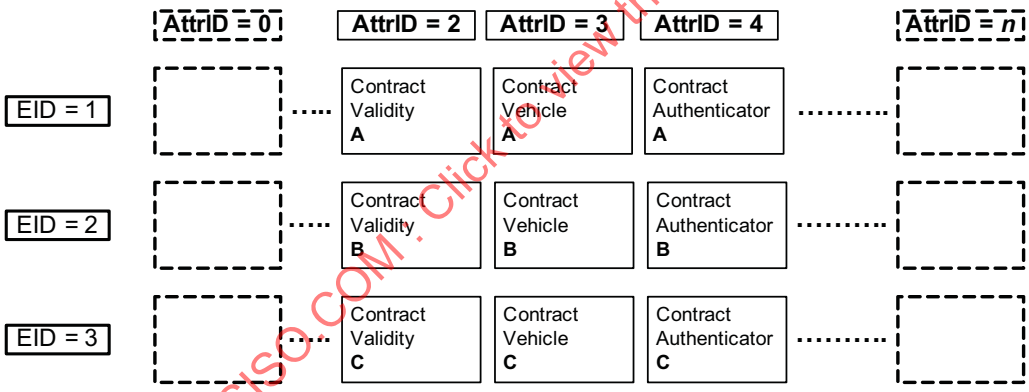

Figure 4 — Role of the EID

EID equals 0 shall be used to address application-independent functions and components, e.g. SET_MMI and TRANSFER_CHANNEL (see 7.2).

5.3.3 Multiple Instances of Attributes

There may be n, where n is an integer, instances of an Attribute available in an OBE.

The maximum number of instances N_{max} of one Attribute may be limited according to the needs of operators and users. The default maximum number of instances is $N_{max}=1$. The value of N_{max} is determined at the time of OBE configuration.

EXAMPLE

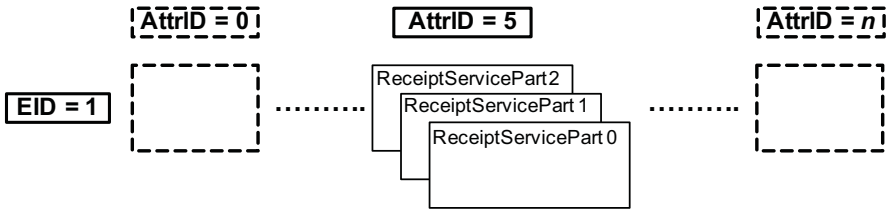

Figure 5 — Multiple instances (0-2) of attribute 5

The handling of multiple instances and the corresponding addressing mechanism are described in detail as part of the behaviour specification of the corresponding functions supporting multiple instances (see 7.2.6 for GET_INSTANCE and 7.2.7 for SET_INSTANCE).

5.4 Addressing of components

Components of an OBE to be addressed via the EFC Application Interface include for example:

— OBU;
— SAM 1;
— SAM 2;
— ICC;
— Display;
— Buzzer;
— Printer;
— Serial interface;
— Parallel interface;
— GPS;
— Tachograph;
— Bluetooth.

Addressing of these components is enabled on two levels, device-specific and device-independent addressing.

The ***device-specific transparent addressing mechanism*** enables the transfer of information, which shall be processed by the addressed device (such as an ICC-command). The addressed device is identified by a *channel Id*. The EFC function TRANSFER_CHANNEL (see 7.2.10) supports this functionality.

EXAMPLE 1 Transfer of a bit string to an ICC.

The ***device-independent addressing mechanism*** uses a set of commands, which describe a certain functionality, which can be performed by various OBE components. In this case, the operating system of the OBE will address the corresponding components. The EFC function SET_MMI supports this functionality (see 7.2.12).

EXAMPLE 2 Invocation of a SET_MMI(EID=0, ContactOperator) function activates an OBE MMI-device, e.g. a buzzer or a display.

NOTE In a specific implementation, specific attributes or data elements may activate some MMI function (e.g. a SET command on the attribute ReceiptText might display the text on an LCD display. A SET command on the attribute ReceiptServicePart with data element SessionResultOperational other than SessionOK might activate an alert beep). Proprietary addressing mechanisms are not defined by this International Standard.

6 EFC Transaction Model

6.1 General

The EFC Transaction Model related to the EFC Application Interface for the DSRC comprises two phases, the initialisation phase and the transaction phase.

NOTE The purpose of the initialisation phase is to set up the communication between the RSE and OBEs that have entered the DSRC zone but have not yet established communication with the RSE, and to notify the application processes. It provides amongst others a multi-application switching mechanism, allowing for execution of several RTTT applications (in parallel) at one RSE station.

The transaction phase can only be reached after completion of the initialisation phase. The EFC functions, as defined in Clause 7, can be performed in the transaction phase. The GET and SET services (DSRC application layer functions) as defined in EN 12834/ISO 15628 (in 6.2) may also be used in an EFC transaction phase.

6.2 Initialisation Phase

6.2.1 Overview

This clause provides an overview of the functionality of, and the information exchanges in, the initialisation phase.

The Initialisation procedures, by means of beacon service table (BST) and vehicle service table (VST) exchanges, are defined in EN 12834/ISO 15628 6.2.2 and 6.2.3 below account for the EFC application-specific information that shall be included in the BST and VST, respectively.

NOTE The OBE evaluates the received BST, and selects the applications that it wishes to perform out of the lists of applications supported by the RSE. If the OBE does not support any of application(s) supported by the RSE, then it is recommended that the OBE does not exchange any information with the RSE. If the OBE supports at least one of the application(s) supported by the RSE, then it is recommended that the OBE informs the RSE of which application it wishes to execute in its corresponding VST.

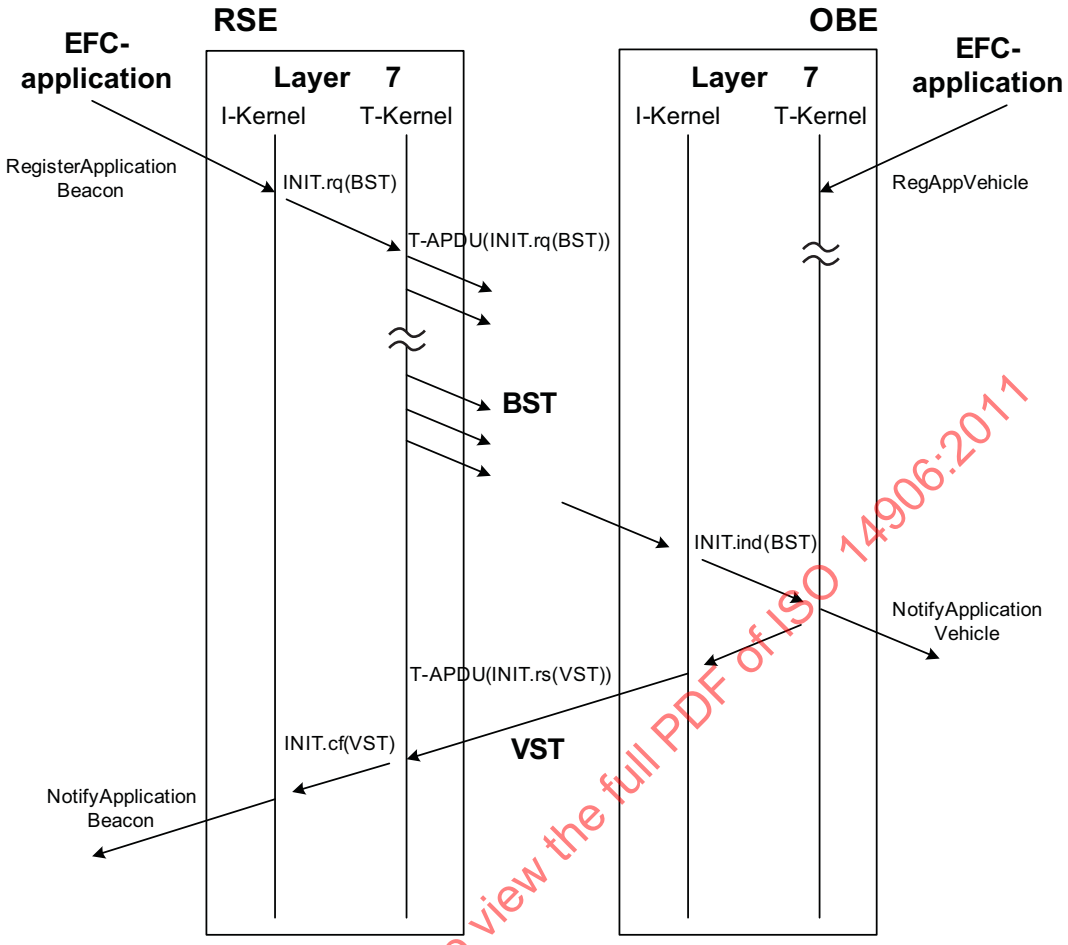

Figure 6 — Initialisation phase: BST - VST exchanges

The Initialisation service associated with the initialisation phase is only used by the Initialisation Kernel (of EN 12834/ISO 15628), which in its turn is configured by the application(s) wishing to execute applications over a DSRC link. The Initialisation Kernels of the RSE and of the concerned OBE shall have been configured, according to EN 12834/ISO 15628, prior to the invocation of the Initialisation service by the RSE.

6.2.2 EFC application-specific contents of the BST

An RSE supporting EFC shall have configured its Initialisation Kernel to carry the following information related specifically to the EFC application(s):

— the application identifier (AID) shall be equal to 1 (i.e. the value assigned for EFC);

— the EFC application shall be qualified as a mandatory application;

— EID shall not be transmitted in the BST related to the EFC application;

— no Parameter shall be transmitted in the BST related to the EFC application.

NOTE 1 AID equals to 14 identifies the multi-purpose payment context. In Japan, ISO 14906 specifies the application interface for DSRC used for multi-purpose payment (when the Aid=14 is used in Japan, the EID and parameter fields are defined through the BST).

There shall be only one EFC application present in the BST (i.e. there shall be only one instance of AID=1 in the BST) regardless of whether the RSE supports more than one EFC-ContextMark (see also 6.2.3).

NOTE 2 The above is the EFC application-specific contents of the BST. The complete BST is defined in EN 12834/ISO 15628 and is given below for readability of this International Standard:

```
BST ::= SEQUENCE {
  beacon    BeaconID,
  time Time,
  profile      Profile,
  mandApplications    ApplicationList,
  nonmandApplications      ApplicationList       OPTIONAL,
  profileList       SEQUENCE(0...127, ...) OF Profile
  }
```

where

ApplicationList ::= SEQUENCE (0..127,...) OF

```
        SEQUENCE{
        aid             DSRCApplicationEntityID,    -- AID=1
        eid             Dsrc-EID        OPTIONAL,   -- empty
        parameter       Container       OPTIONAL    -- empty
        }
```

6.2.3 EFC application-specific contents of the VST

Each EFC application and corresponding contract shall be associated with an EFC-ContextMark, as defined below. An OBE may support several EFC applications.

NOTE 1 It is outside the scope of this International Standard to define in which order EFC-ContextMarks shall be presented in order to indicate a user's order of preference, in case his OBE supports several EFC applications. Such rules for indicating the user's order of preference may be subject to agreements between operators.

An OBE supporting EFC shall have configured its Initialisation Kernel to carry the following information related specifically to the concerned EFC application:

— the AID shall be equal to 1;

— the EID value shall be logically associated with the corresponding EFC-ContextMark, contained in the Parameter, and shall be unique within the OBE throughout the complete DSRC session;

— the Parameter shall be of Container CHOICE type OCTET STRING and shall comprise the EFC-ContextMark as defined below, and may also be configured to carry additional EFC attributes (as defined in Clause 8 and Annex A).

```
EFC-ContextMark::=SEQUENCE{
        ContractProvider        Provider,
        TypeOfContract          OCTET STRING (SIZE(2)),
        ContextVersion          INTEGER(0..127,..)
        }
```

The **EFC-ContextMark** denotes a specific EFC context in the OBE, comprising the organization that issued the contract, the type of contract and the context version. ContractProvider, TypeOfContract and ContextVersion are further defined in Clause 8 as data elements of the Attribute EFC-ContextMark.

NOTE 2 The above is the EFC application-specific contents of the VST. The complete VST is defined in EN 12834/ISO 15628 and is given below for readability of this International Standard:

```
VST ::=   SEQUENCE {
                fill                    BIT STRING (SIZE(4))
                profile                 Profile,
                applications            ApplicationList,
                obeConfiguration        ObeConfiguration
                }
```

where:

```
ApplicationList ::= SEQUENCE (0..127,...) OF
        SEQUENCE{
        aid                     DSRCApplicationEntityID,    -- AID =1
        eid                     Dsrc-EID      OPTIONAL,     -- EID = e.g. 2
        parameter               Container     OPTIONAL      -- EFC-ContextMark
                                                            -- plus any EFC Attribute
        }
```

NOTE 3 Means to ensure backwards compatibility and co-existence:

— EfcModule ASN.1 module, including a version number

— Efc-ContextMark (incl. the ContextVersion), denoting the implementation version, provides a means to ensure co-existence of different implementation versions by means of a look-up table and associated appropriate transaction processing.

NOTE 4 Aid equals to 14 identifies the Multi-purpose payment context. In Japan, ISO 14906 specifies the application interface for DSRC used for multi-purpose payment (when the Aid=14 is used in Japan, the EID and parameter fields are defined through the BST).

NOTE 5 An EFC application provider retains the ultimate control of his security domain, i.e. the security level and the associated security mechanisms to be used within his system.

The data element obeStatus contained in the data element obeConfiguration shall always be present and may indicate the status of the OBE's battery. This information may be used by the RSE to notify the driver (e.g. using the SET_MMI code “contact operator”). See Annex A for details.

NOTE 6 Retrofit DSRC OBE are mostly battery powered and the battery usually has a lifetime which is dependent on the actual usage of the OBE (number of transactions per day, activation of MMI, etc.). In an interoperable environment, the Toll Charger(s) can access to the OBE via the DSRC interface and can can check the status of the battery and indicate it to the driver.

6.3 Transaction phase

After completion of the Initialisation phase, the appropriate RSE application is informed (by means of the Notify Application RSU service) of the EFC-ContextMark and associated EID. The RSE shall use the functions defined in Clause 7 to complete the EFC transaction.

The RSE may invoke any sequence of EFC functions to complete the EFC transaction, provided that they are supported by the EFC-ContextMark. The OBE shall respond to the EFC functions invoked by the RSE, and shall not initiate any EFC functions (by usage of a request service primitive, see further Clause 7) on its side.

EXAMPLE A transaction may consist of the following steps:

— GET(EID, ContractValidity, ContractVehicle, ReceiptServicePart, PaymentMeansBalance)

— DEBIT(EID, DebitPaymentFee)

— SET(EID, ReceiptServicePart)

Due to the construction of the EFC part of the VST, each EID identifies a certain EFC-ContextMark and shall be used by the RSE as parameter of every function to unambiguously address data elements within the context given by the EFC-ContextMark. More than one EID may be used in one session.

Both which attributes that are to be implemented and the maximum number of instances of an attribute is defined at time of configuration of the OBE, and is outside the scope of this International Standard. These implementation dependent aspects are referenced unambiguously by the EFC-ContextMark (for each element present in the OBE).

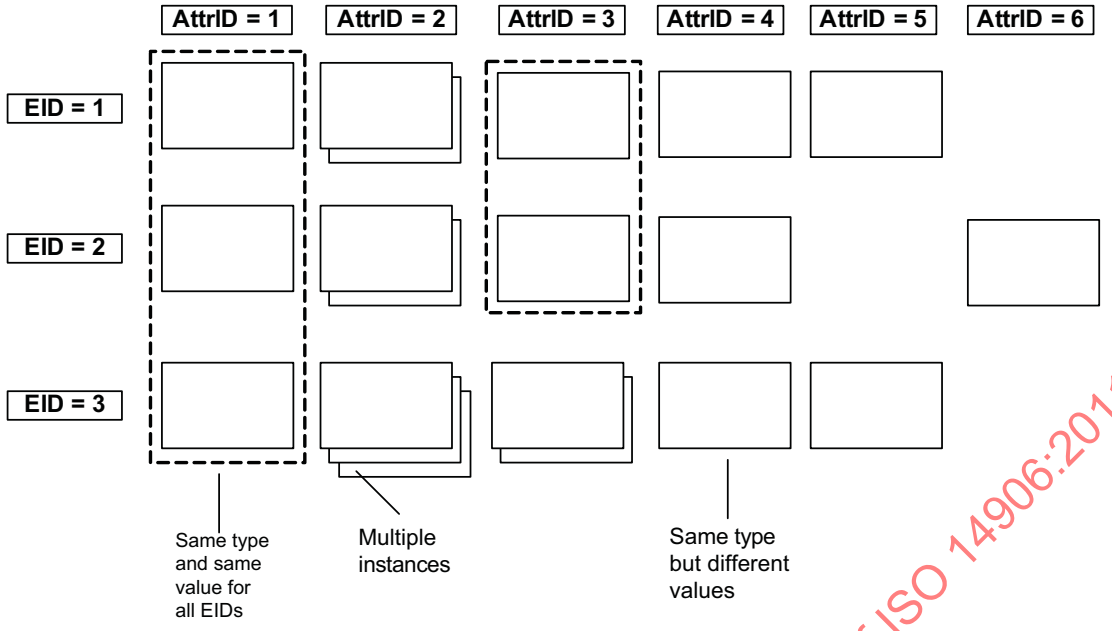

Figure 7 — Context of attributes given by EFC-ContextMark and identified by the EID

NOTE 1 This construction of contexts being identified by EIDs allows amongst others to implement the following transactions:

— Booking from two contracts in one transaction. There is sometimes the need to book from two contracts in one session, e.g. when a customer has a contract with a reduced price (e.g. a commuter contract) for part of the route being tolled, plus an ordinary (not reduced) contract for the rest of the route. This may be implemented by having all data groups identical between two EIDs, except for the data group contract.

— Having either two instances of the data group Vehicle to accommodate a pulling vehicle plus a trailer, or having two Elements containing same attribute one for the pulling vehicle and one for the trailer (and probably also separate Contract).

NOTE 2 This EFC transaction model and associated procedures allow for different levels of co-existence and interoperability between operators:

— No agreement between operators - each operator has a completely separate application domain, i.e. there are no common data groups. Each operator uses the attributes associated with “his” EFC-Context Mark.

— Agreement to share some data groups, but not others. E.g. the data groups Vehicle, Receipt and Payment are shared, but not Contract. Different security measures (algorithms) are used by the two operators (or more. Or, all data groups are shared, except for Payment - each operator books from an account issued by himself.

— Agreement between operators (Toll Chargers and ETS Provider): each Toll Charger has the possibility to select only the attributes needed for a given toll context (principle of “Pick what you need”) amongst common Data Group.

7 EFC Functions

7.1 Overview and general concepts

7.1.1 EFC functions and service primitives

This clause describes the EFC functions invoked by T-ASDUs exchanged between peer applications communicating via a DSRC link. The T-ASDUs are exchanged by means of service primitives of the DSRC Application Layer (EN 12834/ISO 15628). Exchanges of service primitives (and the corresponding T-ASDUs) associated with EFC functions adhere to the following basic pattern:

— xxx.rq (request) service primitive invoked by the RSE application to DSRC Application Layer;

— xxx.ind (indication) service primitive issued by the DSRC Application Layer to the OBE application;

— xxx.rs (response) service primitive invoked by the OBE application to DSRC Application Layer;

— xxx.cf (confirmation) service primitive issued by the DSRC Application Layer to RSE application.

The last two steps are either mandatory or optional, depending on the nature of the service primitive and on the setting of the Mode parameter (see 6.2.4 in EN 12834/ISO 15628).

The logical sequence of a successful service primitives exchange (for Mode = TRUE) is illustrated in Figure 8 below. Service primitives that occur earlier in time, and are connected by dotted lines in the figure, are the logical antecedents of subsequent service primitives.

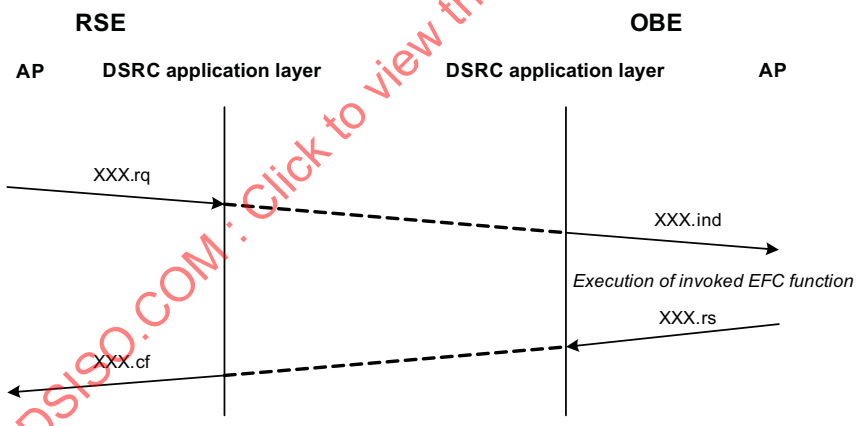

Figure 8 — The logical sequencing of service primitive exchanges

For the purposes of this International Standard, the DSRC link is seen as completely transparent, i.e. in the absence of exceptions the XXX.ind is identical in content and meaning to the XXX.rq, and the XXX.cf is identical in content and meaning to the XXX.rs. For the purpose of conciseness there will be:

— one description for *request*, i.e. XXX.rq, covering both request and indication service primitives;

— one description for *response*, i.e. XXX.rs, covering both response and confirmation service primitives.

The format and the parameters of the service primitives of the DSRC application layer are defined in EN 12834/ISO 15628, 6.2.2 (T Kernel Services).

7.1.2 Overview of EFC functions

This subclause provides an overview of the EFC functions as shown in Table 1 (based on the ACTION service primitive of EN 12834/ISO 15628) that are defined in 7.2 of this International Standard. Each EFC function

comprises a pair of service primitives, a request and its associated response service primitive, which are accounted for in 7.2.

Table 1 — Overview of EFC functions

Function Name	Action Type	Action Parameter	Response Parameter	Remarks
GET_STAMPED	0	GetStampedRq	GetStampedRs	retrieves data with an authenticator from the OBE
SET_STAMPED	1	SetStampedRq	OCTET STRING	sets data in the OBE, which generates an authenticator
GET_SECURE	2	OCTET STRING	OCTET STRING	gets data securely from the OBE
SET_SECURE	3	OCTET STRING	OCTET STRING	sets data securely in the OBE
GET_INSTANCE	4	GetInstanceRq	GetInstanceRs	retrieves a number of entries out of an attribute's multiple instances
SET_INSTANCE	5	SetInstanceRq	n.a.	sets one entry at a specified position in an attribute's multiple instances
GET_NONCE	6	n.a.	OCTET STRING	retrieves a nonce - typically used against replay attacks
SET_NONCE	7	OCTET STRING	n.a.	sets a nonce - typically used against replay attacks
TRANSFER_CHANNEL	8	ChannelRq	ChannelRs	sets and/or retrieves data from the addressed OBE component (e.g. an ICC)
COPY	9	CopyRq	n.a.	copies data from a source EID to a destination EID
SET_MMI	10	SetMMIRq	n.a.	invokes an MMI function (e.g. signal Ok via buzzer)
SUBTRACT	11	SubRq	n.a.	subtracts the given value to the addressed value
ADD	12	AddRq	n.a	adds the given value to the addressed value
DEBIT	13	DebitRq	DebitRs	debits purse
CREDIT	14	CreditRq	CreditRs	credits purse
ECHO	15	OCTET STRING	OCTET STRING	OBE echoes received data

The GET and SET services (DSRC application layer functions) as defined in EN 12834/ISO 15628 (in 6.2) may also be used in an EFC transaction phase.

NOTE GET is used to retrieve (i.e. read) value(s) of the addressed attribute(s), a reply is always expected. SET is used to set (i.e. write) value(s) of the addressed attribute(s).

7.1.3 Handling of multiple instances

For the purpose of description, the number of instances is denoted by *n*. In general the EFC functions operating on multiple instances of OBE Attributes can be divided into the following groups:

— GET, GET_STAMPED : These functions shall always access the last instance (i.e. instance at position 0). If no instance is available, the result is undefined but may lead to the return of an error code.

— GET_INSTANCE : This function carries parameters *N*1 and *N*2, both ≥ 0. It shall return the following:

— if *N*2<*N*1, or *N*1>*n,* then the empty list;

— if *N2* >= *N1* then the values of the instances numbered *N*1, *N*1+1, ... up to and including min(*N*2, n).

NOTE 1 The case that zero instances are returned is legal, in this case the response carries an empty list.

— SET, SET_STAMPED : These functions shall always set the value of instance at position 0. In addition, the previous instance number p (where p is an integer between 0 and N_{max}-1) shall become instance number *p*+1, and instance number N_{max} shall no longer be available.

NOTE 2 A cyclic buffer is as acceptable as a dynamic memory allocation scheme.

NOTE 3 The description above also covers the common case for N_{max} =1. In this case it leads to overwriting the old value of the single instance.

— SET_INSTANCE : This function carries a parameter *N* ≥0, and a value for the addressed attribute. It shall always set the value of instance number *N*.

EXAMPLE 1 Behaviour for a static memory allocation scheme - a cyclic buffer.

Assume N_{max} = 3. Table 2 shows the effects of a certain sequence of functions.

Table 2 — Behaviour for a static memory allocation scheme

Function	*N*	Buffer content Instance position 0	1	2	Result
GET	3	X	X	X	returns X
GET_INSTANCE(1,0)	3	X	X	X	returns empty list
SET (A)	3	A	X	X	
GET	3	A	X	X	returns value A
SET(B)	3	B	A	X	
GET	3	B	A	X	returns value B
GET_INSTANCE(0,7)	3	B	A	X	returns list (B,A,X)
SET(C)	3	C	B	A	
GET	3	C	B	A	returns value C
GET_INSTANCE(1,2)	3	C	B	A	returns list (B, A)
SET(D)	3	D	C	B	value A is no longer available
SET_INSTANCE(1,E)	3	D	E	B	value C is no longer available

EXAMPLE 2 Behaviour for a dynamic memory allocation scheme.

Assume N_{max} = 3. Let n = 0 initially. Table 3 shows the effects of a certain sequence of functions.

Table 3 — Behaviour for a dynamic memory allocation scheme

Function	*N*	Buffer content Instance position 0	1	2	Result
GET	0				undefined (implementation dependent)
GET_INSTANCE(1,0)	0				returns empty list
SET (A)	1	A			
GET	1	A			returns value A
SET(B)	2	B	A		
GET	2	B	A		returns value B
GET_INSTANCE(0,6)	2	B	A		returns list(B, A)
SET(C)	3	C	B	A	
GET	3	C	B	A	returns value C
GET_INSTANCE(1,2)	3	C	B	A	returns list (B, A)
SET(D)	3	D	C	B	value A is no longer available
SET_INSTANCE(1,E)	3	D	E	B	value C is no longer available

7.1.4 Security

7.1.4.1 General

Whilst security is an essential part of EFC applications, the actual mechanisms are outside the scope of this International Standard. It is generally recognised that security mechanisms involve many parameters, like encryption algorithm and keys (if the security mechanism is encryption based at all), hash function, key length, padding method, redundancy data etc. It is assumed that the EFC application communicating parties know everything they need, either by implementation or by deriving information from the VST. This information should suffice for the RSE to determine how to proceed. The OBE, in general, supports only a limited number of security mechanisms.

In this International Standard, only a framework is defined permitting security mechanisms to be specified unambiguously at the discretion of the application provider. It includes the Access Credentials defined in EN 12834/ISO 15628. In addition security values, like authenticators, will often be needed for additional protection.

7.1.4.2 Use of access credentials and authenticators

Access Credentials and Authenticators are defined as being of ASN.1 type OCTET STRING. This only pertains to the ASN.1 syntax; the semantics are implicit in the context given by the EFC-Context Mark as specified in the VST, and as selected by the EID.

Access Credentials shall be used to manage access to attributes. Different access conditions can apply for different attributes, and if so different access credentials should be associated with these access conditions.

EXAMPLE The VehicleDimensions EFC attribute may be associated with no access conditions whilst the ContractSerialNumber and ContractValidity EFC attributes may be subject to access conditions (e.g. requiring the correct password to be presented).

Authenticators shall primarily pertain to values, and prove the source and/or the integrity of the data unit and protect against forgery. Authenticators are used in cryptography related EFC functions such as GET_STAMPED and SET_STAMPED. Authenticators can be transmitted from the RSE to the OBE, as Access Credentials in order to prove the authenticity of the RSE, or from the OBE to the RSE to prove the source and/or integrity of the data unit.

The security mechanisms to be applied, and the exact role of Access Credentials and Authenticators, will be determined by the owner of the EFC-ContextMark and is outside the scope of this International Standard.

NOTE ISO/TS 17574 provides guidelines for preparation and evaluation of appropriate security measures.

7.2 EFC functions

7.2.1 General

In this section, the EFC functions are specified in detail. The format and the parameters of the EFC functions shall adhere to the Action service primitives of the DSRC application layer as defined in 6.2.2 (Transfer Kernel Services) in EN 12834/ISO 15628. Not all parameters associated with the EFC functions are accounted for in this clause, as they are either not specifically needed for the EFC applications or have the same meaning for all functions.

The Return Codes (RET) are explicitly specified whenever additional precision is needed on top of the specifications given in 6.2.4 of EN 12834/ISO 15628.

The ASN.1 type specifications of the ActionParameters and ResponseParameters are provided in the normative Annex A.

7.2.2 GET_STAMPED

GET_STAMPED is used to retrieve the value(s) of the addressed attribute(s), with an authenticator appended to the retrieved data. The authenticator generation involves transformations (notably encryption) that may include a nonce value (e.g. a random number or a sequence number).

Table 4 — GET_STAMPED.request

Parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	0	
AccessCredentials	OCTET STRING		optional use
ActionParameter	GetStampedRq ::= SEQUENCE { attributeIdList AttributeIdList, nonce OCTET STRING, keyRef INTEGER(0..255) }		always to be present
Mode	BOOLEAN	TRUE	

GET_STAMPED.request shall request the retrieval of the value(s) of the attributes addressed by the attributeIdList, with an authenticator given in the response. A response shall always be expected (Mode = TRUE). The parameter keyRef shall contain a reference to the key to be used for the calculation of the authenticator in the response. See Table 4.

NOTE 1 The AccessCredentials are only needed if the data attributes addressed by EID and attributeIdList require authentication of the RSE.

Table 5 — GET_STAMPED.response

Parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	GetStampedRs::= SEQUENCE { attributeList AttributeList, authenticator OCTET STRING }		always to be present when Return Code is present and its value is 0
Return Code (Ret)	ReturnStatus		optional use

GET_STAMPED.response shall carry the retrieved value(s) of the addressed attribute(s) in the attributeList, as the result of the corresponding GET_STAMPED.request command. An authenticator over the retrieved values shall be carried in the authenticator parameter, with the keyRef parameter of the GET_STAMPED.request being used as a reference to the (cryptographic) key to be used. When a nonce of non-zero length is given in the request, the nonce value shall be included in the cryptographic transformation. See Table 5.

NOTE 2 GET_STAMPED can be used with an empty attributeIdList to request an authenticator from the OBE to authenticate the OBE EFC application.

7.2.3 SET_STAMPED

SET_STAMPED is used to set the value(s) of the addressed attribute(s), with the OBE returning an authenticator as a proof that the data has been set. The authenticator generation involves transformations (notably encryption) which may include a nonce value (e.g. a random number or a sequence number).

Table 6 — SET_STAMPED.request

Parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	1	
AccessCredentials	OCTET STRING		optional use
ActionParameter	SetStampedRq::= SEQUENCE { attributeList AttributeList, nonce OCTET STRING, keyRef INTEGER(0..255) }		always to be present
Mode	BOOLEAN	TRUE	

SET_STAMPED.request shall request the setting of the value(s) of the attributes addressed by the attributeList, with an authenticator given in the response. A response shall always be expected (Mode = TRUE). See Table 6.

Table 7 — SET_STAMPED.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	OCTET STRING		always to be present
Return Code (Ret)	ReturnStatus		optional use

SET_STAMPED.response shall carry an authenticator as the response parameter (being of ASN.1 type OCTET STRING) to the corresponding request to convey that the data in the attribute list of the request have

been set. The authenticator shall be calculated over the values given in the request attributeList, with the keyRef parameter of the request being used as a reference to the (cryptographic) key to be used. When a nonce of non-zero length is given in the request, the nonce value shall be included in the cryptographic transformation. See Table 7.

7.2.4 GET_SECURE

GET_SECURE is used to retrieve the value(s) of attribute(s) subject to security measures defined implicitly by the context identification set in the initialisation phase. These measures may involve any kind of transformations (notably encryption).

Table 8 — GET_SECURE.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	2	
AccessCredentials	OCTET STRING		optional use
ActionParameter	OCTET STRING		always to be present
Mode	BOOLEAN	TRUE	

GET_SECURE.request shall request the retrieval of attributes subject to security measures implicit in the context set in the VST, amongst others by explicit reference given in the action parameter. The ActionParameter (being of ASN.1 type OCTET STRING) shall carry the AttributeIds of the requested attributes plus any information (nonce, key reference) required by the algorithm providing the security measures. A reply is always expected (Mode = TRUE). See Table 8.

NOTE 1 The accessCredentials are only needed if the data attributes addressed by EID and AttributeIdList require them.

NOTE 2 The interpretation of the actionParameter is defined by the security mechanism in effect, which is implicit in the context identification in the initialisation phase. The parameter includes a AttrIDList (possibly encrypted), and it may e.g. also contain an authenticator for non-repudiation purposes.

Table 9 — GET_SECURE.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	OCTET STRING		always to be present when Return Code is present and its value is 0
Return Code (Ret)	ReturnStatus		optional use

GET_SECURE.response shall carry as the responseParameter (being of ASN.1 type OCTET STRING) to the corresponding request the requested value(s) of the addressed attribute(s) in the form (e.g. encrypted) implicitly defined in the context set in the VST, amongst others by explicit reference given in the action parameter. See Table 9.

NOTE 3 The interpretation of the responseParameter is defined by the security mechanism that is in effect. The parameter includes a (possibly encrypted, or otherwise transformed) AttributeList. It may in addition contain, e.g., an authenticator for non-repudiation purposes.

7.2.5 SET_SECURE

SET_SECURE is used to set the value(s) of attribute(s) subject to security measures defined implicitly by the context identification set in the initialisation phase. These measures may involve any kind of transformations and additions (e.g. checking of authenticators).

Table 10 — SET_SECURE.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	3	
AccessCredentials	OCTET STRING		optional use
ActionParameter	OCTET STRING		always to be present
Mode	BOOLEAN		

SET_SECURE.request shall request the setting of attributes subject to security measures implicit in the context set in the VST, amongst others by explicit reference given in the action parameter. The ActionParameter (being of ASN.1 type OCTET STRING) shall carry the attributes to be set plus any information (authenticator, nonce, key reference) required by the algorithm providing the security measures. *SET_SECURE.request* can be used in confirmed or non-confirmed mode; a reply shall always be expected in the former case. See Table 10.

NOTE 1 The interpretation of the ActionParameter is defined by the security mechanism in effect, which is implicit in the context identification in the initialisation phase. The parameter includes a attrList (possibly encrypted), and it may e.g. also contain an authenticator for non-repudiation purposes.

Table 11 — SET_SECURE.response

Parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	OCTET STRING		optional use
Return Code (Ret)	ReturnStatus		optional use

SET_SECURE.response shall, if used in the confirmed mode, carry as the ResponseParameter (being of ASN.1 type OCTET STRING) the confirmation of the corresponding request. The confirmation shall be in the form implicitly defined in the context set in the VST, amongst others by explicit reference given in the action parameter. See Table 11.

NOTE 2 The interpretation of the ResponseParameter is entirely defined by the security mechanism that is in effect, which is implicit in the context identification in the initialisation phase. It may, e.g., be empty or contain an authenticator to be used for non-repudiation of receipt.

7.2.6 GET_INSTANCE

GET _INSTANCE is used to retrieve a number of values from multiple instances of the addressed attributes (see 7.1.3 for the handling of multiple instances).

Table 12 — GET_INSTANCE.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	4	
AccessCredentials	OCTET STRING		optional use
ActionParameter	GetInstanceRq ::= SEQUENCE { posOfFirstInstance INTEGER(0..255), posOfLastInstance INTEGER(0..255), attributeIdList AttributeIdList}		always to be present
Mode	BOOLEAN	TRUE	

GET_INSTANCE.request shall request the retrieval of a number of instances of the value(s) of the addressed attribute(s). The ActionParameter contains the position of the first instance and the last instance of the instances of the specified attribute(s) to be retrieved. GET_INSTANCE.*request* can only be used in confirmed mode; a reply shall always be expected. See Table 12.

Table 13 — GET_INSTANCE.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	GetInstanceRs ::= SEQUENCE (0..127,...) OF SEQUENCE { attributeId INTEGER(0..127,...), attributeValues Container::=OCTET STRING } }		always to be present when Return Code is present and its value is 0
Return Code (Ret)	ReturnStatus		optional use

GET_INSTANCE.response shall, as response to the corresponding request, contain all available values of the requested attributes, starting from the value at the first position (posOfFirstInstance) up to the value at the last position (posOfLastInstance), as asked for in the request. The ResponseParameter shall for each requested attribute in turn contain first the attribute Id of the requested attribute, followed by the values of the attribute. The value(s) of an attribute at the first position shall be transferred first in the parameter attributeValues. See Table 13.

7.2.7 SET_INSTANCE

SET _INSTANCE is used to set the value of a specified entry from multiple instances of the addressed attribute (see 7.1.3 for the handling of multiple instances).

Table 14 — SET_INSTANCE.request

Parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	5	
AccessCredentials	OCTET STRING		optional use
ActionParameter	SetInstanceRq ::= SEQUENCE { posOfInstance INTEGER(0..255), attribute Attr }		always to be present
Mode	BOOLEAN		

SET_INSTANCE.request shall request the replacement of a selected instance of the addressed attribute. The ActionParameter contains the value (posOfInstance) attempted to be replaced and the attribute Id. SET_INSTANCE.*request* can be used in confirmed or non-confirmed mode (i.e. Mode equal to TRUE or FALSE, respectively); a reply shall always be expected in the former case. See Table 14.

Table 15 — SET_INSTANCE.response

Parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	None		
Return Code (Ret)	ReturnStatus		optional use

SET_INSTANCE.response shall explicitly convey the result of the corresponding *SET_INSTANCE.request*. If the addressed value could not be replaced, the Return Code shall indicate a failure. See Table 15.

7.2.8 GET_NONCE

GET_NONCE is used by the RSE to obtain a nonce value (e.g. a random number, a sequencing number or a time stamp) to be used for guaranteeing a unique relationship between a number of related data items. The retrieved value shall remain “active” throughout the session or until a new GET_NONCE service has been successfully completed within the same session.

EXAMPLE GET_NONCE can be used to get a challenge value, which is used as an input parameter when computing the applicable access credentials (e.g. an authenticator). The resulting data can subsequently be included as access credentials in another request command.

Guaranteeing uniqueness implies certain requirements on the value to be produced, e.g. sequencing numbers and random numbers should have a sufficiently large period, time stamps should have sufficiently high resolution. In addition, random numbers may have to be generated by a cryptographic algorithm to be “unpredictable” enough. These additional requirements are outside the scope of this International Standard.

Table 16 — GET_NONCE.request

Parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	6	
AccessCredentials	OCTET STRING		Not to be used
ActionParameter	n.a.		Not to be used
Mode	BOOLEAN	TRUE	

GET_NONCE.request shall request the retrieval of a value to be used for guaranteeing a unique relationship between a number of related data items. *GET_NONCE.request* shall always be used in confirmed mode; a reply shall always be expected. See Table 16.

Table 17 — GET_NONCE.response

Parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	OCTET STRING		always to be present when Return Code is present and its value is 0
Return Code (Ret)	ReturnStatus		optional use

GET_NONCE.response shall, as response to the corresponding request, carry as the ResponseParameter (being of ASN.1 type OCTET STRING) a value to be used for guaranteeing a unique relationship between a number of related data items. The retrieved value shall remain ”active” throughout the session or until a new GET_NONCE service has been successfully completed within the same session. See Table 17.

When the *GET_NONCE.request* is not supported by the OBE EFC application, then the Return Code shall indicate complexityLimitation, and ResponseParameter shall be empty.

7.2.9 SET_NONCE

SET_NONCE is used by the RSE to present a value (e.g. a sequencing number, a random number or a time stamp) to the OBE, to be used for guaranteeing a unique relationship between a number of related data items. The set value remains ”active” throughout the session or until a new SET_NONCE service has been successfully completed within the same session.

Table 18 — SET_NONCE.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	7	
AccessCredentials	OCTET STRING		n.a.
ActionParameter	OCTET STRING		always to be present
Mode	BOOLEAN		

SET_NONCE.request shall request setting a nonce value to be used for guaranteeing a unique relationship between a number of related data items. The ActionParameter (being of ASN.1 type OCTET STRING) shall carry the nonce value. *SET_NONCE.request* can be used in confirmed or non-confirmed mode (i.e. Mode equal to TRUE or FALSE, respectively); a reply shall always be expected in the former case. See Table 18.

Table 19 — SET_NONCE.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	n.a.		n.a.
Return Code (Ret)	ReturnStatus		optional use

SET_NONCE.response shall be issued as a response to the corresponding request to convey the result of the request. A receiving peer entity supporting no nonce shall return a Return Code indicating Complexity Limitation. See Table 19.

7.2.10 TRANSFER_CHANNEL

TRANSFER_CHANNEL is used to send and/or retrieve data from a dedicated channel of the OBE.

Table 20 — TRANSFER_CHANNEL.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID	0	
ActionType	INTEGER(0..127,...)	8	
AccessCredentials	OCTET STRING		not to be used
ActionParameter	ChannelRq::= SEQUENCE { channelId ChannelId, apdu OCTET STRING }		
Mode	BOOLEAN		

TRANSFER_CHANNEL.request shall request data transfer from and/or to a dedicated channel of the OBE. The channel shall be addressed by the parameter channelled. Data to be transferred to the OBE channel shall be contained in the parameter apdu in a format recognised by the addressed component. In case no data is to be set to the OBE channel, the apdu can be empty. The direction(s) of transfer is either implicitly given by the context of the addressed channel within the context given in the VST, or is to be conveyed as part of the parameter apdu in a format appropriate for the addressed channel. See Table 20.

NOTE TRANSFER_CHANNEL allows addressing of data residing in components of the OBE using a transparent application layer protocol bridge. The command can e.g. be used to address a serial interface, or an electronic purse that requires a protocol that is not covered by the DEBIT command.

Table 21 — TRANSFER_CHANNEL.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	ChannelRs ::= SEQUENCE { channelId ChannelId, apdu OCTET STRING }		
Return Code (Ret)	ReturnStatus		optional use

TRANSFER_CHANNEL.response shall carry the response to the requested data transfer from and/or to a dedicated channel of the OBE. The parameter channelled shall contain the channel Id of the request. Data requested to be transferred from the OBE channel shall be contained in the parameter apdu in a format specific to the addressed component. In case no data is to be returned from the OBE channel, the apdu can be empty. See Table 21.

7.2.11 COPY

Copy is used to copy the values of the addressed attribute(s) to another EID (i.e. the destination Id).

EXAMPLE 1 When not implemented in the OBE itself, “sharing” of data attributes for common use between two operators (between two different context addressed by two different EIDs) can be performed with a copy command. Note that operators can protect the data under “their” EID by requiring certain Access Credentials only known to parties they have an agreement with.

EXAMPLE 2 The RSE sets (i.e. writes) the last event into the Receipt data group in the OBE, which only stores a limited number of events. The RSE invokes a copy command requesting the OBE to copy the value of the last event to the event log of the ICC.

Table 22 — COPY.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	9	
AccessCredentials	OCTET STRING		optional use
ActionParameter	CopyRq::= SEQUENCE { destinationEID INTEGER(0..127,...), attributeIdList AttributeIdList }		always to be present
Mode	BOOLEAN		

COPY.request shall request the copying of the value(s) of the addressed attribute(s) to the same attribute(s) in a destination EID. *COPY.request* can be used in confirmed or unconfirmed mode, a reply shall always be expected in the former case. See Table 22.

Table 23 — COPY.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	n.a.		n.a.
Return Code (Ret)	ReturnStatus		optional use

COPY.response shall be used to explicitly convey the result of the corresponding COPY.request command. See Table 23.

7.2.12 SET_MMI

SET_MMI is used to perform device-independent MMI functions.

Table 24 — SET_MMI.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID	0	
ActionType	INTEGER(0..127,...)	10	
AccessCredentials	OCTET STRING		not to be used
ActionParameter	SetMMIRq::= INTEGER { ok (0), nok (1), contactOperator (2), reservedForFutureCENUse (3-127), reservedForPrivateUse (128-254), noSignalling (255) } (0..255)		always to be present
Mode	BOOLEAN		

SET_MMI.request shall request to control the MMI in a device-independent way. The ActionParameter shall contain the MMI function that is to be invoked, e.g. signalling of a successful operation (such as a successful

EFC transaction), a non-successful operation or signalling to contact the operator (e.g. due to low balance). *SET_MMI.request* can be used in confirmed or non-confirmed mode; a reply shall always be expected in the former case. See Table 24.

Table 25 — SET_MMI.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	n.a.		n.a.
Return Code (Ret)	ReturnStatus		optional use

SET_MMI.response shall be used to explicitly convey the result of the corresponding SET_MMI.request command. If Mode was set to TRUE in the corresponding request and the operation was successfully executed then the response shall indicate no error. If an error occurred at an attempt to execute the command then the Return Code shall take the appropriate value. See Table 25.

To retain compatibility with existing OBE, future OBE may accept SET_MMI with any value of the EID, and with Container type =0(dec). and 69(dec).

7.2.13 SUBTRACT

SUBTRACT is used to subtract a given INTEGER value from another value of an INTEGER-type attribute.

Table 26 — SUBTRACT.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	11	
Access Credentials	OCTET STRING		optional use
ActionParameter	SubRq::= SEQUENCE { attributeId INTEGER(0..127,..), value INTEGER }		always to be present
Mode	BOOLEAN		

SUBTRACT.request is used to request the subtraction of the value, as contained in the action parameter, from the value of the addressed attribute. *SUBTRACT.request* can be used in confirmed or non-confirmed mode; a reply shall always be expected in the former case. See Table 26.

Table 27 — SUBTRACT.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	n.a.		
Return Code (Ret)	ReturnStatus		optional use

SUBTRACT.response is a response used to explicitly convey the result of the corresponding *SUBTRACT.request* command. *SUBTRACT.response* shall be invoked upon completion of an attempt to execute the corresponding *SUBTRACT.request* command. See Table 27.

7.2.14 ADD

ADD is used to add a given INTEGER value to another value of an INTEGER-type attribute.

Table 28 — ADD.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		
ActionType	INTEGER(0..127,...)	12	
Access Credentials	OCTET STRING		optional use
ActionParameter	AddRq::= SEQUENCE { attributeId INTEGER(0..127,..), value INTEGER }		always to be present
Mode	BOOLEAN		

ADD.request shall request the addition of the value, as contained in the action parameter, to the value of the addressed attribute. *ADD.request* can be used in confirmed or non-confirmed mode; a reply shall always be expected in the former case. See Table 28.

The AccessCredentials protect the value against unauthorised modification.

Table 29 — ADD.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	n.a.		
Return Code (Ret)	ReturnStatus		optional use

ADD.response is a response used to explicitly convey the result of the corresponding *ADD.request* command. *ADD.response* shall be invoked upon completion of an attempt to execute the corresponding *ADD.request* command. See Table 29.

7.2.15 DEBIT

DEBIT is used to perform a debit on the attribute PaymentMeansBalance. The command contains payment related data (a price) and optionally security related data. A (cryptographic) proof of payment can be returned.

Table 30 — DEBIT.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		unequal 0
ActionType	INTEGER(0..127,...)	13	
AccessCredentials	OCTET STRING		optional use
ActionParameter	DebitRq::= SEQUENCE { debitPaymentFee PaymentFee, nonce OCTET STRING, keyRef INTEGER(0..255) }		always to be present
Mode	BOOLEAN	TRUE	

DEBIT.request shall request a debiting transaction to be performed on the attribute PaymentMeansBalance. *DEBIT.request* shall be used in confirmed mode; a reply shall always be expected. The parameter debitPaymentFee shall contain the price, including a currency and a multiplicator, to be subtracted from the attribute PaymentMeansBalance, where PaymentMeansUnit, the unit of PaymentMeansBalance, shall be taken into account. Nonce shall contain a nonce value to be included in the (cryptographic) calculation of the response authenticator or be empty. The parameter keyRef shall contain a reference to the key to be used for the calculation of the authenticator in the response, if required. See Table 30.

Table 31 — DEBIT.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	DebitRs ::= SEQUENCE { debitResult ResultFin, debitAuthenticator OCTET STRING }		always to be present
Return Code (Ret)	ReturnStatus		optional use

DEBIT.response shall contain the response to the corresponding request. On reception of a DEBIT command, in case the currencies of the debitPaymentFee parameter of the request and of the attribute PaymentMeansUnit in the OBE match, the OBE shall attempt to subtract the requested debitPaymentFee from the attribute PaymentMeansBalance, taking the multiplicators of the debitPaymentFee parameter and of the attribute PaymentMeansUnit into account. In case of currency mismatch, the OBE shall not subtract the amount given in debitPaymentFee, and shall return debitResult = 'Transaction not successful, currency not accepted'. See Table 31.

Depending on the context identified by the VST or implicitly given by the type of payment means, the response shall either include a proof of payment in debitAuthenticator, or return an empty debitAuthenticator.

When a nonce of non-zero length is given in the request, the nonce value shall be included in the cryptographic transformation performed to generate debitAuthenticator. If a nonce of zero length is given and if a nonce is required by the cryptographic algorithm, then the nonce given in the most recent SET_NONCE command of the session shall be used.

In case the attempt to debit failed, the parameter debitResult shall be set to the appropriate value.

7.2.16 CREDIT

CREDIT is used to perform a credit (refund) on the attribute PaymentMeansBalance. The command contains payment related data (a refund) and optionally security related data. A (cryptographic) proof can be returned.

Table 32 — CREDIT.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID		unequal 0
ActionType	INTEGER(0..127,...)	14	
AccessCredentials	OCTET STRING		optional use
ActionParameter	CreditRq::= SEQUENCE { refund PaymentFee, nonce OCTET STRING, keyRef INTEGER(0..255) }		always to be present
Mode	BOOLEAN	TRUE	

CREDIT.request shall request a refunding transaction to be performed on the attribute PaymentMeansBalance. CREDIT.request shall be used in confirmed mode; a reply shall always be expected.

The parameter refund shall contain the price, including a currency and a multiplicator, to be added to the attribute PaymentMeansBalance, where PaymentMeansUnit, the unit of PaymentMeansBalance, shall be taken into account. Nonce shall contain a nonce value to be included in the (cryptographic) calculation of the response authenticator or be empty. The parameter keyRef shall contain a reference to the key to be used for the calculation of the authenticator in the response, if required. See Table 32.

Table 33 — CREDIT.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	CreditRs ::= SEQUENCE { creditResult ResultFin, creditAuthenticator OCTET STRING }		always to be present
Return Code (Ret)	ReturnStatus		optional use

CREDIT.response shall contain the response to the corresponding request. On reception of a CREDIT command, in case the currencies of the refund parameter of the request and of the attribute PaymentMeansUnit in the OBE match, the OBE shall attempt to add the requested refund to the attribute PaymentMeansBalance, taking the multiplicators of the debitPaymentFee parameter and of the attribute PaymentMeansUnit into account. In case of currency mismatch, the OBE shall not add the amount given in debitPaymentFee, and shall return creditResult = 'Transaction not successful, currency not accepted'. See Table 33.

Depending on the context identified by the VST or implicitly given by the type of payment means, the response shall either include a proof of refund in creditAuthenticator, or return an empty creditAuthenticator.

When a nonce of non-zero length is given in the request, the nonce value shall be included in the cryptographic transformation performed to generate creditAuthenticator. If a nonce of zero length is given and if a nonce is required by the cryptographic algorithm, then the nonce given in the most recent SET_NONCE command of the session shall be used.

In case the attempt to credit failed, the parameter creditResult shall be set to the appropriate value.

7.2.17 ECHO

ECHO is used for dummy data (i.e. bitstream) being sent to, and returned by, the peer service user. This service may be used for testing purposes and for localisation of the OBE during the passage of the DSRC-EFC station.

Table 34 — ECHO.request

parameter name	ASN.1 type	Value	Remark/Constraints
Element Identifier EID	Dsrc-EID	0	
ActionType	INTEGER(0..127,...)	15	
AccessCredentials	OCTET STRING		not to be used
ActionParameter	OCTET STRING		Always to be present
Mode	BOOLEAN		

ECHO.request shall request dummy data (i.e. an octet string of variable length) to be sent to and returned by the addressed OBE. *ECHO.request* can be used in confirmed or non-confirmed mode; a reply shall always be expected in the former case. See Table 34.

Table 35 — ECHO.response

parameter name	ASN.1 type	Value	Remark/Constraints
ResponseParameter	OCTET STRING		always to be present when Return Code is present and its value is 0
Return Code (Ret)	ReturnStatus		optional use

ECHO.response shall be used to return the data conveyed in ActionParameter, or the result, of the corresponding *ECHO.request* command. See Table 35.

8 EFC Attributes

8.1 General

Within the context of EFC, the following EFC Attributes in Table 36, or a subset thereof, shall be available to perform an EFC transaction.

Table 36 — EFC Attributes

AttributeID	Attribute	Length in Octet	Data Group
0	EFC-ContextMark	6	Contract
1	ContractSerialNumber	4	
2	ContractValidity	6	
35	ValidityOfContract	4	
3	ContractVehicle	Variable	
4	ContractAuthenticator	Variable	
5	ReceiptServicePart	13	Receipt
6	SessionClass	2	
7	ReceiptServiceSerialNumber	3	
36	ReceiptFinancialPart	23	
9	ReceiptContract	9	
10	ReceiptOBUId	Variable	
11	ReceiptICC-Id	Variable	
12	ReceiptText	Variable	
13	ReceiptAuthenticator	Variable	
14	ReceiptDistance	3	
33	ReceiptData1	28	
34	ReceiptData2	28	
15	VehicleIdentificationNumber	Variable	Vehicle
16	VehicleLicencePlateNumber	Variable	
17	VehicleClass	1	
18	VehicleDimensions	3	
19	VehicleAxles	2	
20	VehicleWeightLimits	6	
21	VehicleWeightLaden	2	
22	VehicleSpecificCharacteristics	4	
23	VehicleAuthenticator	Variable	
37	AxleWeightLimits	10	
38	PassengerCapacity	2	
39	Engine	4	
40	SoundLevel	2	
41	ExhaustEmissionValues	8	
42	DieselEmissionValues	4	
43	CO2EmissionValue	2	
44	VehicleTotalDistance	4	
45	TrailerLicencePlateNumber	Variable	
46	TrailerCharacteristics	5	
24	EquipmentOBUId	Variable	Equipment
25	EquipmentICC-Id	Variable	
26	EquipmentStatus	2	
27	DriverCharacteristics	2	Driver
47	ActualNumberOfPassengers	1	
32	PaymentMeans	14	Payment
29	PaymentMeansBalance	3	
30	PaymentMeansUnit	2	
31	PaymentSecurityData	Variable	
48-53	*ReservedForCCC*		
54	*ReservedForLAC*		
55-86	*ReservedForFutureCENuse*		
87-127	*ReservedForPrivateUse*		

In an OBE. any attribute or data element which is present but not explicitly defined shall be filled with '0's (binary). Care should be taken to ensure backward compatibility when an 'old' OBE is read by a 'new' RSE'.

NOTE 1 ReceiptData1, ReceiptData2 and ValidityOfContract have been added since the previous edition of this International Standard (i.e. ISO 14906). The assignment of attribute identifier numbers of this International Standard is backwards compatible with the previous edition.

NOTE 2 The definition of PaymentMeans and ReceiptFinancialPart has changed in the transformation from the pre-standard edition to this edition of the document. The corresponding attribute identifier numbers have changed in order to ensure cohabitation with the previous edition. The attribute id numbers 28 and 8 were assigned to PaymentMeans and ReceiptFinancialPart, respectively, in ISO 14906.

Not every EFC attribute has to be present in any one implementation of this International Standard in order to be compliant except for the EFC ContextMark (in VST). A transaction can be made with a combination of Public and Private Attributes.

NOTE 3 Which EFC attributes are present and which are not is implementation dependent. The implementation is identified by the context given in the EFC-ContextMark of the VST.

In the following tables, EFC Attributes are grouped into data group tables and specified in terms of:

— the name of a data attribute;

— the names of the data elements forming the EFC Attribute - there are no optional data elements within any one EFC attribute;

— the definition of the data element;

— the ASN.1 type;

— the length in octets (PER coded);

— the value range;

— informative remarks, including references to other standards.

EFC Attributes that describe similar aspects of EFC information are grouped into data groups. The grouping has been made to facilitate readability and imply neither any relations with regard to addressing nor to logical interdependence. The specification of the corresponding ASN.1 module is provided in the normative Annex A.

8.2 Data group CONTRACT

Table 37 presents information associated with the service rights of the user of a Toll service.

8.3 Data group RECEIPT

Table 38 presents information associated to a specific session, including both financial and operational data.

8.4 Data group VEHICLE

Table 39 presents information pertaining to the vehicle (i.e. the pulling vehicle) and to the attached the trailer(s). When more sets of vehicle characteristics are needed within the context of an EID, e.g. to cater for several pulling vehicles with different characteristics associated to the OBE or the presence of more than one trailer, then multiple instances of the pertinent EFC attributes shall be used.

Data pertaining to one specific vehicle or trailer and carried in separate attributes shall be contained in the same instance number of those attributes (e.g. data for trailer 1 in Instance 0 of TrailerLicencePLateNumber and TrailerCharacteristics and data for trailer 2 in instance 1 of the same attributes).

8.5 Data group EQUIPMENT

Table 40 presents information pertaining to the OBE.

8.6 Data group DRIVER

Table 41 presents driver/user-related information (groups/individuals) used to calculate the tariff to be applied. The tariff may depend on the characteristics of the driver, as for example a specific group of drivers (category) or driving behaviour.

8.7 Data group PAYMENT

Table 42 presents data associated with the payment transaction.

The following requirements have been taken into account for the definition of the payment attribute. Various payment types should be supported:

— on-board account;

— central account;

— pre- and post-payment;

— purse/token-based payment;

— open payment system/'closed' payment system;

— no/zero payment;

— refunding.

The EFC transaction can be divided into an EFC service transaction and an EFC payment transaction (for example the visit in a restaurant also involves a service transaction (ordering and serving of meals) and a payment transaction (cash, credit card)). The contract provides the link between service transaction and payment transaction.

The EFC service provider may be independent of the payment system operator/issuer. There may be also different security domains. In order to accommodate open payment systems, the payment attribute may be transmitted transparently (as OCTET STRING) from the OBE to the RSE. The Balance and the currency may be accessed independently, in order not to disclose the balance when only currency is needed to debit the right price.

Table 37 — Data group Contract

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
EFC-ContextMark	ContractProvider	Identifies the organization that issued the service rights given in the Contract, i.e. the Toll Service Provider. Numbers shall be assigned on a national basis. It is outside the scope of this International Standard to identify the data that specify the service rights.	Provider	3	AA..ZZ & 0..16383	ASN.1 Type and Value assignment defined in Annex A. Note that the Attribute EFC-ContextMark is part of the VST.
	TypeOfContract	ContractProvider-specific designation of the rules that apply to the Contract.	OCTET STRING(SIZE(2))	2		Allows, e.g., for the determination of the tariff or designating the type of purse associated with the contract.
	ContextVersion	ContextVersion denotes the implementation version of the concerned contract within the context of the given ContractProvider, value assigned at the discretion of the ContractProvider.	INTEGER(0..127,..)	1		The ContextVersion may also be used as a security key reference.
ContractSerial Number	ContractSerial Number	Number designating the individual contract, value assigned at the discretion of the ContractProvider.	INT4	4	0... 4294967295	
ContractValidity	Contract Restrictions	ContractProvider specific coding of the validity restrictions of a contract.	OCTET STRING(SIZE(4))	4		Allows for finer validity restrictions in addition to TypeOfContract, like applicable vehicle classes, zones of the network, duration of validity. (TypeOfContract is given in the VST and is to be kept short.) Note that this attribute/element is present for compatibility with the ISO/TR 14906:1998 version. ValitidityOfContract is intended to replace this attribute in new systems.
	ContractExpiry Date	End date of the validity of the contract. Contract validity ends at 24 h of ContractExpiryDate.	DateCompact	2	[01.01.1990].. [31.12.2117]	Start date not given - it is usually implicitly given by the type of contract. When necessary it may be calculated, since duration of validity may be coded in ContractValidity or follows implicitly from TypeOfContract.

Table 37 (*continued*)

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
ValidityOf Contract	Issuer Restrictions	ContractProvider issuer specific coding of the validity restrictions of a contract.	OCTET STRING(SIZE(2))	2		Allows for finer validity restrictions in addition to TypeOfContract, like applicable vehicle classes, zones of the network, duration of validity. (TypeOfContract is given in the VST and is to be kept short.)
	ContractExpiry Date	End date of the validity of the contract. Contract validity ends at 24 h of ContractExpiryDate.	DateCompact	2	[01.01.1990].. [31.12.2117]	Start date not given - it is usually implicitly given by the type of contract. When necessary it may be calculated, since duration of validity may be coded in ContractValidity or follows implicitly from TypeOfContract.
ContractVehicle	ContractVehicle	For vehicle bound contracts, ContractVehicle gives the licence plate number to which the contract is restricted.	LPN	variable		Contracts valid for two or three vehicles may be handled with multiple instances of ContractVehicle.
Contract Authenticator	Contract Authenticator	Authenticator calculated by the ContractProvider when issuing the Contract, to prevent tampering with contract data.	OCTET STRING	variable		It is not specified over which attributes of the data group the authenticator is to be calculated.

Table 38 — Data group Receipt

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
Receipt ServicePart	SessionTime	Time of session with a two-seconds resolution.	DateAndTime	4	[01.01.1990, 00:00:00] ... [31.12.2117, 23:59:58], then rollover.	Easy to decode into a displayable format by OBE.
	SessionService Provider	Organization that provides the service of the session.	Provider	3	AA..ZZ & 0..16383	Type defined in Annex A
	StationLocation	Service provider specific coding of the station location.	INTEGER (0..1048575)	2.5		e.g.: toll plaza no.
	SessionLocation	Service provider specific coding of the session location within the station location.	BIT STRING(SIZE(8))	1		e.g.: equipment no. (lane no., beacon no.) at the toll plaza.
	TypeOfSession	Designates the type of service station.	StationType (=ENUMERATED)	0.5		*See Annex A for value assignment.*
	SessionResult Operational	Code designating whether a session has been completed successfully or not with regard to operational issues.	ResultOp	1		
	SessionResult Financial	Code designating whether a session has been completed successfully or not with regard to financial reasons.	ResultFin	1		
Session Class	Session TariffClass	Service provider specific tariff class applied in the session.	INT1	1		Enables to reproduce the price calculation (e.g. claimed or measured vehicle class that was applied.)
	Session ClaimedClass	Service provider specific vehicle class derived from claimed characteristics in the data group Vehicle.	INT1	1		Claimed class and applied class (tariff class) may differ.
ReceiptService SerialNumber	ReceiptService SerialNumber	ServiceProvider specific serial number of the session given by the RSE.	INT3	3	0 .. 16777215	

Table 38 (*continued*)

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
Receipt FinancialPart	PersonalAccount Number	Coded according to financial institutions.	PersonalAccount Number	10		Personal account number shall be in accordance with ISO/IEC 7812-1.
	SessionPayment Fee	The amount paid for the service.	PaymentFee	4		Both PaymentFee and Balance contain a currency designation plus a multiplicator.
	SessionCurrent Balance	Balance of the payment means after the session.	PurseBalance	5		In case SessionCurrentBalance is not applicable, the value 0 shall be used.
	ReceiptFinancial SerialNumber	Serial number of the financial receipt	INT4	4	0 .. 4294967295	
ReceiptContract	SessionContract Provider	Organization that issued the contract applied in the session.	Provider	3	AA..ZZ & 0..16383	Type defined in Annex A.
	SessionTypeOf Contract	TypeOfContract applied in the session.	OCTET STRING(SIZE(2))	2		
	SessionContract SerialNumber	ContractSerialNumber of the contract applied in the session.	INT4	4	0 .. 4294967295	
ReceiptOBUId	ReceiptOBUId	Serial number of the OBE used in the session, unique within the context of the manufacturer.	OCTET STRING	Variable		The manufacturer ID is always exchanged as a part of the VST.
ReceiptICC-Id	ReceiptICC-Id	Identification number of smart card used in the session.	ICC-Id	Variable		Multiple instances for multiple ICCs.
ReceiptText	ReceiptText	Plain text decodeable by the OBE.	OCTET STRING	Variable		May be used to display session information to user (e.g. session location).
Receipt Authenticator	Receipt Authenticator	Authenticator over some Attributes of the data group Receipt, calculated by the SessionServiceProvider.	OCTET STRING	Variable		
Receipt Distance	Receipt Distance	Total distance covered by the vehicle, since the beginning of its existence. The distance unit shall be 100 metres.	INT3	3	0..16777215	Vehicle distance readings, e.g. via an interface to a tachograph, may be used to the determine the PaymentFee based on the travelled distance.

Table 38 (*continued*)

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
ReceiptData1	SessionTime	Date and Time of session with a two-seconds resolution.	DateAndTime	4	[01.01.1990, 00:00:00] ... [31.12.2117, 23:59:58], then rollover.	Easy to decode into a displayable format by OBE. Date and time value assignment – Octet Aligned.
	SessionService Provider	Operator that provides the service of the session.	Provider	3	AA..ZZ & 0..16383	Identifier of an operator.
	LocationOf Station	Service provider specific coding of the station location.	INT2	2	0..65535	Toll plaza code defined by the service provider.
	SessionLocation	Service provider specific coding of the session location within the station location.	SessionLocation	1	0/1 + 0..127	Travel direction + Lane Code
	SessionType	Designates the type of service station.	INT1	1		
	SessionResult	Code designating whether a session has been completed successfully or not.	ResultOp	1		
	SessionTariff Class	Service provider specific tariff class applied in the session.	INT1	1		Enables to reproduce the price calculation (e.g. claimed or measured vehicle class that was applied).
	SessionClaimedClass	Service provider specific vehicle class derived from claimed characteristics in the data group Vehicle.	INT1	1		Claimed class and applied class (tariff class) may differ. The use of the attribute it is an operator decision.
	SessionFee	The amount paid for the service.	PaymentFee	4	0..65535 & mulitplier & currency code (12 bits)	Contains a currency designation plus a multiplicator. Currency code shall be according to ISO 4217.
	SessionContractProvider	Organization that provides the contract used in the session.	Provider	3	AA..ZZ & 0..16383	As ContractProvider defined in EFCContextMark attribute.
	SessionTypeOf Contract	ContractProvider-specific designation of the rules that apply to the Contract.	OCTET STRING (SIZE(2))	2		As TypeOfContract defined in EFCContextMark attribute.
	SessionContext Version	It identifies the version of the transaction model used for formatting the data.	INTEGER(0..127,..)	1		As ContextVersion defined in EFCContextMark attribute.
	ReceiptData Authenticator	Authenticator over all other data elements that are part of this Attribute, calculated by the SessionServiceProvider.	OCTET STRING (SIZE(4))	4		Each operator shall define a cryptographic algorithm to calculate the authenticator.

Table 38 (*continued*)

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
ReceiptData2	SessionTime	Date and Time of session with a two-seconds resolution.	DateAndTime	4	[01.01.1990, 00:00:00] ... [31.12.2117, 23:59:58], then rollover.	Easy to decode into a displayable format by OBE. Date and time value assignment – Octet Aligned.
	SessionService Provider	Operator that provides the service of the session.	Provider	3	AA..ZZ & 0..16383	Identifier of an operator.
	LocationOf Station	Service provider specific coding of the station location.	INT2	2	0..65535	Toll plaza code defined by country organization.
	SessionLocation	Service provider specific coding of the session location within the station location.	SessionLocation	1	0/1 + 0..127	Travel direction + Lane Code.
	SessionType	Designates the type of service station.	INT1	1		
	SessionResult	Code designating whether a session has been completed successfully or not.	ResultOp	1		
	SessionTariff Class	Service provider specific tariff class applied in the session.	INT1	1		Enables to reproduce the price calculation (e.g. claimed or measured vehicle class that was applied).
	SessionClaimedClass	Service provider specific vehicle class derived from claimed characteristics in the data group Vehicle.	INT1	1		Claimed class and applied class (tariff class) may differ. The use of the attribute it is an operator decision
	SessionFee	The amount paid for the service.	PaymentFee	4	0..65535 & mulitplier & currency code (12 bits)	Contains a currency designation plus a multiplicator. Currency code shall be according to ISO 4217.
	SessionContractProvider	Organization that provides the contract used in the session.	Provider	3	AA..ZZ & 0..16383	As ContractProvider defined in EFCContextMark attribute.
	SessionTypeOf Contract	ContractProvider-specific designation of the rules that apply to the Contract.	OCTET STRING (SIZE(2))	2		As TypeOfContract defined in EFCContextMark attribute.
	SessionContext Version	It identifies the version of the transaction model used for formatting the data.	INTEGER (0..127,..)	1		As ContextVersion defined in EFCContextMark attribute.
	ReceiptData Authenticator	Authenticator over all other data elements that are part of this Attribute, calculated by the SessionServiceProvider.	OCTET STRING (SIZE(4))	4		Each operator shall define a cryptographic algorithm to calculate the authenticator.

Table 39 — Data group Vehicle

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
VehicleLicence PlateNumber	VehicleLicence PlateNumber	Claimed licence plate of the vehicle	LPN	Variable		
Vehicle Identification Number	VehicleIdentification Number	Identification number of vehicle shall be according ISO 3779	CS5	Variable		Imported from ISO 14816
VehicleClass	VehicleClass	Service provider specific information pertaining to the vehicle.	INT1	1		
Vehicle Dimensions	VehicleLength Overall	Nominal maximum overall length of the vehicle, which shall be according to ISO 612, in dm, rounded to the next dm.	INT1	1		
	VehicleHeight Overall	Nominal overall unladen height, which shall be according to ISO 612, in dm, rounded to the next dm.	INT1	1		
	VehicleWidthOverall	Nominal overall width, which shall be according to ISO 612, in dm, rounded to the next dm	INT1	1		
VehicleAxles	VehicleFirstAxle Height	Bonnet height, measured over the front axle, in dm, rounded to the next dm.	INT1	1		
	VehicleAxles Number	Tyre type and number of axles, including drop axles.	VehicleAxles	1		2 bits for dual tyre. 6 bits are used for the definition of number of axles :3 bits to encode the number of all axles of the tractor and 3 to encode the number of all axles of the trailer. See NOTE

Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Nr of axles of Trailer			Nr of axles of Tractor		
0 to 7			0 to 7		

Table 39 (*continued*)

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
VehicleWeight Limits	VehicleMaxLaden Weight	Maximum permissible total weight including payload, which shall be according to ISO 1176. 10 kg units, rounded down to the next 10kg step.	INT2	2		
	VehicleTrain MaximumWeight	Maximum permissible weight of the complete vehicle train, which shall be as defined in ISO 1176. 10 kg units, rounded down to the next 10kg step.	INT2	2		ISO 1176 Code ISO-M18 maximum design mass of vehicle combination
	VehicleWeight Unladen	Nominal unladen weight, which shall be according to ISO 1176 in 10 kg units, rounded down to the next 10 kg step.	INT2	2		
VehicleWeight Laden	VehicleWeightLaden	Actual weight of vehicle including load in 10 kg units, rounded down to the next 10 kg step.	INT2	2		
AxleWeight Limits	MaxLadenWeightOn Axle1	Technically permissible maximum laden weight on axle 1 of the vehicle, 10 kg units, rounded down to the next 10 kg step.	INT2	2		
	MaxLadenWeightOn Axle2	Technically permissible maximum laden weight on axle 2 of the vehicle, 10 kg units, rounded down to the next 10 kg step.	INT2	2		
	MaxLadenWeightOn Axle3	Technically permissible maximum laden weight on axle 3 of the vehicle, 10 kg units, rounded down to the next 10 kg step.	INT2	2		
	MaxLadenWeightOn Axle4	Technically permissible maximum laden weight on axle 4 of the vehicle, 10 kg units, rounded down to the next 10 kg step.	INT2	2		

Table 39 (*continued*)

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
	MaxLadenWeightOnAxle5	Technically permissible maximum laden weight on axle 5 of the vehicle, 10 kg units, rounded down to the next 10 kg step.	INT2	2		
Passenger Capacity	NumberOfSeats	Number of seats of the vehicle, including the driver's seat.	INT1	1	0...255	
	NumberOfStandingPlaces	Number of standing places of the vehicle	INT1	1	0...255	
VehicleSpecificCharacteristics	VehicleSpecificCharacteristics	Further vehicle characteristics. Each enumerated value has a specific meaning assigned. The meaning of some values are defined in this International Standard, others are reserved for future needs.	VehicleSpecificCharacteristics	4		Assignment of meaning to the unassigned enumerated values is subject to registration according to the registration procedure specified in EN 12834/ISO 15628.
Engine	EngineCapacity	Capacity of the vehicle's engine in cm^3	INT2	2		
	EnginePower	Maximum net power of the vehicle's engine, in KW	INT2	2		
SoundLevel	SoundStationary	Stationary Sound of the vehicle, according to vehicle registration documents in dB(A)	INT1	1	0...255	
	SoundDriveBy	Sound of the vehicle when driving, according to vehicle registration documents in dB(A)	INT1	1	0...255	

Table 39 (*continued*)

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
Exhaust EmissionValues	EmissionCO	Exhaust emission of CO, according to vehicle registration documents, in 10^{-3} g/km or g/kWh.	INTEGER (0…32766)			If the emissions are measured directly on the engine test bed the value is declared in g/kWh
	EmissionHC	Exhaust emission of HC, according to vehicle registration documents, in 10^{-3} g/km or g/kWh.	INT 2	2	0...65535	If the emissions are measured directly on the engine test bed the value is declared in g/kWh
	EmissionNOX	Exhaust emission of NOX, according to vehicle registration documents, in 10^{-3} g/km or g/kWh.	INT 2	2	0...65535	If the emissions are measured directly on the engine test bed the value is declared in g/kWh
	EmissionHCNOX	Exhaust emission of HCNOX, according to vehicle registration documents, in 10^{-3} g/km or g/kWh.	INT 2	2	0...65535	If the emissions are measured directly on the engine test bed the value is declared in g/kWh
DieselEmission Values	Particulate	Particulates for diesel, according to vehicle registration documents, in 10^{-3} g/km or g/kWh.	INTEGER(0…32766)	2		If the emissions are measured directly on the engine test bed the value is declared in g/kWh
	AbsorptionCoeff	Corrected absorption coefficient for diesel, according to vehicle registration documents, in 10^{-3} m^{-1}.	INT 2	2	0...65535	
CO2Emission Value	CO2EmissionValue	Vehicle's CO2 emission value according to vehicle registration documents, in g/km.	INT 2	2	0...65535	
VehicleTotal Distance	VehicleTotal Distance	Total distance as measured by the vehicle, in 10 meters resolution, continuously incremented.	INT4	4	0… 4294967294	The initial value of this attribute may be either the value zero or the vehicle's kilometer reading at the moment of personalisation of the OBU
TrailerLicence PlateNumber	TrailerLicencePlate Number	Claimed licence plate of the trailer	LPN	Variable		

Table 39 (*continued*)

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
Trailer Characteristics	TrailerDetails	Indication provided on trailer presence, type and number of axles.	TrailerDetails	1		5 bits are used for the trailer presence and type. 3 bits are used for the number of axles. If only one trailer is present, the presence and the number of axles of this single trailer is available in VehicleAxles and may not be included in this attribute.
	TrailerMaxLaden Weight	Maximum permissible total weight of the trailer including payload, which shall be according to ISO 1176. 10 kg units, rounded down to the next 10 kg step.	INT2	2		
	TrailerWeight Unladen	Nominal unladen weight of the trailer, which shall be according to ISO 1176 in 10 kg units, rounded down to the next 10 kg step.	INT2	2		
Vehicle Authenticator	VehicleAuthenticator	Authenticator calculated by the entity entering the data elements at time of entry or modification.	OCTET STRING	Variable		
European Directive 2003/127/EC provides the requirements on data available in the registration document for vehicle. When available, the data from the registration document should be used by the service provider for the encoding of the OBE.						
NOTE Since the first version of this International Standard, the definition of this data element has been changed. This requires taking into account the potential backward compatibility issues related to equipment and software compatible with the first generation of equipment.						

Table 40 — Data group Equipment

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
EquipmentOBUId	EquipmentOBUId	Unique Identification number of OBE within the context of the associated manufacturer.	OCTET STRING	Variable		The manufacturer ID is always exchanged as a part of the VST
EquipmentICC-Id	EquipmentICC-Id	Identification number of smart card	ICC-Id	Variable		Multiple instances shall be used for multiple smart cards
EquipmentStatus	EquipmentStatus	Operator-specific EFC application-related information pertaining to the status of the equipment	BIT STRING (SIZE(16))	2		Boolean information to support an operator's handling of an OBE on application level. (E.g. “next suitably equipped gantry should take an enforcement picture”).

Table 41 — Data group Driver

EFC Attribute	*Data Element*	*Definition*	*Type*	*Length in octet*	*Value Range*	*Informative remarks*
Driver Characteristics	DriverClass	Description of the driver's characteristics as pertinent to the calculation of the tariff; contract provider specific coding.	INT1	1	0...255	Information that may affect the tariff to be applied.
	TripPurpose	Parameter indicating the purpose of the trip of the user as pertinent to the calculation of the tariff; contract provider specific coding.	INT1	1	0...255	
ActualNumberOf Passengers	ActualNumberOf Passengers	Actual number of passengers (i.e. human beings) present in the vehicle, incl. the driver.	INT1	1	0...255	Information that may affect the applicability of tolls or the value of the tariff to be applied, e.g. in High Occupancy Tolling or High Occupancy Vehicle lanes.

Table 42 — Data group Payment

EFC Attribute	*Data element*	*Definition*	*Type*	*Length in octet*	*Value range*	*Informative remarks*
PaymentMeans	PersonalAccount Number	Coded according to financial institutions.	PersonalAccount Number	10		Personal account number shall be in accordance with ISO/IEC 7812-1.
	PaymentMeans ExpiryDate	Expiring date of payment means. Payment means expires at 24 h of PaymentMeans ExpiryDate..	DateCompact	2	[01.01.1990].. [31.12.2117]	
	PaymentMeans UsageControl	Indicates issuer's specified restrictions on the geographic usage and services allowed for the applications	OCTET STRING (SIZE(2))	2		
PaymentMeans Balance	PaymentMeans Balance	Balance of payment means in units of PaymentMeansUnit.	SignedValue	3	-8388608... +8388607	
PaymentMeans Unit	PaymentMeans Unit	The unit of the payment means value in multiples or fractions of a currency or in units of tokens.	PayUnit	2		
PaymentSecurity Data	PaymentSecurity Data	Security-related data for the authentication of the data integrity.	OCTET STRING	Variable		

Annex A
(normative)

EFC data type specifications

The EFC data types and associated coding related to the EFC action parameters, response parameters and attributes, described in Clauses 7 and 8, are defined using the Abstract Syntax Notation One (ASN.1) technique according to ISO/IEC 8824-1. The packed encoding rules according to ISO/IEC 8825-2 are applied.

```
EfcModule {iso standard 14906 modules(0) efc(0) version(1)} DEFINITIONS
AUTOMATIC TAGS
::= BEGIN
      IMPORTS CountryCode, CS5, IssuerIdentifier
      FROM  AVIAEINumberingAndDataStructures
            {iso(1) standard(0) 14816 }
            -- defined in ISO 14816 --
      Container, AttributeIdList, Attributes, AttributeList FROM DSRCData
            {iso standard 14906 modules (0) dsrc (1) version (1)};

-- NOTE: The following are the definitions of the action and response
-- parameters

ActualNumberOfPassengers ::= Int1

AxleWeightLimits ::= SEQUENCE{
      maxLadenweightOnAxle1  Int2,
      maxLadenweightOnAxle2  Int2,
      maxLadenweightOnAxle3  Int2,
      maxLadenweightOnAxle4  Int2,
      maxLadenweightOnAxle5  Int2
      }

AddRq::= SEQUENCE {
      attributeId       INTEGER(0..127,...),
      value             INTEGER
      }

ChannelId::= INTEGER {
      obu                     (0),
      sam1                    (1),  -- secure application module
      sam2                    (2),
      icc                     (3),  -- integrated circuit(s) card
      display                 (4),
      buzzer                  (5),
      printer                 (6),
      serialInterface         (7),  -- serial interface: eg. RS232 and RS485
      parallelInterface       (8),
      gPS                     (9),
      tachograph              (10),
      privateUse1             (11), -- free for proprietary use
      privateUse2             (12), -- free for proprietary use
      privateUse3             (13), -- free for proprietary use
      privateUse4             (14), -- free for proprietary use
      privateUse5             (15), -- free for proprietary use
      bluetooth               (16)
      -- (17-255) are reserved for future CEN use
      } (0..255)

ChannelRq::= SEQUENCE{
      channelId   ChannelId,
      apdu        OCTET STRING
                  -- format according to the interface
                  -- of the channelId
      }
```

```
ChannelRs::= SEQUENCE{
      channelId   ChannelId,
      apdu        OCTET STRING
                  -- format according to the interface
                  -- of the channelId
      }

CopyRq::= SEQUENCE {
      destinationEID    INTEGER(0..127,...),
      attributeIdList   AttributeIdList
      }

CreditRq::= SEQUENCE {
      refund      PaymentFee,
      nonce       OCTET STRING,
      key         INTEGER(0..255)
      }

CreditRs ::= SEQUENCE {
      creditResult           ResultFin,
      creditAuthenticator    OCTET STRING
      }

DebitRq::= SEQUENCE {
      debitPaymentFee        PaymentFee,
      nonce                  OCTET STRING,
      keyRef                 INTEGER(0..255)
      }

DebitRs ::= SEQUENCE {
      debitResult            ResultFin,
      debitAuthenticator     OCTET STRING
      }

GetInstanceRq ::= SEQUENCE {
      posOfFirstInstance           INTEGER(0..255),
            -- position of first instance to be retrieved
      posOfLastInstance            INTEGER(0..255),
            -- position last instance to be retrieved
      attributeIdList              AttributeIdList
            -- Ids of attributes to be retrieved
      }

GetInstanceRs::= SEQUENCE (SIZE (0..127,...)) OF SEQUENCE {
      attributeId INTEGER(0..127,...),
            -- number of instances retrieved
      attributeValues  Container (WITH COMPONENTS {octetstring PRESENT})
            -- The octetstring shall contain the contatenation of
            -- the unaligned PER encodings of the values of the
            -- instances, with each encoding padded to an integral
            -- of octets as specified for a top-level type in
            -- ISO/IEC 8825-2
      }

GetStampedRq::= SEQUENCE {
      attributeIdList  AttributeIdList,
      nonce            OCTET STRING, -- e.g. a random number
      keyRef           INTEGER(0..255)
      }

GetStampedRs::= SEQUENCE {
      attributeList    AttributeList,
      authenticator    OCTET STRING
      }

SetInstanceRq ::= SEQUENCE {
      posOfInstance    INTEGER(0..255),
      attribute        Attributes
      }
```

```
SetMMIRq::= INTEGER {
      ok    (0),        -- operation / transaction successfully completed
      nok   (1),        -- operation / transaction not successfully completed
      contactOperator   (2),  -- e.g. due to low balance or battery
      noSignalling      (255) -- no signalling
      -- (3-127) are reserved for future CEN use
      -- (128-254) are reserved for private use
      } (0..255)

SetStampedRq::= SEQUENCE {
      attributeList     AttributeList,
      nonce             OCTET STRING,
      keyRef            INTEGER(0..255)
      }

SubRq::= SEQUENCE {
      attributeId       INTEGER(0..127,...),
      value             INTEGER
      }

-- NOTE: The following are the definitions of EFC attributes

CO2EmissionValue ::= Int2

ContractSerialNumber ::= Int4

ContractAuthenticator ::= OCTET STRING

ContractValidity ::= SEQUENCE {
      contractRestrictions          OCTET STRING (SIZE(4)),
      contractExpiryDate            DateCompact
      }  -- intended to support ENV implemented systems

ContractVehicle ::= LPN

DateCompact::= SEQUENCE {
      year        INTEGER (1990..2117),
      month       INTEGER (0..12)  -- Value zero shall not be used
                                   -- except with 1990 - see below.
      day         INTEGER (0..31)  -- Value zero shall not be used
                                   -- except with 1990 - see below.
      }
      -- The value "{year 1990, month 0, day 0}" is a 16-bit all-zero
      -- encoding, and is used to represent "no date".

DescriptiveCharacteristics ::= INTEGER {
      noEntry                 (0),
      vehicleShape1           (1),
      vehicleShape2           (2),
      vehicleShape3           (3),
      vehicleShape4           (4),
      vehicleShape5           (5),
      vehicleShape6           (6),
      vehicleShape7           (7),
      vehicleShape8           (8),
      vehicleShape9           (9),
      vehicleShape10          (10),
      vehicleShape11          (11),
      vehicleShape12          (12),
      vehicleShape13          (13),
      vehicleShape14          (14),
      vehicleShape15          (15),
      vehicleShape16          (16),
      vehicleShape17          (17),
      vehicleShape18          (18),
      vehicleShape19          (19),
      vehicleShape20          (20),
      vehicleShape21          (21),
      vehicleShape22          (22),
      vehicleShape23          (23),
      vehicleShape24          (24),
```

```
        vehicleShape25          (25),
        vehicleShape26          (26),
        vehicleShape27          (27),
        vehicleShape28          (28),
        vehicleShape29          (29),
        vehicleShape30          (30),
        vehicleShape31          (31),
        vehicleShape32          (32),
        vehicleShape33          (33),
        vehicleShape34          (34),
        vehicleShape35          (35),
        vehicleShape36          (36),
        vehicleShape37          (37),
        vehicleShape38          (38),
        vehicleShape39          (39),
        vehicleShape40          (40),
        vehicleShape41          (41),
        vehicleShape42          (42),
        vehicleShape43          (43),
        vehicleShape44          (44),
        vehicleShape45          (45),
        vehicleShape46          (46),
        vehicleShape47          (47),
        vehicleShape48          (48),
        vehicleShape49          (49),
        vehicleShape50          (50)
        -- (51..255) are reserved for future CEN use
        } (0..255)  -- vehicle shape x as defined in prENV/278/8/1/5 for silhouette

DieselEmissionValues ::= SEQUENCE {
        particulate SEQUENCE{
                unitType    ENUMERATED  {
                        g-km  (0),
                        g-kWh (1)
                        },
                value       INTEGER (0..32766)
                },
                absorptionCoeff   Int2
        }

DriverCharacteristics ::= SEQUENCE {
        driverClass       Int1,
        tripPurpose       Int1
        }

EFC-ContextMark ::= SEQUENCE {
        contractProvider        Provider,
        typeOfContract          OCTET STRING (SIZE(2)),
        contextVersion          INTEGER(0..127,...)
        }

EnvironmentalCharacteristics::= SEQUENCE {
        euroValue         ENUMERATED {
                noEntry                 (0),
                euro-1                  (1),
                euro-2                  (2),
                euro-3                  (3),
                euro-4                  (4),
                euro-5                  (5),
                euro-6                  (6),
                reservedForUse1         (7)
                },    -- 4 bits, EURO-Clases as defined in EC directive 88/77/EEC, annex 1
                      -- and in 91/542/EEC, 96/1/EC, 1999/96/EC, 2001/27/EC
        copValue          ENUMERATED {
                noEntry                 (0),
                co2class1               (1), -- below 101 g/km
                co2class2               (2), -- 101 to 120 g/km
                co2class3               (3), -- 121 to 140 g/km
                co2class4               (4), -- 141 to 160 g/km
```

```
            co2class5               (5), -- 161 to 200 g/km
            co2class6               (6), -- 201 to 250 g/km
            co2class7               (7) -- above 250 g/km
            } -- 4 bits, reserved for carbon dioxide pollution values as defined in
              -- EC directive 2003/127/EC
      }

EngineCharacteristics::= INTEGER {
      noEntry           (0),
      noEngine          (1),
      petrolUnleaded    (2),
      petrolLeaded      (3),
      diesel            (4),
      lPG               (5),
      battery           (6),
      solar             (7)
      -- (8-255) are reserved for future CEN use
      } (0..255)

Engine ::= SEQUENCE{
      engineCapacity    Int2,
      enginePower       Int2
      }

EquipmentOBUId ::= OCTET STRING

EquipmentStatus ::= BIT STRING (SIZE(16))

ExhaustEmissionValues ::= SEQUENCE {
      unitType    ENUMERATED {
            g-km  (0),
            g-kWh (1)
            },
      emissionCO        INTEGER (0..32766),
      emissionHC        Int2,
      emissionNOX       Int2,
      emissionHCNOX     Int2
      }

FutureCharacteristics ::= INTEGER {
      noEntry           (0),
      airSuspension     (1)
      -- (2..255) are reserved for future CEN use
      } (0..255)

ICC-Id ::= OCTET STRING

Int1 ::= INTEGER(0..255)

Int2 ::= INTEGER(0..65535)

Int3 ::= INTEGER(0..16777215)

Int4 ::= INTEGER(0..4294967295)

LPN::= SEQUENCE {
      countryCode             CountryCode,
      alphabetIndicator       ENUMERATED {
            latinAlphabetNo1        (1),  -- encoded as 00 00 00'B
            latinAlphabetNo2        (2),  -- encoded as 00 00 01'B etc
            latinAlphabetNo3        (3),
            latinAlphabetNo4        (4),
            latinCyrillicAlphabet   (5),
            latinArabicAlphabet     (6),
            latinGreekAlphabet      (7),
            latinHebrewAlphabet     (8),
            latinAlphabetNo5        (9),
            latinAlphabetNo6        (10),
            twoOctetBMP             (11),
            fourOctetCanonical      (12),
            reservedForUse1         (13),
            reservedForUse2         (14),
```

```
            reservedForUse3         (15),
            reservedForUse4         (16),
            reservedForUse5         (17),
            reservedForUse6         (18),
            reservedForUse7         (19),
            reservedForUse8         (20),
            reservedForUse9         (21),
            reservedForUse10        (22),
            reservedForUse11        (23),
            reservedForUse12        (24),
            reservedForUse13        (25),
            reservedForUse14        (26),
            reservedForUse15        (27),
            reservedForUse16        (28),
            reservedForUse17        (29),
            reservedForUse18        (30),
            reservedForUse19        (31),
            reservedForUse20        (32),
            reservedForUse21        (33)
            } -- 6 bits, latinAlphabetNo1 recommended -- ,
              -- refer to Annex E for conversion from LatinAlphabetNo 2
              -- and 5 to Latin AlphabetNo1
      licencePlateNumber     OCTET STRING
      }

PassengerCapacity ::= SEQUENCE{
      numberOfSeats                Int1,
      numberOfStandingPlaces       Int1
      }

PaymentFee ::= SEQUENCE {
      -- The fee (toll, charge or fare) which is requested by the
      -- service provider for the service provided or to be provided.
      paymentFeeAmount       Int2,
      -- paymentFeeAmount is the value of the fee being charged for the
      -- service. If no unit (payment fee unit) is specified, then
      -- it is known by default.
      paymentFeeUnit         PayUnit
      -- paymentFeeUnit is the unit in which the fee is expressed.
      }

PaymentMeans ::= SEQUENCE {
      personalAccountNumber        PersonalAccountNumber,
      paymentMeansExpiryDate       DateCompact,
      pamentMeansUsageControl      OCTET STRING(SIZE(2))
            -- issuer's specified restrictions, on the geographic usage
            -- and services allowed for the applications
      }

PaymentMeansBalance ::= SignedValue


SignedValue ::= CHOICE {
positive INTEGER (0..8388607),
negative INTEGER (-8388608..-1)
      }
      -- corresponds to a "3 octets Signed Integer", associated with the following
      -- examples of line codes:
      -- -8'388'608 : 80 00 00'H
      -- -1 : FF FF FF'H
      --  0 : 00 00 00'H
      --  1 : 00 00 01´H
      --  8'388'607 : 7F FF FF'H


PaymentMeansUnit ::= PayUnit

PaymentSecurityData ::= OCTET STRING
```

```
PayUnit ::= OCTET STRING (SIZE(2))
      -- The unique designation of a Currency as defined in ISO 4217
      -- using the ISO numeric binary coded decimal representation.
      -- The code can also express a company specific token or a
      -- "charging unit code" as used in the freight.unit in which
      -- the fee is expressed.
      -- Value Assignment :
            -- '0xxx'H  Currency in main units
            -- '1xxx'H  Currency in minor units of 10 :1 ('dime')
            -- '2xxx'H  Currency in minor units of 100 :1 ('cents')
            -- '3xxx'H  Currency in minor units of 1000 :1
            -- '4xxx'H  Currency in 'major' units / 10
                        -- (e.g. 10 Belgian Francs)
            -- '5xxx'H  Currency in 'major' units / 100
                        -- (e.g. 100 Italian Lire)
            -- '6xxx'H  Currency in 'major' units / 1000
            -- '7xxx'H  Currency in 'major' units / 10000
            -- '8xxx'H  Currency in 'major' units / 100000
                  -- where xxx is the BCD representation of "Currency"
                  -- as defined in ISO 4217
            -- '9xxx'H  Tokens
                  -- where xxx is Purse Provider specific coding.
            -- 'Axxx'H  Charging Unit Codes,
            -- denoting quantification of the service provided
            -- (e.g. man-hours)

PersonalAccountNumber ::= OCTET STRING (SIZE(10))
      -- Personal account number structure - according to ISO/IEC 7812-1
            -- Issuer identifier number ("BIN")
                  -- Major industry identifier (MII, 1 binary coded decimal, BCD)
                        -- 0 : reserved for future use by ISO/TC68
                        -- 1 : airline sector
                        -- 2 : extended airline sector
                        -- 3 : travel and tourism sector
                        -- 4 : financial banking sector
                        -- 5 : financial banking sector
                        -- 6 : commerce and banking sector
                        -- 7 : petrol industry sector
                        -- 8 : telecommunication sector
                        -- 9 : reserved for national use
                  -- Issuer identifier (5 BCD in the second edition of ISO/IEC 7812-1)
      -- Account number (max 12 BCD)
      -- Control digit (1 BCD)
      -- Padding bits, set to 1'B, in order to accomplish a
      -- total length of 10 octets.

Provider ::= SEQUENCE {
      countryCode             CountryCode,
      providerIdentifier      IssuerIdentifier
      }

PurseBalance ::=SEQUENCE  {
            -- The balance on the (electronic) purse, consisting of
            -- the value and the unit in which it is expressed.
      purseValue        SignedValue,
            -- The size of a balance expressed in a currency.
            -- This may be positive or negative.
      purseUnit         PayUnit
      }

ReceiptContract ::= SEQUENCE {
      sessionContractProvider      Provider,
      sessionTypeOfContract        OCTET STRING(SIZE(2)),
      sessionContractSerialNumber  Int4
      }
```

```
ReceiptData1 ::= ReceiptData

ReceiptData2 ::= ReceiptData

ReceiptData ::= SEQUENCE {
      sessionTime                   DateAndTime,
      sessionServiceProvider        Provider,
      locationOfStation             Int2,
      sessionLocation               SessionLocation,
      sessionType                   Int1,
      sessionResult                 ResultOp,
      sessionTariffClass            Int1,
      sessionClaimedClass           Int1,
      sessionFee                    PaymentFee,
      sessionContractProvider       Provider,
      sessionTypeOfContract         OCTET STRING (SIZE(2)),
      sessionContextVersion         INTEGER (0..127,...),
      receiptDataAuthenticator      OCTET STRING(SIZE(4))
      }

ReceiptDistance ::= Int3

ReceiptFinancialPart ::= SEQUENCE {
      personalAccountNumber             PersonalAccountNumber,
      sessionPaymentFee                 PaymentFee,
      sessionCurrentBalance             PurseBalance,
      receiptFinancialSerialNumber      Int4
      }

ReceiptICC-Id ::= ICC-Id

ReceiptOBUId ::= OCTET STRING

ReceiptServicePart ::= SEQUENCE {
      sessionTime                   DateAndTime,
      sessionServiceProvider        Provider,
      stationLocation               INTEGER(0..1048575),
      sessionLocation               BIT STRING (SIZE(8)),
      typeOfSession                 StationType,
      sessionResultOperational      ResultOp,
      sessionResultFinancial        ResultFin
      }

ReceiptServiceSerialNumber ::= Int3

ReceiptAuthenticator ::= OCTET STRING

ReceiptText ::= OCTET STRING

ResultFin ::= OCTET STRING (SIZE(1))
      -- A code designating whether a card transaction was completed successfully
      -- or not. Value Assignment : Hexadecimal
      -- Most significant 4 bits: 0 OK :
      -- '0x'H OK
      -- Most significant 4 bits > 0 Not OK :
      -- '1x'H Not OK, not specified further
      -- '2x'H Not OK, Abnormal (First or Previous) Event
      -- '3x'H Not OK, Contract not accepted
      -- '4x'H Not OK, Account or Purse not accepted
      -- 'x0'H not specified further
      -- 'x1'H Balance close to zero
      -- 'x2'H Balance now negative
      -- 'x3'H Balance Overflow
      -- 'x4'H Provider not accepted
      -- 'x5'H Authentication failure
      -- x6'H Vehicle Class incorrect
```

```
ResultOp ::= INTEGER {
      correctTransaction                              (0), -- transaction correct
      obeStatusNotAccepted                            (1),
      equipmentStatusNotAccepted                      (2),
      contractNotInWhiteList                          (3),
                        -- VST contract data not in white list
      contractIdentifierInBlackList                   (4),
      contractIdentifierNotCorrect                    (5),
                        -- Luhn algorithm verification failure
      expiredContract                                 (6), -- contract expired
      contractRestrictionsNotFulfilled                (7),
      claimedVehicleCharacteristicsNotValid           (8),
      vehicleClassAuthenticationFailed                (9),
      entryVehicleClassDifferentFromExitVehicleClass (10),
      entryReceiptMissing                             (11),
      entryReceiptNotValid                            (12),
      entryTollStationNotValid                        (13),
      equipmentNotCertified                           (14),
                        -- manufacturer or EquipClass not recognised
      timeDifference                                  (15),
                        -- problem with the time diff of the two latest receipts
      accessCredentialsNotAccepted                    (16),
      contractAuthenticatorNotAccepted                (17),
      receiptAuthenticatorNotAccepted                 (18),
      claimedVehicleCharacteristicsMissing            (19),
      paymentMeansNotAccepted                         (20),
      paymentAuthenticatorNotAccepted                 (21)
      paymentMeansInBlackList                         (22),
      PaymentMeansNotCorrect                          (23),
                        -- Luhn algorithm verification failure
      expiredPaymentMeans                             (24),
                        -- PaymentMeans expired
      PaymentMeansRestrictionsNotFulfilled            (25),
      -- (25-255) are reserved for future CEN use
      } (0..255)

SessionClass ::= SEQUENCE {
      sessionTariffClass      Int1,
      sessionClaimedClass     Int1
      }

SessionLocation ::= SEQUENCE {
      ascendingKilometrage          BOOLEAN,    -- travel direction indicator
      laneCodeNumber                INTEGER(0..127) -- lane code number
      }

StationType ::= ENUMERATED {
      unspecified                   (0),
      closedEntryWithPayment        (1),
      closedEntryWithoutPayment     (2),
      closedTransit                 (3),
      closedExit                    (4),
      closedCredit                  (5),
      mixed                         (6),
      passage                       (7), -- open exit
      checkpoint                    (8),
      reload                        (9),
      reservedForFutureCENUse1      (10),
      reservedForFutureCENUse2      (11),
      reservedForFutureCENUse3      (12),
      reservedForFutureCENUse4      (13),
      privateUse5                   (14),
      privateUse6                   (15)
      }
```

```
DateAndTime ::= SEQUENCE {
      timeDate          DateCompact,
      timeCompact       SEQUENCE {
                        -- expresses time of the day in
                        -- hours, min, and sec
                        hours             INTEGER (0..23),
                              -- number of hours after midnight
                        mins              INTEGER (0..59),
                              -- number of minutes after the hour
                        double-secs       INTEGER (0..30)
                              -- number of two-seconds after the minute
                        }
      -- Midnight at the start of a day cannot be represented.
      -- Midnight at the end of a day is represented by
      -- {hours 23, mins 59, double-secs 30}
      -- The 16 bit zero value {hours 0, mins 0, double-secs 0}
      -- denotes "no time"
      }

SoundLevel ::= SEQUENCE{
      soundstationary   Int1,
      sounddriveby      Int1
      }

TrailerCharacteristics ::= SEQUENCE {
      trailerDetails              TrailerDetails,
      trailerMaxLadenWeight       Int2,
      trailerWeightUnladen        Int2
      }

TrailerDetails::= SEQUENCE {
      trailerType       INTEGER{
            notPresent  (0), -- trailer not attached or only one trailer attached,
                             see
                             -- VehicleAxlesNumber for more information
            trailer     (1), -- also known as pull-bar trailer
            semitrailer (2) -- also known as articulate trailer
            -- (3..31) reserved for future CEN/ISO use
            } (0..31),
      trailerAxles      TrailerAxles
      }

TrailerLicencePlateNumber ::= LPN

ValidityOfContract ::= SEQUENCE {
      issuerRestrictions      OCTET STRING (SIZE(2)),
      contractExpiryDate      DateCompact
      }

VehicleAuthenticator ::= OCTET STRING

VehicleAxles ::= SEQUENCE {
      vehicleFirstAxleHeight        Int1,
      vehicleAxlesNumber            SEQUENCE {
            tyreType                ENUMERATED{
                  notSpecified      (0),
                  singleTyre        (1),  -- single tyre on all axles
                  dualTyres         (2),  -- dual tyres on at least one axle
                  reservedForUse    (3)   -- reserved for future CEN use
                  },
            numberOfAxles           SEQUENCE {
                  trailerAxles      TrailerAxles,
                  tractorAxles      TractorAxles
                  }
            }
      }

TrailerAxles ::= INTEGER (0..7)   -- number of axles of the trailer when available

TractorAxles ::= INTEGER (0..7)   -- number of axles of the tractor
```

```
VehicleClass ::= Int1

VehicleDimensions ::= SEQUENCE {
      vehicleLengthOverall    Int1,
      vehicleHeigthOverall    Int1,
      vehicleWidthOverall     Int1
      }

VehicleLicencePlateNumber ::= LPN

VehicleIdentificationNumber ::= CS5

VehicleSpecificCharacteristics ::= SEQUENCE {
      environmentalCharacteristics      EnvironmentalCharacteristics,
      engineCharacteristics             EngineCharacteristics,
      descriptiveCharacteristics        DescriptiveCharacteristics,
      futureCharacteristics             FutureCharacteristics
      }

VehicleTotalDistance ::= Int4

VehicleWeightLaden ::= Int2

VehicleWeightLimits ::= SEQUENCE {
      vehicleMaxLadenWeight        Int2,
      vehicleTrainMaximumWeight    Int2,
      vehicleWeightUnladen         Int2
      }

END

DSRCData {iso standard 14906 modules (0) dsrc (1) version (1)}
      DEFINITIONS AUTOMATIC TAGS ::= BEGIN
      EXPORTS ALL;
      IMPORTS GetStampedRq,GetStampedRs,SetStampedRq,GetInstanceRq,
      GetInstanceRs,SetInstanceRq,SetMMIRq,ChannelRq,ChannelRs,CopyRq,
      SubRq,AddRq,DebitRq,DebitRs,CreditRq,CreditRs,
      EFC-ContextMark,ContractSerialNumber,ContractValidity,
      ContractVehicle,ContractAuthenticator,ICC-Id,
      ReceiptServicePart,SessionClass,SignedValue,ReceiptServiceSerialNumber,
      ReceiptFinancialPart,ReceiptContract,ReceiptOBUId,ReceiptICC-Id,
      ReceiptText,ReceiptAuthenticator,ReceiptDistance,LPN,VehicleClass,
      VehicleDimensions,VehicleAxles,VehicleWeightLimits,
      VehicleWeightLaden,VehicleSpecificCharacteristics,
      VehicleAuthenticator,EquipmentOBUId,
      EquipmentStatus,DriverCharacteristics,
      PaymentMeans,PaymentMeansBalance,PaymentMeansUnit,
      PaymentSecurityData,PersonalAccountNumber,ReceiptData1,
      ReceiptData2,SessionLocation,ValidityOfContract, AxleWeightLimits,
      PassengerCapacity, Engine, SoundLevel, ExhaustEmissionValues,
      DieselEmissionValues, CO2EmissionValue, VehicleTotalDistance,
      TrailerLicencePlateNumber, TrailerCharacteristics, ActualNumberOfPassengers
      FROM EfcModule
                  {iso standard 14906 modules (0) efc (0) version (1)}
      CS5
      FROM  AVIAEINumberingAndDataStructures
            {iso(1) standard(0) 14816 };
            -- defined in ISO 14816 --
      -- EXPORTS everything;

Action-Request::= SEQUENCE{
      mode                    BOOLEAN,
      eid                     Dsrc-EID,
      actionType              ActionType,
      accessCredentials       OCTET STRING (SIZE (0..127,...))
                                    OPTIONAL,
      actionParameter         Container   OPTIONAL,
      iid                     Dsrc-EID    OPTIONAL
      }
```

```
Action-Response::= SEQUENCE{
      fill                    BIT STRING (SIZE(1)),
      eid                     Dsrc-EID,
      iid                     Dsrc-EID          OPTIONAL,
      responseParameter       Container         OPTIONAL,
      ret                     ReturnStatus      OPTIONAL
      }

ActionType::=INTEGER (0..127,...)
      --  0 : getStamped
      --  1 : setStamped
      --  2 : getSecure
      --  3 : setSecure
      --  4 : getInstance
      --  5 : setInstance
      --  6 : getNonce
      --  7 : setNonce
      --  8 : transferChannel
      --  9 : copy
      -- 10 : setMMI
      -- 11 : substract
      -- 12 : add
      -- 13 : debit
      -- 14 : credit
      -- 15 : echo
      -- (16..118) Reserved for ISO/CEN use
      -- (119-127) Reserved for private use

ApplicationContextMark::=Container
      (WITH COMPONENTS {octetstring           PRESENT})
      -- The contents of the octetstring shall be an aligned PER
      -- encoding of EFC-Contextmark, but this encoding may be followed
      -- by non-standardised encodings. Illustrations of the inclusion
      -- of non-standardised encodings are shown in annex B.

ApplicationList::=SEQUENCE (SIZE (0..127,...)) OF
      SEQUENCE {
            aid         DSRCApplicationEntityID,
            eid         Dsrc-EID                OPTIONAL,
            parameter   ApplicationContextMark OPTIONAL
      }

AttributeIdList::=SEQUENCE (SIZE(0.. 127,...)) OF INTEGER(0..127,...)

AttributeList::=SEQUENCE (SIZE(0..127,...)) OF Attributes

Attributes::=SEQUENCE {
      attributeId     INTEGER (0..127,...),
      attributeValue  Container
      }

BeaconID::=SEQUENCE{
      manufacturerid    INTEGER(0.. 65535),
      individualid      INTEGER(0..134217727)
      } -- for registration of manufacturerid see  www.nni.nl/cen278

BroadcastPool::=SEQUENCE{
      directoryvalue    Directory,
      content           SEQUENCE (SIZE(0..127,...)) OF File
      }

BST::=SEQUENCE{
      rsu                     BeaconID,
      time                    Time,
      profile                 Profile,
      mandApplications        ApplicationList,
      nonmandApplications     ApplicationList  OPTIONAL,
      profileList             SEQUENCE (SIZE(0..127,...)) OF Profile
      }
```

```
Container::=CHOICE{
      -- The alternative for container and its value is determined
      -- from the service primitives.
      integer           [0]   INTEGER,
      bitstring         [1]   BIT STRING,
      octetstring       [2]   OCTET STRING (SIZE (0..127), ...),
      universalString   [3]   UniversalString,
      beaconId          [4]   BeaconID,
      t-apdu            [5]   T-APDUs,
      dsrcApplicationEntityId      [6]   DSRCApplicationEntityID,
      dsrc-Ase-Id       [7]   Dsrc-EID,
      attrIdList        [8]   AttributeIdList,
      attrList          [9]   AttributeList,
      broadcastPool     [10]  BroadcastPool,
      directory         [11]  Directory,
      file              [12]  File,
      fileType          [13]  FileType,
      record            [14]  Record,
      time              [15]  Time,
      vector            [16]  SEQUENCE (SIZE(0..255)) OF
                              INTEGER(0..127,...),
      gstrq             [17] GetStampedRq,
      gstrs             [18] GetStampedRs,
      sstrq             [19] SetStampedRq,
      ginrq             [20] GetInstanceRq,
      ginrs             [21] GetInstanceRs,
      sinrq             [22] SetInstanceRq,
      charq             [23] ChannelRq,
      chars             [24] ChannelRs,
      cpprq             [25] CopyRq,
      subrq             [26] SubRq,
      addrq             [27] AddRq,
      debrq             [28] DebitRq,
      debrs             [29] DebitRs,
      crerq             [30] CreditRq,
      crers             [31] CreditRs,
      efccontext        [32] EFC-ContextMark,
      contser           [33] ContractSerialNumber,
      contval           [34] ContractValidity,
      contveh           [35] ContractVehicle,
      contauth          [36] ContractAuthenticator,
      recspt            [37] ReceiptServicePart,
      sessioncls        [38] SessionClass,
      recservserialno   [39] ReceiptServiceSerialNumber,
      recfinptENV       [40] NULL,
      reccont           [41] ReceiptContract,
      recOBUId          [42] ReceiptOBUId,
      recICCId          [43] ReceiptICC-Id,
      rectext           [44] ReceiptText,
      recauth           [45] ReceiptAuthenticator,
      recdist           [46] ReceiptDistance,
      vehlpn            [47] LPN, -- vehicle licence plate number
      vehid             [48] CS5, -- vehicle identification number
      vehclass          [49] VehicleClass,
      vehdims           [50] VehicleDimensions,
      vehaxles          [51] VehicleAxles,
      vehwtlims         [52] VehicleWeightLimits,
      vehwtladen        [53] VehicleWeightLaden,
      vehspchars        [54] VehicleSpecificCharacteristics,
      vehauth           [55] VehicleAuthenticator,
      equOBUId          [56] EquipmentOBUId,
      equICCId          [57] ICC-Id,
      equstat           [58] EquipmentStatus,
      dvrchars          [59] DriverCharacteristics,
      paymeansENV       [60] NULL,
      paymbal           [61] PaymentMeansBalance,
      paymunit          [62] PaymentMeansUnit,
      paysecdata        [63] PaymentSecurityData,
```

```
        paymeans          [64] PaymentMeans,
        recdata1          [65] ReceiptData1,
        recdata2          [66] ReceiptData2,
        valofcon          [67] ValidityOfContract,
        recfinpt          [68] ReceiptFinancialPart,
        setmmirq          [69] SetMMIRq,
        awl               [70]  AxleWeightLimits,
        paca              [71]  PassengerCapacity,
        eng               [72]  Engine,
        sl                [73]  SoundLevel,
        eev               [74]  ExhaustEmissionValues,
        dev               [75]  DieselEmissionValues,
        co2ev             [76]  CO2EmissionValue,
        vtd               [77]  VehicleTotalDistance,
        tlpn              [78]  TrailerLicencePlateNumber,
        tch               [79]  TrailerCharacteristics,
        anp               [80]  ActualNumberOfPassengers,
        rfuCenISO48       [81] NULL,
        rfuCenISO49       [82] NULL,
        rfuCenISO50       [83] NULL,
        rfuCenISO51       [84] NULL,
        rfuCenISO52       [85] NULL,
        rfuCenISO53       [86] NULL,
        -- Container CHOICE type values [81..86]are reserved for
        -- attribute Ids 48 to 53 which are used in CCC
        -- Container CHOICE type values [87..127] are reserved for private EFC use and
        -- intended for the addressing of the corresponding private
        -- attribute identifiers. Note that container type 87 is also used in LAC
        ...         -- extension marker
        }

FileType ::= NULL  -- not defined

Directory::= SEQUENCE (SIZE(0..127,...)) OF FileName

Dsrc-EID::=INTEGER(0..127, ...)

DSRCApplicationEntityID::=INTEGER {
        system                          (0),
        electronic-fee-collection       (1),
        freight-fleet-management        (2),
        public-transport                (3),
        traffic-traveller-information   (4),
        traffic-control                 (5),
        parking-management              (6),
        geographic-road-database        (7),
        medium-range-preinformation     (8),
        man-machine-interface           (9),
        intersystem-interface           (10),
        automatic-vehicle-identification (11),
        emergency-warning               (12),
        private                         (13),
        multi-purpose-payment           (14),
        dsrc-resource-manager           (15),
        after-theft-systems             (16)
        -- (17..28) are reserved for ISO/CEN DSRC applications
        -- (29..30) are reserved for private use
        -- 31 is reserved for ISO/CEN-DSRC-applications
        }(0..31,...)
        -- For the latest standard use of application definition, refer
        -- to www.nni.nl/cen278
        -- As an example, the application "electronic-fee-collection (1)"
        -- is standardised by EN/ISO 14906
```

```
Event-Report-Request::= SEQUENCE{
            mode                    BOOLEAN,
            eid                     Dsrc-EID,
            eventType               EventType,
            accessCredentials       OCTET STRING (SIZE(0..127,...)) OPTIONAL,
            eventParameter          Container   OPTIONAL,
            iid                     Dsrc-EID    OPTIONAL
      }

Event-Report-Response::= SEQUENCE{
            fill        BIT STRING (SIZE(2)),
            eid         Dsrc-EID,
            iid         Dsrc-EID    OPTIONAL,
            ret         ReturnStatus     OPTIONAL
      }

EventType::=INTEGER{
                  release     (0)
            -- (1..118) are reserved for ISO/CEN use
            -- (119..127) are reserved for private use
      }(0..127,...)

File::=SEQUENCE(SIZE(0..127,...)) OF Record

FileName::=SEQUENCE{
      aseID       Dsrc-EID,
      fileID      INTEGER(0..127,...)
      }

Get-Request::=SEQUENCE{
      fill                    BIT STRING (SIZE(1)),
      eid                     Dsrc-EID,
      accessCredentials       OCTET STRING (SIZE(0..127,...)) OPTIONAL,
      iid                     Dsrc-EID          OPTIONAL,
      attrIdList              AttributeIdList   OPTIONAL
      }

Get-Response::=SEQUENCE{
      fill              BIT STRING (SIZE(1)),
      eid               Dsrc-EID,
      iid               Dsrc-EID          OPTIONAL,
      attributelist     AttributeList     OPTIONAL,
      ret               ReturnStatus      OPTIONAL
      }

Initialisation-Request::= BST

Initialisation-Response::= VST

NamedFile::=SEQUENCE{
      name        FileName,
      file        File
      }

ObeConfiguration::=SEQUENCE{
      equipmentClass    INTEGER(0..32767),
      manufacturerID    INTEGER(0..65535),
      obeStatus         INTEGER(0..65535) OPTIONAL
      -- obeStatus shall always be present. Bit nr 5 of the first octet may indicate
      the
      -- status of the battery: 0 indicates OK, 1 indicates low (xxxB xxxx'H)
      }

Profile::= INTEGER {
      profile0                (0),  -- DSRC Profile 0 as defined in EN 13372
      profile1                (1)   -- DSRC Profile 1 as defined in EN 13372
      -- (2..118) are reserved for ISO/CEN use,
      -- (119..127) are reserved for private use
      }(0..127,...)
```

```
Record::=CHOICE{simple VisibleString,
      ...
      }

ReturnStatus::=INTEGER {
      noError                 (0),
      accessDenied            (1),
      argumentError           (2),
      complexityLimitation    (3),
      processingFailure       (4),
      processing              (5),
      chainingError           (6)
      -- (7..127) are reserved for future CEN use
      }(0..127,...)

Set-Request::=SEQUENCE{
      fill                    BIT STRING (SIZE(1)),
      mode                    BOOLEAN,
      eid                     Dsrc-EID,
      accessCredentials       OCTET STRING (SIZE(0..127,...)) OPTIONAL,
      attrList                AttributeList,
      iid                     Dsrc-EID OPTIONAL
      }

Set-Response::=SEQUENCE{
      fill              BIT STRING (SIZE(2)),
      eid               Dsrc-EID,
      iid               Dsrc-EID OPTIONAL,
      ret               ReturnStatus OPTIONAL
      }

Time::=INTEGER(0..4294967295)
      -- "UNIX time": number of seconds since midnight at the
      -- start of 1st January 1970

T-APDUs::= CHOICE{
      action-request              [0]   Action-Request,
      action-response             [1]   Action-Response,
      event-report-request        [2]   Event-Report-Request,
      event-report-response       [3]   Event-Report-Response,
      set-request                 [4]   Set-Request,
      set-response                [5]   Set-Response,
      get-request                 [6]   Get-Request,
      get-response                [7]   Get-Response,
      initialisation-request      [8]   Initialisation-Request,
      initialisation-response     [9]   Initialisation-Response
      }

VST::=SEQUENCE{
      fill              BIT STRING (SIZE(4)),
      profile           Profile,
      applications      ApplicationList,
      obeConfiguration  ObeConfiguration
      }

END
-- Below imported data from ISO 14816's ASN.1 module
      -- AVIAEINumberingAndDataStructures {iso(1) standard(0) 14816 }
      -- DEFINITIONS AUTOMATIC TAGS ::= BEGIN
      -- EXPORTS ALL;

      -- CS5 ::= VisibleString

      -- CountryCode ::= BIT STRING(SIZE(10))
            -- Value assignment is done in accordance with ISO 3166-1 and by
            -- using the ITA.2 alphabet.

      -- IssuerIdentifier ::= INTEGER(0 .. 16383)
            -- See Annex A of ISO 14816 for registration
      -- END
```

Annex B
(informative)

CARDME transaction

B.1 General

Annex B provides an informative example of a transaction by specifying the CARDME transaction. First a transaction overview is given in B.2. The transaction phases and the data exchanges are subsequently defined in B.3. Finally, the bit-level specification is accounted for in B.4.

B.2 Overview

B.2.1 The four phases

B.2.1.1 General

When a user enters a manual tolling station, four phases can be discerned. The electronic CARDME transaction consists of the same four phases, as shown in Table B.1.

Table B.1 — CARDME's four transaction phases

Phase	Icon	Short description
Initialisation		**"Hello, welcome, where do you come from, how do you want to pay"** Negotiation of the EFC contract to use.
Presentation		**"Please give me your payment details and your entry ticket"** The RSE reads OBE data (details on contract, account, vehicle classification, last transaction, etc.).
Receipt		**"Here is your receipt"** The RSE writes an electronic receipt (which may also serve as an entry ticket).
Tracking and closing		**"Thank you and good bye"** The RSE tracks the vehicle through the communication zone and eventually closes the transaction.

Irrespective of EFC station type (passage in open system, entry or exit in closed system) the transaction performed is always the same. Although the functionality of the different station types is quite different, there is a single CARDME transaction, which is identical at all locations.

B.2.1.2 Initialisation - Say Hello

EFC beacons continually emit a signal in order to make contact to newly approaching vehicles. The data in this periodic signal is called the Beacon Service Table (BST).

As soon as a vehicle receives a BST, it answers with its Vehicle Service Table (VST). The VST contains a list of all EFC-contracts present in the OBE.

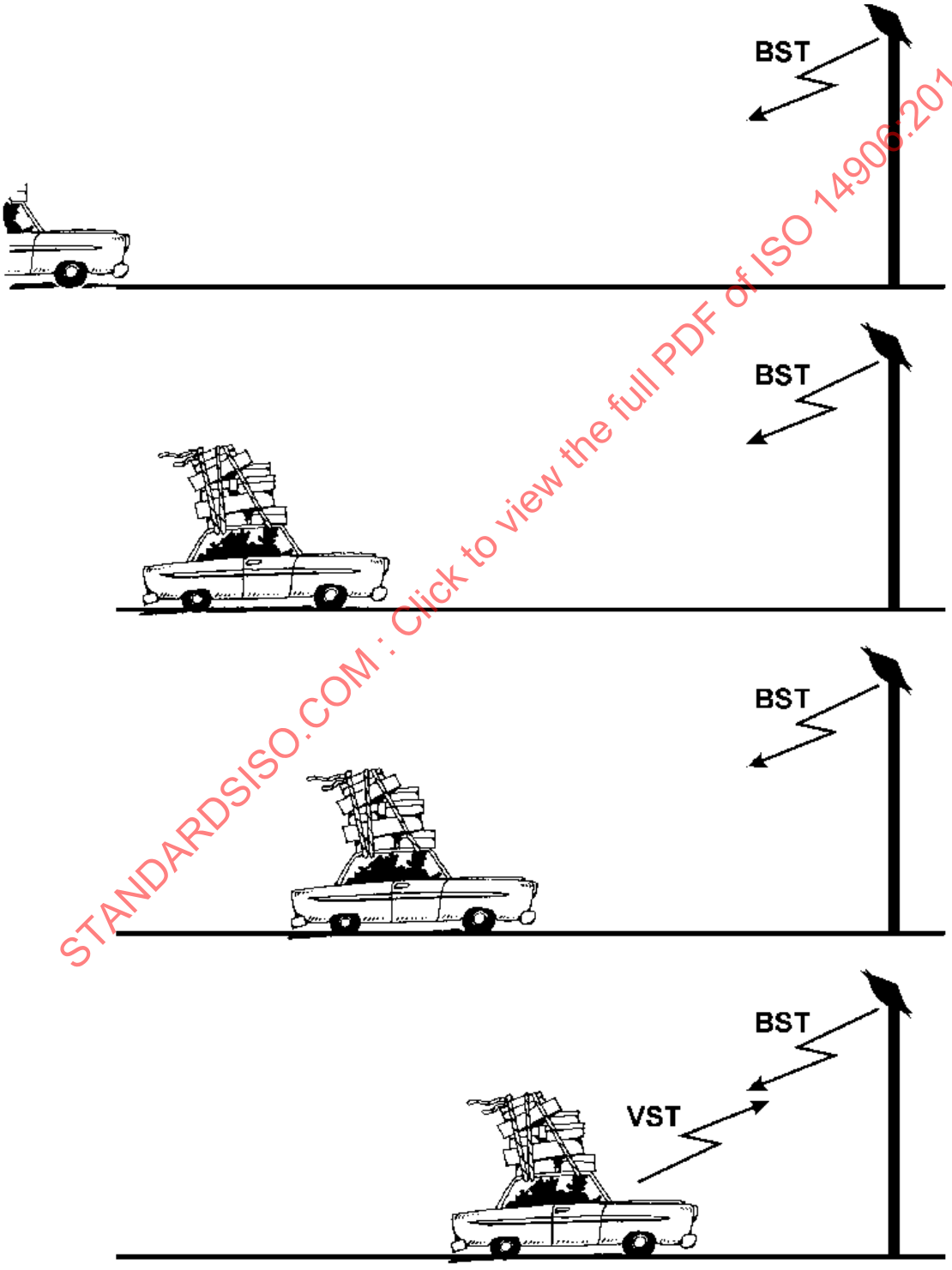

Figure B.1 — Initialisation

Upon reception of the VST the RSE analyses its contents and decides whether it accepts one of the EFC contracts presented by the OBE. In case the RSE accepts a contract, it knows exactly what to do from then on. The RSE knows which organization has issued the contract, and hence, where to send the claim and which transaction type is supported by the OBE. Although the RSE may have software available for several different EFC applications (e.g. software routines for the local EFC application and the CARDME application), only one piece of software executes at a time. The initialisation phase can be seen as a switch where the RSE decides which path to follow. From the initialisation onwards, the RSE will (for a certain OBE) address a single EFC contract only.

If however, the RSE cannot accept one of the EFC contracts presented by the OBE, the transaction will be terminated. As no information regarding the identity of the user has been exchanged at this point, the local exception handling procedures will be initiated.

An example of the information exchange in the initialisation phase - between a beacon at a French tolling station and an OBE inside a Norwegian vehicle – is given in Table B.2 below.

Table B.2 — Initialisation phase — Information exchange

Road Side Equipment	On-Board Equipment
BST: *"Hello, here is an EFC Station"*	
BST: *"Hello, here is an EFC Station"*	(A vehicle is approaching. OBE wakes up and replies)
	VST: *"Hello, I can offer the following EFC contracts and transactions:"* 1. *Transaction type "AUTOPASS"* *Central account with the Operator "NorwegTrans"* 2. *Transaction type "CARDME"* *Central account with the Operator "Öresundskonsortiet"*
The roadside thinking: According to my tables, I have the following transactions available and recognise the accounts with the following operators: *Transaction* — *Operator* TIS transaction — AREA, COFIROUTE, ESCOTA, SANEF CARDME transaction — AustroToll, BelgiaPay, NorwegTrans, PagaMadrid When I compare my table with the VST, I see that I recognise the second option offered by the OBE. Hence, I will from now on use "CARDME/NorwegTrans"	

B.2.1.3 Presentation - Read OBE Data

In order to know which tariff to apply and which account to charge, the RSE requires some additional information from the passing OBE. The RSE obtains this information via “read commands” sent over the DSRC link.

NOTE The RSE addresses only data from the contract that it has chosen to use in the preceding initialisation phase (“CARDME/NorwegTrans” in our example).

Table B.3 — Presentation phase — Information exchange

Road Side Equipment	On-Board Equipment
“Please give me the following information about your CARDME contract with NorwegTrans: - your payment means, including the personal account number (with signature) - your previous receipts - your vehicle classification details“	
	“With pleasure, here are the data you have asked for. *I have added my signature to show that my data are correct and that you can trust to receive money:* - payment means, with signature - my previous receipts (entry ticket) - my vehicle classification details”

The RSE uses the retrieved data for the following purposes:

— Payment means including the personal account number: The account held at the issuer of the contract is identified through the personal account number. The personal account number points to exactly one customer account held with a Contract Issuer in Europe. This information enables the EFC Operator to draw money from the account of a local user or to claim money from the Contract Issuer of a foreign user.

— Previous receipts: Two receipts, associated with the two most recent passages through EFC CARDME stations, are read from the OBE memory. (When an OBE passes an EFC station, a new Receipt is written into the OBE memory. See also the explanation of the Receipt Phase in B.2.1.4 below).

— In a classical manual closed tolling system a user takes a ticket from an automatic ticket dispensing machine when he enters the motorway. At the exit the user shows this ticket to the tolling personnel, who calculates the fee from the distance matrix entry-exit. The same thing happens electronically. Some systems also require the last but one receipt to determine the fee. This is especially the case when there are alternative routes through the (motorway) network.

— In an open toll system, where one pays per passage of a bridge, a mountain pass or a stretch of motorway, reading the last receipt is of little use to the RSE. In CARDME it is done anyway, in order to have the same transaction everywhere, regardless of station type.

— Vehicle classification details: In some systems, the applicable tariff is determined from the vehicle class measured at the tolling station. In other systems, vehicle class is determined from the data in OBE (the so called 'declared classification”). These OBE-declared vehicle-related data are read out here. The declared vehicle characteristics are sufficient for any RSE to determine the applicable tariff. Systems that measure class can simply ignore these data.

— Signature: The OBE adds several security-related data to the tolling data, here simply called 'Signature'. CARDME foresees several different such security data, and even an optional second read-command for roaming users, in order cover all security needs. These security measures are discussed in a separate

chapter. In CARDME it is mandatory for OBEs to produce these security-related data. It is important to note, however, that using the security data is optional in the sense that the roadside may simply ignore them. From a technical point of view, every operator is free to decide which of the security data he wants to check, when and where he wants to check them, and whether he wants to check them at all.

B.2.1.4 Receipt - Write New OBE Data

In the previous phases the RSE has read all data that are required to charge the user (either directly for local users or indirectly for roaming users, which are charged via their contract issuers).

The receipt phase is used to write all data to the OBE that are to be carried to the next tolling station ("you can only read what you have written before"). It is also time to inform the user about the success of the tolling transaction.

Table B.4 — Receipt phase – Information exchange

Road Side Equipment	On-Board Equipment
"Please store the following information in your memory: - Transaction receipt (entry ticket) *Inform the user about the success of the transaction"*	
	"I confirm. I have stored the ticket and I have given the user a signal".

The most important data that have to be written into the OBE is the **entry ticket**. In closed tolling systems it is essential that this information is carried from one tolling station to the next. Also for other system types it makes sense to give an **electronic receipt**. This receipt is not primarily intended as direct information for the user since only very few OBEs have the capability to display the rather complex receipt information. The receipt rather serves as a record of past transactions in case a dispute arises.

It is recommended to store the two latest receipts in the OBE. These are transmitted over the DSRC link as "ReceiptData1" and "ReceiptData2". The RSE keeps track of what is old and what is new, and not the OBE in order to have a simple OBE design. The RSE always reads and writes both receipts. When writing, the RSE writes the new receipt to ReceiptData1 and it copies the data just read under ReceiptData1 (in the presentation phase) to ReceiptData2.

The information in the receipt or in the entry ticket comprises:

— passage data and time;

— passage location (EFC operator, station number, lane number, station type);

— passage result (OK/not OK, wrong class, blacklisted, security error, etc.);

— applied vehicle/tariff class;

— used contract.

In addition, the **user is informed** about the transaction result. The OBE signals to the user one of three messages: "OK", "not OK" and "Contact Operator". Also in the Write-Phase there is security-related information added to the data.

B.2.1.5 Tracking and Closing - End the Transaction

At this stage in the transaction all tolling–related data exchange is completed. A failure of the transaction is no longer possible.

There are only some technical housekeeping tasks required, namely to track the vehicle through the communication zone (mainly required in free-flow installations with video-enforcement) and/or to formally close the transaction, i.e. telling the OBE that there is no more to come.

B.3 CARDME transaction phases

B.3.1 Overview

Table B.5 below provides an overview of the CARDME transaction in terms of the sequence of DSRC application layer functions (such as INITIALISATION, GET, SET and EVENT-REPORT), EFC functions and data exchanged between an RSE and an OBE. The subsequent subclauses give details of exchanged EFC application data.

Table B.5 — CARDME transaction overview

Phase	*Roadside Equipment*		*On-board Equipment*	*Remarks*
Initialisation (BST – VST)	INITIALISATION.request (BST)	→		RSE periodically sends BST.
		←	INITIALISATION.response (VST) • EFC-ContextMark • AC_CR-KeyReference • RndOBE	A newly arrived OBE answers with VST. AC-CR-KeyReference is the ref. to the Access Credential Keys to be used by the RSE. RndOBE is a random number that the RSE uses when calculating the Access Credentials.
Presentation (GET)	GET_STAMPED.request AC_CR [Access Credential Key] • PaymentMeans (incl. PersonalAccountNumber) (RndRSE, KeyRef_Op) GET.request AC_CR [Access Credential Key] • ReceiptData1 • ReceiptData2 • EquipmentStatus • Classification data: – VehicleClass – VehicleDimensions – VehicleAxles – VehicleLicencePlateNumber – VehicleWeightLimits – VehicleSpecificCharacteristics	→		OBE to calculate an Authenticator that proves its authenticity. OBE will give access only when RSE provides the correct Access Credentials AC_CR. Personal Account Number, pointing to the user contract/account at the contract issuer. Random number and key reference for the authenticator that the OBE calculates. Read last and penultimate receipts (entry ticket or latest transaction). Read the equipment status (which includes a transaction counter). Read declared classification data. Vehicle Class also gives information on trailer presence. Vehicle Axles includes information on presence of dual tyres. Vehicle Specific Characteristics include information on emission class, engine type, etc.
		←	GET_STAMPED.response • Operator_Authenticator (Auth_Op) GET.response	OBE responds with the data asked for, plus an Authenticator calculated with the interoperable key, i.e. with a key known to all EFC Operators
Optional Presentation for Foreign OBEs	GET_STAMPED.request AC_CR [Access Credential Key] • PaymentMeans incl. PersonalAccountNumber (RndRSE, KeyRef_Iss)	→		For OBEs from a foreign Contract Issuer, the RSE asks for the calculation of an additional authenticator over the Personal Account Number with keys only known to the Contract Issuer, so that one can prove that the vehicle actually has passed.
		←	GET_STAMPED.response • Issuer_Authenticator (Auth_Iss)	
Receipt (SET)	SET.request • ReceiptData1 • ReceiptData2 • EquipmentStatus • ReceiptText SET_MMI.request	→		Write new receipts (or entry ticket). Write new status information and increment transaction counter. Transmit textual information, which may be displayed to the user. Give an „OK“ indication to the user (normally the OBE will beep).
		←	SET.response Set_MMI.response	
Tracking And *Closing*	ECHO.request	→		Track OBE by exchanging dummy information. The usage of Echo is by optional, at the discretion of the RSE, and may be repeated.
		←	ECHO.response	
	EVENT_REPORT.request (Release)	→		RSE closes transaction and releases OBE.

B.3.2 Initialisation Phase

Table B.6 provides details of the data exchanged between an RSE and an OBE in the initialisation phase.

Table B.6 — Initialisation phase

Phase	*RSE*		*OBE*
Initialisation	INITIALISATION.request (BST)	→	
		←	INITIALISATION.response (VST) Data for EFC Contract #1 (e.g. from local EFC system): • EFC-ContextMark: – ContractProvider – TypeOfContract – ContextVersion • (optional additional data) — — Data for EFC Contract #2 (e.g. CARDME European Service) • EFC-ContextMark: – ContractProvider – TypeOfContract – ContextVersion • AC_CR-KeyReference: – AC_CR-MasterKeyReference – AC_CR-Diversifier • RndOBE

NOTE1 The above data exchange is described on application level. There are also more technical negotiations going on between RSE and OBE, e.g. in order to arrive at a mutually agreed DSRC communication profile, determining amongst other parameters which subcarrier frequency the OBE shall use on uplink.

NOTE2 RSE may issue a command Event-Report Release, according to Layer 7, in case it does not recognizes the EFC Context mark received by the OBE to avoid disturbance created by unknown OBE.

Initialisation request (BST) data
For EFC, the application identification (AID) code equals 1. The BST is fully standardised and contains nothing CARDME-specific.

Initialisation response (VST) data
Every EFC transaction supported by the OBE is represented via the data element “EFC-Context Mark”. An EFC-Context Mark indicates which operator issued the contract in the OBE, the type of contract and a version number (context version, i.e. application/software/key versions).

CARDME's ContextMark contains the mandatory EFC-ContextMark information:

— ContractProvider: A code standing for the Contract Issuer (the code contains the country of residence of the Contract Issuer and a national assigned number identifying the individual issuer).

— TypeOfContract: A data element that gives the RSE basic information on the EFC contract residing in the OBE (e.g. central account or on-board purse; pre- or post-paid; unlimited or restricted to a certain concession area; discount tariff applies). Within CARDME currently only one type of contract is defined, namely the “CARDME European central account transaction”.

— ContextVersion: This data element is used by the OBE to tell the RSE the Version of some data it contains. For CARDME it says something like “I am built according to CARDME-4 Specification V1.0, and I have been initialised using interoperable Security Key Version 3”.

In addition to the mandatory contents above, the ContextMark may contain further information. CARDME makes use of this feature and has defined the following additional data as part of the OBE response in the CARDME ContextMark.

— AC_CR-KeyReference: A reference number that tells the RSE which keys to use when calculating the Access Credentials. CARDME supports both key generations and key diversification.

— RndOBE: This data element (“random number generated by the OBE”) contains a number that is freely chosen by the OBE and not predictable by the RSE. The number has to be used by the RSE when calculating the Access Credentials. This guarantees that the RSE has to calculate the Access Credentials afresh for each session. By this it is avoided that someone erects an un-authorised beacon using Access Credentials he obtained through listening to correct EFC transactions.

Note that the VST contains no data that might have privacy implications. The VST is accessible to any standards-conformant beacon. It can neither be forbidden nor technically prevented that any interested party reads VST information. Hence, in CARDME the VST contains no information that allows identification of the vehicle.

B.3.3 Presentation Phase

B.3.3.1 General

Table B.7 provides details of the data exchanged between an RSE and an OBE in the presentation phase.

Table B.7 — Presentation phase

Phase	*RSE*		*OBE*
Presentation	GET_STAMPED.request AC_CR [Access Credential Key] • PaymentMeans incl. PersonalAccountNumber (RndRSE, KeyRef_Op) GET.request AC_CR [Access Credential Key] • ReceiptData1 • ReceiptData2 • EquipmentStatus • Classification data: – VehicleClass – VehicleDimensions – VehicleAxles – VehicleLicencePlateNumber – VehicleWeightLimits – VehicleSpecificCharacteristics	→	
		←	GET_STAMPED.response — Operator_Authenticator (Auth_Op) GET.response

Functionally, four groups of data can be discerned in the presentation phase:

1) **Account information - *static***. Data that allow the EFC Operator to claim money from a user account held with a financial institution or with a Contract Issuer.

2) **Information about the last passage - *dynamic***. The RSE reads data that have been written at the last tolling station. At the exit of a closed tolling system these data constitute the entry ticket, which was written by the DSRC beacon on entry. On other stations the data are normally ignored.

3) **Vehicle classification information - *static***. In tolling systems which rely on declared classification to determine tariff, vehicle classification information has to be read from the OBE. Also in systems relying on measured class, classification information read from the OBE is sometimes used for verification.

4) **Security related information** – ***dynamic***. The CARDME transaction enables several security mechanisms, without making them mandatory to use by the RSE. The different mechanisms address different security requirements. If one decides not to use some of the mechanisms, the related security data can simply be ignored by the RSE, respectively dummy data can be transferred.

B.3.3.2 Account and contract information

Account and contract related information is contained in the following attributes:

— PaymentMeans incl. PersonalAccountNumber: Points to a user account held at the Contract Issuer (which is already known from the Initialisation Phase). The EFC Operator will send his claim to this account. Contains also expiry date and restrictions of the payment means.

B.3.3.3 Information about the last passage

Information about the last passage is mainly required when exiting a closed tolling system. In this case the information sent by the OBE is the “entry ticket” written when the vehicle entered the tolled system.

— ReceiptData1: The “Ticket” written by the last beacon passed. This attribute contains both the class that was declared at the last transaction and the class that was actually used (e.g. measured). When used as part of an entry ticket, this information enables the exit station to find out whether the class has changed during the trip. The attribute ReceiptData1 also contains the data element ReceiptAuthenticator, which contains data that have been calculated by the last beacon in order to “sign” the ticket given. The RSE may use this authenticator to check that the (entry-)ticket has not been manipulated, especially that the entry point has not been changed. In addition the authenticator can be used as a kind of “indirect authentication of the previous beacon”. The Receipt Authenticator is both produced and checked with a local algorithm and with local keys, i.e. with keys that are not distributed to any third party. The procedure is fully in the realm of an EFC operator's local system, and the authenticators are both calculated and checked by his own beacons only. The OBE merely transports this authenticator from one beacon to the next. Operators are free to use this security service, simply ignore it (i.e. by writing empty authenticators in the Receipt Phase and not checking the authenticators read in the Presentation Phase) or even use it for other purposes. In this use, the Receipt Authenticator serves as a free field where an EFC Operator may transport some local data from one beacon to the next.

— ReceiptData2: The “Ticket” of the penultimate beacon (last but one). In some systems two receipts are required to find which of two alternative routes the vehicles has passed.

B.3.3.4 Vehicle classification information

CARDME foresees a list of declared classification data that tries to cover a maximum of needs while remaining as short as possible. Keeping the list of classification data short is not required for technical reasons – a few bytes more or less over the DSRC link make little difference – but for cost reasons upon personalisation of the OBE. In order to be as flexible as possible, CARDME has devised an adaptable concept to treat classification.

— VehicleClass: Vehicle Class is a very simple and well known data element which in CARDME is used in the following way: Vehicle Class covers three purposes: (1) it gives the local class in the “home system” of the Contract Issuer; (2) for clear-cut cases it gives simple “European harmonised classes”, avoiding to have the more complex extended declared characteristics present; (3) it states whether a trailer is present or not.

— VehicleDimensions, VehicleAxles, VehicleLicencePlateNumber, VehicleWeightLimits, VehicleSpecificCharacteristics: These extended declared vehicle characteristics are only present if required by the contract or when the vehicle does not fall into a “European harmonised class”. The RSE may read only those classification data it requires.

B.3.3.5 Security related information

CARDME enables several security mechanisms, which are designed to protect the individual security requirements of the different entities in the CARDME architecture. The security level is adaptable from the RSE point of view – all security measures can either be used or be disregarded by the RSE.

— AccessCredentials: In CARDME all access to OBE data (both read and write) is protected with Access Credentials (AC_CR). The RSE has to send the right Access Credentials before the OBE accepts a command. The RSE calculates the Access Credentials dynamically using a challenge produced by the OBE (see B.3.1 Initialisation phase). The required keys need to be known by all partners of the MoU. Because of the wide distribution of the keys, it is indispensable that these keys are diversified (different OBE have different keys) and that there are key generations (new keys are used from time to time).

NOTE 1 How to do without AccessCredentials? CARDME foresees one generation of Access Credentials ("Generation 0") which is openly known to all. OBEs or rather contracts with these Access Credentials can be read by everybody. It is left to the Contract Issuer policy (and presumably to agreements in the MoU) whether he issues such OBEs or not.

— Operator_Authenticator: The CARDME Transaction uses a GET_STAMPED command to retrieve the Contract Identifier. The purpose of this command is to ask the OBE to "stamp" the data it sends back with an Authenticator. This special Authenticator is called the Operator_Authenticator since it can be interpreted by any Operator. The Authenticator can be checked to make sure that the OBE passing is a "genuine one", i.e. part of the interoperability scheme (and not a forged one). It is dynamically calculated by the OBE with the interoperable keys, i.e. with keys known to all EFC operators (KeyRef_Op).

NOTE 2 How to do without OBE_Authenticator? There is no need from technically reasons to check this Authenticator. It is automatically produced by the OBE, but EFC Operators that do not wish to perform such a security check are free not to check the authenticator in their RSE (at least from a technical viewpoint – they might be obliged to do so contractually, say through contracts with associated issuers or through the interoperability MoU). One possible reason why an operator does not to want check this authenticator is when he adds the CARDME transaction onto his existing RSE which has no appropriate key storage and security handling facilities. At some point in time, e.g. when he routinely replaces some of his older equipment, he can decide to go for higher security.

— ReceiptAuthenticator (contained in the ReceiptData-attributes): This RSE signature under the receipt has already been treated above under "Information about the last passage".

— EquipmentStatus: In CARDME this data element serves as a very simply but effective security measure, namely as a simple transaction counter. According to this EFC application standard (ISO 14906) the coding of the data element 'EquipmentStatus" is left to the operator (it says 'operator-specific EFC-application information pertaining to the status of the equipment"). The data element provides for 16 bits. CARDME recommends that operators agree to reserve 4 bits for their private local use (e.g. for management of the transaction in their own system, containing information like 'next suitably equipped gantry should make an enforcement picture"), and to leave 12 bits for a transaction counter (0...4095). CARDME proposes that each communication between RSE and OBE should be counted and a record of this maintained in the OBE, thereby increasing the practicalities of proving some instance of fraud (e.g. on the part of an operator by duplicating transactions at RSE, especially when he makes use of the low-security mode of operation optionally allowed for in CARDME. The transaction counter also helps identifying instances when cryptographic security is broken – a very important system performance monitoring facility.).

  The RSE of every EFC operator signed up to the MoU reads the Equipment Status in the Presentation Phase, increments the counter, and writes the new value back to the OBE in the Receipt Phase.

B.3.4 Optional Presentation Phase

Table B.8 provides details of the data exchanged between an RSE and an OBE in the optional presentation phase.

Table B.8 — Optional presentation phase

Phase	*RSE*		*OBE*
Optional Presentation	GET_STAMPED.request AC_CR [Access Credential Key] — PaymentMeans incl. PersonalAccountNumber (RndRSE, KeyRef_Iss)	→	
		←	GET_STAMPED.response — OBE Authenticator (Auth_Iss)

Nowadays it is very often the case that the EFC Operator and the Contract Issuer is one and the same organization. Very often the operator issues the OBE with the contract inside. In this case, the security information passed in the presentation phase (see B.3.3) is sufficient.

In case these two organizations are different, and especially when they are organizationally strongly separated, e.g. reside in different countries (which is normal in a roaming environment) or are totally different entities altogether (e.g. a tolling operator and a bank) a new security requirement arises. In the presentation phase, the EFC Operator was able to check through the Operator_Authenticator that the OBE (or rather the Contract in the OBE) is a genuine one, i.e. from an organization belonging to the MoU. He will then send a claim to the Contract Issuer asking for money. The Contract Issuer now has no means to really check this claim. He has no indisputable proof for his customer, the user that drove the vehicle, that the money he asks for actually corresponds to a passage somewhere.

Every RSE knows from the data element “ContractProvider” in the ContextMark of the VST, whether it is communicating with a local or a foreign vehicle. Only for foreign vehicles the RSE executes this optional presentation phase. The sole purpose of this phase is to obtain an Authenticator from the OBE. This Authenticator should be calculated by the OBE with keys only known to the Contract Issuer. The (foreign) EFC Operator, where the vehicle passes, can neither check nor forge this authenticator. The EFC Operator simply adds this authenticator to the transaction record he sends as a claim to the Contract Issuer in order to be reimbursed. The Contract Issuer checks the Issuer_Authenticator in order to have proof that a vehicle for that he is obliged to pay actually has passed a certain (foreign) EFC station. This both serves as a proof against users trying to deny the passage and checks for a correct claim made by the foreign operator.

The challenge sent by the RSE (RndRSE) should not be a number freely chosen by the RSE. It should to be prescribed and constructed in such a way that the RSE cannot influence its value. A concatenation of Date and Time of passage would serve this purpose perfectly. This challenge has to be passed to the Contract Issuer in the transaction record, together with the authenticator calculated by the OBE, in order to enable the Contract Issuer to check the authenticator.

B.3.5 Receipt Phase

Table B.9 provides details of the data exchanged between an RSE and an OBE in the receipt phase.

Table B.9 — Receipt phase

Phase	*RSE*		*OBE*
Receipt	SET.request — ReceiptData1 — ReceiptData2 — EquipmentStatus — ReceiptText — SET_MMI.request	→	
		←	SET.response Set_MMI.response

The data already treated in the presentation phase (see B.3.3), which are most of the data associated with the Receipt Phase, are not treated again in this subclause. The data written to the OBE in the Receipt Phase and treated in the presentation phase are:

— ReceiptData1: The “Ticket” given by the RSE.

— ReceiptData2*:* The data read in ReceiptData1 the presentation phase are now written as ReceiptData2 (“old ticket”).

— EquipmentStatus: Includes a transaction counter.

In addition, **the user is informed** about the success of the transaction. This is done twofold:

1) **ReceiptText**: The RSE may use this data element to send a short text message to the passing OBE (maximum length 12 characters). It is not prescribed what the RSE shall say. The text might contain some cost information (“EURO 2.00”), some station information (“ENTRY 24”), added value information (“A55 CLOSED”), or may even be left blank.

NOTE Nowadays only few OBEs have a display, and very few OBEs will have the possibility (during the lifetime of this version of the standard) to display this text information, so normally OBEs will simply disregard the information. CARDME believes nevertheless that in many emerging systems it will be increasingly important to have the option to send at least a short text message, consisting of a few text characters. Especially in systems with complex, time-dependent tariffs for demand management purposes the user has to be informed about the actual cost of the current passage. Otherwise variable tariffication would become pretty meaningless.

2) **SET_MMI**: With this command the RSE instructs the OBE to use its man-machine interface (MMI), to signal the user one of three pre-defined messages, namely “OK”, “not OK” and “Contact Operator”. It is up to the OBE how to signal these messages. Depending on OBE make, the OBE may beep, light a signal lamp or even write something onto a display.

B.3.6 Tracking and Closing Phases

Table B.10 provides details of the sequence of data exchanged between an RSE and an OBE in the tracking/closing phase.

Table B.10 — Tracking and Closing phase

Phase	*RSE*		*OBE*
Tracking	ECHO.request	→	
	Etc...	←	ECHO.response
Closing	EVENT_REPORT.request (Release)	→	

In full multi-lane systems, a tracking phase is usually used to keep track of the vehicle after the EFC transaction is finished (using the Echo service, which may be used several times). In systems requiring no tracking, the session is closed with an explicit release in the Closing Phase.

Tracking and Closing are optional and used by RSE where required locally. All OBEs need to support these functionalities.

B.4 Bit-level specification

B.4.1 General

Tables B.11 to B.23 below provides for the bit-level specification of the CARDME transaction. The specification accounts for the complete frame content (excluding the zero-bit insertions) of the data exchanged, including protocol information related to DSRC-L1, -L2 and –L7 in order to ensure a maximum unambiguity of the CARDME transaction specification. The data that are associated with this International Standard are highlighted in grey.

B.4.2 Initialisation

B.4.2.1 Initialisation request (BST)

Table B.11 — Initialisation request (BST) frame content

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Broadcast LID	1111 1111	Link address for broadcast
3	MAC control field	1010 0000	The frame contains a command LPDU
4	LLC control field	0000 0011	UI command
5	Fragmentation header	1xxx x001	No fragmentation. PDU no shall never be set to 0000_2 or 0001_2.
6	BST SEQUENCE { OPTION indicator	1000 0	INITIALISATION.request NonmandApplications not present.
	BeaconId SEQUENCE { ManufacturerId INTEGER (0..65535)	000	Manufacturer identifier- Example 1 (=Kapsch). See ISO 14816.
7		0000 0000	Register at www.nen.nl/cen278 for value assignment.
8		0000 1	
	IndividualId INTEGER (0..134217727)	000	27 bit ID available for manufacturer. Example: Id=1052_{10}
9		0000 0000	
10		0000 0100	
11	}	0001 1100	
12	Time INTEGER(0..4294967295)	0100 0001	32 bit UNIX real time. Example: 1103790512_{10}
13		1100 1010	
14		1000 0001	
15		1011 0000	
16	Profile INTEGER (0..127,...)	0000 0000	No extension, Profile. Example : Profile = 0
17	MandApplications SEQUENCE (SIZE(0..127,...)) OF {	0000 0001	No extension, Number of mandApplications = 1_{10}
18	SEQUENCE { OPTION indicator	 0	 EID not present
	OPTION indicator	0	Parameter not present
	AID DSRCApplicationEntityID } }	00 0001	No extension. AID = 1_{10}, EFC
19	ProfileList SEQUENCE (0..127,...) OF Profile }	0000 0000	No extension, number of profiles in list = 0.
20	FCS	xxxx xxxx	Frame check sequence
21		xxxx xxxx	
22	FLAG	0111 1110	End Flag

B.4.2.2 Private window request

Table B.12 — Private window request frame content

Octet #	Attribute / Field	Bits in Octet B_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Private LID	xxxx xxx0	Link address of a specific OBE
3		xxxx xxx0	
4		xxxx xxx0	
5		xxxx xxx1	
6	MAC control field	0110 0000	Private window request
7	FCS	xxxx xxxx	Frame check sequence
8		xxxx xxxx	
9	FLAG	0111 1110	End Flag

B.4.2.3 Private window allocation

Table B.13 — Private window allocation frame content

Octet #	Attribute/Field	Bits in Octet B_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Private LID	xxxx xxx0	Link address of a specific OBE
3		xxxx xxx0	
4		xxxx xxx0	
5		xxxx xxx1	
6	MAC control field	0010 s000	Private window allocation
7	FCS	xxxx xxxx	Frame check sequence
8		xxxx xxxx	
9	FLAG	0111 1110	End Flag

B.4.2.4 Initialisation response (VST)

Table B.14 — Initialisation response (VST) frame content

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Private LID	xxxx xxx**0**	Link address of a specific OBE
3		xxxx xxx**0**	
4		xxxx xxx**0**	
5		xxxx xxx**1**	
6	MAC control field	1100 0000	The frame contains a command LPDU
7	LLC control field	0000 0011	UI command
8	Fragmentation header	1xxx x001	No fragmentation. PDU no shall never be set to 0000_2 or 0001_2.
9	VST SEQUENCE {	1001	INITIALISATION.response
	Fill BIT STRING (SIZE(4))	0000	Set to 0
10	Profile INTEGER (0..127,...)	**0**000 0000	No extension, Profile. Example : 0_{10}
11	Applications SEQUENCE (SIZE((0..127,...)) OF {	**0**000 0010	No extension, 2 applications
12	SEQUENCE { OPTION indicator	1	EID present
	OPTION indicator	1	Parameter present
	AID DSRCApplicationEntityID	**0**0 0001	No extension, AID = 1, EFC
13	EID	0000 0010	Associated with a context mark. Example : 2_{10}
14	Parameter CONTAINER {	**0**000 0010	No extension, Container Choice = 2_{10}, Octet string
15		**0**000 0110	No extension, octet string length = 6_{10}
16	EFC-ContextMark SEQUENCE {		
	ContractProvider SEQUENCE {		
	CountryCode BIT STRING (SIZE(10))	0011 0000	10 bit country code according to ISO 3166 with ITA2
17		11	Binary encoding based on ISO 14816. Example : NO
	IssuerIdentifier INTEGER (0..16383) }	00 0000	14 bits issuer identifier. Example : 2_{10}
18		0000 0010	
19	TypeOfContract OCTET STRING (SIZE(2))	0000 0000	Type of contract. Example : 1_{10}
20		0000 0001	
21	ContextVersion INTEGER (0..127,..) } } }	**0**000 0010	No extension, context version. Example : 2_{10}
22	SEQUENCE { OPTION indicator	1	EID present
	OPTION indicator	1	Parameter present
	AID DSRCApplicationEntityID	**0**0 0001	No extension, AID = 1, EFC
23	EID	0000 0101	Associated with a context mark. Example : 5_{10}
24	Parameter CONTAINER {	**0**000 0010	No extension, Container Choice = 2_{10}, Octet string
25		**0**001 0000	No extension, octet string length = 16_{10}
26	EFC-ContextMark SEQUENCE {		
	ContractProvider SEQUENCE {		
	CountryCode BIT STRING (SIZE(10))	1010 0100	10 bit country code according to ISO 3166 with ITA2 binary
27		00	Encoding based on ISO 14816. Example : SE
	IssuerIdentifier INTEGER (0..16383) }	00 0000	14 bits issuer identifier. Example : 1_{10} (Öresundskonsortiet)
28		0000 0001	
29	TypeOfContract OCTET STRING (SIZE(2))	0000 0000	Type of contract. Example : 2_{10}
30		0000 0010	
31	ContextVersion INTEGER (0..127,..) }	**0**000 0001	No extension, context version. Example : 1_{10}
32	CONTAINER {	**0**000 0010	No extension, Container Choice = 2_{10}, Octet string
33		**0**000 0010	No extension, octet string length = 2_{10}
34	AC_CR-Reference SEQUENCE { AC-MasterKeyRef Int1,	0000 0001	AC_CR-Reference to, consisting of AC_CR-MasterKeyRef and AC_CR-Diversifier, used for the computation of AC_CRKey and
35	AC_CR-Diversifier Int1 } }	0000 0001	AC_CR.
36	CONTAINER {	**0**000 0010	No extension, Container Choice = 2_{10}, Octet string
37		**0**000 0100	No extension, octet string length = 4_{10}
38	RndOBE Int4	0000 0000	Random Number (nonce) used together with AC_CRKey to
39		0000 0000	calculate AC_CR. Example : 640_{10}
40		0000 0010	
41	} } } }	1000 0000	
42	ObeConfiguration SEQUENCE {		
	OPTION indicator	1	ObeStatus present
	EquipmentClass INTEGER (0..32767)	000 0000	Example : 3_{10}
43		0000 0011	
44	ManufacturerId INTEGER (0..65535)	0000 0000	Manufacturer identifier. See ISO 14816 Register at
45		0000 0010	www.nen.nl/cen278 for value assignment. Example : 2_{10}.
46	ObeStatus INTEGER(0..65535)	0000 0011	Example : 768_{10}
47	} }	0000 0000	
48	FCS	xxxx xxxx	Frame check sequence
49		xxxx xxxx	
50	FLAG	0111 1110	End Flag

B.4.3 Presentation

B.4.3.1 Presentation request

Table B.15 — Presentation request frame content

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Private LID	xxxx xxx**0**	Link address of a specific OBE
3		xxxx xxx**0**	
4		xxxx xxx**0**	
5		xxxx xxx**1**	
6	MAC control field	1010 s000	The frame contains a command LPDU
7	LLC control field	n111 0111	Polled ACn command, n bit
8	Fragmentation header	1xxx x001	No fragmentation. First service of chain.
9	ACTION.request SEQUENCE {	0000	ACTION.request (GET_STAMPED.request)
	OPTION indicator	1	AccessCredentials present
	OPTION indicator	1	ActionParameter present
	OPTION indicator	0	IID not present
	Mode BOOLEAN	1	Mode = TRUE, Response expected
10	EID INTEGER(0..127,...)	**0**000 0101	No extension, Element EID, uniquely related to a Context mark within the OBE. Example : 5_{16}
11	ActionType INTEGER(0..127,...)	**0**000 0000	No extension, Action type = 0, GET_STAMPED.request
12	AccessCredentials OCTET STRING {	**0**000 0100	No extension, octet string length = 4_{10}
13	AC_CR	aaaa aaaa	Access credentials calculated by RSE using RndOBE and the Access Credentials Key AC_CRKey.
14		aaaa aaaa	
15		aaaa aaaa	
16	}	aaaa aaaa	
17	ActionParameter CONTAINER {	**0**001 0001	No extension, Container Choice = 17_{10}, GetStampedRq
18	AttributeIdList SEQUENCE (SIZE(0..127,...)) OF { INTEGER (0..127,...) AttributeId	**0**000 0001	No extension, number of attribute IDs = 1
19	PaymentMeans }	**0**010 0000	No extension, AttributeId = 32_{10}, PaymentMeans
20	Nonce OCTET STRING {	**0**000 0100	No extension, octet string length = 4_{10}
21	RndRSE	rrrr rrrr	Random number from RSE, containing SessionTime, needed to calculate OperatorAuthenticator
22		rrrr rrrr	
23		rrrr rrrr	
24	}	rrrr rrrr	
25	KeyRef_Op(h) } }	xxxx xxxx	h = Reference to AuKey_Op used for the computation of Operator Authenticator.
26	Fragmentation header	1xxx x001	No fragmentation. Same PDU no as before (concatenation).
27	GET.request SEQUENCE {	0110	GET.request
	OPTION indicator	1	AccessCredentials present
	OPTION indicator	0	IID not present
	OPTION indicator	1	AttributeIdList present
	Fill BIT STRING(SIZE(1))	0	Set to 0
28	EID INTEGER(0..127,...)	**0**000 0101	No extension, EID, Example : 5_{10}
29	AccessCredentials OCTET STRING {	**0**000 0100	No extension, octet string length = 4_{10}
30	AC_CR	aaaa aaaa	Access credentials calculated by RSE using RndOBE and the Access Credentials Key AC_CRKey.
31		aaaa aaaa	
32		aaaa aaaa	
33	}	aaaa aaaa	
34	AttributeIdList SEQUENCE (SIZE(0..127,...)) OF { INTEGER (0..127,...) AttributeId	**0**000 0110	No extension, number of attribute IDs = 6_{10}
35	VehicleLicencePlateNumber	**0**001 0000	No extension, AttributeId = 16_{10}, VehicleLicencePlateNr
36	VehicleClass	**0**001 0001	No extension, AttributeId = 17_{10}, VehicleClass
37	VehicleWeightLimits	**0**001 0100	No extension, AttributeId = 20_{10}, VehicleWeightLimits
38	EquipmentStatus	**0**001 1010	No extension, AttributeId = 26_{10}, EquipmentStatus
39	ReceiptData1	**0**010 0001	No extension, AttributeId = 33_{10}, ReceiptData1
40	ReceiptData2 } }	**0**010 0010	No extension, AttributeId = 34_{10}, ReceiptData2
41	FCS	xxxx xxxx	Frame check sequence
42		xxxx xxxx	
43	FLAG	0111 1110	End Flag

NOTE VehicleSpecificCharacteristics, VehicleDimensions and Vehicle Axles are not included in the examples in this table.

B.4.3.2 Presentation response

Table B.16 — Presentation response frame content

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Private LID	xxxx xxx**0**	Link address of a specific OBE
3		xxxx xxx**0**	
4		xxxx xxx**0**	
5		xxxx xxx**1**	
6	MAC control field	1101 0000	The frame contains a response LPDU
7	LLC control field	n111 0111	Response available, ACn command n bit
8	LLC status field	0000 0000	Response available and command accepted
9	Fragmentation header	1xxx x001	No fragmentation. First service of chain.
10	ACTION.response SEQUENCE {	0001	ACTION.response (GET STAMPED.response)
	OPTION indicator	0	IID not present
	OPTION indicator	1	ResponseParameter present
	OPTION indicator	0	ReturnStatus not present
	Fill BIT STRING(SIZE(1))	0	Set to 0
11	EID INTEGER (0..127,...)	**0**000 0101	No extension, EID, Example : 5_{10}
12	ResponseParameter CONTAINER {	**0**001 0010	No extension, Container Choice = 18_{10}, GetStampedRs
13	AttributeList SEQUENCE (SIZE(0..127,...)) OF {	**0**000 0001	No extension, number of attributes: 1
14	Attributes SEQUENCE { AttributeId INTEGER (0..127,...)	**0**010 0000	No extension, AttributeId = 32_{10}, PaymentMeans
15	AttributeValue CONTAINER {	**0**100 0000	No extension, Container Choice = 64_{10}
16	PaymentMeans SEQUENCE { PersonalAccountNumber	xxxx xxxx	PersonalAccountNumber
17		xxxx xxxx	
18		xxxx xxxx	
19		xxxx xxxx	
20		xxxx xxxx	
21		xxxx xxxx	
22		xxxx xxxx	
23		xxxx xxxx	
24		xxxx xxxx	
25		xxxx xxxx	
26	PaymentMeansExpiryDate	0001 1110	DateCompact. Example : 2005-03-01
27		0110 0001	
28	PaymentMeansUsageControl	0000 0000	Example : 1
29	} } } }	0000 0001	
30	Authenticator OCTET STRING {	**0**000 0100	No extension, octet string size = 4_{10}
31	OperatorAuthenticator	xxxx xxxx	Operator Authenticator over AttributeLIst (containing PaymentMeans) and RndRSE (containing SessionTime) calculated using AuKey_Op(h).
32		xxxx xxxx	
33		xxxx xxxx	
34	} } }	xxxx xxxx	
35	Fragmentation header	1xxx x001	No fragmentation. Same PDU no as before (concatenation).
36	GET.response SEQUENCE {	0111	GET.response
	OPTION indicator	0	IID not present
	OPTION indicator	1	AttributeList present
	OPTION indicator	0	ReturnStatus not present
	Fill BIT STRING(SIZE(1))	0	Set to 0
37	EID INTEGER(0..127,...)	**0**000 0101	No extension, EID, Example : 5_{10}
38	AttributeList SEQUENCE (SIZE(0..127,...)) OF {	**0**000 0110	No extension, 6 attributes in list.
39	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	**0**001 0000	No extension, AttributeId = 16_{10}, VehicleLicencePlateNo
40	Attribute Value CONTAINER {	**0**010 1111	No extension, Container choice = 47_{10}
41	VehicleLicencePlateNumber SEQUENCE { CountryCode,	1010 0100	Example : countrycode: SE
42		00	
	AlphabetIndicator,	00 0000	Example : alphabet indicator no 1
43	LicencePlateNumber	0000 0110	Length, Example : 6_{10}
44		0100 1111	'OCD560'
45		0100 0011	
46		0100 0100	
47		0011 0101	
48		0011 0110	
49	} } }	0011 0000	
50	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	**0**001 0001	No extension, AttributeId = 17_{10}, VehicleClass
51	Attribute Value CONTAINER {	**0**011 0001	No extension, Container choice = 49_{10}
52	VehicleClass Int1 } }	xxxx xxxx	VehicleClass value

Table B.16 (*continued*)

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
53	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	0001 0100	No extension, AttributeId = 20_{10} = VehicleWeightLimits
54	Attribute Value CONTAINER {	0011 0100	No extension, Container choice = 52_{10}
55	VehicleWeightLimits SEQUENCE { VehicleMaxLadenWeight Int2	xxxx xxxx	VehicleMaxLadenWeight
56		xxxx xxxx	
57	VehicleTrainMaxWeight Int2	xxxx xxxx	VehicleTrainMaxWeight
58		xxxx xxxx	
59	VehicleWeightUnladen Int2	xxxx xxxx	VehicleWeightUnladen
60	} } }	xxxx xxxx	
61	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	0001 1010	No extension, AttributeId = 26_{10}, EquipmentStatus
62	Attribute Value CONTAINER {	0011 1010	No extension, Container choice = 58_{10}
63	EquipmentStatus BIT STRING(SIZE(16))	0000 0000	EquipmentStatus value
64	} }	0000 0001	
65	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	0010 0001	No extension, AttributeId = 33_{10}, ReceiptData1
66	Attribute Value CONTAINER {	0100 0001	No extension, Container choice = 65_{10}
67	ReceiptData1 SEQUENCE { SessionTime	xxxx xxxx	SessionTime
68		xxxx xxxx	
69		xxxx xxxx	
70		xxxx xxxx	
71	SessionServiceProvider	xxxx xxxx	SessionServiceProvider
72		xxxx xxxx	
73		xxxx xxxx	
74	LocationOfStation	xxxx xxxx	LocationOfStation
75		xxxx xxxx	
76	SessionLocation	xxxx xxxx	SessionLocation
77	SessionType	xxxx xxxx	SessionType
78	SessionType	xxxx xxxx	SessionResult
79	SessionTariffClass	xxxx xxxx	SessionTariffClass
80	SessionClaimedClass	xxxx xxxx	SessionClaimedClass
81	SessionFee	xxxx xxxx	SessionFee
82		xxxx xxxx	
83		xxxx xxxx	
84		xxxx xxxx	
85	SessionContractProvider	xxxx xxxx	SessionContractProvider
86		xxxx xxxx	
87		xxxx xxxx	
88	SessionTypeOfContract	xxxx xxxx	SessionTypeOfContract
89		xxxx xxxx	
90	SessionContextVersion	xxxx xxxx	SessionContextVersion
91	ReceiptDataAuthenticator	xxxx xxxx	ReceiptDataAuthenticator
92		xxxx xxxx	
93		xxxx xxxx	
94	} } }	xxxx xxxx	
95	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	0010 0010	No extension, AttributeId = 34_{10}, ReceiptData2
96	Attribute Value CONTAINER {	0100 0001	No extension, Container choice = 65_{10}
97	ReceiptData2	xxxx xxxx	ReceiptData2. Same format as ReceiptData1 (see octets # 67-94)
...		...	
124	} } } }	xxxx xxxx	
125	FCS	xxxx xxxx	Frame check sequence
126		xxxx xxxx	
127	FLAG	0111 1110	End Flag

NOTE VehicleSpecificCharacteristics, VehicleDimensions and Vehicle Axles are not included in the examples in this table.

B.4.4 Optional presentation

B.4.4.1 Optional presentation request

Table B.17 — Optional presentation request frame content

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Private LID	xxxx xxx0	Link address of a specific OBE
3		xxxx xxx0	
4		xxxx xxx0	
5		xxxx xxx1	
6	MAC control field	1010 s000	The frame contains a command LPDU
7	LLC control field	n111 0111	Polled ACn command n bit
8	Fragmentation header	1xxx x001	No fragmentation. First service of chain.
9	ACTION.request SEQUENCE {	0000	ACTION.request (GET_STAMPED.request)
	OPTION indicator	1	AccessCredentials present
	OPTION indicator	1	ActionParameter present
	OPTION indicator	0	IID not present
	Mode BOOLEAN	1	Mode = TRUE, Response expected
10	EID INTEGER(0..127,...)	0000 0101	No extension, Element EID, uniquely related to a Context mark within the OBE. Example : 5_{10}
11	ActionType INTEGER(0..127,...)	0000 0000	No extension, Action Type = 0, GET_STAMPED.request
12	AccessCredentials OCTET STRING {	0000 0100	No extension, octet string length = 4_{10}
13	AC_CR	aaaa aaaa	Access credentials calculated by RSE using RndOBE and the Access Credentials Key AC_CRKey.
14		aaaa aaaa	
15		aaaa aaaa	
16	}	aaaa aaaa	
17	ActionParameter CONTAINER {	0001 0001	No extension, Container Choice = 17_{10}, GetStampedRq
18	AttributeIdList SEQUENCE (SIZE(0..127,...)) OF { INTEGER (0..127,...) AttributeId	0000 0001	No extension, number of attribute IDs = 1
19	PaymentMeans }	0010 0000	No extension, AttributeId = 32_{10}, PaymentMeans
20	Nonce OCTET STRING {	0000 0100	No extension, octet string length = 4_{10}
21	RndRSE	rrrr rrrr	Random number from RSE, containing SessionTime, needed to calculate IssuerAuthenticator
22		rrrr rrrr	
23		rrrr rrrr	
24	}	rrrr rrrr	
25	KeyRef_Iss(i) } }	xxxx xxxx	i = Reference to AuKey_Iss used in the computation of Issuer Authenticator.
26	FCS	xxxx xxxx	Frame check sequence
27		xxxx xxxx	
28	FLAG	0111 1110	End Flag

B.4.4.2 Optional presentation response

Table B.18 — Optional presentation response frame content

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Private LID	xxxx xxx**0**	Link address of a specific OBE
3		xxxx xxx**0**	
4		xxxx xxx**0**	
5		xxxx xxx**1**	
6	MAC control field	1101 0000	The frame contains a response LPDU
7	LLC control field	n111 0111	ACn command n bit
8	LLC status field	0000 0000	Response available and command accepted
9	Fragmentation header	1xxx x001	No fragmentation. First service of chain.
10	ACTION.response SEQUENCE {	0001	ACTION.response (GET STAMPED.response)
	OPTION indicator	0	IID not present
	OPTION indicator	1	ResponseParameter present
	OPTION indicator	0	ReturnStatus not present
	Fill BIT STRING(SIZE(1))	0	Set to 0
11	EID INTEGER (0..127,...)	**0**000 0101	No extension, EID, Example : 5_{10}
12	ResponseParameter CONTAINER {	**0**001 0010	No extension, Container Choice = 18_{10}, GetStampedRs
13	AttributeList SEQUENCE (SIZE(0..127,...)) OF {	**0**000 0001	No extension, number of attributes: 1
14	Attributes SEQUENCE { AttributeId INTEGER (0..127,...)	**0**010 0000	No extension, AttributeId = 32_{10}, PaymentMeans
15	AttributeValue CONTAINER {	**0**100 0000	No extension, Container Choice = 64_{10}
16	PaymentMeans SEQUENCE { PersonalAccountNumber	xxxx xxxx	PersonalAccountNumber
17		xxxx xxxx	
18		xxxx xxxx	
19		xxxx xxxx	
20		xxxx xxxx	
21		xxxx xxxx	
22		xxxx xxxx	
23		xxxx xxxx	
24		xxxx xxxx	
25		xxxx xxxx	
26	PaymentMeansExpiryDate	0001 1110	DateCompact. Example : 2005-03-01
27		0110 0001	
28	PaymentMeansUsageControl	0000 0000	Example : 1
29	} } } }	0000 0001	
30	Authenticator OCTET STRING {	**0**000 0100	No extension, octet string size = 4_{10}
31	IssuerAuthenticator	xxxx xxxx	Issuer Authenticator over AttributeLIst (containing PaymentMeans) and RndRSE (containing SessionTime) calculated using AuKey_Iss(i).
32		xxxx xxxx	
33		xxxx xxxx	
34	} } }	xxxx xxxx	
35	FCS	xxxx xxxx	Frame check sequence
36		xxxx xxxx	
37	FLAG	0111 1110	End Flag

B.4.5 Receipt

B.4.5.1 Set receipt request

Table B.19 — Set receipt request frame content

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Private LID	xxxx xxx0	Link address of a specific OBE
3		xxxx xxx0	
4		xxxx xxx0	
5		xxxx xxx1	
6	MAC control field	1010 s000	The frame contains a command LPDU
7	LLC control field	n111 0111	Polled ACn command n bit
8	Fragmentation header	1xxx x001	No fragmentation. First service of chain
9	SET.request SEQUENCE {	0100	SET.request
	OPTION indicator	1	AccessCredentials present
	OPTION indicator	0	IID not present
	Fill BIT STRING(SIZE(1))	0	Set to 0
	Mode BOOLEAN	1	Mode = TRUE, Response expected
10	EID INTEGER(0..127,...)	0000 0101	No extension, EID. Example : 5_{10}
11	AccessCredentials OCTET STRING {	0000 0100	No extension, octet string length = 4_{10}
12	AC_CR	aaaa aaaa	Access credentials calculated by RSE using RndOBE and the Access Credentials Key AC_CRKey.
13		aaaa aaaa	
14		aaaa aaaa	
15	}	aaaa aaaa	
16	AttributeList SEQUENCE (SIZE(0..127,...) OF {	0000 0100	No extension, number of attributes in list = 4_{10}
17	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	0000 1100	No extension, AttributeId = 12_{10}, ReceiptText
18	Attribute Value CONTAINER {	0010 1100	No extension, Container choice = 44_{10}
19	Indicator	0000 1010	No extension, octet string length = 10_{10}
20	ReceiptText	xxxx xxxx	ReceiptText value
21		xxxx xxxx	
22		xxxx xxxx	
23		xxxx xxxx	
24		xxxx xxxx	
25		xxxx xxxx	
26		xxxx xxxx	
27		xxxx xxxx	
28		xxxx xxxx	
29	} }	xxxx xxxx	
30	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	0001 1010	No extension, AttributeId = 26_{10}, EquipmentStatus
31	Attribute Value CONTAINER {	0011 1010	No extension, Container choice = 58_{10}
32	EquipmentStatus BIT STRING(SIZE(16))	xxxx xxxx	EquipmentStatus value
33	} }	xxxx xxxx	
34	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	0010 0001	No extension, AttributeId = 33_{10}, ReceiptData1
35	Attribute Value CONTAINER {	0100 0001	No extension, Container choice = 65_{10}
36	ReceiptData1 SEQUENCE { SessionTime	xxxx xxxx	SessionTime
37		xxxx xxxx	
38		xxxx xxxx	
39		xxxx xxxx	
40	SessionServiceProvider	xxxx xxxx	SessionServiceProvider
41		xxxx xxxx	
42		xxxx xxxx	
43	LocationOfStation	xxxx xxxx	LocationOfStation
44		xxxx xxxx	
45	SessionLocation	xxxx xxxx	SessionLocation
46	SessionType	xxxx xxxx	SessionType
47	SessionType	xxxx xxxx	SessionResult
48	SessionTariffClass	xxxx xxxx	SessionTariffClass
49	SessionClaimedClass	xxxx xxxx	SessionClaimedClass

Table B.19 (*continued*)

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
50	SessionFee	xxxx xxxx	SessionFee
51		xxxx xxxx	
52		xxxx xxxx	
53		xxxx xxxx	
54	SessionContractProvider	xxxx xxxx	SessionContractProvider
55		xxxx xxxx	
56		xxxx xxxx	
57	SessionTypeOfContract	xxxx xxxx	SessionTypeOfContract
58		xxxx xxxx	
59	SessionContextVersion	xxxx xxxx	SessionContextVersion
60	ReceiptDataAuthenticator	xxxx xxxx	ReceiptDataAuthenticator
61		xxxx xxxx	
62		xxxx xxxx	
63	} } }	xxxx xxxx	
64	Attributes SEQUENCE { AttributeId INTEGER(0..127,...)	0010 0010	No extension, AttributeId = 34_{10}, ReceiptData2
65	Attribute Value CONTAINER {	0100 0001	No extension, Container choice = 65_{10}
66	ReceiptData2	xxxx xxxx	ReceiptData2. Same format as ReceiptData1 (see octets #36-63)
....		....	
93	} } } }	xxxx xxxx	
94	Fragmentation header	1xxx x001	No fragmentation. Same PDU no as before (concatenation).
95	ACTION.request SEQUENCE {	0000	ACTION.request (SET_MMI.request)
	OPTION indicator	0	AccessCredential not present
	OPTION indicator	1	ActionParameter present
	OPTION indicator	0	IID not present
	Mode BOOLEAN	1	Mode = TRUE, Response expected
96	EID INTEGER(0..127,...)	0000 0000	No extension, EID = 0 (System Element)
97	ActionType INTEGER(0..127,...)	0000 1010	No extension, Action Type = 10_{10}, SET_MMI.request
98	ActionParameter CONTAINER {	0100 0101	No extension, Container choice = 69_{10}, SETMMIRq
99	SetMMI INTEGER (0..255) } }	0000 0000	Example : ok (0_{10})
100	FCS	xxxx xxxx	Frame check sequence
101		xxxx xxxx	
102	FLAG	0111 1110	End Flag

B.4.5.2 Set receipt response

Table B.20 — Set receipt response frame content

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Private LID	xxxx xxx**0**	Link address of a specific OBE
3		xxxx xxx**0**	
4		xxxx xxx**0**	
5		xxxx xxx**1**	
6	MAC control field	1101 0000	The frame contains a response LPDU
7	LLC control field	n111 0111	ACn command n bit
8	LLC status field	0000 0000	Response available and command accepted
9	Fragmentation header	1xxx x001	No fragmentation. First service of chain.
10	SET.response SEQUENCE {	0101	SET.response
	OPTION indicator	0	IID not present
	OPTION indicator	0	ReturnStatus not present
	Fill BIT STRING (SIZE(2))	00	Set to 0
11	EID INTEGER (0..127,...) }	0000 0101	No extension, EID, Example : 5_{10}
12	Fragmentation header	1xxx x001	No fragmentation.
13	ACTION.response SEQUENCE {	0001	ACTION.response (SET_MMI.response)
	OPTION indicator	0	IID not present
	OPTION indicator	0	ResponseParameter not present
	OPTION indicator	0	ReturnStatus not present
	Fill BIT STRING (SIZE(1))	0	Set to 0
14	EID INTEGER (0..127,...) }	0000 0000	No extension, EID = 0 (System Element)
15	FCS	xxxx xxxx	Frame check sequence
16		xxxx xxxx	
17	FLAG	0111 1110	End Flag