INTERNATIONAL STANDARD

ISO 14906

Fourth edition
2022-12

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

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

Reference number
ISO 14906:2022(E)

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

Published in Switzerland

Contents

Page

Foreword

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

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

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

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

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

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

This fourth edition cancels and replaces the third edition (ISO 14906:2018), which has been technically revised. It also incorporates the Amendment ISO 14906:2018/Amd 1:2020.

The main changes are as follows:

— updating of terms and definitions, including reference to ISO/TS 17573-2 as the primary source;

— updating of data definitions, including reference to ISO 17573-3:—[1)] as the primary source;

— addition of a second level of the version identifier (i.e. minor version) of the abstract syntax notation one (ASN.1) module to provide enhanced support to standards that import data types from this document;

— introduction of use of imported ASN.1 types with successors (i.e. including all future minor versions).

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

1) Under preparation. Stage at the time of publication: ISO/DIS 17573-3:2022.

Introduction

This document specifies an application interface for electronic fee collection (EFC) systems, which is based on dedicated short-range communication (DSRC). It supports interoperability between EFC systems on an EFC-DSRC application interface level. This document 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 can potentially 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 document 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 the 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 document. 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 document provides the basis for agreements between operators, which are needed to achieve interoperability. Based on the tools specified in this document, interoperability can be reached by operators recognizing each others' EFC transactions (including the exchange of security algorithms and keys) and implementing the EFC transactions in each others' RSE, 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 edition of this document, users are faced with issues related to backward compatibility. Such issues can be managed by using the following:

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

— EfcContextMark (incl. the ContextVersion), denoting the implementation version; this 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 its capability to accept the new features introduced by this edition of ISO 14906.

This application interface definition can also be used with other DSRC media which do not use 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 provides a detailed description of a transaction for central account systems, this document can also be used for on-board account systems, in conjunction with ISO 25110, which provides examples of systems based on on-board accounts.

This document also includes:

— Annex A, which provides the normative ASN.1 data type specifications (EFC action parameters and attributes);

— Annex B, which presents an example of a transaction based on the CARDME specification, including bit-level specification;

— Annex C, which presents examples of EFC transaction types, using the specified EFC functions and attributes;

— Annex D, which presents a mapping table from LatinAlphabetNo2 & 5 to LatinAlphabetNo1 to facilitate for a service provider the use of LatinAlphabetNo1 to encode an OBE for data available written with non-Latin1 characters;

— Annex E, which presents a mapping table between EFC vehicle data attributes and European registration certificates to facilitate the task of a service provider in OBE-personalization with vehicle data;

— Annex F, which presents the security calculations according to the data encryption standard (DES).

NOTE 1 Annex F is based on EN 15509:2014, Annex B.

— Annex G, which presents security computations examples for DES;

NOTE 2 Annex G is based on EN 15509:2014, Annex E.

— Annex H, which presents the security calculations for advanced encryption standard (AES);

NOTE 3 Annex H is an adaptation of EN 15509:2014, Annex B for the case of AES.

— Annex I, which presents the security computations examples for AES.

NOTE 4 Annex I is an adaptation of EN 15509:2014, Annex E for the case of AES.

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

1 Scope

This document specifies the application interface in the context of electronic fee collection (EFC) systems using 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. This document 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. integrated circuit(s) card);

— 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.

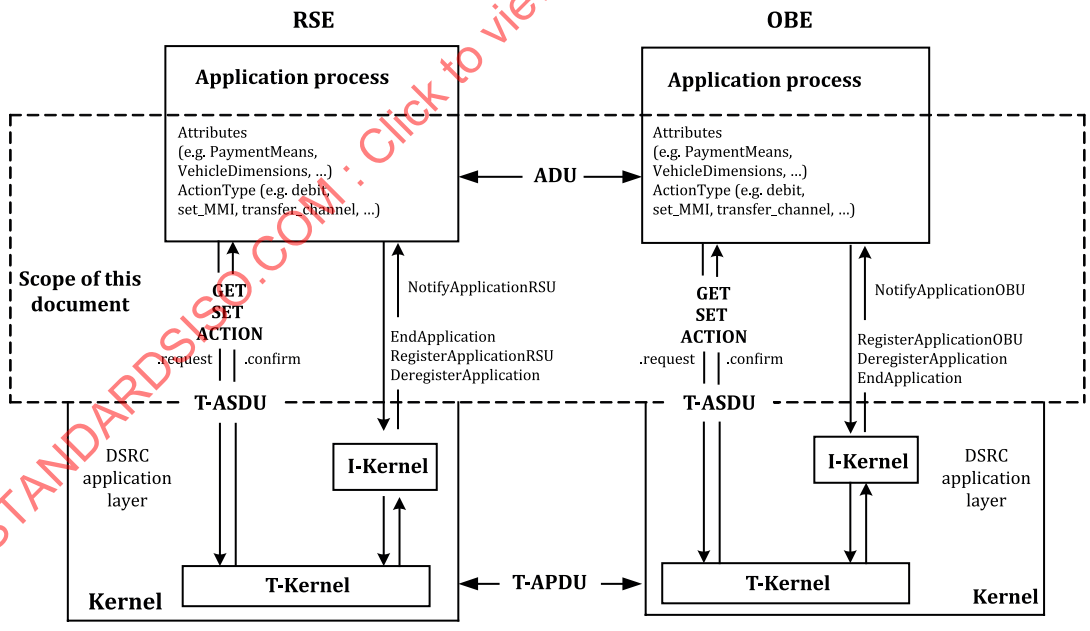


Figure 1 — The EFC application interface

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

This document 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 document.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

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

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

ISO 4217, *Codes for the representation of currencies*

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

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

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

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

ISO 15628:2013, *Intelligent transport systems — Dedicated short range communication (DSRC) — DSRC application layer*

ISO 17573-3:—[2)], *Electronic fee collection — System architecture for vehicle-related tolling — Part 3: Data dictionary*

ISO/IEC 18033-3:2010, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*

EN 12834, *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.

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

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

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

3.1
access credentials
trusted attestation or secure module that establishes the claimed identity of an object or application

[SOURCE: ISO/TS 17573-2:2020, 3.4]

2) Under preparation. Stage at the time of publication: ISO/DIS 17573-3:2022.

3.2
attribute
addressable package of data consisting of a single data element or structured sequences of data elements

[SOURCE: ISO/TS 17573-2:2020, 3.13]

3.3
authenticator
data, possibly encrypted, that is used for authentication

[SOURCE: ISO/TS 17573-2:2020, 3.16]

3.4
channel
information transfer path

[SOURCE: ISO/TS 17573-2:2020, 3.30]

3.5
cryptography
principles, means and methods for the transformation of data in order to hide its information content, prevent its undetected modification or prevent its unauthorized use

[SOURCE: ISO/TS 17573-2:2020, 3.53]

3.6
data group
class of closely related *attributes* (3.2)

[SOURCE: ISO/TS 17573-2:2020, 3.55]

3.7
data integrity
information integrity
property that data has not been altered or destroyed in an unauthorized manner

[SOURCE: ISO/TS 17573-2:2020, 3.56]

3.8
Element
DSRC directory containing application information in the form of *attributes* (3.2)

3.9
on-board equipment
all required equipment on-board a vehicle for performing required EFC functions and communication services

[SOURCE: ISO/TS 17573-2:2020, 3.126]

3.10
on-board unit
single electronic unit on-board a vehicle for performing specific EFC functions and for communication with external systems

[SOURCE: ISO/TS 17573-2:2020, 3.127]

3.11
roadside equipment
fixed or movable electronic fee collection equipment located along or on the road

Note 1 to entry: Roadside equipment (RSE) and roadside unit (RSU) have been used as terms with the same or similar meaning in the standardization of DSRC and DSRC-based EFC within CEN and ETSI. Previously-developed relevant standards used the term "RSU" with the meaning "RSE" as defined in this document.

[SOURCE: ISO/TS 17573-2:2020, 3.161]

3.12
toll charger
entity which levies toll for the use of vehicles in a *toll domain* (3.13)

[SOURCE: ISO/TS 17573-2:2020, 3.194]

3.13
toll domain
area or part of a road network where a toll regime is applied

[SOURCE: ISO/TS 17573-2:2020, 3.201]

3.14
toll service
service enabling users to pay toll

[SOURCE: ISO/TS 17573-2:2020, 3.205]

3.15
toll service provider
entity providing *toll services* (3.14) in one or more *toll domains* (3.13)

[SOURCE: ISO/TS 17573-2:2020, 3.206]

3.16
transaction
whole of the exchange of information between two physically separated communication facilities

[SOURCE: ISO/TS 17573-2:2020, 3.211]

3.17
transaction model
functional model describing the structure of electronic payment *transactions* (3.16)

[SOURCE: ISO/TS 17573-2:2020, 3.213]

4 Abbreviated terms

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

ADU	application data unit
AES	advanced encryption standard
AP	application process
APDU	application protocol data unit
ASN.1	abstract syntax notation one
BST	beacon service table

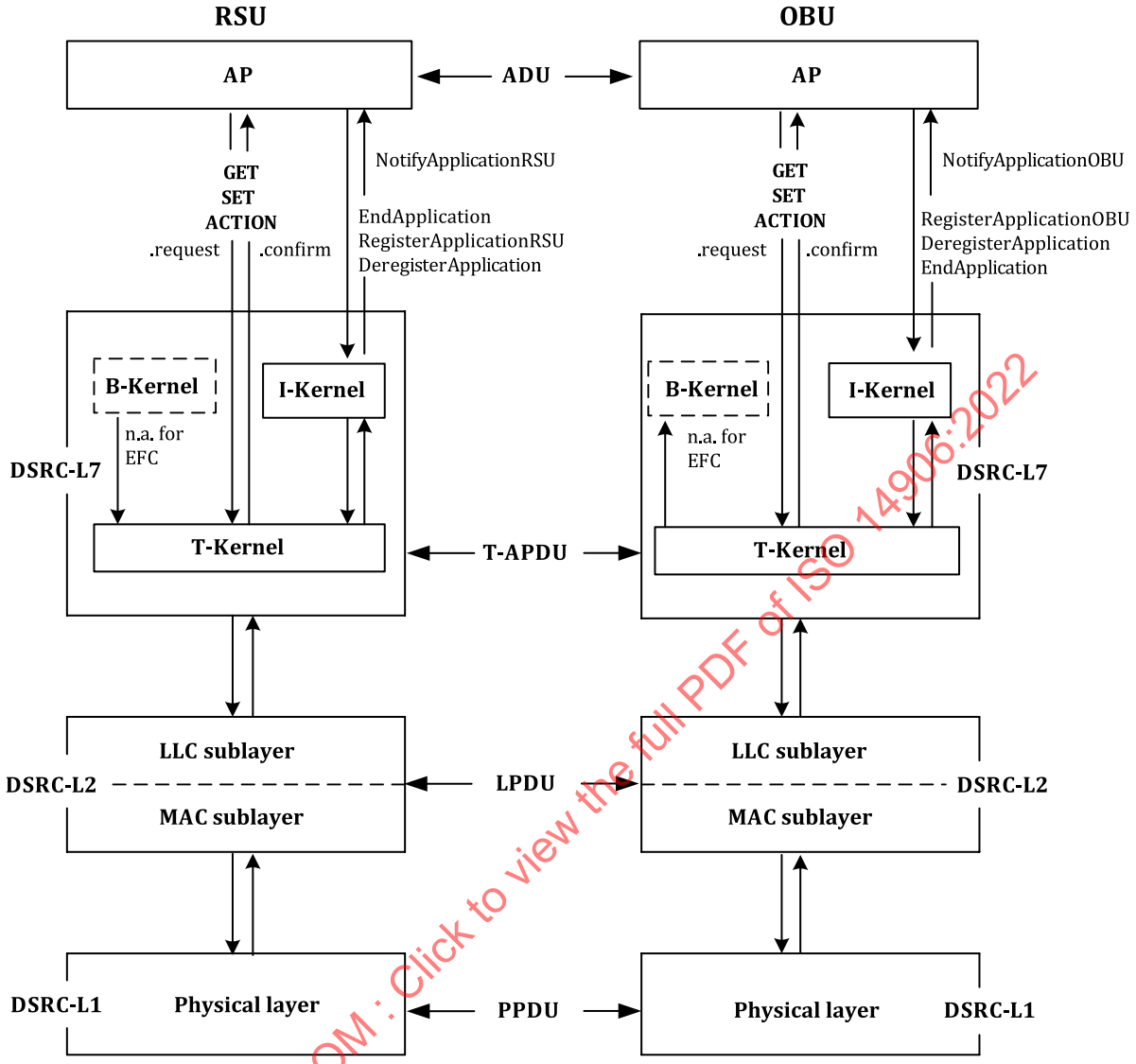
CCC	compliance check communication
cf	confirm
DES	data encryption standard
DSRC	dedicated short-range communication
EFC	electronic fee collection
EID	Element Identifier
GNSS	global navigation satellite system
ICC	integrated circuit(s) card
IID	invoker identifier
I-Kernel	initialisation kernel
ind	indication
L1	layer 1 of DSRC (physical layer)
L2	layer 2 of DSRC (data link layer)
L7	application layer core of DSRC
LAC	localisation augmentation communication
LID	logical link control identifier
LLC	logical link control
LPDU	LLC protocol data unit
MAC	medium access control
MMI	man-machine interface
n.a.	not applicable
OBE	on-board equipment
OBU	on-board unit
PDU	protocol data unit
PER	packed encoding rules
PPDU	physical protocol data unit
req	request
rs	response
RSE	roadside equipment
RSU	roadside unit
SAM	secure application module

T-APDU	transfer-application protocol data unit
T-ASDU	transfer-application service data unit
T-Kernel	transfer kernel
TC	toll charger
TSP	toll service provider
VST	vehicle service table
XOR	exclusive or (logical operation)

5 EFC application interface architecture

5.1 Relation to the DSRC communication architecture

The DSRC services are provided to an application process (AP) 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) as shown in Figure 2.



NOTE The abbreviated terms used in Figure 2 are defined in Clause 4.

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

The transfer kernel (T-Kernel)of DSRC application layer offers the following services to application processes (see Figure 2):

— 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 an 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 ISO 15628):

— The INITIALISATION services (listed below) are used to realize the EFC-specific initialisation mechanism (see Clause 6):

  — 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),

— 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 realize additional EFC specific functionality needed to support EFC application processes, such as TRANSFER_CHANNEL, SET_MMI and ECHO (see 7.2).

In the following subclauses, 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. For example, it is 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 document.

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 abstract syntax notation one (ASN.1) types. Each data element is associated with, within the context of this document, 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.

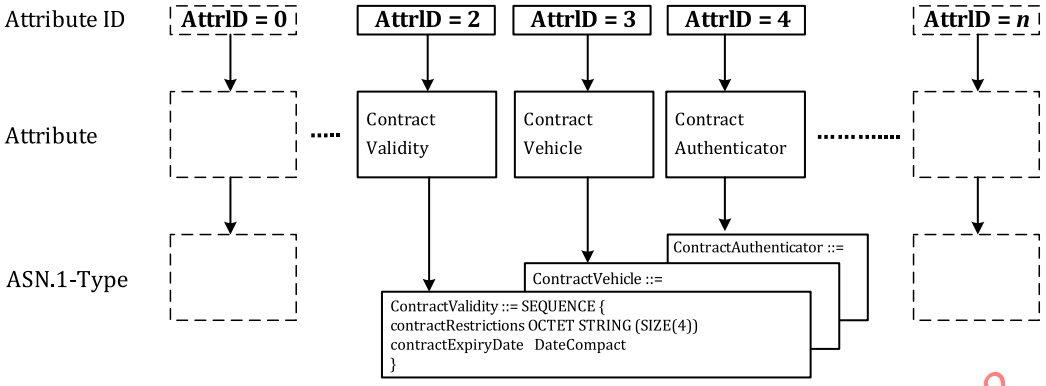


Figure 3 — Basic addressing mechanism

5.3.2 Role of the Element Identifier (EID)

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

EXAMPLE Figure 4 illustrates the role of the EID.

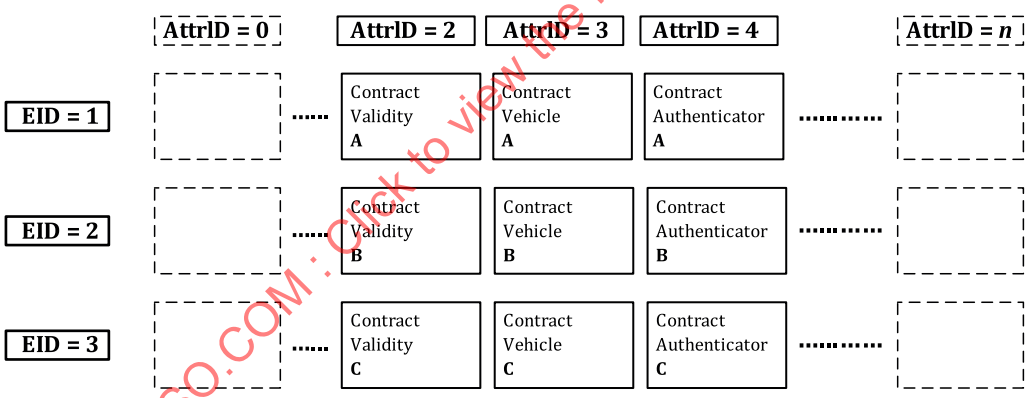


Figure 4 — Role of the EID

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 the 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 illustrates multiple instances (0-2) of attribute 5.

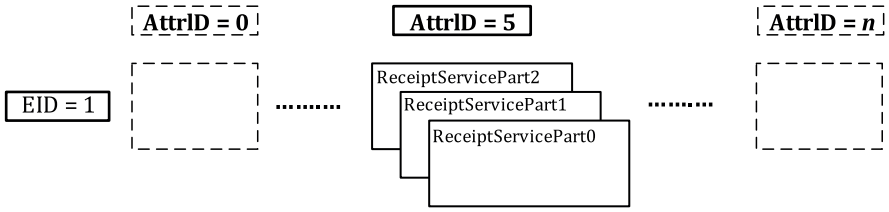


Figure 5 — Multiple instances (0-2) of attribute 5

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;

— secure application module (SAM) 1;

— SAM 2;

— ICC;

— display;

— buzzer;

— printer;

— serial interface;

— parallel interface;

— global navigation satellite system (GNSS);

— 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 integrated circuit(s) card (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 can activate some man-machine interface (MMI) function (e.g. a SET command on the attribute ReceiptText can display the text on a display. A SET command on the attribute ReceiptServicePart with data element SessionResultOperational other than SessionOK can activate an alert beep). Proprietary addressing mechanisms are not defined by this document.

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 OBE 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 ITS 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 15628:2013, 6.2, may also be used in an EFC transaction phase.

6.2 Initialisation phase

6.2.1 Overview

This subclause 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. Subclauses 6.2.2 and 6.2.3 account for the EFC application-specific information that shall be included in the BST and VST, respectively, as shown in Figure 6.

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 the OBE should not exchange any information with the RSE. If the OBE supports at least one of the application(s) supported by the RSE, then the OBE should inform the RSE of which application it wishes to execute in its corresponding VST.

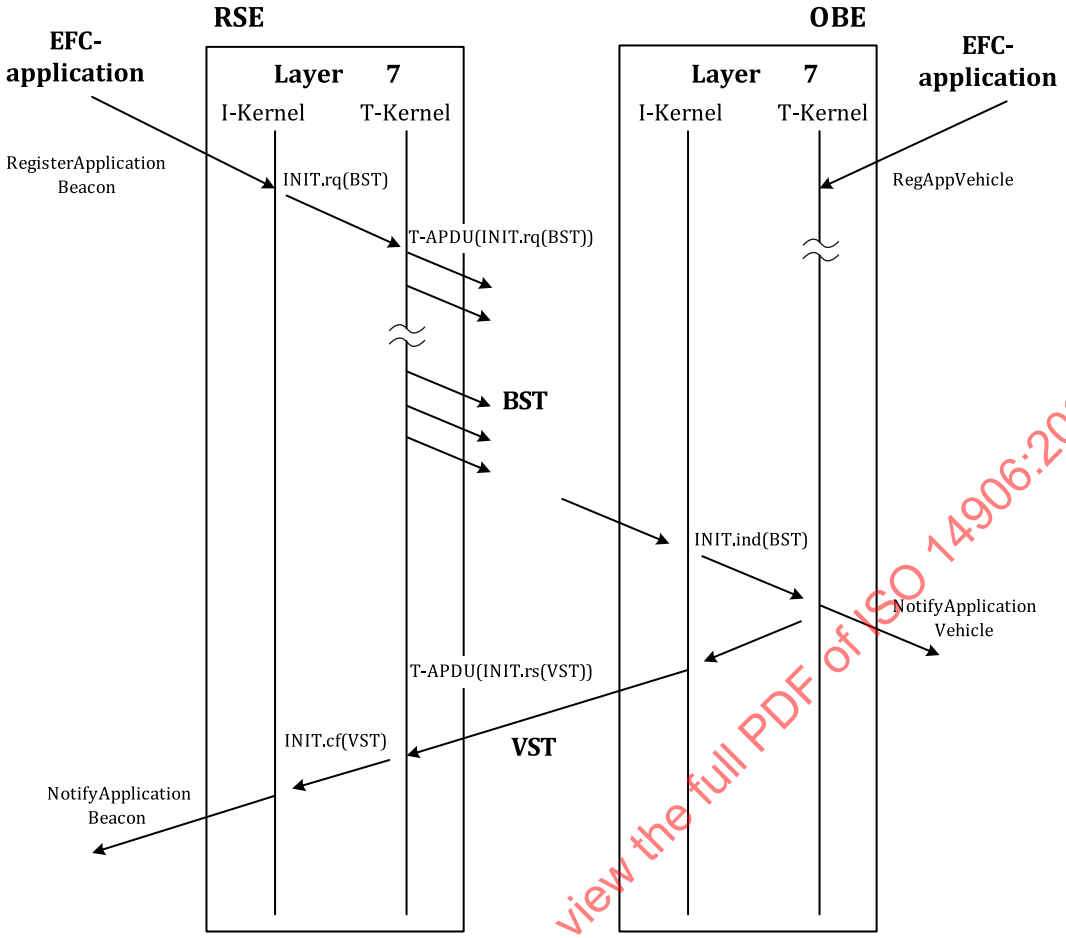


Figure 6 — Initialisation phase: BST - VST exchanges

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 equal to 14 identifies the multi-purpose payment context. In Japan, this document specifies the application interface for DSRC used for multi-purpose payment (when 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 EfcContextMark (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 here for readability of this document:

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

where

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

6.2.3 EFC application-specific contents of the VST

Each EFC application and corresponding contract shall be associated with an EfcContextMark, as defined in this subclause. An OBE may support several EFC applications.

NOTE 1 It is outside the scope of this document to define the presentation order of EfcContextMarks in order to indicate a user's order of preference, in case the user's OBE supports several EFC applications. Such rules for indicating the user's order of preference can 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 EfcContextMark, 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 EfcContextMark as defined here, and may also be configured to carry additional EFC attributes (as defined in Clause 8 and in accordance with Annex A).

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

The extensibility of the ContextVersion should not be used. ContextVersion is coded as a single octet.

The EfcContextMark 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 EfcContextMark.

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 here for readability of this document:

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

where

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

NOTE 3 Means to ensure backwards compatibility and co-existence are as follows.

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

— EfcContextMark (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 An EFC application provider retains the ultimate control of its security domain, i.e. the security level and the associated security mechanisms to be used within its 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”). Annex A applies.

NOTE 5 Retrofit DSRC OBE are mostly battery powered, and the battery usually has a lifetime that is dependent on the actual usage of the OBE (number of transactions per day, activation of MMI, etc.). In an interoperable environment, toll charger (TCs) can access the OBE via the DSRC interface, check the status of the battery and signal this status 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 EfcContextMark 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 EfcContextMark. 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 can 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 EfcContextMark and shall be used by the RSE as a parameter of every function to unambiguously address data elements within the context given by the EfcContextMark. More than one EID may be used in one session.

The attributes to be implemented and the maximum number of instances of an attribute are defined at time of configuration of the OBE, which is outside the scope of this document. These implementation-dependent aspects are referenced unambiguously by the EfcContextMark (for each element present in the OBE) shown in Figure 7.

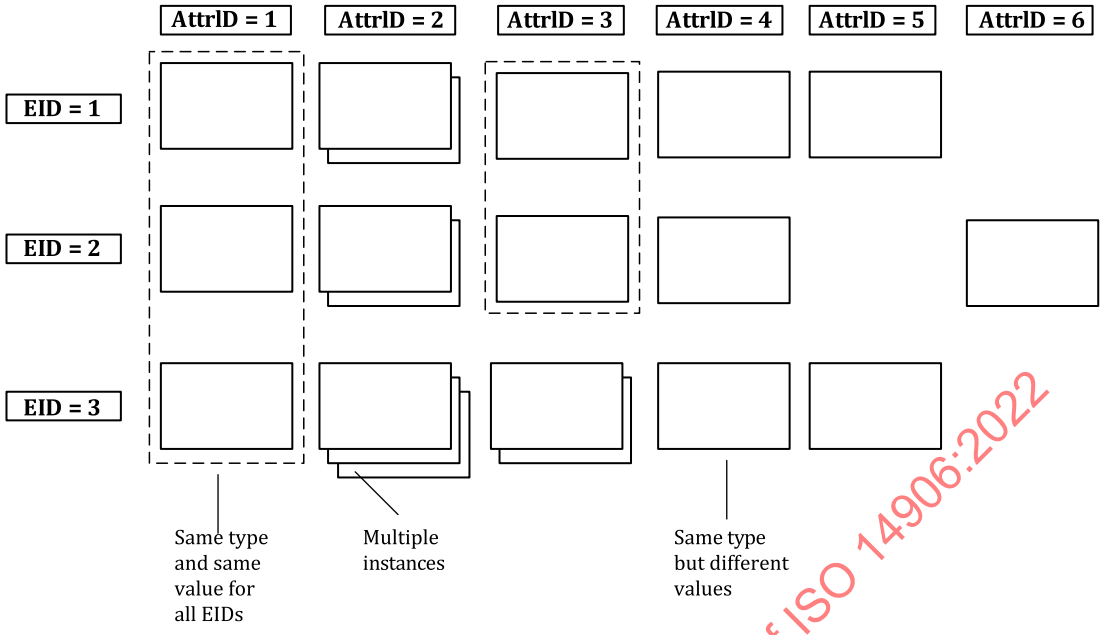


Figure 7 — Context of attributes given by EfcContextMark and identified by the EID

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 can 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 "its" 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 itself.

— Agreement between operators (TCs and ETS Provider): each TC 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 transfer-application service data units (T-ASDU) 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 EN 12834:2003, 6.2.4/ISO 15628:2013, 6.2.4).

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

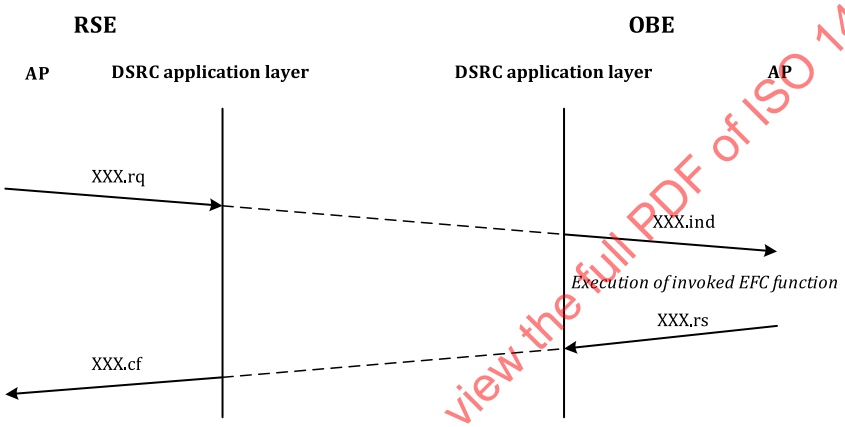


Figure 8 — Logical sequencing of service primitive exchanges

For the purposes of this document, 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 15628:2013, 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. 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

Table 1 *(continued)*

Function name	Action type	Action parameter	Response parameter	Remarks
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 15628:2013, 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 $N_2 \geq N_1$, then the values of the instances numbered N_1, N_1+1, ... up to and including min. (N_2, n).

  The case that zero instance is returned is permissible. 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 1 A cyclic buffer is as acceptable as a dynamic memory allocation scheme.

NOTE 2 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 $N \geq 0$, 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

Security is an essential part of EFC applications. The actual mechanisms used by an application provider are outside the scope of this document. It is generally recognized that security mechanisms involve many parameters, e.g. encryption algorithm and keys (if the security mechanism is encryption based), 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 document, for calculation of access credentials using DES, 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. This document describes security mechanisms for access credentials and authenticators, based on data encryption standard (DES) and advanced encryption standard (AES) primitives.

The security mechanisms to be applied and the exact role of access credentials and authenticators are outside the scope of this document.

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

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. Access credentials are used in transactions, in order to protect against non-authorized access to sensitive user data and against (commercial) use of the OBU by non-authorized RSE.

EXAMPLE The VehicleDimensions EFC attribute can be associated with no access conditions whilst the ContractSerialNumber and ContractValidity EFC attributes can 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 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.

7.1.4.3 Principle of access credentials

The principle of access credentials to the OBU information is shown in Figure 9. When an OBU, having entered the communication zone, responds to a polling message (BST) from the RSE, it returns a VST that for each available contract contains information about an access credential reference (AC_CR-KeyReference) and a random number (RndOBE). Access credential reference contains the diversifier and a reference to a secret master key (MAcK) that shall be used for the computation of the secret key (AcK). This key shall be used for the computation of the access credentials (AC_CR) using the RndOBE number as input. The RSE returns the access credential calculated and the OBU compares the access credential with its own calculation. If they are equal, the OBU accepts the RSE as a genuine RSE and reading data from the OBU is allowed.

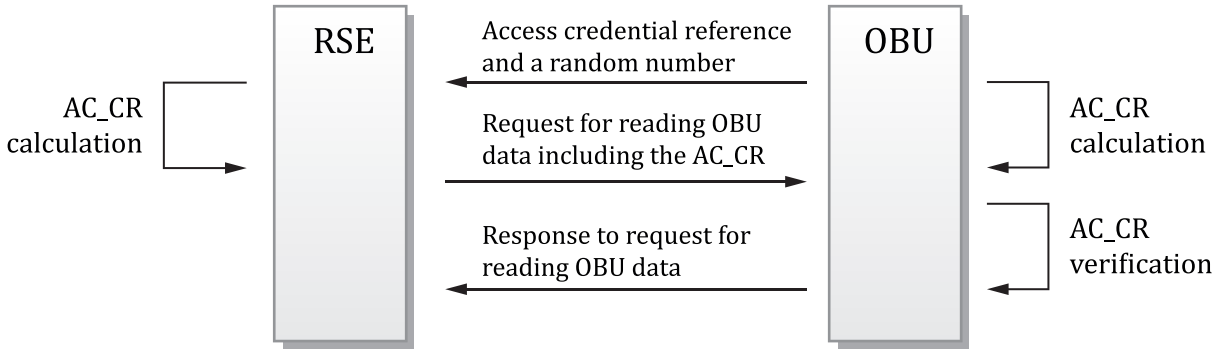


Figure 9 — Principle of access control to the OBU data

Parameters needed (AC_CR-KeyReference and RndOBE) shall be included in the VST application parameter. The encoding of the EfcContextMark and the security parameters into `OCTET STRING` is done with an ASN.1 data type defintion equivalent to:

```
ApplicationContextMark ::= SEQUENCE {
    efcContextMark      EfcContextMark,
    ac-cr-keyReference  EfcContainer (WITH COMPONENTS {octetstring PRESENT}),
    rndOBE              EfcContainer (WITH COMPONENTS {octetstring PRESENT})
}
```

The size of the RndOBE may vary with the requirements of the access credential method and the application.

7.1.4.4 Access credentials with DES

If the OBU supports calculation of access credentials for protection of user related data on the OBU based on DES, these calculations shall be performed according to F.3.

The security related data elements listed in Table 4 shall be implemented in the OBU.

Table 4 — OBU security related data for handling of access credentials

Name	Length (octets)	Remarks
AccessKey	8	Private.
AC_CR	4	Access credentials calculated by the RSE and the OBU using RndOBE and the access key AC_CRKey.
AC_CR-KeyReference	2	Reference to the key generation and the diversifier for the computation of AC_CRKey.
RndOBE	4	Random number (nonce) used together with the access key (referenced through AC_CR-KeyReference) to calculate the access credentials.

7.1.4.5 Access credentials with AES

If the OBU supports calculation of access credentials for the protection of user-related data on the OBU based on AES, these calculations shall be performed according to H.3.

The security-related data elements listed in Table 5 shall be implemented in the OBU.

Table 5 — Overview of the OBU security related data

Name	Length (octets)	Remarks
AccessKey	**16**	Private.
AC_CR	**8**	Access credentials calculated by the RSE and the OBU using RndOBE and the access key AC_CRKey.
AC_CR-KeyReference	**2**	Reference to the key generation and the diversifier for the computation of AC_CRKey.
RndOBE	**8**	Random number (nonce) used together with the access key (referenced through AC_CR-KeyReference) to calculate the access credentials.

7.1.4.6 Authenticator with DES

The security-related data elements listed in Table 6 shall be implemented in the OBU.

Table 6 — Overview of the OBU security related DES data

Name	Length (octets)	Remarks
AuthenticationKey1	**8**	Private.
AuthenticationKey2	**8**	Private.
AuthenticationKey3	**8**	Private.
AuthenticationKey4	**8**	Private.
AuthenticationKey5	**8**	Private.
AuthenticationKey6	**8**	Private.
AuthenticationKey7	**8**	Private.
AuthenticationKey8	**8**	Private.
KeyRef	**1**	Reference to AuKey used for the computation of the authenticators, e.g. toll service provider (TSP) and TC authenticators. The TSP decides which keys are shared with (MoU) TCs (referenced through AuKey_Op), and which are only known by the TSP (referenced through AuKey_Iss).
RndRSE	**4**	Random number, containing SessionTime, from RSE used for the computation of authenticator.

If the OBU supports calculation of an authenticator (i.e. supports the GET_STAMPED function operating on one attribute list considering that the response shall fit into one Layer 2 frame) to validate data integrity and origin of the application data, these calculations shall be performed according to F.2.

7.1.4.7 Authenticator with AES

The security-related data elements listed in Table 7 shall be implemented in the OBU.

Table 7 — Overview of the OBU security related AES data

Name	Length (octets)	Remarks
AuthenticationKey1	**16**	Private.
AuthenticationKey2	**16**	Private.
AuthenticationKey3	**16**	Private.
AuthenticationKey4	**16**	Private.

Table 7 *(continued)*

Name	Length (octets)	Remarks
AuthenticationKey5	**16**	Private.
AuthenticationKey6	**16**	Private.
AuthenticationKey7	**16**	Private.
AuthenticationKey8	**16**	Private.
KeyRef	**1**	Reference to AuKey used for the computation of the authenticators, e.g. TSP and TC authenticators. The TSP decides which keys are shared with (MoU) TCs (referenced through AuKey_Op), and which are only known to the TSP (referenced through AuKey_Iss).
RndRSE	**8**	Random number from RSE used for the computation of authenticator.
RndOBE2	**8**	Random number (nonce) used together with AttributeList and RndRSE to calculate the authenticator.

If the OBU supports calculation of an authenticator (i.e. support the GET_STAMPED function operating on one attribute list considering that the response shall fit into one Layer 2 frame) to validate data integrity and origin of the application data, these calculations shall be performed according to H.2.

7.2 EFC functions

7.2.1 General

In this subclause, 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 EN 12834:2003, 6.2.2 (Transfer Kernel Services)/ISO 15628:2013, 6.2.2. Not all parameters associated with the EFC functions are accounted for in this subclause, 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 EN 12834/ISO 15628:2013, 6.2.4.

The ASN.1 type specifications of the ActionParameters and ResponseParameters are provided in 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).

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 8.

Table 8 — 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

Table 8 *(continued)*

Parameter name	ASN.1 type	Value	Remark/constraints

ActionParameter	`GetStampedRq ::= SEQUENCE {`
	`attributeIdList AttributeIdList,`
	`nonce OCTET STRING,`
	`keyRef Int1Unsigned`
	`}`
			always to be present

Mode	`BOOLEAN`	TRUE	

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

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 9.

Table 9 — 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

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).

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 10.

Table 10 — 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 Int1Unsigned }`
			always to be present

Table 10 *(continued)*

Parameter name	ASN.1 type	Value	Remark/constraints
Mode	BOOLEAN	TRUE	

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 11.

Table 11 — SET_STAMPED.response

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

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).

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 12.

Table 12 — 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	

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 an AttrIDList (possibly encrypted), and it can, for example, also contain an authenticator for non-repudiation purposes.

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 13.

Table 13 — 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

Table 13 *(continued)*

parameter name	ASN.1 type	Value	Remark/constraints
Return Code (Ret)	ReturnStatus		optional use

NOTE 3 The interpretation of the responseParameter is defined by the security mechanism that is in effect. The parameter includes an (possibly encrypted, or otherwise transformed) AttributeList. It can in addition contain, for example, 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).

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 14.

Table 14 — 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		

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 an attrList (possibly encrypted), and it can, for example, also contain an authenticator for non-repudiation purposes.

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 15.

Table 15 — SET_SECURE.response

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

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 can, for example, 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).

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 16.

Table 16 — 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  Int1Unsigned,`
	`  posOfLastInstance  Int1Unsigned,`
	`  attributeIdList  AttributeIdList}`
			always to be present

Mode	`BOOLEAN`	TRUE	

GET_INSTANCE.response shall, as a 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 17.

Table 17 — 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

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).

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 18.

Table 18 — 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  Int1Unsigned,`
	`  attribute  Attr }`
			always to be present

Table 18 *(continued)*

Parameter name	ASN.1 type	Value	Remark/constraints
Mode	BOOLEAN		

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 19.

Table 19 — SET_INSTANCE.response

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

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 need to have a sufficiently large period, time stamps need to have sufficiently high resolution. In addition, random numbers may have to be generated by a cryptographic algorithm to be sufficiently "unpredictable". These additional requirements are outside the scope of this document.

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 20.

Table 20 — 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.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 21.

Table 21 — 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

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.

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 22.

Table 22 — 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.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 the Complexity Limitation. See Table 23.

Table 23 — SET_NONCE.response

Parameter name	ASN.1 type	Value	Remark/constraints
ResponseParameter	n.a.		n.a.
Return Code (Ret)	`ReturnStatus`		optional use

7.2.10 TRANSFER_CHANNEL

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

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 application protocol data unit (APDU) in a format recognized 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 24.

Table 24 — 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

Table 24 *(continued)*

parameter name	ASN.1 type	Value	Remark/constraints
ActionParameter	`ChannelRq ::= SEQUENCE { channelId ChannelId, apdu OCTET STRING }`		
Mode	`BOOLEAN`		

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

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 25.

Table 25 — 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

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 contexts 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 with whom they have an agreement.

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.

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 26.

Table 26 — 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.response shall be used to explicitly convey the result of the corresponding COPY.request command. See Table 27.

Table 27 — COPY.response

Parameter name	ASN.1 type	Value	Remark/constraints
ResponseParameter	n.a.		n.a.
Return Code (Ret)	`ReturnStatus`		optional use

7.2.12 SET_MMI

SET_MMI is used to perform device-independent MMI functions.

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 28.

Table 28 — 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.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 29.

Table 29 — SET_MMI.response

Parameter name	ASN.1 type	Value	Remark/constraints
ResponseParameter	n.a.		n.a.
Return Code (Ret)	`ReturnStatus`		optional use

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.

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 30.

Table 30 — 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.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 31.

Table 31 — SUBTRACT.response

Parameter name	ASN.1 type	Value	Remark/constraints
ResponseParameter	n.a.		
Return Code (Ret)	`ReturnStatus`		optional use

7.2.14 ADD

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

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 32.

Table 32 — 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`		

The AccessCredentials protect the value against unauthorized modification.

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 33.

Table 33 — ADD.response

Parameter name	ASN.1 type	Value	Remark/constraints
ResponseParameter	n.a.		
Return Code (Ret)	`ReturnStatus`		optional use

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.

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 PayUnit, 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 34.

Table 34 — 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, PaymentFee`
	`nonce OCTET STRING,`
	`keyRef Int1Unsigned }`
			always to be present

Mode	`BOOLEAN`	TRUE	

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 PayUnit 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 PayUnit 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 35.

Table 35 — 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

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.

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 PayUnit, 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 36.

Table 36 — 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  Int1Unsigned }`
			always to be present

Mode	`BOOLEAN`	TRUE	

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 PayUnit 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 PayUnit 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 37.

Table 37 — 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

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 localization of the OBE during the passage of the DSRC-EFC station.

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 38.

Table 38 — 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.response shall be used to return the data conveyed in ActionParameter, or the result, of the corresponding ECHO.request command. See Table 39.

Table 39 — 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

8 EFC Attributes

8.1 General

The specification of data elements is specified in ISO 17573-3:—. Within the context of EFC, the following EFC Attributes in Table 40, or a subset thereof, shall be available to perform an EFC transaction.

Table 40 — EFC Attributes

AttributeID	Attribute	Length (octets)	Data group
0	`EfcContextMark`	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	

Table 40 *(continued)*

AttributeID	Attribute	Length (octets)	Data group
9	ReceiptContract	9	
10	ReceiptObuId	Variable	
11	ReceiptIccId	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	EngineDetails	4	
40	SoundLevel	2	
41	ExhaustEmissionValues	8	
42	DieselEmissionValues	4	
43	CO2EmissionValue	2	
44	VehicleTotalDistance	4	
45	TrailerLicencePlateNumber	Variable	
46	TrailerCharacteristics	5	
55	VehicleCurrentMaxTrainWeight	2	
24	EquipmentObuId	5 (1 + 4)	Equipment
25	EquipmentEquipmentIccId	Variable	
26	EquipmentStatus	2	
27	DriverCharacteristics	2	Driver
47	ActualNumberOfPassengers	1	
32	PaymentMeans	14	Payment
29	PaymentMeansBalance	3	
30	PaymentMeansUnit	2	
31	PaymentSecurityData	Variable	
48-53,60-64	*Reserved for ISO 12813*		
54	*Reserved for ISO 13141*		
8,56-59,65-86	*Reserved for future CEN use*		
87-127	*Reserved for private use*		

In an OBE, any attribute or data element which is present shall be filled with '0's (binary) unless otherwise explicitly differently defined.

Not every EFC attribute has to be present in any one implementation of this document in order to be compliant except for the EfcContextMark (in VST). A transaction can be made with a combination of public and private attributes.

NOTE Which EFC attributes are present and which are not is implementation-dependent. The implementation is identified by the context given in the EfcContextMark 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 implies neither any relations with regard to addressing nor to logical interdependence. The specification of the corresponding ASN.1 module is provided in the Annex A.

In case of discrepancies between the ASN.1 types and the length in octets in Clause 8 and the ASN.1 module, the ASN.1 module shall take precedence.

8.2 Data group contract

Table 41 presents information associated with the service rights of the user of a toll service.

Table 41 — Data group contract

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
EfcContextMark	ContractProvider	Identifies the organization that issued the service rights given in the Contract, i.e. the TSP. Numbers shall be assigned on a national basis. It is outside the scope of this document 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 EfcContextMark 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 ISO/TR 14906:1998 (previous edition of this document). ValidityOfContract 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 follow implicitly from TypeOfContract.
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 follow 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.

Table 41 *(continued)*

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
Contract Authenticator	Contract Authenticator	Authenticator calculated by the Contract-Provider 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.

8.3 Data group receipt

Table 42 presents information associated to a specific session, including both financial and operational data.

Table 42 — Data group receipt

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
Receipt Service-Part	SessionTime	Time of session with a two-second resolution.	DateAndTime	4	[01.01.1990, 00:00:00] ... [31.12.2117, 23:59:58], then rollo-ver.	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 reproduction of the price calculation (e.g. claimed or measured vehicle class that was applied.)
	Session Claimed-Class	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	
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.

Table 42 *(continued)*

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
	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.
ReceiptIccId	ReceiptIccId	Identification number of smart card used in the session.	EquipmentIccId	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 m.	INT3	3	0..16777215	Vehicle distance readings (e.g. via an interface to a tachograph) may be used to determine the PaymentFee based on the travelled distance.
ReceiptData1	SessionTime	Date and time of session with a two-second 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.

Table 42 *(continued)*

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
	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 reproduction of the price calculation (e.g. claimed or measured vehicle class that was applied).
	SessionClaimed-Class	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.
	SessionContract-Provider	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 42 *(continued)*

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
ReceiptData2	SessionTime	Date and time of session with a two-second 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 reproduction of the price calculation (e.g. claimed or measured vehicle class that was applied).
	SessionClaimed-Class	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 is an operator decision.
	SessionFee	The amount paid for the service.	PaymentFee	4	0..65535 & multiplier & currency code (12 bits)	Contains a currency designation plus a multiplicator. Currency code shall be according to ISO 4217.
	SessionContract-Provider	Organization that provides the contract used in the session.	Provider	3	AA..ZZ & 0..16383	As ContractProvider defined in Efc-ContextMark attribute.
	SessionTypeOf Contract	ContractProvider-specific designation of the rules that apply to the Contract.	OCTET STRING (SIZE(2))	2		As TypeOfContract defined in Efc-ContextMark attribute.

Table 42 *(continued)*

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
	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.

8.4 Data group vehicle

Table 43 presents information pertaining to the vehicle (i.e. the pulling vehicle) and to the attached trailer(s), if applicable. When more sets of vehicle characteristics are needed within the context of an EID, for example, 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).

Table 43 — Data group vehicle

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
VehicleLicence PlateNumber	VehicleLicence PlateNumber	Claimed licence plate of the vehicle	Lpn	Variable		
Vehicle Identification Number	VehicleIdentification Number	Identification number of vehicles shall be according to 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		Vehicle Axles substructure: DDTTTttt'B where: DD'B°= 2 bits for dual tyre. TTT'B= 3 bits to encode the number of all axles of the tractor. ttt'B = 3 bits to encode the number of all axles of the trailer.
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 10 kg 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 10 kg step.	INT2	2		ISO 1176 Code ISO-M18 maximum design mass of vehicle combination.
European Directive 2003/127/EC[8] 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.						

Table 43 *(continued)*

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
	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*		
VehicleCurrent-MaxTrainWeight	VehicleCurrentMax-TrainWeight	Maximum permissible weight of the complete vehicle train that is currently in operation, which shall be as defined in ISO 1176. 10 kg units, rounded down to the next 10 kg step.	INT2	2		This weight may be lower than VehicleTrainMaximumWeight as it represents the current maximum train weight and not the maximum design mass.
AxleWeight Limits	MaxLadenWeight-OnAxle1	Technically permissible maximum laden weight on axle 1 of the vehicle, 10 kg units, rounded down to the next 10 kg step.	INT2	2		
	MaxLadenWeight-OnAxle2	Technically permissible maximum laden weight on axle 2 of the vehicle, 10 kg units, rounded down to the next 10 kg step.	INT2	2		
	MaxLadenWeight-OnAxle3	Technically permissible maximum laden weight on axle 3 of the vehicle, 10 kg units, rounded down to the next 10 kg step.	INT2	2		
	MaxLadenWeight-OnAxle4	Technically permissible maximum laden weight on axle 4 of the vehicle, 10 kg units, rounded down to the next 10 kg step.	INT2	2		
	MaxLadenWeight-OnAxle5	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	
	NumberOfStanding-Places	Number of standing places of the vehicle	INT1	1	0...255	
European Directive 2003/127/EC[8] 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.						

Table 43 *(continued)*

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
VehicleSpecific Characteristics	VehicleSpecific Characteristics	Further vehicle characteristics. Each enumerated value has a specific meaning assigned. The meaning of some values are defined in this document, others are reserved for future needs.	VehicleSpecific Characteristics	4		Assignment of meaning to the unassigned enumerated values is subject to registration according to the registration procedure specified in EN 12834/ ISO 15628.
EngineDetails	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	
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	
European Directive 2003/127/EC[8] 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.						

Table 43 *(continued)*

EFC Attribute	Data element	Definition	Type	Length (octets)	Value range	Informative remarks
CO2Emission Value	CO2EmissionValue	Vehicle's CO_2 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 m resolution, continuously incremented.	INT4	4	0... 4294967294	The initial value of this attribute may be either the value zero or the vehicle's km reading at the moment of personalization of the OBU.
TrailerLicence PlateNumber	TrailerLicencePlate-Number	Claimed licence plate of the trailer.	Lpn	Variable		
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[8] 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.						

8.5 Data group equipment

Table 44 presents information pertaining to the OBE.

Table 44 — Data group equipment

EFC Attribute	Data element	Definition	Type	Length (octets)	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.
EquipmentEquipmentIccId	EquipmentEquipmentIccId	Identification number of smart card	`EquipmentIccId`	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”).

8.6 Data group driver

Table 45 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.

Table 45 — Data group driver

EFC Attribute	Data Element	Definition	Type	Length (octets)	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.

8.7 Data group payment

Table 46 presents data associated with the payment transaction.

The following provisions 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 to not disclose the balance when only currency is needed to debit the right price.

Table 46 — Data group payment

EFC Attribute	Data element	Definition	Type	Length (octets)	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 PayUnit.	`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 unaligned packed encoding rules according to ISO/IEC 8825-2 shall be applied.

The actual ASN.1 module is contained in files "ISO14906(2022)EfcDsrcApplicationv9.1.asn" and "ISO14906(2022)EfcDsrcGenericv10.1.asn", which can be directly imported in a compiler.

The syntax and semantics of the data types in the ASN.1 types in the above-mentioned files that are imported shall conform to ISO 3166-1, ISO 14816, ISO 15628, ISO 17573-3:— and EN 12834, respectively.

NOTE 1 The abovementioned files are available for download via https://standards.iso.org/iso/14906.

Table A.1 provides the SHA-256 cryptographic hash digests for the referenced files, offering a means to verify the integrity of the referenced files. The SHA-256 algorithm is specified in NIST 180-4.[10]

Table A.1 — SHA-256 cryptographic hash digests

File Name	SHA-256 cryptographic hash digest
ISO14906(2022)EfcDsrcApplicationv9.1.asn	F8D9AAF8415330C7C60045B8BCBBE917500579E83837C18F6B-223FD8EBB0C160
ISO14906(2022)EfcDsrcGenericv10.1.asn	9131BD3FA01DDAA4C302335C70F837C8EC25E3E9A91A1C36DF-7C37F1EB283ABD

NOTE 2 Pasting the text of the file into one of the hash digest computation pages available on the web can result in a non-matching hash digest due to changes in the underlying coding.

Annex B
(informative)

CARDME transaction

B.1 General

Annex B provides an informative example of a transaction by specifying the CARDME transaction. A transaction overview is given in B.2. The transaction phases and the data exchanges are then 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

The initialisation phase is illustrated in Figure B.1.

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.

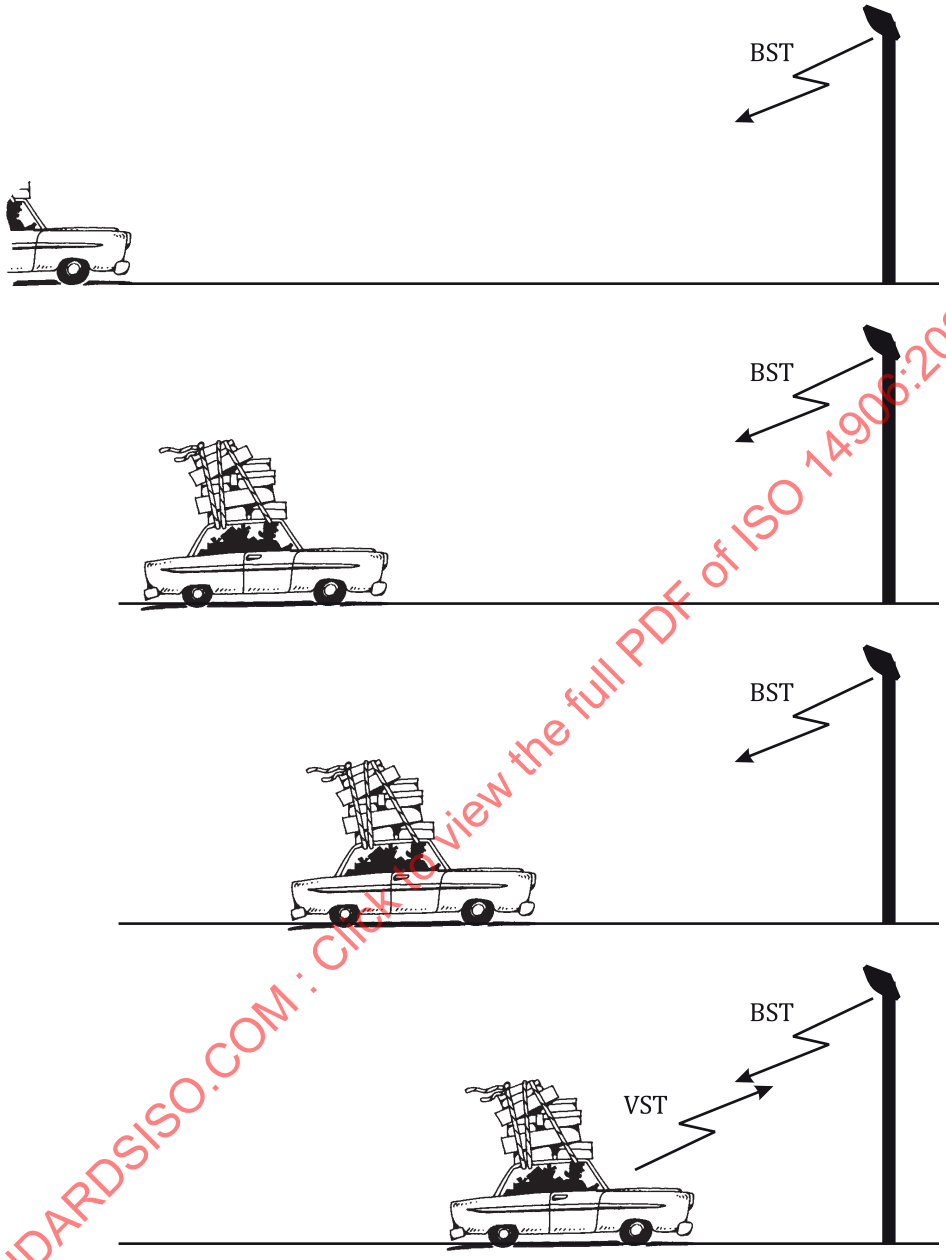


Figure B.1 — Initialisation

Upon reception of the VST the RSE analyses its contents and decides whether it accepts one of the EFC contracts presented by the OBE. If 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 can 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 phase 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.

Table B.2 — Initialisation phase — Information exchange

Roadside 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 "Öresundskonsortiet"* 2) *Transaction type "CARDME" Central account with the Operator "NorwegTrans"*
The RSE thinking: According to my tables, I have the following transactions available and recognize 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 recognize 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 shown in Table B.3.

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

Table B.3 — Presentation phase — Information exchange

Roadside 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"*	

Table B.3 *(continued)*

Roadside equipment	On-board equipment
	"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).

— In a classical manual closed tolling system, a user takes a ticket from an automatic ticket dispensing machine when entering 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 to cover all security needs. These security measures are discussed separately. In CARDME it is mandatory for OBE 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 can simply ignore them. From a technical point of view, every operator is free to decide which of the security data it wishes to control, when and where it wishes to control it and whether it wishes to control it at all.

B.2.1.4 Receipt — Write new OBE data

In the previous phases the RSE has read all data that are required for charging the user (either directly for local users or indirectly for roaming users, which are charged via their contract issuers) shown in Table B.4.

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

Roadside 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 are the entry ticket. In closed tolling systems it is essential that this information be 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 OBE have the capability to display the rather complex receipt information. Rather, the receipt 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 (rather than the OBE) keeps track of what is old and what is new 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, blocklisted, 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". Additionally, 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 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 shown in Table B.5.

Table B.5 — CARDME transaction overview

Phase	Roadside equipment		On-board equipment	Remarks
	INITIALISATION.request (BST)	→		RSE periodically sends BST.
Initialisation (BST – VST)		←	INITIALISATION.response (VST) — EfcContextMark — AC_CR-KeyReference — RndOBE	A newly-arrived OBE answer with VST. AC-CR-KeyReference is the reference 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 OBE	GET_STAMPED.request AC_CR [Access Credential Key] — PaymentMeans including PersonalAccountNumber (RndRSE, KeyRef_Iss)	→		For OBE 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)	

Table B.5 *(continued)*

Phase	Roadside equipment		On-board equipment	Remarks
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.
		←	ECHO.response	The usage of Echo is by optional, at the discretion of the RSE, and may be repeated.
	EVENT_REPORT.request (Release)	→		The RSE closes the transaction and releases the 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): — EfcContextMark: — ContractProvider — TypeOfContract — ContextVersion — (optional additional data) — Data for EFC Contract #2 (e.g. CARDME European Service) — EfcContextMark: — ContractProvider — TypeOfContract — ContextVersion — AC_CR-KeyReference: — AC_CR-MasterKeyReference — AC_CR-Diversifier — RndOBE

NOTE 1 The above data exchange is described on application level. There are also more technical negotiations which occur between RSE and OBE, for example, in order to arrive at a mutually agreed DSRC communication profile determining, amongst other parameters, which subcarrier frequency the OBE is to use on uplink.

NOTE 2 The RSE can issue a command Event-Report Release, according to Layer 7, if it does not recognize the EFC Context mark received by the OBE to avoid disturbance created by an unknown OBE.

1) **Initialisation request (BST) data**

For EFC, the AID code equals 1. The BST is fully standardized and contains nothing CARDME-specific.

2) **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 EfcContextMark information:

— ContractProvider: A code standing for the contract issuer (the code contains the country of residence of the ontract Isisuer 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 can say, for example, “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 is used by the RSE when it calculates the access credentials. This ensures that the RSE calculates the access credentials afresh for each session. This prevents the possibility of someone setting up an unauthorized beacon using access credentials obtained by intercepting correct EFC transactions.

Note that the VST contains no data that can potentially 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	→	

Table B.7 *(continued)*

Phase	RSE		OBE
		←	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 including PersonalAccountNumber: Points to a user account held at the contract issuer (which is already known from the initialisation phase). The EFC operator will send its claim to this account. Is also contains the 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, and in particular 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 only by its own beacons. 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 the two alternative routes the vehicles have 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 personalization 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 harmonized classes", avoiding the necessity of having 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 harmonized 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. Due to the wide distribution of the keys, it is indispensable that these keys be diversified (different OBE have different keys) and that there are key generations (new keys are used from time to time).

NOTE 1 CARDME foresees one generation of access credentials ("Generation 0") which is openly known to all. OBE, or rather contracts with these access credentials, can be read by everybody. It is left to the contract issuer policy (and to agreements) whether it issues such OBE 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 From a technical point of view, there is no need 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 (although it is possible that they can be obliged to do so, for example, through contracts with associated issuers). One possible reason for an operator not wanting to check this authenticator is when they add the CARDME transaction onto their existing RSE which has no appropriate key storage and security handling facilities. At some point in time, for example, when routinely replacing some of their older equipment, they can decide to go for higher security.

— ReceiptAuthenticator (contained in the ReceiptData-attributes): This RSE signature under the receipt has already been covered in B.3.3.3.

— EquipmentStatus: In CARDME this data element serves as a very simple but effective security measure, namely as a simple transaction counter. According to this document, the coding of the data element 'EquipmentStatus" is left to the operator. 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 instances of fraud (e.g. on the part of an operator by duplicating transactions at RSE, especially when using 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.

If these two organizations are different, and especially when they are organizationally strongly separated, for example, 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. The EFC operator will then send a claim to the contract issuer asking for money. The contract issuer now has no means to ready check this claim. The contract issuer has no indisputable proof from its customer, the user that drove the vehicle, that the money asked for indeed 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. The RSE executes this optional presentation phase only for foreign vehicles. 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 sent 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 that contract issuer is obliged to pay for 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 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 covered 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 in 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). What the RSE says is not prescribed. The text may 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 a few OBE have a display, and very few OBE will have the possibility (during the lifetime of this version of this document) to display this text information, so normally OBE 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 meaningless.

2) **SET_MMI**: With this command the RSE instructs the OBE to use its man-machine interface (MMI), to signal the user of 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 the make of the OBE, it 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 and closing phase.

Table B.10 — Tracking and closing phase

Phase	RSE		OBE
Tracking	ECHO.request	→	

Table B.10 *(continued)*

Phase	RSE		OBE
	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 can 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 are used by RSE where required locally. All OBE need to support these functionalities.

B.4 Bit-level specification

B.4.1 General

Tables B.11 to B.23 provide 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 document are highlighted in grey.

B.4.2 Initialisation

B.4.2.1 General

Tables B.11, B.12, B.13 and B.14 provide details of the data exchanged between an RSE and an OBE in the initialisation phase.

B.4.2.2 Initialisation request (BST)

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

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
1	FLAG	0111 1110	Start flag
2	Broadcast logical link control identifier (LID)	1111 1111	Link address for broadcast
3	Medium access control (MAC) control field	1010 0000	The frame contains a command LPDU
4	Logical link control (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 {	1000	INITIALISATION.request
	OPTION indicator	0	NonmandApplications not present.
	BeaconId SEQUENCE { ManufacturerId INTEGER (0..65535)	000	Manufacturer identifier- Example 1 (=Kapsch). See ISO 14816.
7		0000 0000	Register at https://www.itsstandards.eu/registries/ for value assignment.
8		0000 1	

Table B.11 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
	IndividualId INTEGER (0..134217727)	000	27 bits 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 bits 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.4.2.3 Private window request

Table B.12 — Private window request frame content

Octet #	Attribute/field	Bits (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.4.2.4 Private window allocation

Table B.13 — Private window allocation frame content

Octet #	Attribute/field	Bits (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.4.2.5 Initialisation response (VST)

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

Octet #	Attribute/field	Bits (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	1100 0000	The frame contains a command LPDU
7	Logical link control (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,...)	0000 0000	No extension, Profile. Example: 0_{10}
11	Applications SEQUENCE (SIZE((0..127,...)) OF {	0000 0010	No extension, 2 applications
12	SEQUENCE { OPTION indicator	1	EID present
	OPTION indicator	1	Parameter present
	AID DSRCApplicationEntityID	00 0001	No extension, AID = 1, EFC
13	EID	0000 0010	Associated with a context mark. Example: 2_{10}
14	Parameter CONTAINER {	0000 0010	No extension, Container choice = 2_{10}, OCTET STRING
15		0000 0110	No extension, OCTET STRING length = 6_{10}
16	EfcContextMark SEQUENCE {		

Table B.14 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
	ContractProvider SEQUENCE {		
	CountryCode BIT STRING (SIZE(10))	0011 0000	10 bits 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,...) } } }	0000 0010	No extension, context version. Example: 2_{10}
22	SEQUENCE { OPTION indicator	1	EID present
	OPTION indicator	1	Parameter present
	AID DSRCApplicationEntityID	00 0001	No extension, AID = 1, EFC
23	EID	0000 0101	Associated with a context mark. Example: 5_{10}
24	Parameter CONTAINER {	0000 0010	No extension, Container choice = 2_{10}, OCTET STRING
25		0001 0000	No extension, OCTET STRING length = 16_{10}
26	EfcContextMark SEQUENCE {		
	ContractProvider SEQUENCE {		
	CountryCode BIT STRING (SIZE(10))	1010 0100	10 bits 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,...) }	0000 0001	No extension, context version. Example: 1_{10}
32	CONTAINER {	0000 0010	No extension, Container Choice = 2_{10}, OCTET STRING
33		0000 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 {	0000 0010	No extension, Container Choice = 2_{10}, OCTET STRING
37		0000 0100	No extension, OCTET STRING length = 4_{10}

Table B.14 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
38	RndOB E Int4	0000 0000	Random Number (nonce) used together with AC_CRKey to calculate AC_CR. Example: 640_{10}
39		0000 0000	
40		0000 0010	
41	} } } }	1000 0000	
42	ObeConfiguration SEQUENC E {		
	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 https://www.itsstandards.eu/registries/ for value assignment. Example: 2_{10}.
45		0000 0010	
46	ObeStatus INT EGER(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 General

Tables B.15 and B.16 provide details of the data exchanged between an RSE and an OBE in the presentation phase.

B.4.3.2 Presentation request

Table B.15 — Presentation request frame content

Octet #	Attribute/field	Bits (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 LLC protocol data unit (LPDU)
7	LLC control field	n111 0111	Polled ACn command, *n* bit
8	Fragmentation header	1xxx x001	No fragmentation. First service of chain.
NOTE VehicleSpecificCharacteristics, VehicleDimensions and VehicleAxles are not included in the examples in this table.			

Table B.15 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
9	ACTION.request SEQUENCE {	0000	ACTION.request (GET_STAMPED.request)
	OPTION indicator	1	AccessCredentials present
	OPTION indicator	1	ActionParameter present
	OPTION indicator	0	Invoker identifier (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 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 OperatorAuthenticator
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,...)	0000 0101	No extension, EID, Example: 5_{10}
29	AccessCredentials OCTET STRING {	0000 0100	No extension, OCTET STRING length = 4_{10}

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

Table B.15 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
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	0000 0110	No extension, number of attribute IDs = 6_{10}
35	VehicleLicencePlateNumber	0001 0000	No extension, AttributeId = 16_{10}, VehicleLicencePlateNr
36	VehicleClass	0001 0001	No extension, AttributeId = 17_{10}, VehicleClass
37	VehicleWeightLimits	0001 0100	No extension, AttributeId = 20_{10}, VehicleWeightLimits
38	EquipmentStatus	0001 1010	No extension, AttributeId = 26_{10}, EquipmentStatus
39	ReceiptData1	0010 0001	No extension, AttributeId = 33_{10}, ReceiptData1
40	ReceiptData2 } }	0010 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 VehicleAxles are not included in the examples in this table.			

B.4.3.3 Presentation response

Table B.16 — Presentation response frame content

Octet #	Attribute/field	Bits (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	1101 0000	The frame contains a response LPDU
7	LLC control field	n111 0111	9_{10} = 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
NOTE VehicleSpecificCharacteristics, VehicleDimensions and VehicleAxles are not included in the examples in this table.			

Table B.16 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
	Fill BIT STRING(SIZE(1))	0	Set to 0
11	EID INTEGER (0..127,...)	0000 0101	No extension, EID, Example: 5_{10}
12	ResponseParameter CONTAINER {	0001 0010	No extension, Container choice = 18_{10}, GetStampedRs
13	AttributeList SEQUENCE (SIZE(0..127,...)) OF {	0000 0001	No extension, number of attributes: 1
14	Attributes SEQUENCE { AttributeId INTEGER (0..127,...)	0010 0000	No extension, AttributeId = 32_{10}, PaymentMeans
15	AttributeValue CONTAINER {	0100 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 {	0000 0100	No extension, OCTET STRING size = 4_{10}
31	OperatorAuthenticator	xxxx xxxx	OperatorAuthenticator 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

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

Table B.16 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
	Fill BIT STRING(SIZE(1))	0	Set to 0
37	EID INTEGER(0..127,...)	0000 0101	No extension, EID, Example: 5_{10}
38	AttributeList SEQUENCE (SIZE(0..127,...)) OF {	0000 0110	No extension, 6 attributes in list.
39	Attributes SEQUENCE { AttributeId INTEGER (0..127,...)	0001 0000	No extension, AttributeId = 16_{10}, VehicleLicencePlateNo
40	Attribute Value CONTAINER {	0010 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,...)	0001 0001	No extension, AttributeId = 17_{10}, VehicleClass
51	Attribute Value CONTAINER {	0011 0001	No extension, Container choice = 49_{10}
52	VehicleClass Int1 } }	xxxx xxxx	VehicleClass value
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
NOTE VehicleSpecificCharacteristics, VehicleDimensions and VehicleAxles are not included in the examples in this table.			

Table B.16 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
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	ReceiptData 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
NOTE VehicleSpecificCharacteristics, VehicleDimensions and VehicleAxles are not included in the examples in this table.			

Table B.16 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
96	Attribute Value CONTAINER {	0100 0001	No extension, Container choice = 65_{10}
97	ReceiptData	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 VehicleAxles are not included in the examples in this table.			

B.4.4 Optional presentation

B.4.4.1 General

Tables B.17 and B.18 provide details of the data exchanged between an RSE and an OBE in the optional presentation phase.

B.4.4.2 Optional presentation request

Table B.17 — Optional presentation request frame content

Octet #	Attribute/field	Bits (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

Table B.17 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
14		aaaa aaaa	Access credentials key AC_CRKey.
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 IssuerAuthenticator.
26	FCS	xxxx xxxx	Frame check sequence
27		xxxx xxxx	
28	FLAG	0111 1110	End flag

B.4.4.3 Optional presentation response

Table B.18 — Optional presentation response frame content

Octet #	Attribute/field	Bits (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	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

Table B.18 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
	Fill BIT STRING(SIZE(1))	0	Set to 0
11	EID INTEGER (0..127,...)	0000 0101	No extension, EID, Example: 5_{10}
12	ResponseParameter CONTAINER {	0001 0010	No extension, Container choice = 18_{10}, GetStampedRs
13	AttributeList SEQUENCE (SIZE(0..127,...)) OF {	0000 0001	No extension, number of attributes: 1
14	Attributes SEQUENCE { AttributeId INTEGER (0..127,...)	0010 0000	No extension, AttributeId = 32_{10}, PaymentMeans
15	AttributeValue CONTAINER {	0100 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 {	0000 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 General

Tables B.19 and B.20 provide details of the data exchanged between an RSE and an OBE in the Receipt phase.

B.4.5.2 Set receipt request

Table B.19 — Set receipt request frame content

Octet #	Attribute/field	Bits (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
13		aaaa aaaa	Access credentials key AC_CRKey.
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}, Receipt-Text
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	

Table B.19 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
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
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	

Table B.19 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
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	HD 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	SetMMIRq 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.4.5.3 Set receipt response

Table B.20 — Set receipt response frame content

Octet #	Attribute/field	Bits (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	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

Table B.20 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
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

B.4.6 Tracking and closing

B.4.6.1 General

Tables B.21, B.22 and B.23 provide details of the data exchanged between an RSE and an OBE in the tracking and closing phases.

B.4.6.2 Tracking request (ECHO.request)

Table B.21 — Tracking request frame content

Octet #	Attribute/field	Bits (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
9	ACTION.request SEQUENCE {	0000	ACTION.request (ECHO.request)

Table B.21 *(continued)*

Octet #	Attribute/field	Bits (octet) b_7 b_0	Description
	OPTION indicator	0	No access credentials
	OPTION indicator	1	ActionParameter present
	OPTION indicator	0	IID not present
	Mode BOOLEAN	1	Mode = TRUE, Response expected
10	EID INTEGER (0..127,...)	0000 0000	No extension, EID = 0 (System Element)
11	ActionType INTEGER (0..127,...)	0000 1111	No extension, Action Type = 15_{10}, ECHO.request
12	ActionParameter CONTAINER {	0000 0010	No extension, Container choice = 2_{10}, OCTET STRING
13	} }	0000 0000	No extension. String length = 0 octets
14	FCS	xxxx xxxx	Frame check sequence
15		xxxx xxxx	
16	FLAG	0111 1110	End flag

B.4.6.3 Tracking response (ECHO.response)

Table B.22 — Tracking response frame content

Octet #	Attribute/field	Bits (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	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
10	ACTION.response SEQUENCE {	0001	ACTION.response (ECHO.response)
	OPTION indicator	0	No IID
	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,...)	0000 0000	No extension, EID = 0 (system element)
12	ResponseParameter CONTAINER {	0000 0010	No extension, Container choice = 2_{10}, OCTET STRING
13	} }	0000 0000	No extension. String length = 0 octets
14	FCS	xxxx xxxx	Frame check sequence
15		xxxx xxxx	
16	FLAG	0111 1110	End flag

B.4.6.4 Closing

Table B.23 — Closing frame content

Octet #	Attribute/field	Bits (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	1000 0000	The frame contains a command LPDU
7	LLC control field	0000 0011	UI command
8	Fragmentation header	1xxx x001	No fragmentation
9	EVENT_REPORT.request SEQUENCE {	0010	EVENT_REPORT.request (RELEASE)
	OPTION indicator	0	AccessCredential not present
	OPTION indicator	0	EventParameter not present
	OPTION indicator	0	IID not present
	Mode BOOLEAN	0	Mode = FALSE, No response expected
10	EID INTEGER (0..127,...)	0000 0000	No extension, EID = 0 (system element)
11	EventType INTEGER (0..127,...) }	0000 0000	No extension, Event Type = 0, RELEASE
12	FCS	xxxx xxxx	Frame check sequence
13		xxxx xxxx	
14	FLAG	0111 1110	End flag

Annex C
(informative)

Examples of EFC transaction types

C.1 General

Annex C provides informative examples EFC transaction types, using the specified EFC functions and attributes in this document. Examples are given for the following transaction types:

— read-only EFC transaction;

— read and write EFC transaction;

— EFC purse transaction using the DEBIT function;

— on-board account transaction using ICC and the TRANSFER_CHANNEL function;

— multiple contracts EFC transactions.

The purpose of these examples is to demonstrate various transaction concepts, and to illustrate how they are supported by this document.

NOTE Annex B partly overlaps with the "Read and write EFC transaction" (Clause C.3), as both are based on a "read and write" centrally-held account transaction. CARDME's transaction, in Annex B, includes dynamic security measures and a more detailed specification.

C.2 Read-only EFC transaction

The example described in Figure C.1 is based on a centrally held account read only transaction, without any dynamic security measure.

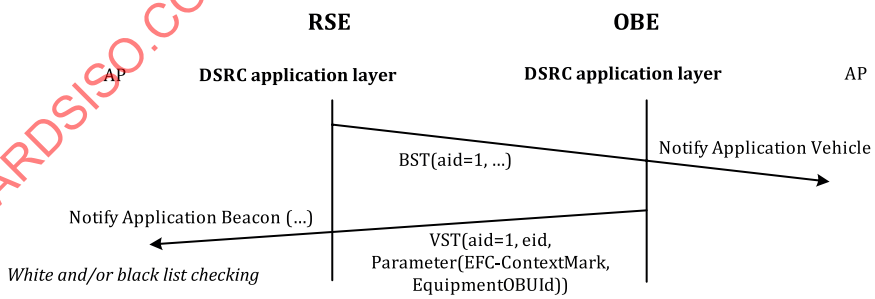


Figure C.1 — Read-only EFC transaction

C.3 Read and write EFC transaction

The example described in Figure C.2 is based on a centrally-held account transaction, without any dynamic security measure, performed at an exit station within a closed system.

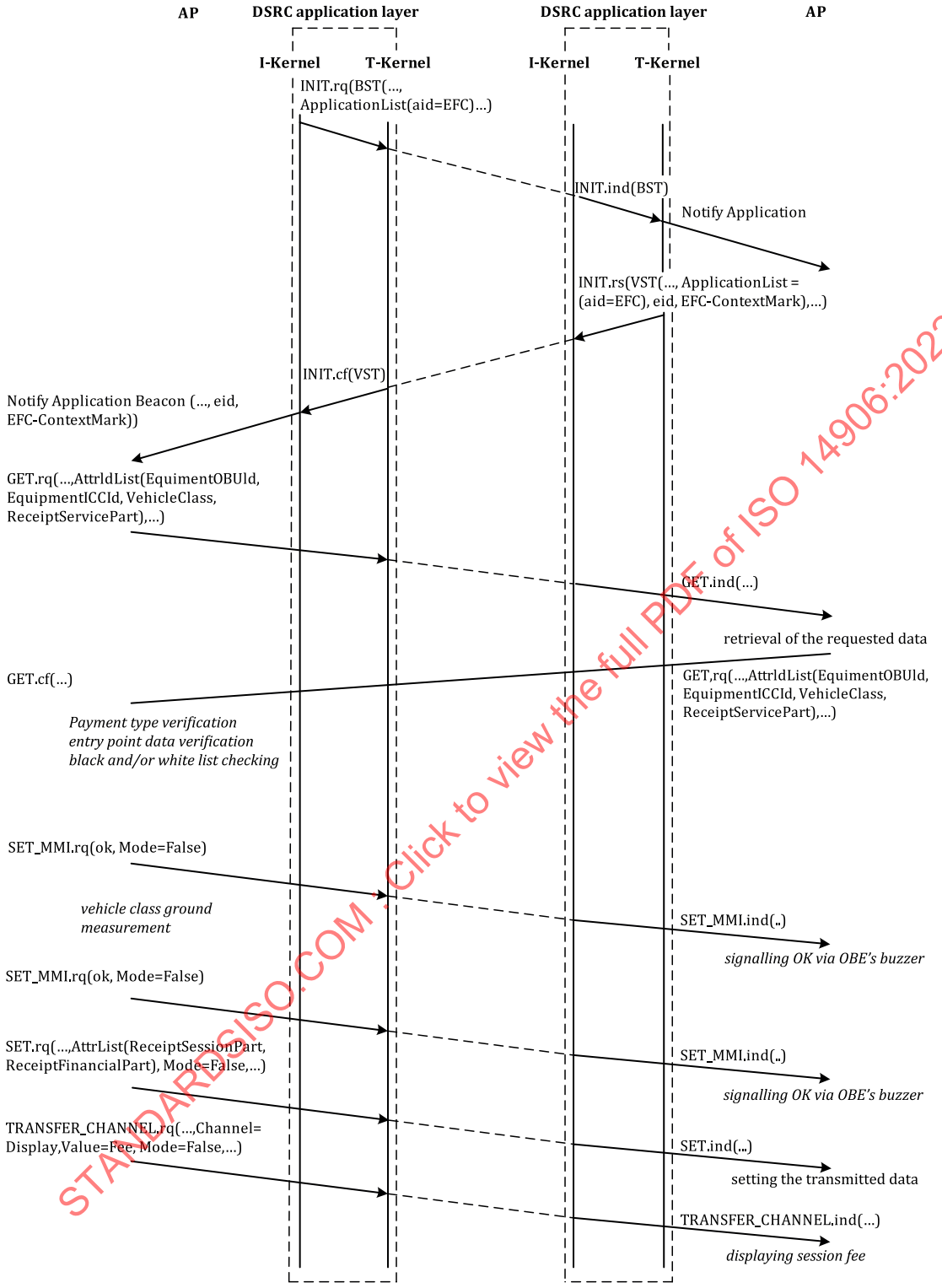


Figure C.2 — Read and write EFC transaction

C.4 EFC purse transaction using the DEBIT function

The example in Figure C.3 demonstrates an EFC purse transaction using the DEBIT function.

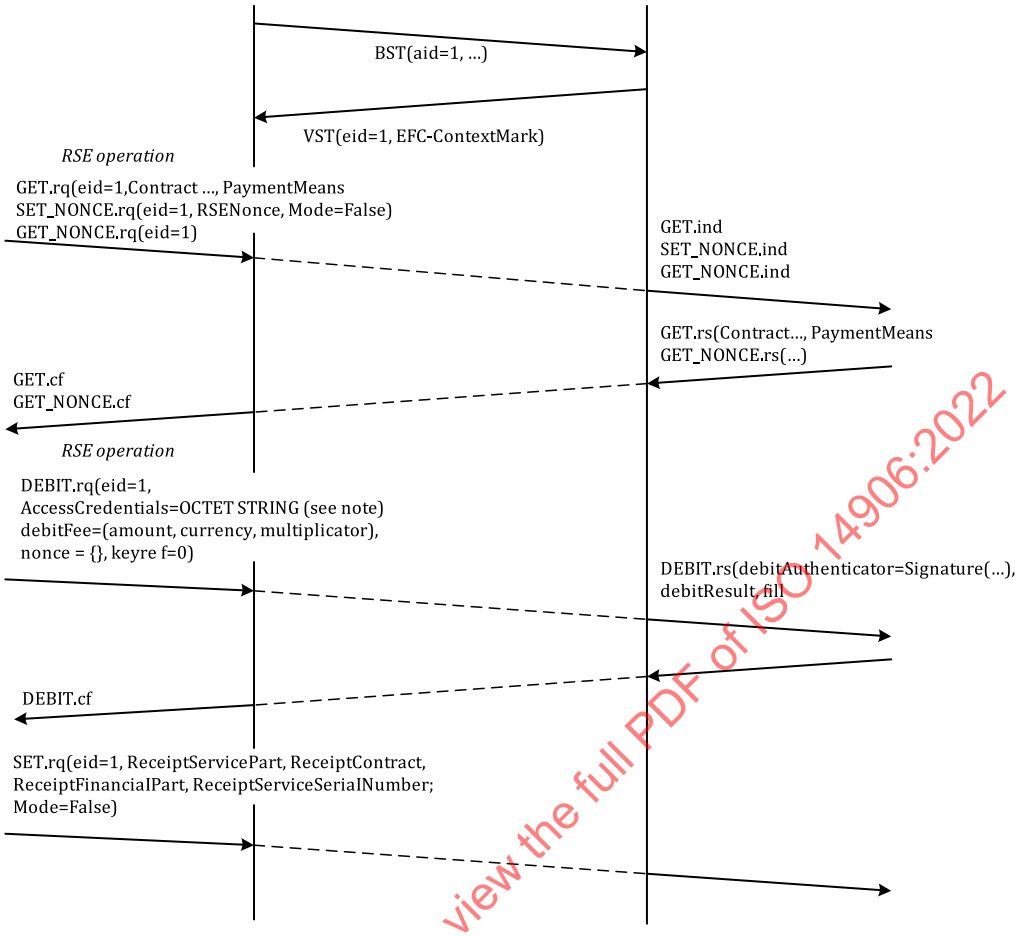


Figure C.3 — EFC purse transaction using the DEBIT function

C.5 On-board account transaction using an ICC and the TRANSFER_CHANNEL function

C.5.1 General

Clause C.5 provides an example of an on-board account transaction using an ICC and the Transfer_Channel function. An overview of the transaction is given in C.5.2. The transaction phases and the data exchanges are described in C.5.3, reflecting typical exchanges used in EFC systems in Japan, Korea, Singapore, Malaysia and China.

This example also illustrates how an ICC can be used as a means to carry the payment method. The user may possess several payment acounts, e.g. a business account and a private account. The user can choose which payment account to use by inserting the corresponding ICC into the OBU, prior to making use of the EFC service.

C.5.2 Overview

C.5.2.1 General

The EFC transaction consists of the five phases as described in Table C.1.

Table C.1 — Phases of on-board account transaction using an ICC

Phase	Short description
Preparation	The vehicle user enters an agreement with a TSP and obtains an OBU and an ICC, which carries the payment method.
Initialisation	The vehicle user needs to insert the ICC into the OBU in order to use the EFC service. The OBU activates the payment method at the time of insertion of the ICC or upon the passage of a tolling station.
Presentation	The RSE reads vehicle data in the OBU and payment data on the ICC (details on contract, account, vehicle classification, last transaction, etc.).
Receipt	The RSE writes an electronic receipt into the OBU and/or the ICC (which may also serve as an entry ticket).
Tracking and closing	The RSE tracks the vehicle through the communication zone and eventually closes the transaction. The OBU de-activates the ICC if necessary.

Irrespective of EFC station type (passage in open system, entry or exit in closed system), the transaction performed is always the same.

C.5.2.2 Preparation phase

The vehicle user enters an agreement with a TSP and obtains an OBU and an ICC, which carries the payment method. The OBU stores vehicle-related data such as vehicle classification and OBU account and contract information at personalization processing.

The ICC stores payment-related data such as tolling account and contract information at personalization processing.

C.5.2.3 Initialisation phase

Table C.2 shows an example of the information exchange in the initialisation phase.

Table C.2 — Initialisation phase — Information exchange

Roadside equipment	OBU and ICC
	Before driving, OBU is ready to read account information on the ICC upon insertion of the ICC into the OBU.
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 "Korean on-board account with the Operator KEC"* 2) *Transaction type "Japanese on-board account with the Operator NEXCO"*

Table C.2 *(continued)*

Roadside equipment		OBU and ICC
The roadside thinking: According to my tables, I have the following transactions available and recognize the accounts with the following operators: Transaction — Operator CN transaction — BeijinExpress TianjinExpress HerbeiExpress JP transaction — NEXCO MEX HEX When I compare my table with the VST, I see that I support the second option offered by the OBE. Hence, I will from now on use "JP/NEXCO"		

C.5.2.4 Presentation phase

An example of the information exchange in the presentation phase is given in Table C.3.

Table C.3 — Presentation phase — Information exchange

Roadside equipment	OBU and ICC
"Please give me the following information about your JP contract with NEXCO: — *your payment means on the ICC, including the personal account number and stored value (with signature)* — *your previous receipts in the OBU and/or on the ICC* — *your vehicle classification details in the OBU"*	
	"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 on the ICC, with signature* — *my previous receipts in the OBU and/or on the ICC (entry ticket)* — *my vehicle classification details in the OBU"*

The RSE uses the retrieved data for the following purposes in addition to the ones explained in B.2.1.3:

— Authentication: RSE verifies the validity of both the OBU and the ICC;

— Signature: Both the OBU and the ICC add several security-related data to the tolling data to confirm the integrity by issuer and/or operator;

C.5.2.5 Receipt phase

The receipt phase is used to write all data into an OBU and/or an ICC that are to be carried to the next tolling station. An example of the information exchange in the receipt phase is given in Table C.4.

Table C.4 — Presentation phase — Information exchange

Roadside Equipment	OBU and ICC
"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 in the OBU and/or on the ICC and I have given the user a signal".

Storage of the entry ticket on the ICC provides the following flexibility for the tolling service:

— in closed tolling systems, the entry information is carried from one tolling station to the next with all combinations of following station types;

— also for other system types, it makes sense to give an electronic receipt to the vehicle user personally.

Station types are as follows.

— Only simple media reader is equipped at tolling station due to low traffic volume.

— Single lane charging at tolling station.

— Free-flow charging at tolling station for high traffic volume.

C.5.2.6 Tracking and closing phase

During the transaction or after the transaction, the OBU performs some technical housekeeping tasks with inserted ICC including flushing and write-back cash and formally closing the transaction.

C.5.3 Transaction using an ICC

Table C.5 provides an overview of the transaction using an ICC 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, an OBU and an ICC. This is a typical examaple of a transaction using a media in the transparent access method as described in ISO 25110, and with the same functionality as with the CARDME transaction described in B.3 when a media is used.

Table C.5 — Overview of on-board account transaction using an ICC

Phase	Roadside equipment		On-board unit	ICC	Remarks
Preparation	not applicable (n.a).		Activate an inserted ICC and retrieve an ICC ID.		Vehicle user inserts an ICC when ride is on and checks the condition.
	INITIALISATION.request (BST)	→			RSE periodically sends BST at toll station.
Initialisation (BST – VST)		←	INITIALISATION.response (VST) — EfcContextMark in OBU — AC_CR-KeyReference — RndOBE — EfcContextMark in ICC — ICC ID		A newly-arrived OBU answers with VST. RSU retrieves an ICC ID and contract data from an ICC via OBU.

Table C.5 *(continued)*

Phase	Roadside equipment		On-board unit	ICC	Remarks
Presentation (GET)	GET_STAMPED.request AC_CR [Access Credential Key] — EquipmentStatus — Classification data: — VehicleClass — VehicleAxles — VehicleLicencePlateNumber — VehicleSpecificCharacteristics TRANSFER_CHANNEL.request (channel ID=ICC, ICC command list in apdu to retrieve following attributes from ICC) — PaymentMeans (including PersonalAccountNumber) — ReceiptData1 — ReceiptData2	→		ICC command – response is performed sequentially.	Authentication processing between the ICC and RSU (or OBU) is performed if necessary before accessing memory area in an ICC; this procedure depends on the ICC specification by issuer. OBU transfers the ICC command to the ICC and buffers the ICC response from the ICC sequentially. OBU set all ICC response in APDU of TRANSFER_CHANNEL.response. This phase may be divided into several DSRC functions.
		←	GET_STAMPED.response — Operator_Authenticator (Auth_Op) TRANSFER_CHANNEL.response		OBE responds with the data including ICC response date.
Tracking	ECHO.request	→			