



BSI Standards Publication

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

National foreword

This British Standard is the UK implementation of EN ISO 14906:2018+A1:2020. It is identical to ISO 14906:2018, incorporating amendment 1:2020. It supersedes BS EN ISO 14906:2018, which is withdrawn.

The start and finish of text introduced or altered by amendment is indicated in the text by tags. Tags indicating changes to ISO text carry the number of the ISO amendment. For example, text altered by ISO amendment 1 is indicated by **A1** **A1**.

The UK participation in its preparation was entrusted to Technical Committee EPL/278, Intelligent transport systems.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2020
Published by BSI Standards Limited 2020

ISBN 978 0 539 04797 4

ICS 35.240.60; 03.220.20

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 December 2018.

Amendments/corrigenda issued since publication

Date	Text affected
31 May 2020	Implementation of ISO amendment 1:2020 with CEN endorsement A1:2020

English Version

Electronic fee collection - Application interface definition
for dedicated short-range communication (ISO
14906:2018)

Perception du télépéage - Définition de l'interface
d'application relative aux communications dédiées à
courte portée (ISO 14906:2018)

Elektronische Gebührenerhebung -
Anwendungsschnittstelle zur dezidierten Nahbereich-
Kommunikation (ISO 14906:2018)

This European Standard was approved by CEN on 6 September 2018.

CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration. Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CEN member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.



EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

European foreword

This document (EN ISO 14906:2018) has been prepared by Technical Committee ISO/TC 204 "Intelligent transport systems" in collaboration with Technical Committee CEN/TC 278 "Intelligent transport systems" the secretariat of which is held by NEN.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by June 2019, and conflicting national standards shall be withdrawn at the latest by June 2019.

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

This document supersedes EN ISO 14906:2011.

This document has been prepared under a mandate given to CEN by the European Commission and the European Free Trade Association, and supports essential requirements of EU Directive(s).

According to the CEN-CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

Endorsement notice

The text of ISO 14906:2018 has been approved by CEN as EN ISO 14906:2018 without any modification.

Foreword to amendment A1

This document (EN ISO 14906:2018/A1:2020) has been prepared by Technical Committee ISO/TC 204 "Intelligent transport systems" in collaboration with Technical Committee CEN/TC 278 "Intelligent transport systems" the secretariat of which is held by NEN.

This Amendment to the European Standard EN ISO 14906:2018 shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by October 2020, and conflicting national standards shall be withdrawn at the latest by October 2020.

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

This document has been prepared under a mandate given to CEN by the European Commission and the European Free Trade Association.

According to the CEN-CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

Endorsement notice

The text of ISO 14906:2018/Amd 1:2020 has been approved by CEN as EN ISO 14906:2018/A1:2020 without any modification.

Contents

Page

Foreword	v
Introduction	vi
Introduction to Amendment A1	viii
1 Scope	1
2 Normative references	2
3 Terms and definitions	2
4 Abbreviated terms	4
5 EFC application interface architecture	5
5.1 Relation to the DSRC communication architecture	5
5.2 Usage of DSRC application layer by the EFC application interface	7
5.3 Addressing of EFC attributes	7
5.3.1 Basic mechanism	7
5.3.2 Role of the EID	8
5.3.3 Multiple Instances of Attributes	8
5.4 Addressing of components	9
6 EFC Transaction Model	10
6.1 General	10
6.2 Initialisation Phase	10
6.2.1 Overview	10
6.2.2 EFC application-specific contents of the BST	11
6.2.3 EFC application-specific contents of the VST	12
6.3 Transaction phase	13
7 EFC functions	14
7.1 Overview and general concepts	14
7.1.1 EFC functions and service primitives	14
7.1.2 Overview of EFC functions	15
7.1.3 Handling of multiple instances	16
7.1.4 Security	18
7.2 EFC functions	21
7.2.1 General	21
7.2.2 GET_STAMPED	21
7.2.3 SET_STAMPED	22
7.2.4 GET_SECURE	23
7.2.5 SET_SECURE	24
7.2.6 GET_INSTANCE	25
7.2.7 SET_INSTANCE	25
7.2.8 GET_NONCE	26
7.2.9 SET_NONCE	27
7.2.10 TRANSFER_CHANNEL	27
7.2.11 COPY	28
7.2.12 SET_MMI	29
7.2.13 SUBTRACT	29
7.2.14 ADD	30
7.2.15 DEBIT	30
7.2.16 CREDIT	31
7.2.17 ECHO	32
8 EFC Attributes	33
8.1 General	33
8.2 Data group CONTRACT	35
8.3 Data group RECEIPT	38
8.4 Data group VEHICLE	44
8.5 Data group EQUIPMENT	51

8.6	Data group DRIVER.....	53
8.7	Data group PAYMENT.....	55
Annex A (normative) EFC data type specifications.....		57
Annex B (informative) CARDME transaction.....		58
Annex C (informative) Examples of EFC transaction types.....		92
Annex D  (informative) Mapping table from LatinAlphabetNo2, LatinCyrillicAlphabet, LatinGreekAlphabet and LatinAlphabetNo10 to LatinAlphabetNo1 .....		104
Annex E (informative) Mapping table between EFC Vehicledata attribute and European registration certificate		105
Annex F (normative) Security calculations for DES.....		108
Annex G (informative) Security computation examples for DES.....		113
Annex H (normative) Security calculations for AES.....		116
Annex I (informative) Security computation examples for AES.....		121
Bibliography		123

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 on 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 the following URL: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*.

This third edition cancels and replaces the second edition (ISO 14906:2011), which has been technically revised. It also incorporates the Corrigendum ISO 14906:2011/Cor1:2013 and the Amendment ISO 14906:2011/Amd1:2015.

The main changes compared to the previous edition are as follows:

- Inclusion of security calculations according to advanced encryption standard, as recommended in CEN/TR 16968 on security mechanisms (revision of [Clause 7](#) and new [Annexes F, G, H and I](#));
- Update of the normative references, terms and definitions and abbreviated terms clauses and the Bibliography;
- Conversion of the ASN.1 module into an electronic insert;
- Revision of [Annex C](#);
- Removal of [Annex D](#) (informative) on functional requirements.

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

Introduction

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 might be used for other DSRC applications (e.g. compliance checking communication) and/or on other interfaces (e.g. the application interface of autonomous systems).

This 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 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 recognising each others' EFC transactions (including the exchange of security algorithms and keys) and implementing the EFC transactions in each others' RSEs, or they can reach an agreement to define a new transaction (and contract) that is common to both. Considerations should also be made by each operator so that the RSE has sufficient resources to implement such additional EFC transactions.

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

- which optional features are actually being implemented and used,
- access rights and ownership of EFC application data in the OBE,
- security policy (including encryption algorithms and key management, if applicable),
- operational issues, such as how many receipts may be stored for privacy reasons, how many receipts are necessary for operational reasons (for example as entry tickets or as proof of payment),
- the agreements needed between operators in order to regulate the handling of different EFC transactions.

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

- EfcModule ASN.1 module, including a version number;
- Efc-ContextMark (incl. the ContextVersion), denoting the implementation version, provides a means to ensure co-existence of different implementation versions by means of a look-up table and associated appropriate transaction processing. This will enable the software of the RSE to determine the version of the OBE and his capability to accept the new features introduced by this edition of ISO 14906.

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

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

[Annex C](#) presents informative examples of EFC transaction types, using the specified EFC functions and attributes.

[Annex D](#) presents an informative mapping table from LatinAlphabetNo2 & 5 to LatinAlphabetNo1 to ease for a Service Provider the use of LatinAlphabetNo1 to encode an OBE for data available written with non-Latin1 characters.

[Annex E](#) presents an informative mapping table between EFC vehicle data attributes and European registration certificates to ease the task of a service provider in the OBE personalisation with vehicle data.

[Annex F](#) presents the security calculations according to the data encryption standard (DES). This annex is based on EN 15509:2014, Annex B.

[Annex G](#) presents the security computations examples for DES. This annex is based on EN 15509:2014, Annex E.

[Annex H](#) presents the security calculations for advanced encryption standard (AES). This annex is the adaptation of EN 15509:2014, Annex B for the case of AES.

[Annex I](#) presents the security computations examples for AES. This annex is the adaptation of EN 15509:2014, Annex E for the case of AES.

This application interface definition can also be used with other DSRC media which do not use a layer 7 according to ISO 15628/EN 12834. Any DSRC medium which provides services to read and write data, to initialise communication and to perform actions is suitable to be used as a basis for this application interface. Adaptations are medium specific and are not further covered here. As [Annex B](#) describes in detail a transaction for central account systems, this 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.

Introduction to Amendment 1

NOTE The syntax of the ASN.1 code in the attached file ISO14906(2020)EfcDsrcApplicationv8.asn or www.itsstandards.eu/index.php/efc#EFCstandards and also at <https://standards.iso.org/iso/14906/ed-3/amd/1/en> takes precedence to the ASN.1 code included in the main body of this document.

Replace the following text (i.e. lines 248 to 260) in ISO14906(2018)EfcDsrcApplicationv6.asn:

```
EngineCharacteristics ::= INTEGER {  
  noEntry (0),  
  noEngine (1),  
  petrolUnleaded (2),  
  petrolLeaded (3),  
  diesel (4),  
  LPG (5),  
  battery (6),  
  solar (7),  
  hybrid (8),  
  hydrogen (9)  
  -- (10-255) are reserved for future CEN use  
} (0..255)
```

with the following text (i.e. lines 248 to 304) in ISO14906(2020)EfcDsrcApplicationv8.asn:

```
EngineCharacteristics ::= INTEGER {  
  noEntry (0),  
  noEngine (1),  
  petrolUnleaded (2),  
  petrolLeaded (3),  
  diesel (4),  
  LPG (5),  
  battery (6), -- vehicle powered exclusively by battery  
  solar (7),  
  hybrid (8), -- kept for legacy compatibility, more differentiated  
  values are available  
  hydrogen (9),  
  multi-fuel (10), -- multi fuel engine  
  bivalent-petrol-lpg (11), -- bivalent operating engine with petrol  
  or liquefied petroleum gas  
  bivalent-petrol-cng (12), -- bivalent operating engine with petrol  
  or compressed natural gas  
  combined-petrol-electric (13), -- combined operation with petrol and  
  electric engine  
  cng (14), -- compressed natural gas  
  lng (15), -- liquefied natural gas  
  combined-diesel-electric (16), -- combined operation of diesel and  
  electric engine  
  combined-hydrogen-electric (17), -- combined operation of hydrogen  
  and electric engine  
  bivalent-hydrogen-petrol (18), -- bivalent operating engine with  
  hydrogen or petrol  
  bivalent-hydrogen-petrol-electric-engine (19), -- bivalent operating  
  engine with hydrogen or petrol combined with electric engine  
  fuel-cell-hydrogen (20), -- fuel cell with hydrogen as primary energy  
  source and electric engine  
  fuel-cell-petrol (21), -- fuel cell with petrol as primary energy  
  source and electric engine  
  fuel-cell-methanol (22), -- fuel cell with methanol as primary energy  
  source and electric engine
```

fuel-cell-ethanol (23), -- fuel cell with ethanol as primary energy source and electric engine
 fuel-cell-diesel (24), -- fuel cell with diesel as primary energy source and electric engine
 combined-multi-fuel-electric-engine (25), -- combined operation of multi fuel and electric engine
 combined-cng-electric-engine (26), -- combined operation with compressed natural gas and electric engine
 combined-lng-electric-engine (27), -- combined operation with liquefied natural gas and electric engine
 petrol-ethanol (28), -- fuel mix of petrol and mainly ethanol, e.g. E85
 combined-lpg-electric-engine (29), -- combined operation of LPG and electric engine
 hybrid-petrol-external-battery (30), -- hybrid drive with petrol and external rechargeable battery (plug-in hybrid)
 hybrid-diesel-external-battery (31), -- hybrid drive with diesel and external rechargeable battery (plug-in hybrid)
 hybrid-lpg-external-battery (32), -- hybrid drive with LPG and external rechargeable battery (plug-in hybrid)
 hybrid-hydrogen-external-battery (33), -- hybrid drive with hydrogen and external rechargeable battery (plug-in hybrid)
 hybrid-multi-fuel-external-battery (34), -- hybrid drive with multi fuel and external rechargeable battery (plug-in hybrid)
 hybrid-cng-external-battery (35), -- hybrid drive with compressed natural gas and external rechargeable battery (plug-in hybrid)
 hybrid-lng-external-battery (36), -- hybrid drive with liquefied natural gas and external rechargeable battery (plug-in hybrid)
 hybrid-bivalent-hydrogen-petrol-external-battery (37), -- hybrid drive with bivalent operating hydrogen and petrol engine and external rechargeable battery (plug-in hybrid)
 hydrogen-cng (38), -- fuel mix of hydrogen and compressed natural gas
 hydrogen-lng (39), -- fuel mix of hydrogen and liquefied natural gas
 hybrid-hydrogen-cng-external-battery (40), -- hybrid drive with hydrogen and compressed natural gas and external chargeable battery (plug-in hybrid)
 hybrid-hydrogen-lng-external-battery (41), -- hybrid drive with hydrogen and liquefied natural gas and external chargeable battery (plug-in hybrid)
 ethanol (42), -- ethanol or fuel mix of ethanol and other fuel (except petrol) or additives, e.g. E95
 hybrid-fuel-cell-hydrogen (43), -- hybrid drive with fuel cell (electric engine) and hydrogen (combustion engine)
 hybrid-fuel-cell-hydrogen-external-battery (44), -- hybrid drive with fuel cell (electric engine) and hydrogen (combustion engine) and external chargeable battery (plug-in hybrid)
 dual-fuel-lng-diesel (45), -- dual operation with LNG and diesel
 electric-external (46), -- electric engine with external power supply
 biogas (47), -- mixture of different gases produced by the breakdown of organic matter
 bioDiesel (48), -- vegetable oil- or animal fat-based diesel fuel
 bioPetrol (49), -- petrol fully or partly based on vegetable sources
 bivalent-petrol-biogas (50), -- bivalent operating engine with petrol or biogas
 combined-biogas-electric-engine (51), -- combined operation of biogas and electric engine
 dual-fuel-cng-diesel (52), -- dual operation with CNG and diesel
 -- (53-254) are reserved for future CEN and ISO use
 other (255)
 } (0..255)

A1

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 the dedicated short-range communication (DSRC).

The EFC application interface is the EFC application process interface to the DSRC application layer, as can be seen in [Figure 1](#) below. This 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. ICC and MMI),
- EFC application functions, i.e. further qualification of actions by definitions of the concerned services, assignment of associated ActionType values and content and meaning of action parameters,
- the EFC transaction model, which defines the common elements and steps of any EFC transaction,
- the behaviour of the interface so as to ensure interoperability on an EFC-DSRC application interface level.

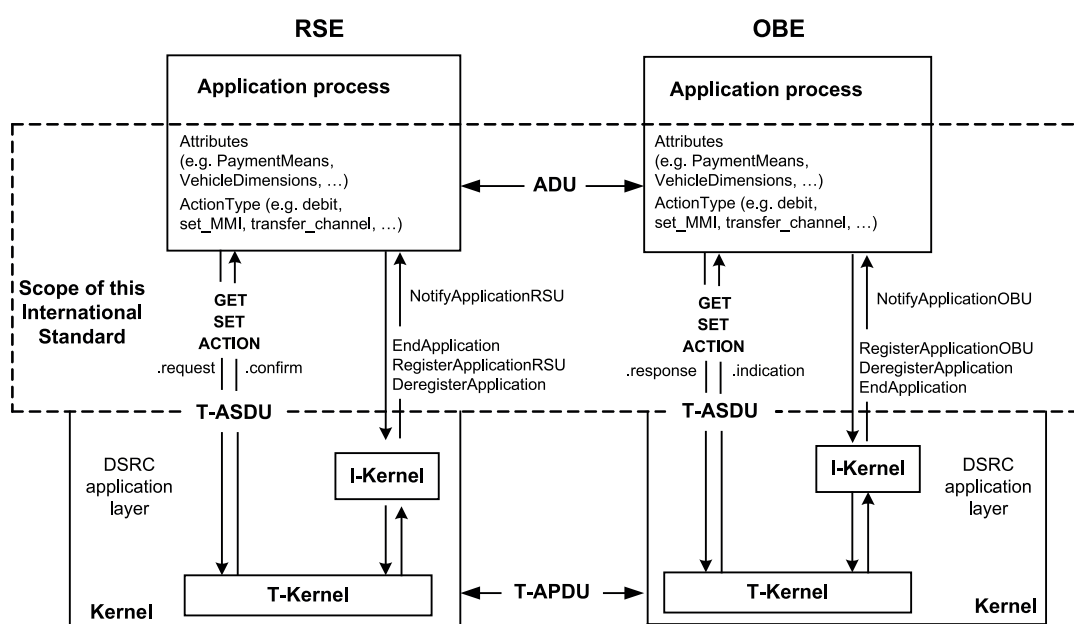


Figure 1 — The EFC application interface

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

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: Specification of Packed Encoding Rules (PER) — Part 2*

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

ISO 14816:2005, *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/IEC 18033-3:2010, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*

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

3 Terms and definitions

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

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

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

3.1 access credentials

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

3.2 attribute

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

[SOURCE: ISO 17575-1:2016, definition 3.2]

3.3 authenticator

data, possibly encrypted, that is used for authentication

3.4**channel**

information transfer path

[SOURCE: ISO 7498-2:1989, definition 3.3.13]

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

3.6**data group**

class of closely related attributes

[SOURCE: ISO 17575-1:2016, definition 3.10]

3.7**data integrity**

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

3.8**Element**

DSRC directory containing application information in the form of attributes

3.9**on-board equipment**

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

3.10**on-board unit**

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

3.11**roadside equipment**

equipment located along the road, either fixed or mobile

3.12**toll charger**

entity which levies toll for the use of vehicles in a toll domain

3.13**toll domain**

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

[SOURCE: ISO 17573:2010, definition 3.18]

3.14**toll service**

service enabling users to pay toll

3.15**toll service provider**

entity providing toll services in one or more toll domains

3.16**transaction**

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

[SOURCE: ISO 17575-1:2016, definition 3.21]

3.17
transaction model

functional model describing the structure of electronic payment transactions

4 Abbreviated terms

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

AP	Application Process
APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation One (ISO/IEC 8824-1)
BST	Beacon Service Table
CCC	Compliance check communication
cf	Confirm
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
PDU	Protocol Data Unit
PER	Packed Encoding Rules (ISO/IEC 8825-2)

req	Request
rs	Response
RSE	Roadside Equipment
SAM	Secure Application Module
T-APDU	Transfer-Application Protocol Data Unit
T-ASDU	Transfer-Application Service Data Unit
T-Kernel	Transfer Kernel
VST	Vehicle Service Table

5 EFC application interface architecture

5.1 Relation to the DSRC communication architecture

The DSRC services are provided to an application process by means of the DSRC Application Layer service primitives, which are abstract implementation interactions between a communication service user and a provider. The services are offered by the DSRC communication entities by means of its DSRC Application Layer (EN 12834/ISO 15628) as shown in [Figure 2](#).

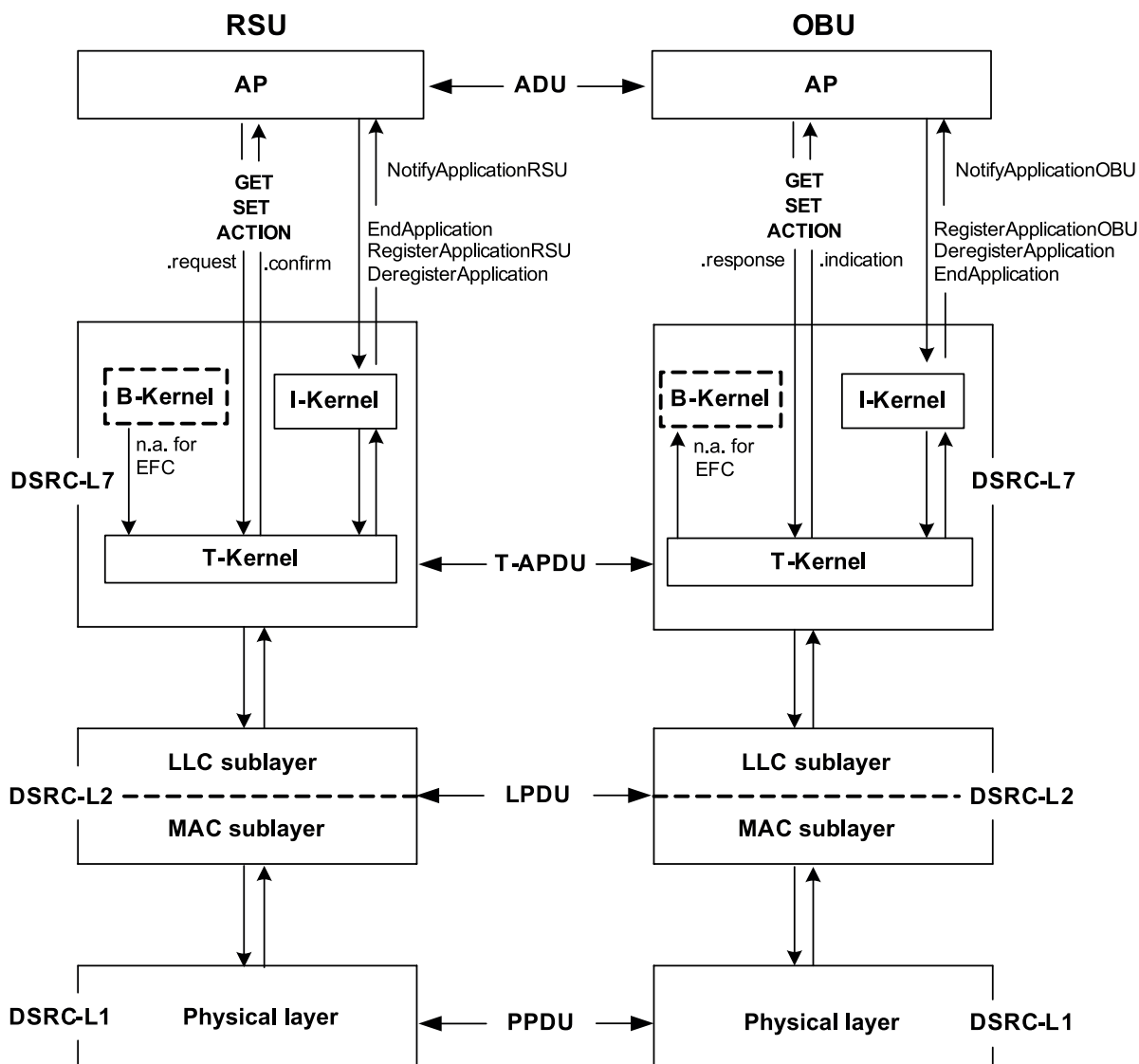


Figure 2 — The EFC application process on top of the DSRC communication stack

NOTE The abbreviated terms used in [Figure 2](#) are defined in [Clause 4](#).

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

- GET: The invocation of a GET service request results in retrieval (i.e. reading) of application information (i.e. Attributes) from the peer service user (i.e. the OBE application process), a reply is always expected.
- SET: The invocation of a SET service request results in modification (i.e. writing) of application information (i.e. Attributes) of the peer service user (i.e. the OBE application process). This service may be requested in confirmed or non-confirmed mode, a reply is only expected in the former case.
- ACTION: The invocation of an ACTION service request results in a performance of an action by the peer service user (i.e. the OBE application process). An action is further qualified by the value of the ActionType. This service may be requested in confirmed or non-confirmed mode, a reply is only expected in the former case.
- EVENT-REPORT: The invocation of an EVENT-REPORT service request forwards a notification of an event to the peer service user.

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

5.2 Usage of DSRC application layer by the EFC application interface

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

- The INITIALISATION services:
 - Notify Application RSU (at RSE);
 - End Application (at RSE);
 - Register Application RSU (at RSE);
 - Deregister Application (at RSE and OBE);
 - Notify Application OBU (at OBE);
 - Register Application OBU (at OBE)
 are used to realise the EFC-specific initialisation mechanism (see [Clause 6](#));
- The GET service is used to retrieve EFC attributes (For attribute specifications see [Clause 8](#));
- The SET service is used to set EFC attributes;
- The ACTION services are applied to realise additional EFC specific functionality needed to support EFC application processes, such as TRANSFER_CHANNEL, SET_MMI and ECHO (see [7.2](#)).

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

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

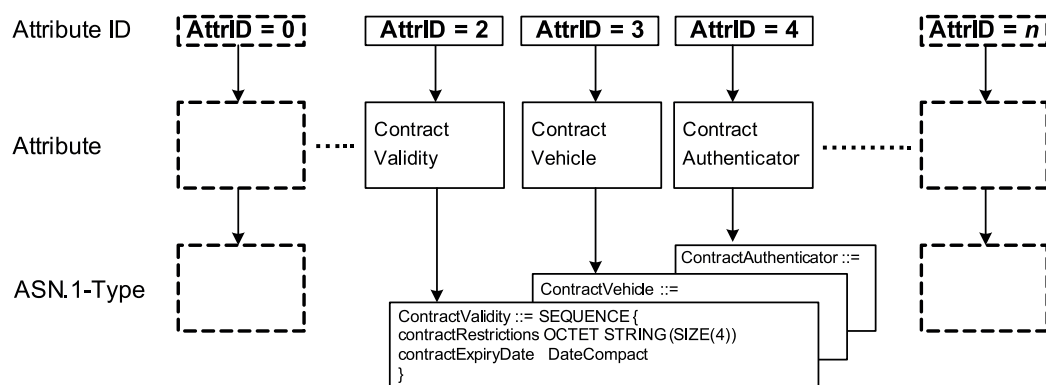


Figure 3 — Basic addressing mechanism

5.3.2 Role of the EID

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

EXAMPLE [Figure 4](#) illustrates the role of the EID.

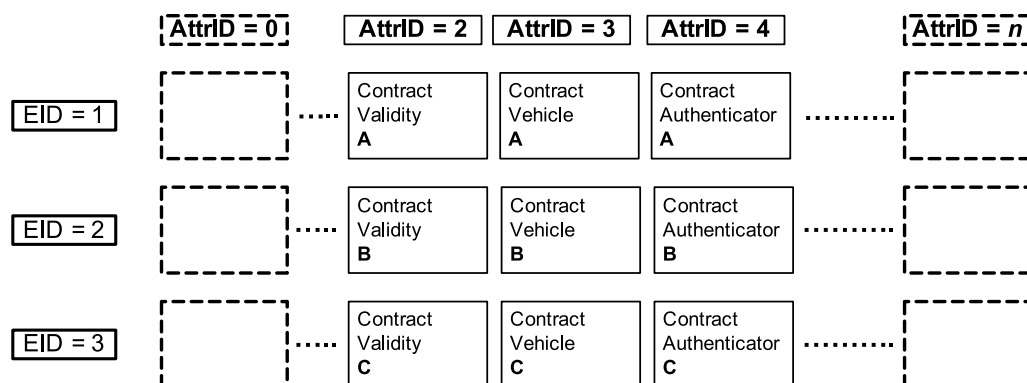


Figure 4 — Role of the EID

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.

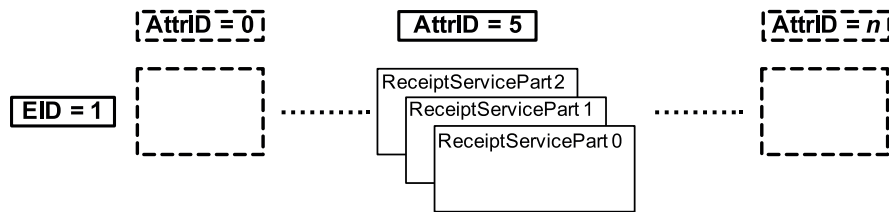


Figure 5 — Multiple instances (0-2) of attribute 5

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;
- SAM 1;
- SAM 2;
- ICC;
- Display;
- Buzzer;
- Printer;
- Serial interface;
- Parallel interface;
- 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 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 MMI function (e.g. a SET command on the attribute ReceiptText might display the text on a liquid crystal display. A SET command on the attribute ReceiptServicePart with data element SessionResultOperational other than SessionOK might activate an alert beep). Proprietary addressing mechanisms are not defined by this 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 OBEs that have entered the DSRC zone but have not yet established communication with the RSE, and to notify the application processes. It provides amongst others a multi-application switching mechanism, allowing for execution of several 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 (in [6.2](#)) may also be used in an EFC transaction phase.

6.2 Initialisation Phase

6.2.1 Overview

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

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

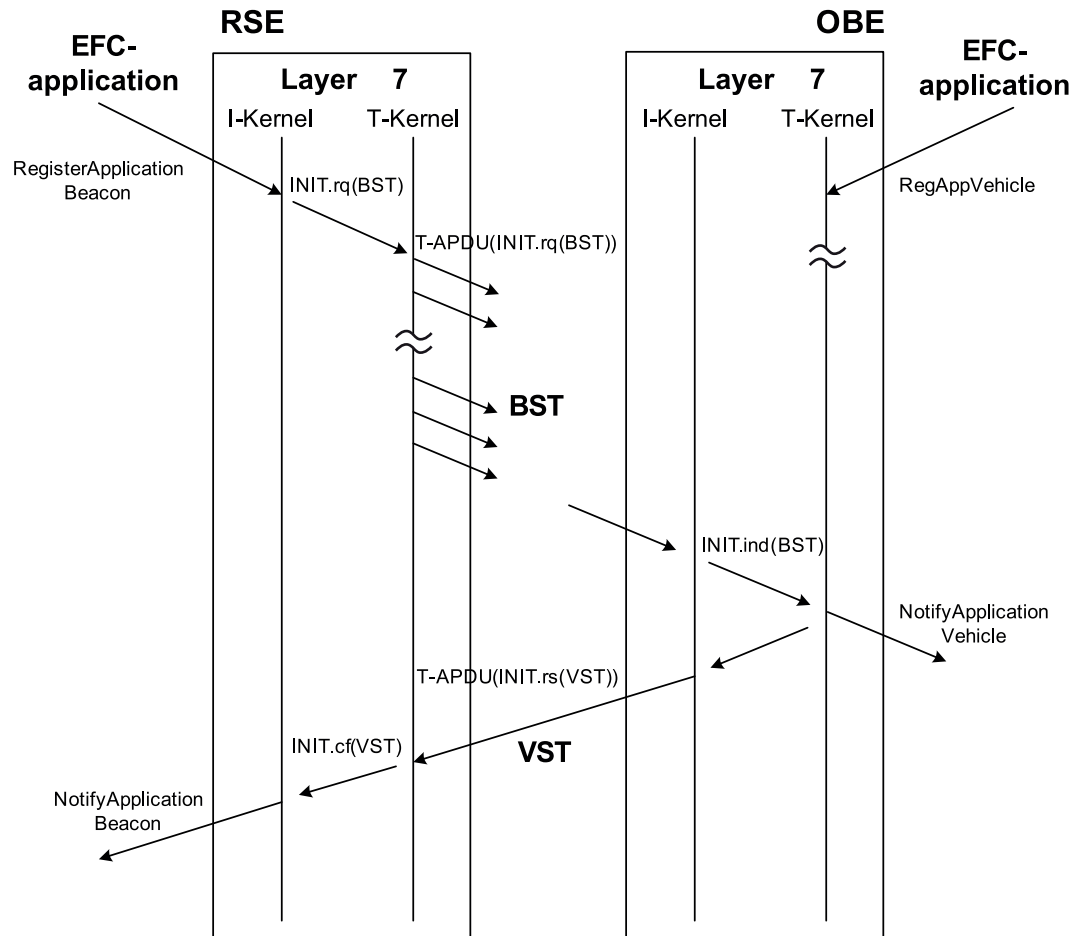


Figure 6 — Initialisation phase: BST - VST exchanges

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

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

NOTE 2 The above is the EFC application-specific contents of the BST. The complete BST is defined in EN 12834/ISO 15628 and is given below for readability of this 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,             -- empty
    parameter          ApplicationContextMark OPTIONAL -- empty
}
```

6.2.3 EFC application-specific contents of the VST

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

NOTE 1 It is outside the scope of this document to define in which order EFC-ContextMarks shall be presented in order to indicate a user's order of preference, in case his OBE supports several EFC applications. Such rules for indicating the user's order of preference 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 EFC-ContextMark, contained in the Parameter, and shall be unique within the OBE throughout the complete DSRC session;
- the Parameter shall be of Container CHOICE type OCTET STRING and shall comprise the EFC-ContextMark as defined below, and may also be configured to carry additional EFC attributes (as defined in [Clause 8](#) and [Annex A](#)).

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

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

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

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

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

where

```

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

```

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

- EfcModule ASN.1 module, including a version number
- Efc-ContextMark (incl. the ContextVersion), denoting the implementation version, provides a means to ensure co-existence of different implementation versions by means of a look-up table and associated appropriate transaction processing.

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

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

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

NOTE 6 Retrofit Dsrc OBE are mostly battery powered, and the battery usually has a lifetime that is dependent on the actual usage of the OBE (number of transactions per day, activation of MMI, etc.). In an interoperable environment, the Toll Charger(s) can access 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 EFC-ContextMark and associated EID. The RSE shall use the functions defined in [Clause 7](#) to complete the EFC transaction.

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

EXAMPLE A transaction may consist of the following steps:

- GET(EID, ContractValidity, ContractVehicle, ReceiptServicePart, PaymentMeansBalance)
- DEBIT(EID, DebitPaymentFee)
- SET(EID, ReceiptServicePart)

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

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

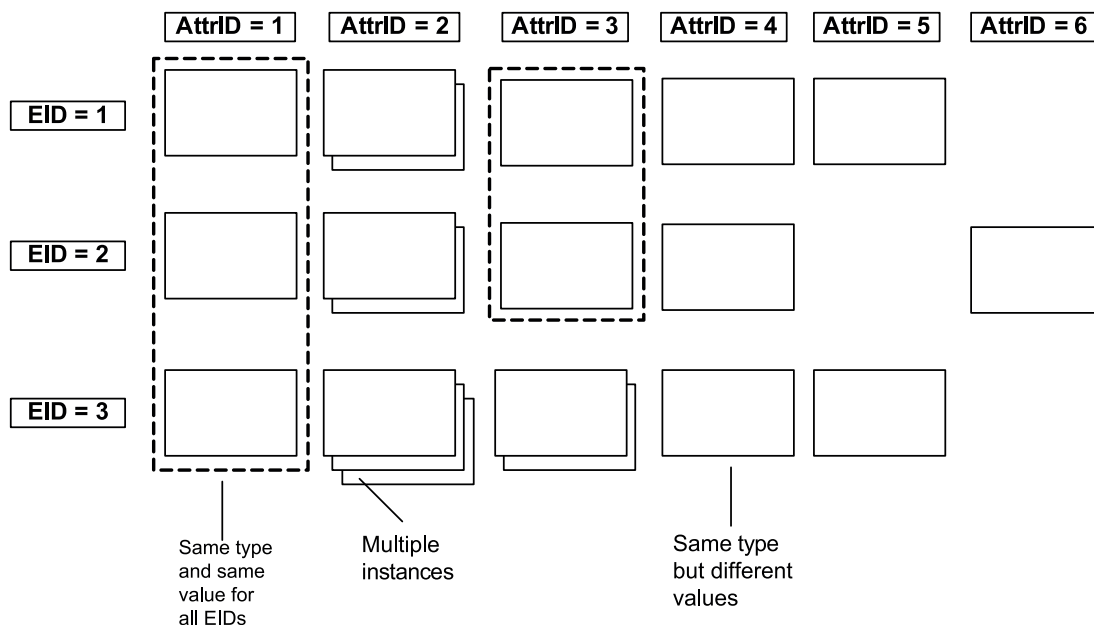


Figure 7 — Context of attributes given by EFC-ContextMark and identified by the EID

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

- Booking from two contracts in one transaction. There is sometimes the need to book from two contracts in one session, e.g. when a customer has a contract with a reduced price (e.g. a commuter contract) for part of the route being tolled, plus an ordinary (not reduced) contract for the rest of the route. This 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 “his” EFC-Context Mark.
- Agreement to share some data groups, but not others, e.g. the data groups Vehicle, Receipt and Payment are shared, but not Contract. Different security measures (algorithms) are used by the two operators (or more or, all data groups are shared, except for Payment - each operator books from an account issued by himself.
- Agreement between operators (Toll Chargers and ETS Provider): each Toll Charger has the possibility to select only the attributes needed for a given toll context (principle of “Pick what you need”) amongst common Data Group.

7 EFC functions

7.1 Overview and general concepts

7.1.1 EFC functions and service primitives

This clause describes the EFC functions invoked by T-ASDUs exchanged between peer applications communicating via a DSRC link. The T-ASDUs are exchanged by means of service primitives of the DSRC

Application Layer (EN 12834/ISO 15628). Exchanges of service primitives (and the corresponding T-ASDUs) associated with EFC functions adhere to the following basic pattern:

- xxx.rq (request) service primitive invoked by the RSE application to DSRC Application Layer;
- xxx.ind (indication) service primitive issued by the DSRC Application Layer to the OBE application;
- xxx.rs (response) service primitive invoked by the OBE application to DSRC Application Layer;
- xxx.cf (confirmation) service primitive issued by the DSRC Application Layer to RSE application.

The last two steps are either mandatory or optional, depending on the nature of the service primitive and on the setting of the Mode parameter (see 6.2.4 in EN 12834/ISO 15628, 6.2.4:2013).

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

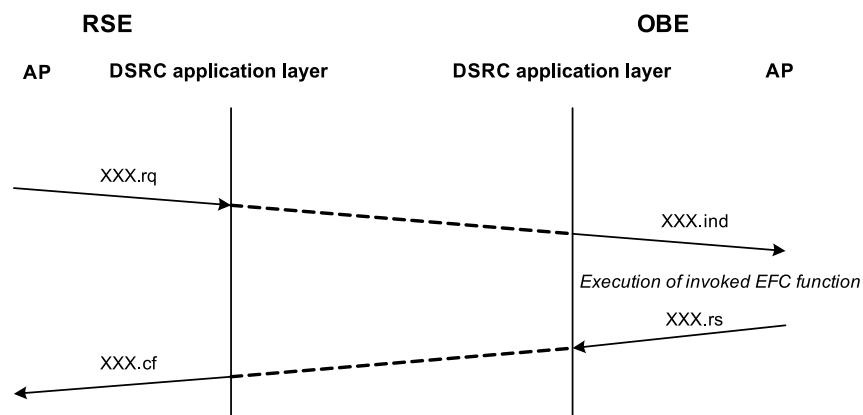


Figure 8 — Logical sequencing of service primitive exchanges

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, 6.2.2:2013 (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
GET_SECURE	2	OCTET STRING	OCTET STRING	gets data securely from the OBE
SET_SECURE	3	OCTET STRING	OCTET STRING	sets data securely in the OBE
GET_INSTANCE	4	GetInstanceRq	GetInstanceRs	retrieves a number of entries out of an attribute's multiple instances
SET_INSTANCE	5	SetInstanceRq	n.a.	sets one entry at a specified position in an attribute's multiple instances
GET_NONCE	6	n.a.	OCTET STRING	retrieves a nonce - typically used against replay attacks
SET_NONCE	7	OCTET STRING	n.a.	sets a nonce - typically used against replay attacks
TRANSFER_CHANNEL	8	ChannelRq	ChannelRs	sets and/or retrieves data from the addressed OBE component (e.g. an ICC)
COPY	9	CopyRq	n.a.	copies data from a source EID to a destination EID
SET_MMI	10	SetMMIRq	n.a.	invokes an MMI function (e.g. signal Ok via buzzer)
SUBTRACT	11	SubRq	n.a.	subtracts the given value to the addressed value
ADD	12	AddRq	n.a.	adds the given value to the addressed value
DEBIT	13	DebitRq	DebitRs	debits purse
CREDIT	14	CreditRq	CreditRs	credits purse
ECHO	15	OCTET STRING	OCTET STRING	OBE echoes received data

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

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

7.1.3 Handling of multiple instances

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

- GET, GET_STAMPED: These functions shall always access the last instance (i.e. instance at position 0). If no instance is available, the result is undefined but may lead to the return of an error code.
- GET_INSTANCE: This function carries parameters $N1$ and $N2$, both ≥ 0 . It shall return the following:
 - if $N2 < N1$, or $N1 > n$, then the empty list;
 - if $N2 \geq N1$ then the values of the instances numbered $N1$, $N1+1$, ... up to and including $\min(N2, n)$.

The case that zero instances are 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			Result
		0	1	2	
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			Result
		0	1	2	
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

Table 3 (continued)

Function	N	Buffer content Instance position			Result
		0	1	2	
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 recognised 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 DES and 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 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-authorised access to sensitive user data and against (commercial) use of the OBU by non-authorised RSE.

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

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

7.1.4.3 Principle of Access Credentials

The principle of access control 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 RndOBU number as input. The RSE returns the access credential calculated and the OBU compares the access credential with its own calculation. In case they are equal, the OBU accepts the RSE as a genuine RSE and reading data from the OBU is allowed.

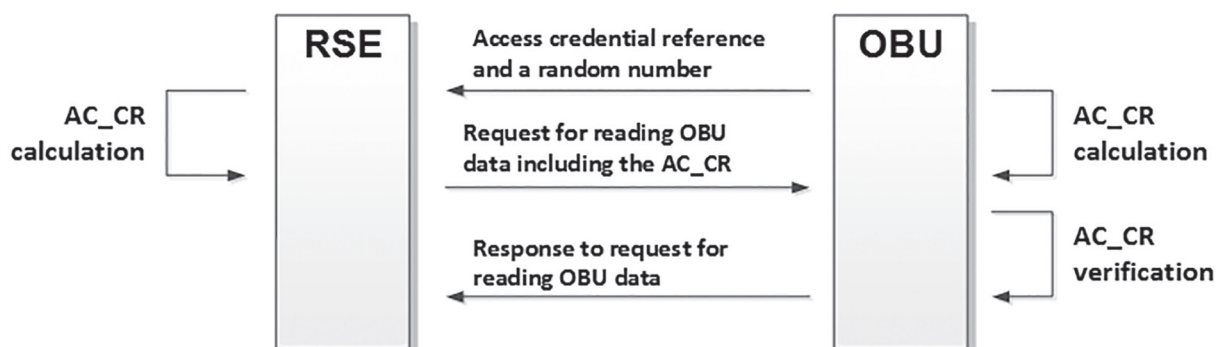


Figure 9 — Principle of access control to the OBU data

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

```

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

The size of the RndOBU 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 (in octets)	Remarks
AccessKey	8	Private.
AC_CR	4	Access credentials calculated by the RSE and the OBU using RndOBU and the Access Key AC_CRKey.
AC_CR-KeyReference	2	Reference to the key generation and the Diversifier for the computation of AC_CRKey.
RndOBU	4	Random number (nonce) used together with AccessKey (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 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 (in octets)	Remarks
AccessKey	16	Private.
AC_CR	8	Access credentials calculated by the RSE and the OBU using RndOBU and the Access Key AC_CRKey.
AC_CR-KeyReference	2	Reference to the key generation and the Diversifier for the computation of AC_CRKey.
RndOBU	8	Random number (nonce) used together with 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 (in 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 and Toll Charger Authenticators. The Toll Service Provider decides which keys are shared with (MoU) Toll chargers (referenced through AuKey_Op), and which are only known by himself (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. 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 [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 (in octets)	Remarks
AuthenticationKey1	16	Private
AuthenticationKey2	16	Private
AuthenticationKey3	16	Private
AuthenticationKey4	16	Private
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. Toll Service Provider and Toll Charger Authenticators. The Toll Service Provider decides which keys are shared with (MoU) Toll chargers (referenced through AuKey_Op), and which are only known by himself (referenced through AuKey_Iss).
RndRSE	8	Random number from RSE used for the computation of Authenticator.
RndOBU2	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 Clause, the EFC functions are specified in detail. The format and the parameters of the EFC functions shall adhere to the Action service primitives of the DSRC application layer as defined in [6.2.2](#) (Transfer Kernel Services) in EN 12834/ISO 15628. Not all parameters associated with the EFC functions are accounted for in this clause, as they are either not specifically needed for the EFC applications or have the same meaning for all functions.

The Return Codes (RET) are explicitly specified whenever additional precision is needed on top of the specifications given in EN 12834/ISO 15628, 6.2.4:2013.

The ASN.1 type specifications of the ActionParameters and ResponseParameters are provided in the normative [Annex A](#).

7.2.2 GET_STAMPED

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

Table 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
ActionParameter	GetStampedRq ::= SEQUENCE { attributeIdList AttributeIdList, nonce OCTET STRING, keyRef INTEGER(0..255) }		always to be present
Mode	BOOLEAN	TRUE	

GET_STAMPED.request shall request the retrieval of the value(s) of the attributes addressed by the attributeIdList, with an authenticator given in the response. A response shall always be expected (Mode = TRUE). The parameter keyRef shall contain a reference to the key to be used for the calculation of the authenticator in the response. See [Table 8](#).

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

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

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](#).

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

7.2.3 SET_STAMPED

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

Table 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 INTEGER(0..255) }		always to be present
Mode	BOOLEAN	TRUE	

SET_STAMPED.request shall request the setting of the value(s) of the attributes addressed by the attributeList, with an authenticator given in the response. A response shall always be expected (Mode = TRUE). See [Table 10](#).

Table 11 — SET_STAMPED.response

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

SET_STAMPED.response shall carry an authenticator as the response parameter (being of ASN.1 type OCTET STRING) to the corresponding request to convey that the data in the attribute list of the request have been set. The authenticator shall be calculated over the values given in the request attributeList, with the keyRef parameter of the request being used as a reference to the (cryptographic) key to be used. When a nonce of non-zero length is given in the request, the nonce value shall be included in the cryptographic transformation. See [Table 11](#).

7.2.4 GET_SECURE

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

Table 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	

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](#).

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, e.g. also contain an authenticator for non-repudiation purposes.

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
Return Code (Ret)	ResponseStatus		optional use

GET_SECURE.response shall carry as the responseParameter (being of ASN.1 type OCTET STRING) to the corresponding request the requested value(s) of the addressed attribute(s) in the form (e.g. encrypted) implicitly defined in the context set in the VST, amongst others by explicit reference given in the action parameter. See [Table 13](#).

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, e.g., an authenticator for non-repudiation purposes.

7.2.5 SET_SECURE

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

Table 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		

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](#).

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

Table 15 — SET_SECURE.response

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

SET_SECURE.response shall, if used in the confirmed mode, carry as the ResponseParameter (being of ASN.1 type OCTET STRING) the confirmation of the corresponding request. The confirmation shall be in the form implicitly defined in the context set in the VST, amongst others by explicit reference given in the action parameter. See [Table 15](#).

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, e.g. be empty or contain an authenticator to be used for non-repudiation of receipt.

7.2.6 GET_INSTANCE

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

Table 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 INTEGER(0..255), posOfLastInstance INTEGER(0..255), attributeIdList AttributeIdList}		always to be present
Mode	BOOLEAN	TRUE	

GET_INSTANCE.request shall request the retrieval of a number of instances of the value(s) of the addressed attribute(s). The ActionParameter contains the position of the first instance and the last instance of the instances of the specified attribute(s) to be retrieved. *GET_INSTANCE.request* can only be used in confirmed mode; a reply shall always be expected. See Table 16.

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

GET_INSTANCE.response shall, as response to the corresponding request, contain all available values of the requested attributes, starting from the value at the first position (posOfFirstInstance) up to the value at the last position (posOfLastInstance), as asked for in the request. The ResponseParameter shall for each requested attribute in turn contain first the attribute Id of the requested attribute, followed by the values of the attribute. The value(s) of an attribute at the first position shall be transferred first in the parameter attributeValues. See Table 17.

7.2.7 SET_INSTANCE

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

Table 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 INTEGER(0..255), attribute Attr }		always to be present
Mode	BOOLEAN		

SET_INSTANCE.request shall request the replacement of a selected instance of the addressed attribute. The ActionParameter contains the value (posOfInstance) attempted to be replaced and the attribute Id. *SET_INSTANCE.request* can be used in confirmed or non-confirmed mode (i.e. Mode equal to TRUE or FALSE, respectively); a reply shall always be expected in the former case. See [Table 18](#).

Table 19 — SET_INSTANCE.response

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

SET_INSTANCE.response shall explicitly convey the result of the corresponding *SET_INSTANCE.request*. If the addressed value could not be replaced, the Return Code shall indicate a failure. See [Table 19](#).

7.2.8 GET_NONCE

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

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

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

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.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 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

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](#).

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

7.2.9 SET_NONCE

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

Table 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.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 23 — SET_NONCE.response

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

SET_NONCE.response shall be issued as a response to the corresponding request to convey the result of the request. A receiving peer entity supporting no nonce shall return a Return Code indicating Complexity Limitation. See [Table 23](#).

7.2.10 TRANSFER_CHANNEL

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

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
ActionParameter	ChannelRq ::= SEQUENCE { channelId ChannelId, apdu OCTET STRING }		
Mode	BOOLEAN		

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

is to be conveyed as part of the parameter apdu in a format appropriate for the addressed channel. See [Table 24](#).

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

Table 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

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](#).

7.2.11 COPY

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

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

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

Table 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.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 27 — COPY.response

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

COPY.response shall be used to explicitly convey the result of the corresponding COPY.request command. See [Table 27](#).

7.2.12 SET_MMI

SET_MMI is used to perform device-independent MMI functions.

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.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 29 — SET_MMI.response

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

SET_MMI.response shall be used to explicitly convey the result of the corresponding SET_MMI.request command. If Mode was set to TRUE in the corresponding request and the operation was successfully executed then the response shall indicate no error. If an error occurred at an attempt to execute the command then the Return Code shall take the appropriate value. See [Table 29](#).

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

7.2.13 SUBTRACT

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

Table 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		