INTERNATIONAL STANDARD

ISO 15628

First edition
2007-02-01

Road transport and traffic telematics — Dedicated short range communication (DSRC) — DSRC application layer

Télématique du transport routier et de la circulation (TICS) — Communication de courte portée dédiée (DSRC) — Couche d'application DSRC

Reference number
ISO 15628:2007(E)

Contents

Page

Foreword

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

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

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

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

ISO 15628 was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*.

Introduction

The communication requirements of many ITS applications can be fulfilled by DSRC. The DSRC International Standards enable compliant communication systems to serve multiple ITS applications in parallel.

The small service areas and severe real-time constraints require a specific protocol architecture leading to the reduced protocol stack shown in Figure A, built up by the "application layer", the "data link layer" and the "physical layer". Such architecture is very common for real-time environments.

This International Standard gives the architecture and services offered by the DSRC application layer.

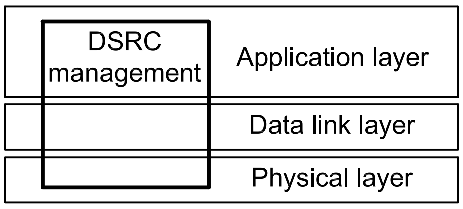


Figure 1 — DSRC protocol stack

This International Standard contains, besides the normative main body, three normative annexes: "Data structures", "Naming and registration", "Declaration of application layer features supported"; plus two informative annexes: "Example of coding" and "Lower layer services".

Road transport and traffic telematics — Dedicated short range communication (DSRC) — DSRC application layer

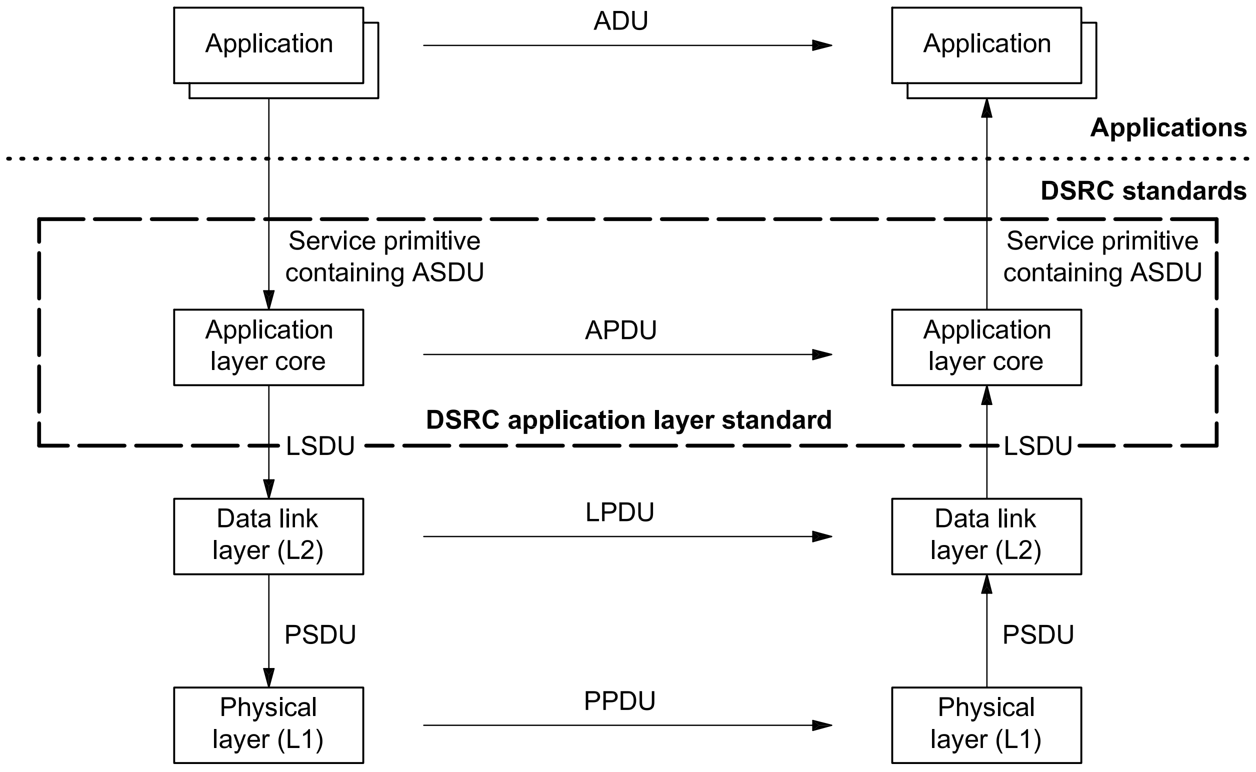
1 Scope

This International Standard specifies the application layer core which provides communication tools for applications based on DSRC. These tools consist of kernels that can be used by application processes via service primitives. The application processes, including application data and application-specific functions, are outside the scope of this International Standard.

This International Standard is named "application layer", although it does not cover all functionality of OSI Layer 7 and it includes functionality from lower layers.

It uses services provided by DSRC data link layer, and covers functionality of intermediate layers of the "OSI Basic Reference Model" (ISO/IEC 7498-1).

Figure 2 illustrates the global data flow between the parts of the DSRC stack (physical, data link and application layers) and the application.



NOTE For definitions of the terms used in Figure 2, see ISO/IEC 7498-1.

Figure 2 — Architecture and data flow of the DSRC stack

The following subjects are covered by this International Standard:

- application layer structure and framework;
- services to enable data transfer and remote operations;
- application multiplexing procedure;
- fragmentation procedure;
- concatenation and chaining procedures;
- common encoding rules to translate data from abstract syntax ASN.1 (ISO/IEC 8824-1) into transfer syntax (ISO/IEC 8825-2:2002) and vice versa;
- communication initialisation and release procedures;
- broadcast service support;
- DSRC management support including communication profile handling; and
- extensibility for different lower layer services and application interfaces.

It is outside the scope of this International Standard to define a security policy. Some transport mechanisms for security-related data are provided.

NOTE No implementation of the “broadcast pool” functionality has become known. “Broadcast pool” functionality is therefore considered untested.

2 Normative references

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

ISO/IEC 7498-1, *Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*

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

ISO/IEC 8825-2:2002, *Information technology — ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*

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

3 Terms and definitions

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

3.1
application
user of the services offered by the DSRC communication stack

3.2
attribute
value, which may have a structure, consisting of a set or sequence of data elements

NOTE The value of an “attribute” can be observed or modified by sending a request to GET (read) or SET (write) the value.

3.3
attribute identifier
identifier which unambiguously distinguishes an attribute from all other attributes within the same element

3.4
beacon service table
data structure transmitted by the RSU indicating available services

3.5
broadcast pool
data structure broadcast from the RSU to the OBUs

3.6
chaining
function performed by the transfer kernel to link the execution of service primitives

3.7
concatenation
function performed by the transfer kernel to map multiple T-APDU fragments into one data link layer service data unit

NOTE The inverse function is called separation or deconcatenation.

3.8
element
coherent set of data and functionality

NOTE Application elements are created by the applications and are addressed using element identifiers.

3.9
element identifier
identifier which unambiguously distinguishes an element from all other elements residing in the same OBU

3.10
fragmentation
function performed by the transfer kernel to map one ASDU on multiple LSDUs

NOTE 1 In ISO/IEC 7498-1, fragmentation is called segmentation.

NOTE 2 The inverse function is called defragmentation or, in ISO/IEC 7498-1, disassembling.

3.11
head of the line
queuing discipline (also referred to as strict or fixed priority queuing), where a number of queues are served in priority order

NOTE A lower priority queue is served if all higher priority queues are empty, each queue is served in “first come, first served” order, and each user goes to the head of the line of the users of lower priorities but behind all users of equal or higher priority.

3.12
management
provides and distributes values for the communication parameters for controlling the DSRC communication stack

3.13
multiplexing
function within the transfer kernel allowing simultaneous support for more than one application in a single OBU

3.14
operation
abstract representation of behaviour invoked in an entity

3.15
profile
information about capabilities and settings in the different DSRC layers

3.16
single-T-APDU fragment
T-APDU that contains a complete PDU

3.17
T-APDU fragment
fragment header followed by part or all of the encoding of a value of the ASN.1 type T-APDUs

3.18
time
number of seconds passed since 1st January 1970, 00:00 (UTC)

3.19
vehicle service table
data structure transmitted by the OBU indicating available services

4 Abbreviations

For the purposes of this document, the following abbreviations apply.

4.1
ADU
application data unit

4.2
AID
application identifier

4.3
APDU
application protocol data unit

4.4
ARIB
Association of Radio Industries and Businesses

4.5
ASDU
application service data unit

4.6
ASN.1
abstract syntax notation one (ISO/IEC 8824-1)

4.7
ASTM
American Society of Testing and Material

4.8
B-Kernel
broadcast kernel

4.9
BST
beacon service table

4.10
CEN
Comité européen de normalisation

4.11
DSRC
dedicated short range communication

4.12
EID
element identifier

4.13
EVENT-RT
event-report

4.14
FCS
frame check sequence

4.15
ID
identifier

4.16
IEEE
Institute of Electrical and Electronic Engineers

4.17
IID
invoker identifier

4.18
I-Kernel
initialisation kernel

4.19
LID
logical link control identifier

4.20
LLC
logical link control

4.21
LPDU
LLC protocol data unit

4.22
LSDU
LLC service data unit

4.23
L1
layer 1 of DSRC (physical layer)

4.24
L2
layer 2 of DSRC (data link layer)

4.25
L7
application layer core of DSRC

4.26
MAC
medium access control

4.27
NEN
Nederlands Normalisatie-instituut

4.28
OBU
on-board unit

NOTE This equipment usually resides on board a vehicle.

4.29
PDU
protocol data unit

4.30
PPDU
physical layer protocol data unit

4.31
PSDU
physical layer service data unit

4.32
PER
packed encoding rules (ISO/IEC 8825-2:2002)

4.33
RSU
road-side unit

NOTE This is often referred to as beacon.

4.34
RTTT
road transport and traffic telematics

4.35
SDU
service data unit

4.36
T-APDU
transfer application protocol data unit

4.37
T-Kernel
transfer kernel

4.38
VST
vehicle service table

5 Structure of the application layer core

The “application layer core” shall consist of the T-Kernel and either the I-Kernel or the B-Kernel, or both.

Figure 3 shows the application layer kernels and the relationships to external entities. The T-Kernel provides the basic transportation facilities that can be used by the I-Kernel, by the B-Kernel and by the applications.

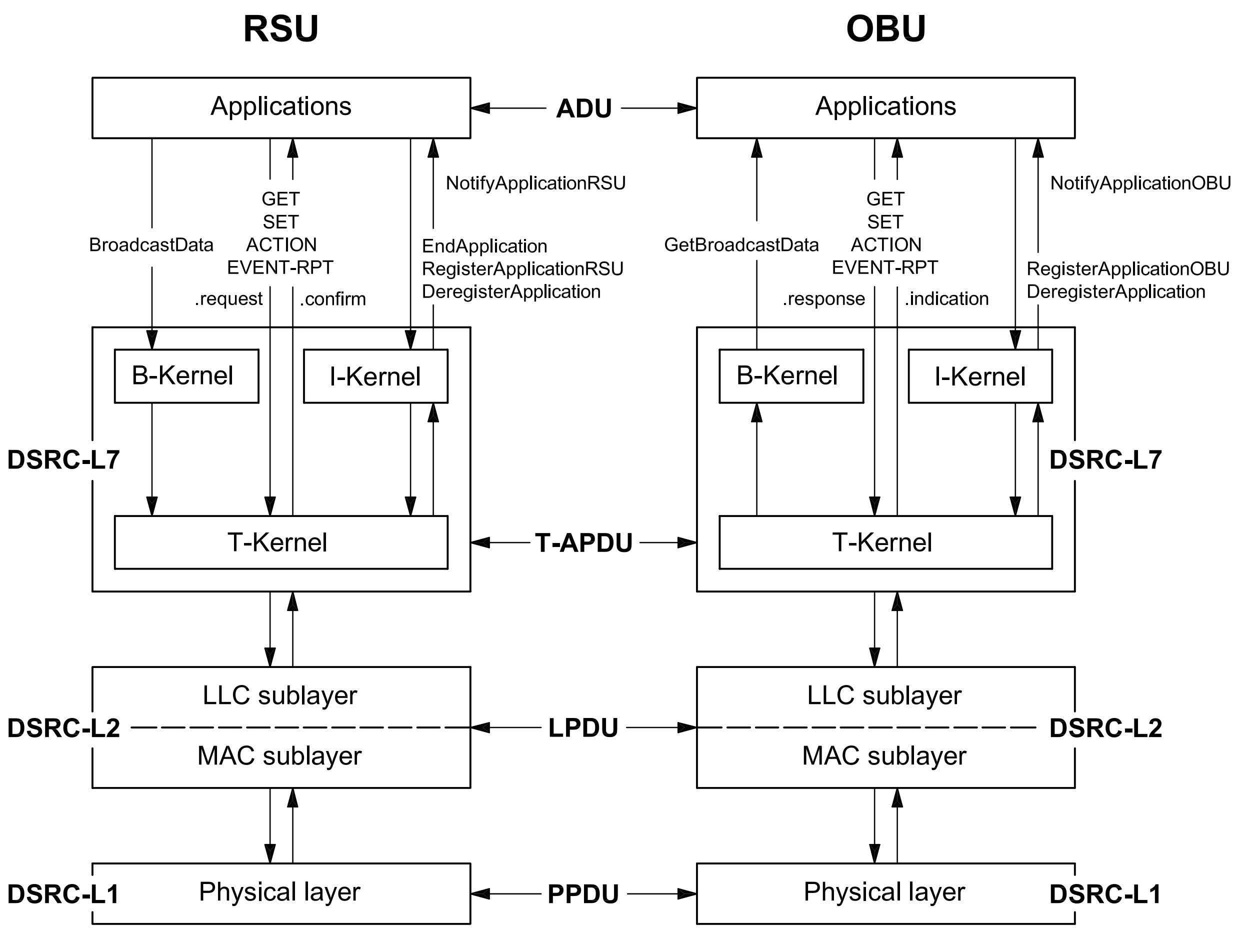


Figure 3 — Context and structure of the application layer core

6 T-Kernel

6.1 General

The T-Kernel shall transfer information between two peer kernels or applications, and shall abstract from the realization of this transfer.

The T-Kernel shall offer its services by means of service primitives defined in 6.2.2.

The T-Kernel shall transfer the information by means of T-APDUs defined in Annex A.

The T-Kernel shall realize the transfer by means of a protocol with the behaviour defined in 6.3.

The T-Kernel shall use the services of the logical link control sub-layer of the DSRC data link layer, which is defined in Clause 9 and Annex E.

NOTE The behaviour defined in 6.3 does not guarantee that the service elements with the same priorities will be delivered to a receiving application in the same order as they were delivered to the T-Kernel on the sending side.

6.2 Services

6.2.1 General

The T-Kernel shall provide the following services:

— GET: The invocation of the "GET" service by an application shall result in the retrieval (reading) of information (i.e. attributes) from a peer application. The service shall only be requested in a confirmed mode, and a reply is expected.

— SET: The invocation of the "SET" service by an application shall result in the modification (writing) of information (i.e. attributes) by a peer application. The service may be requested in confirmed or non-confirmed mode. In confirmed mode, a reply is expected.

— ACTION: The invocation of the "ACTION" service by an application shall result in the performance of an action by a peer application. An action is further qualified by the value of the "ActionType" (see ISO 14906 for examples). The service may be requested in confirmed or non-confirmed mode. In confirmed mode, a reply is expected.

— EVENT-REPORT: The invocation of the "EVENT-REPORT" service by an application or by the I-Kernel shall result in the notification of an event to a peer application or I-Kernel. The service may be requested in confirmed or non-confirmed mode. In confirmed mode, a reply is expected.

— INITIALISATION: The invocation of the "INITIALISATION" service by the I-Kernel shall result in an attempt to initialise the communication between an RSU and each OBU that has not yet established communication with that RSU. The "INITIALISATION" service shall only be used by the I-Kernel.

6.2.2 Service primitives

The T-Kernel shall provide the services given in 6.2.1 by the following service primitives:

— GET.request;

— GET.indication;

— GET.response;

— GET.confirm;

- SET.request;
- SET.indication;
- SET.response;
- SET.confirm;
- ACTION.request;
- ACTION.indication;
- ACTION.response;
- ACTION.confirm;
- EVENT-REPORT.request;
- EVENT-REPORT.indication;
- EVENT-REPORT.response;
- EVENT-REPORT.confirm:
- INITIALISATION.request;
- INITIALISATION.indication;
- INITIALISATION.response;
- INITIALISATION.confirm.

The INITIALISATION.request and INITIALISATION.confirm primitives shall only be used on the RSU side, the INITIALISATION.indication and INITIALISATION.response primitives shall only be used on OBU side.

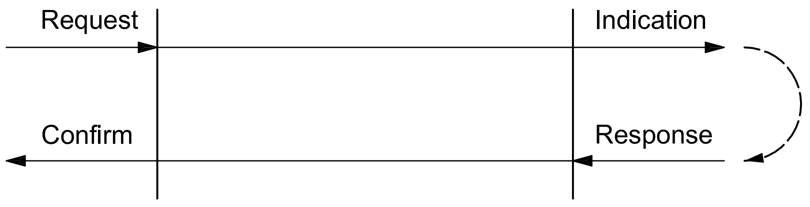


Figure 4 — Services used in confirmed mode

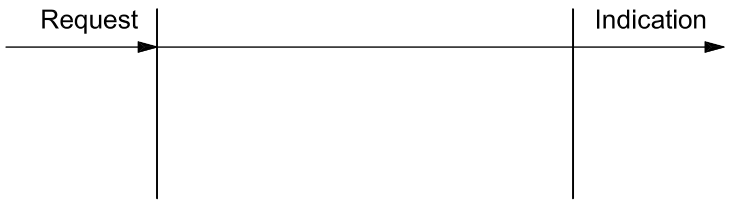


Figure 5 — Services used in non-confirmed mode

6.2.3 Format of service primitives

The T-ASDUs for the service primitives shall have the formats defined in Tables 2 to 6, with the notation defined in Table 1.

Table 1 — Definition of the expressions used in Tables 2 to 5

Symbol	Definition
iid/eid	Mandatory. IID of related indication if present, else EID of related indication.
optional	The use of the optional fields is detailed by the underlying ASN.1 standards.
=	Same as in corresponding request/indication.
—	Not applicable.

Table 2 — Format of the GET service primitives

Parameter name	Request/indication	Response	Confirm	ASN.1 type
Invoker identifier (IID)	optional	=		Dsrc-EID
Link identifier (LID)	mandatory	=		BIT STRING
Chaining	mandatory	=		Boolean
Element identifier (EID)	mandatory	iid/eid		Dsrc-EID
Access credentials	optional	—		OCTET STRING
Attribute Id list (AttrIdList)	optional	—		AttributeIdList
FlowControl	mandatory	mandatory	optional	INTEGER
Attribute list (AttrList)	—	optional		AttributeList
Return code (Ret)	—	optional		ReturnStatus

Table 3 — Format of the SET service primitives

Parameter name	Request/indication	Response	Confirm	ASN.1 type
Invoker identifier (IID)	optional	=		Dsrc-EID
Link identifier (LID)	mandatory	=		BIT STRING
Chaining	mandatory	=		Boolean
Element identifier (EID)	mandatory	iid/eid		Dsrc-EID
Access credentials	optional	—		OCTET STRING
Attribute list (AttrList)	mandatory	—		AttributeList
Mode	mandatory	—		Boolean
FlowControl	mandatory	mandatory	optional	INTEGER
Return code (Ret)	—	optional		ReturnStatus

Table 4 — Format of the ACTION service primitives

Parameter name	Request/indication	Response	Confirm	ASN.1 type
Invoker identifier (IID)	optional	=		Dsrc-EID
Link identifier (LID)	mandatory	=		BIT STRING
Chaining	mandatory	=		Boolean
Element identifier (EID)	mandatory	iid/eid		Dsrc-EID
ActionType	mandatory	—		INTEGER(0..127,..)
Access credentials	optional	—		OCTET STRING
ActionParameter	optional	—		Container
Mode	mandatory	—		Boolean
FlowControl	mandatory	mandatory	optional	INTEGER
ResponseParameter	—	optional		Container
Return code (Ret)	—	optional		ReturnStatus

Table 5 — Format of the EVENT-REPORT service primitives

Parameter name	Request/indication	Response	Confirm	ASN.1 type
Invoker identifier (IID)	optional	=		Dsrc-EID
Link identifier (LID)	mandatory	=		BIT STRING
Chaining	mandatory	=		Boolean
Element identifier (EID)	mandatory	iid/eid		Dsrc-EID
EventType	mandatory	—		INTEGER(0..127,..)
Access credentials	optional	—		OCTET STRING
EventParameter	optional	—		Container
Mode	mandatory	—		Boolean
FlowControl	mandatory	mandatory	optional	INTEGER
Return code (Ret)	—	optional		ReturnStatus

Table 6 — Format of the INITIALISATION service primitives

Parameter name	Request/indication	Response	Confirm	ASN.1 type
Link identifier (LID)	mandatory	mandatory (private)		BIT STRING
Initialisation parameter	mandatory (BST)	mandatory (VST)		BST/VST

6.2.4 Parameters

The parameters shall be set and interpreted as follows:

IID shall be of ASN.1 type Dsrc-EID and carry the identifier of the element initiating the request or the response, respectively. This parameter is not needed if an answer shall be sent to a default invoker. If IID is used, it shall contain the EID of the response to this primitive.

LID shall be the LID chosen by the I-Kernel on the OBU side as specified in 7.3.2.

Chaining shall be a Boolean parameter. If true, “concatenation with chaining”, defined in 6.3.6, is performed.

EID shall be of ASN.1 type Dsrc-EID and carry the identifier of the element that shall receive the indication or confirm related to a request or response, respectively. This EID is used by the T-Kernel on the side of the receiver to deliver an indication or a confirm to the addressed element. When the IID is used in a request, the element invoking a response shall use this IID as the EID.

AccessCredentials shall be of ASN.1 type OCTET STRING and carry security-related information needed to fulfil access conditions in order to perform the operation on the addressed element.

AttrIdList shall be a list of IDs of attributes of the element receiving a GET.indication. The values of these attributes shall be sent via a GET.response and GET.confirm to the element invoking the GET.request, provided that applicable access conditions have been fulfilled.

FlowControl shall be a parameter that represents the behaviour of the underlying communication service. This parameter shall be mapped by the T-Kernel on a certain LLC-service. The relation between FlowControl parameter, application layer behaviour and LLC service shall be as defined in Table 7. Other lower layer (LLC) services related or not related to this table are described in Clause 9 and Annex E.

Table 7 — Relation between FlowControl parameter, application layer behaviour and LLC service

Flow-Control	Application layer	LLC service
1	no flow control, no answer	DL-UNITDATA.request without response request
2	no flow control, answer	DL-UNITDATA.request with response request
3	no flow control	DL-UNITDATA.indication
4	flow control, data unit transmission	DL-DATA-ACK.request
5	flow control, data unit transmission	DL-DATA-ACK.indication
6	flow control, data unit transmission status	DL-DATA-ACK-STATUS.indication
7	flow control, data unit exchange	DL-REPLY.request
8	flow control, data unit exchange	DL-REPLY.indication
9	flow control, data unit exchange status	DL-REPLY-STATUS.indication
10	flow control, data unit exchange preparation	DL-REPLY-UPDATE.request
11	flow control, data unit exchange preparation status	DL-REPLY-UPDATE-STATUS.indication

AttrList shall be a sequence of attributes sent by SET.request/SET.indication or GET.response/GET.confirm. The element receiving a SET.indication shall modify the values of the attributes identified in the AttrIdList to the values given in the AttrIdList, provided that applicable access conditions have been fulfilled. In the case of GET.response/GET.confirm, the element that received the corresponding GET.indication shall send the values of the attributes addressed in the AttrIdList of the GET.indication to the element invoking the GET.request, provided that applicable access conditions have been fulfilled.

Ret shall be a return code issued as an answer to a service.indication. The following codes are predefined:

— noError: the requested operation was performed successfully;

— accessDenied: the requested operation was not performed for reasons pertinent to the security of the system;

— argumentError: one or more attribute values were not accessed because the identifier for the specified attribute was not recognized or the attribute value specified was out of range or otherwise inappropriate for one or more attributes, or the action or event-report invoked was not supported by the receiving entity;

— complexityLimitation: the requested operation was not performed because a parameter was too complex;

— processingFailure: a general failure in processing the operation was encountered;

- processing: the requested operation is being processed, and the result is not yet available;
- chainingError: the requested operation was not performed in accordance with the rule defined in 6.3.6, “concatenation with chaining”.

Mode shall be a Boolean parameter. If true, then there shall be a service.response to a service.indication (confirmed mode).

ActionType shall identify an operation of the element which receives an ACTION.indication and which shall be invoked.

ActionParameter shall be the information needed for the invocation of an operation identified in an ACTION.indication.

ResponseParameter may be information resulting from the execution of the operation invoked by ACTION.indication.

EventType shall identify the message which shall be delivered to an element which receives an EVENT-REPORT.indication.

EventParameter shall be the additional information needed for the message sent via an EVENT-REPORT.request and EVENT-REPORT.indication, respectively.

Initialisation parameter shall be the information needed for the initialisation of the communication (i.e. the BST on the downlink and VST on the uplink) sent via an initialisation service.

6.3 Behaviour

The transfer protocol shall consist of the following steps, described in 6.3.1 to 6.3.12:

a) translating SDU to a T-APDU,

b) encoding of the T-APDU,

c) fragmentation of the T-APDU,

d) multiplexing of T-APDU fragments,

e) concatenation of short T-APDU fragments,

f) access to the LLC,

g) access from the LLC,

h) demultiplexing of T-APDU fragments,

i) defragmentation of non-concatenated T-APDU fragments,

j) decoding and separation of a T-APDU, possibly concatenated with one or more single-T-APDU fragments, and

k) translating PDU to SDU and distribution to addressee.

NOTE 1 The implementation of individual steps is outside the scope of this International Standard as long as their (externally observable) behaviour complies with requirements below. An implementation may even change the order of steps as long as this does not have any implication for its peer implementation and for the overall behaviour of the whole sequence of steps, i.e. for the T-APDUs and the LPDUs.

NOTE 2 The arrows shown in Figures 6 to 8 and 10 to 17 indicate the conversion process.

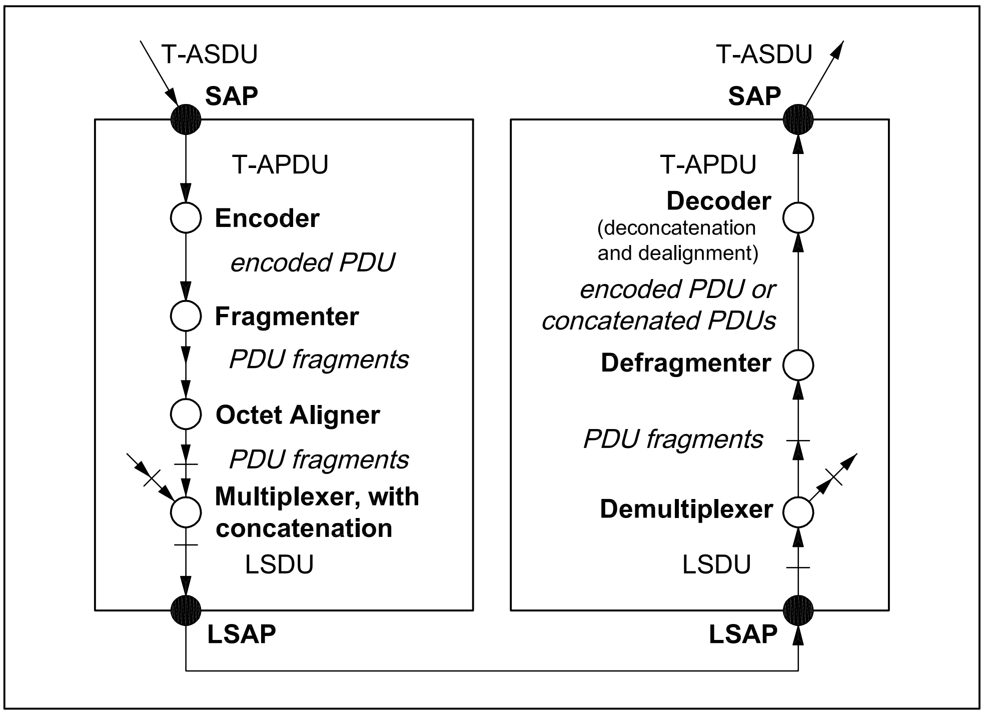


Figure 6 — Functionality of the T-Kernel

6.3.1 SDU to PDU

The T-Kernel shall translate the request and response service primitive into T-APDUs according to the following rules.

A service.request is translated into the corresponding service-request T-APDU defined in Annex A. A service.response is translated into the corresponding service-response T-APDU defined in Annex A. The LID shall be removed and shall be given to the LLC in each LLC service primitive as defined in 6.3.7. In the case of the INITIALISATION.request, the LID field shall be 1111 1111$_2$.

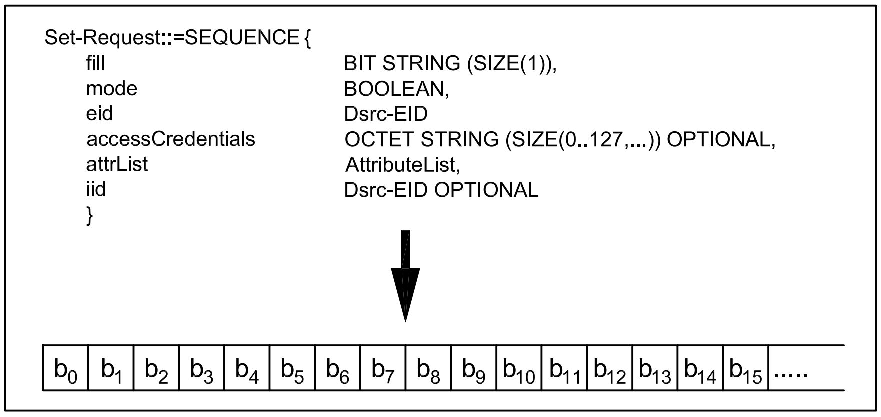
```
SET.request(IID, LID, Chaining, EID, AccessCredentials, AttrList, Mode, FlowControl)

                                   ↓

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

Figure 7 — SDU to PDU

6.3.2 Encoding

The T-Kernel shall encode the request and response PDUs according to ASN.1 BASIC-PER, UNALIGNED (ISO/IEC 8825-2:2002). The encodable ASN.1 types are specified in Annex A. The fill bit of ASN.1-type definitions shall be set to zero.

NOTE In order to mitigate the implication of UNALIGNED encoding, some fill bits are included in the ASN.1-type definition in Annex A. The encoding result is always an integral multiple of eight bits (octet). The octet alignment and octet dealignment procedures are performed in PDU encoding and decoding steps, if it is needed. The octet alignment and octet dealignment procedures are outside the scope of this International Standard. Refer to 10.1 of ISO/IEC 8825-2:2002.



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

b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15} .....

Figure 8 — Encoding

6.3.3 Fragmentation

The T-Kernel shall fragment encoded T-APDUs down to T-APDU fragments, which include a fragmentation header. A fragmentation header shall have a length of at least 1 octet and at most 3 octets. The length of the T-APDU fragments shall not exceed the maximum LLC frame length. The number of bits inside a fragment shall be a multiple of 8 bits. All but the last fragment shall have the same length. Fragmentation shall be made from the most significant bit down to the least significant bit according to ASN.1 BASIC-PER.

NOTE 1 Fragmentation is based on the property that the length of the PER encoded T-APDU is a multiple of eight bits.

Fragmentation of multiple PDUs shall be done in parallel.

NOTE 2 As fragmentation is done in parallel, the fragmentation of a small T-APDU may be finished before a larger one that was received from the SAP before the smaller one. As a consequence, T-APDUs with the same priority might not be sent in the same order as they were received from the SAP.

A T-APDU fragment that contains a complete T-APDU is called a single-T-APDU.

6.3.3.1 Fragmentation header

The first octet of the fragmentation header shall be the first octet of the fragment. If the fragmentation header consists of more than one octet, these octets are given in increasing order directly after the first octet.

6.3.3.2 Structure of the fragmentation header

The fragmentation header shall consist of one fragmentation indicator, one PDU number, one fragment counter, and one fragment number extension indicator. The position of the related bits is described in Figure 9. The bits are numbered from 7 to 0, where bit 7 is the most significant and bit 0 is the least significant bit.

7	6	5	4	3	2	1	0
Fragmentation indicator	PDU number				Fragment counter		Extension indicator

Figure 9 — One-octet fragmentation header

6.3.3.3 Fragmentation indicator

The most significant bit (bit 7) of any fragmentation header shall be the fragmentation indicator. The fragmentation indicator shall be 1_2 if this fragment is the last of a sequence of fragments belonging to one PDU or if the PDU is not fragmented. The fragmentation indicator shall be 0_2 if fragmentation is performed and if it is not the last fragmented frame in a message.

6.3.3.4 PDU number

Bits 6 to 3 of the first octet represent a PDU number, which shall be unique for each LID during the defragmentation time in the receiving entities and the same in all T-APDU fragments belonging to one T-APDU. PDU number 0000_2 and 0001_2 shall only be used by T-APDU fragments that are sent by the B-Kernel.

6.3.3.5 One-octet fragmentation header

A one-octet fragmentation header shall be used if no fragmentation is performed or, if fragmentation is performed, for fragments numbered 0 to 3. A fragment counter shall be used to identify a fragment. Bits 2 and 1 shall be interpreted as an unsigned integer, where the highest significant bit is bit 2 of the first octet and the least significant bit is bit 1 of the first octet. Bit 0 shall be set to 1_2. If fragmentation is performed then the first fragment shall be given the fragment counter value 0, the second fragment the fragment counter value 1, etc. If no fragmentation is performed, then the fragment shall be given the fragment counter value 0.

6.3.3.6 Two-octet fragmentation header

A two-octet fragmentation header shall be used for fragments between 4 and 511; bit 0 of the first octet shall be set to 0_2. Bits 2 and 1 of the first octet and bits 7 to 1 of the second octet shall be interpreted as an unsigned integer, where the highest significant bit is bit 2 of the first octet and the least significant bit is bit 1 of the second octet. Bit 0 of the second octet shall be set to 1_2. Fragment numbers shall be assigned to the fragment as specified in 6.3.3.5.

6.3.3.7 Three-octet fragmentation header

A three-octet fragmentation header shall be used for fragments between 512 and 65535; bit 0 of the first octet shall be set to 0_2. Bits 2 and 1 of the first octet, bits 7 to 1 of the second octet and bits 7 to 1 of the third octet are interpreted as unsigned integers, where the highest significant bit is bit 2 of the first octet and the least significant bit is bit 1 of the third octet. Bit 0 of the second octet shall be set to 0_2 and bit 0 of the third octet shall be set to 1_2. Fragment numbers shall be assigned to the fragment as specified in 6.3.3.5.

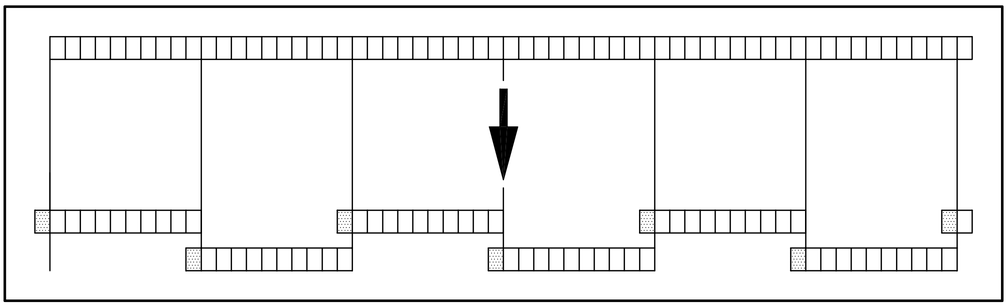


Figure 10 — Fragmentation

6.3.4 Multiplexing

The T-Kernel shall multiplex the T-APDU fragments according to a head-of-the-line strategy where the priorities are given by the I-Kernel (see 7.3.2).

NOTE Apart from respecting priorities, this International Standard does not impose any restriction on how a multiplexer should serve the queues. As a consequence, T-APDUs with the same priority may not be sent in the same order as they were fragmented.

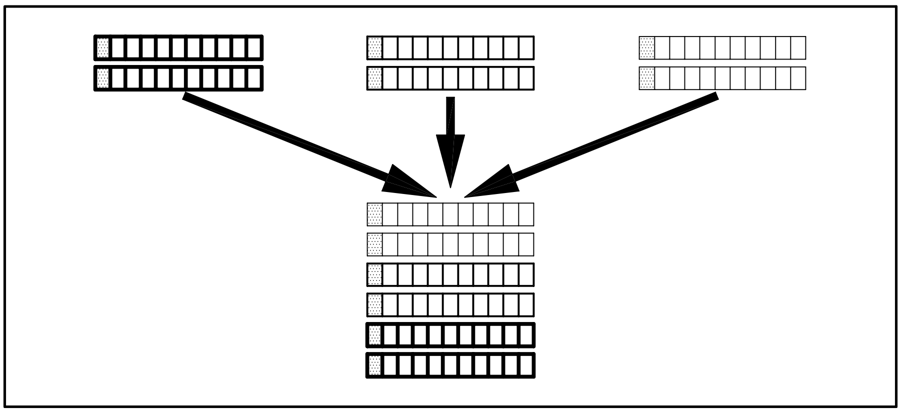


Figure 11 — Multiplexing

6.3.5 Concatenation

Multiple consecutive T-APDU fragments may be mapped on one LLC service if the services and the LID used in the services are the same and the size constraints for the LLC frames are not violated. The order of the T-APDU fragments inside the LSDU shall be the one given by the multiplexing procedure.

NOTE These conditions for concatenation will only occur in the following two situations.

1) One or more T-APDU fragments, each containing one short, unfragmented T-APDU, are mapped on one LSDU.

2) One or more T-APDU fragments, each containing one short, unfragmented T-APDU, are mapped on one LSDU after the last fragment of a series of T-APDU fragments belonging to one T-APDU.

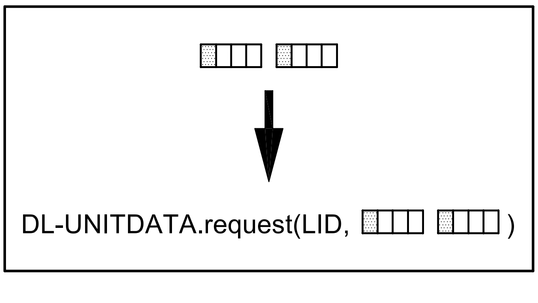


Figure 12 — Concatenation

6.3.6 Concatenation with chaining

A chain consists of consecutive and ordered concatenated T-APDU fragments having the same PDU number.

The execution of operations invoked by a T-APDU belonging to a chain is dependent on the successful execution of the operations invoked in the previous T-APDU fragments of the same chain.

If a chained T-APDU fragment generates a response T-APDU fragment with a ReturnStatus different from “no error”, none of the operations invoked by the subsequent T-APDU fragments of the same chain shall be performed and all associated responses shall contain a ReturnStatus of “chainingError”.

NOTE If a chaining parameter has the value “true”, and the procedures defined in 6.3.6 are ignored, it would not affect the whole peer-to-peer protocol because the chaining parameters are not passed to a peer receiver side.

6.3.7 Access to LLC

The T-Kernel shall use the LLC service assigned in the FlowControl parameter of the T-ASDU. The FlowControl parameter shall be interpreted as defined in 6.2.4. For the INITIALISATION.request service the “DL-UNITDATA.request with response request” service shall be used; for the INITIALISATION.response service, the “DL-UNITDATA.request without response request” service shall be used.

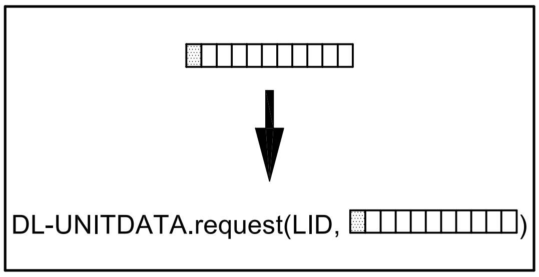


Figure 13 — Access to LLC

6.3.8 Access from LLC

The T-Kernel on the receiving side shall receive the LSDUs in the LLC indication primitives via the LSAP.

6.3.9 Demultiplexing

The T-Kernel shall demultiplex the LSDUs containing one or more concatenated T-APDU fragments according to the PDU number in the first fragmentation header.

The T-Kernel shall demultiplex LSDUs containing concatenated T-APDU fragments according to the PDU number in the first fragmentation header.

An LSDU with an invalid first fragmentation header shall be discarded.

NOTE 1 This results in queues in which all LSDUs have the same PDU number in their first fragment header. As subsequent T-APDU fragments in an LSDU can only be single-T-APDU fragments, all T-APDU fragments from a T-APDU will be put in the same queue.

NOTE 2 As only a decoder that knows the type of PDUs to be decoded can determine the length of a PER-encoded PDU, the demultiplexer cannot determine the length of concatenated T-APDU fragments. Therefore, separation of concatenated T-APDU fragments is postponed until the decoding step. As this demultiplexing step guarantees that any two T-APDU fragments with the same PDU number will always be put in the same queue, this demultiplexing (separation) can indeed be solved later.

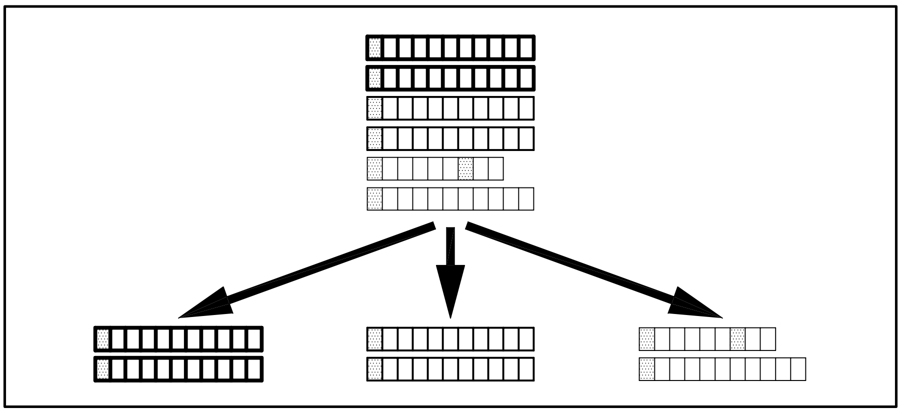


Figure 14 — Demultiplexing

6.3.10 Defragmentation

The T-Kernel shall defragment the LSDUs per queue, i.e. all LSDUs with same PDU number in the first fragmentation header, by removing in each LSDU the first fragmentation header and by concatenating the remaining LSDU parts according to the fragment number in the first fragmentation header. If the fragmentation header is not valid, the LSDU and all other LSDUs with same PDU number in their first fragmentation header shall be discarded.

NOTE 1 This defragmentation results in octet strings that contain one complete T-APDU possibly concatenated with one or more single-T-APDU fragments.

NOTE 2 As only a decoder that knows the type of PDUs to be decoded can determine the length of a PER-encoded PDU, the defragmenter cannot determine the length of the first T-APDU and therefore cannot separate a T-APDU from any concatenated single-T-APDU fragments. The separation of concatenated T-APDU fragments is postponed until the decoding step.

NOTE 3 As one defective first fragmentation header will cause the deletion of all LSDUs with the same PDU number in their first fragmentation header, such a defect will cause the deletion of any concatenated single-T-APDU fragments.

If one or more T-APDU fragment headers of a T-APDU are still missing when T-APDU fragments with the eight higher PDU numbers (out of 14 cyclically used values) are received, all LSDUs with a PDU number equal to the PDU number of a missing T-APDU fragment are discarded.

NOTE 4 As the T-APDU fragments could not be interpreted, a T-Kernel cannot determine whether or not a discarded T-APDU is part of an indication or part of a conformance application service primitive, or whether it was part of an action, an event-report, a set or a get. The T-Kernel is therefore unable to generate or assist an application element in generating a T-APDU with the correct value of the ReturnStatus.

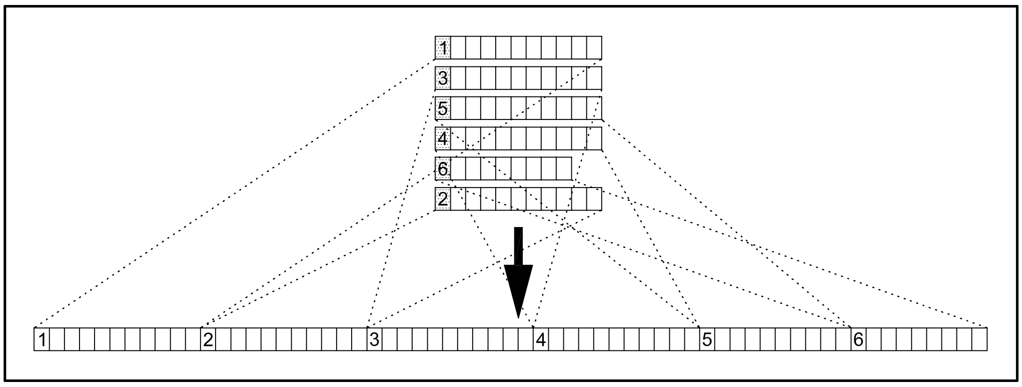


Figure 15 — Defragmentation

6.3.11 Decoding and separation

The T-Kernel shall decode the defragmented T-APDU according to ASN.1 BASIC-PER, UNALIGNED (ISO 8825-2:2002). The decodable ASN.1 types are specified in Annex A.

When provided with an octet string containing a T-APDU with possibly concatenated single-T-APDU fragments, the decoder (separator) proceeds decoding steps as described below.

1) It decodes the T-APDU.

2) If it does not succeed in decoding the T-APDU, the whole octet string is discarded.

3) If it succeeds in decoding the T-APDU, it passes the decoded T-APDU to the next procedure (see 6.3.12).

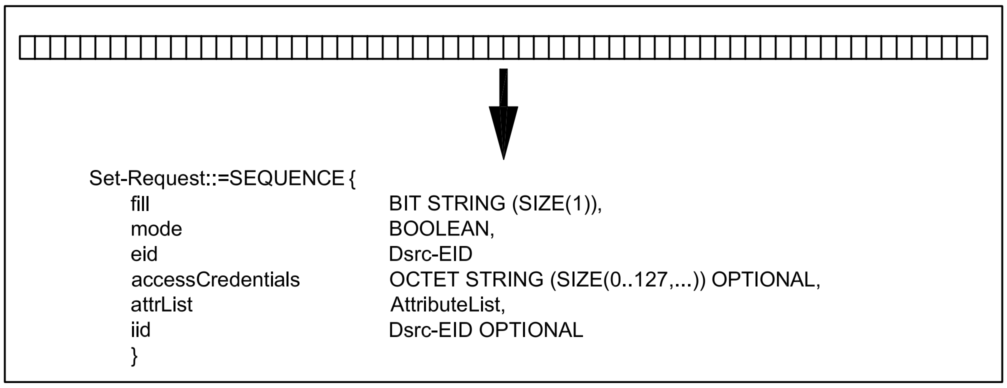
4) It shall look for trailing octets and assume that these remaining octets contain one more single-T-APDU fragments.

5) It shall check the validity of the first fragment header of the remaining octet string (bit 2 to bit 0 shall have the value “001_2”).

6) If the first fragment header is invalid, all the remaining T-APDU fragments shall be discarded.

7) If the first fragment header of the T-APDU is valid, the fragment header shall be removed and the decoder (separator) shall proceed decoding with step 1.

This requires an ASN.1 decoder that does not collapse when it has to decode an octet string longer than the PDU to be decoded. The decoder should even be able to return the remaining octet string. Whether or not such an ASN.1 decoder is commercially available is the outside the scope of this International Standard.



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

Figure 16 — Decoding

6.3.12 PDU to SDU

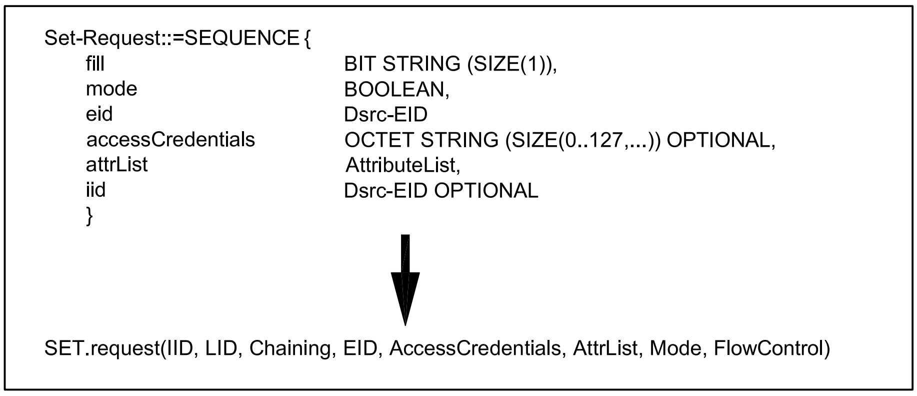
The decoded T-APDU shall be used to build the T-ASDU according to the following rules.

A service-request shall be translated into the corresponding service.indication T-ASDU. A service-response shall be translated into the corresponding service.confirm T-ASDU.

The T-ASDU shall be delivered to the element addressed in the EID parameter of the T-APDU, but not to the T-Kernel itself. The INITIALISATION.indication SDU shall be delivered to the I-Kernel.

If the addressed element is not present, the T-ASDU shall be discarded.

The T-Kernel shall inform the management about the LID of this SDU.



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

SET.request(IID, LID, Chaining, EID, AccessCredentials, AttrList, Mode, FlowControl)
```

Figure 17 — PDU to SDU

7 Initialisation kernel

7.1 General

The I-Kernel shall realize the initialisation of the communication between OBU and RSU by exchanging information concerning profiles and applications with its peer entity. It shall inform the applications inside the OBU about the presence of a peer application inside the RSU. It shall handle the LID of the OBU.

The I-Kernel shall offer its services by means of service primitives defined in 7.2.

The I-Kernel shall initialise the communication by means of a BST as defined in Annex A. The BST shall be transferred in one LLC service primitive.

The I-Kernel may initialise the communication by means of a VST as defined in Annex A.

The I-Kernel shall perform the initialisation by a protocol with the behaviour defined in 7.3.

The I-Kernel shall perform associations by means of the LID, the BST and a release mechanism.

7.2 Services

7.2.1 Description of service primitives

The I-Kernel shall provide the following services to other kernels or applications:

— RegisterApplicationRSU: The invocation of the RegisterApplicationRSU service by an application on the RSU side shall result in the registration of this application in the ApplicationList of the I-Kernel.

— RegisterApplicationOBU: The invocation of the RegisterApplicationOBU service by an application on the OBU side shall result in the registration of this application in the ApplicationList of the I-Kernel.

— DeregisterApplication: The invocation of the DeregisterApplication service by an application shall result in the deletion of the related entry in the ApplicationList.

— NotifyApplicationOBU: The I-Kernel uses the NotifyApplicationOBU service to inform the application on the OBU side about the presence of a potential communication partner (i.e. application on the RSU side) and the LID generated by the OBU.

— NotifyApplicationRSU: The I-Kernel uses the NotifyApplicationRSU service to inform the application on the RSU side about the presence of a potential communication partner (i.e. application) and the LID of the associated OBU.

— EndApplication: The invocation of the EndApplication service by the application shall result in the notification of the I-Kernel that the LID is no longer needed for this application.

7.2.2 Format of service primitives

The initialisation ASDU for the service primitives shall have the formats defined in Tables 8 to 13.

Table 8 — RegisterApplicationRSU

Parameter name	**ASN.1 type**	**Optional or default**
AID	DSRCApplicationEntityID	
MandatoryApplication	BOOLEAN	
Priority	INTEGER	
EID	Dsrc-EID	optional
Profiles	SEQUENCE OF Profile	optional
Parameter	ApplicationContextMark	optional

Table 9 — RegisterApplicationOBU

Parameter name	**ASN.1 type**	**Optional or default**
AID	DSRCApplicationEntityID	
Priority	INTEGER	
EID	Dsrc-EID	
Profiles	SEQUENCE OF Profile	optional
Parameter	ApplicationContextMark	optional

Table 10 — DeregisterApplication

Parameter name	**ASN.1 type**	**Optional or default**
AID	DSRCApplicationEntityID	
EID	Dsrc-EID	optional

Table 11 — NotifyApplicationRSU

Parameter name	**ASN.1 type**	**Condition**
Priority	INTEGER	
EID	Dsrc-EID	if present for AID in VST
LID	BIT STRING	
Parameter	ApplicationContextMark	optional
ObeConfiguration	ObeConfiguration	

Table 12 — NotifyApplicationOBU

Parameter name	ASN.1 type	Condition
RSU	BeaconID	
Priority	INTEGER	
EID	Dsrc-EID	if present for AID in BST
LID	BIT STRING	
Parameter	ApplicationContextMark	optional

Table 13 — EndApplication

Parameter name	ASN.1 type	Condition
EID	Dsrc-EID	
LID	BIT STRING	

7.2.3 Parameters

The parameters shall be interpreted as follows:

— AID identifies the DSRC application.

— MandatoryApplication is “true” if the application is mandatory and “false” if it is non-mandatory.

— Priority is the priority of the application. A small value represents a high priority, a large value represents a low priority. The I-Kernel may take this parameter into account when deciding on the priority of the application.

The EID parameter includes the following:

— For the RegisterApplicationRSU service, the EID identifies the element to be registered. EID is not used if the application is the single-default application.

— For the RegisterApplicationOBU service, the EID identifies the element to be registered.

— For the NotifyApplication services, the EID identifies an element of the peer application, if present.

Profiles, if present, give a list of profiles associated with the application.

The parameter is optional information for the RegisterApplicationRSU and RegisterApplicationOBU services. If present, the parameter is conveyed in the NotifyApplicationOBU or NotifyApplicationRSU services, respectively. Parameter is of type ApplicationContextMark, which shall be of Container CHOICE type OCTET STRING (SIZE(0..127,…)).

RSU indicates the identification of the RSU which offers the service.

ObeConfiguration describes the configuration and the status of the OBU related to the LID given in the NotifyApplicationRSU.

7.2.4 Location of services

On the RSU side, the I-Kernel shall provide the RegisterApplicationRSU, DeregisterApplication, NotifyApplicationRSU and EndApplication services.

On the OBU side, the I-Kernel shall provide the RegisterApplicationOBU, DeregisterApplication, NotifyApplicationOBU and EndApplication services.

7.3 Behaviour

7.3.1 RSU: Repetitive transmission of BST

7.3.1.1 Reason

System implementation requires transmission of BST.

7.3.1.2 Behaviour

On the RSU side, the I-Kernel shall transmit the BST defined in Annex A. It shall use the INITIALISATION.request service of the T-Kernel with the following parameter settings: initialisation parameter = BST.

The time between subsequent BST transmissions is a system implementation issue and is outside the scope of this International Standard.

7.3.2 OBU: Reception of BST and transmission of VST

7.3.2.1 Reason

The OBU I-Kernel receives a BST by means of an INITIALISATION.indication of the T-Kernel with the following parameter settings. Initialisation parameter = BST.

7.3.2.2 Behaviour

If at least one of the following conditions,

a) the BeaconID differs from the last received BeaconID,

b) the time between the last received BST and the current received BST exceeds T seconds, the value of which is defined in Clause 9,

is fulfilled, the OBU I-Kernel shall show the following behaviour:

- The I-Kernel shall inform the management about the profile received in the BST. This profile shall be the default profile for the communication.
- The I-Kernel shall compare the DSRCApplicationEntityIDs given in the ApplicationList inside the BST with the registered DSRCApplicationEntityIDs. For all registered DSRCApplicationEntityIDs which are inside the BST it shall:
  - add the DSRCApplicationEntityIDs and, if present in the related RegisterApplicationOBU, the EID and the parameters into the ApplicationList of the VST;
  - notify the registered element by means of a NotifyApplicationOBU with the following parameters:
    - RSU = BeaconID given in the BST;
    - Priority = position in the mandatory ApplicationList of the BST for all DSRCApplicationEntityIDs inside the mandatory ApplicationList or the number of DSRCApplicationEntityIDs in the mandatory ApplicationList plus the registered priority for all DSRCApplicationEntityIDs inside the non-mandatory ApplicationList;
    - Link identifier = LID selected by the I-Kernel;
    - Parameter = parameter received in the BST if present, else empty.

- The I-Kernel shall choose a random LID having an approximately uniform distribution over the range of possible values according to LLC requirements, as defined in Clause 9.
- The I-Kernel shall inform the T-Kernel via the management about the priorities of the notified applications.
- The I-Kernel shall select a profile out of the profile or the profileList of the BST which is supported in the OBU (i.e. which is known by the I-Kernel), and shall set the profile data element of the VST to this profile.
- The VST is of ASN.1 type VST as defined in Annex A.
- The I-Kernel shall transmit the VST. It shall use the INITIALISATION.response service of the T-Kernel with the following parameter settings:
  - Link identifier = LID;
  - Initialisation parameter = VST.
- The I-Kernel shall store the BeaconID and time.
- The I-Kernel shall store the ApplicationList of the VST together with the LID as long as the LID is valid.

7.3.3 RSU: Answer to VST

7.3.3.1 Reason

The RSU I-Kernel receives a VST by means of an INITIALISATION.confirm of the T-Kernel with the parameters:

- Link identifier = LID;
- Initialisation parameter = VST.

7.3.3.2 Behaviour

For all DSRCApplicationEntityIDs inside the VST, the I-Kernel shall notify the registered element by means of a NotifyApplicationRSU with the following parameters:

- Priority = position in the list of mandatory applications, mandApplications of the BST for all DSRCApplicationEntityIDs inside the mandApplications, or the number of DSRCApplicationEntityIDs in the mandApplications plus the position in the nonmandApplications for all DSRCApplicationEntityIDs inside the nonmandApplications;
- EID = EID received in the VST if present, else empty;
- LID = LID received in the INTITIALISATION.confirm;
- Parameter = Parameter received in the VST if present, else empty;
- ObeConfiguration = ObeConfiguration received in the VST.

The I-Kernel shall inform the management about the relation between LID and the profile given inside the VST. This profile shall be used for further communication with the OBU with this LID, i.e. outgoing data shall be sent using this profile and incoming data shall be received using this profile.

The I-Kernel shall store the ApplicationList of the VST together with the LID given in the INITIALISATION.confirm.

7.3.4 RSU: RegisterApplicationRSU

7.3.4.1 Reason

The reason for this is the invocation by an application (application on the RSU side).

7.3.4.2 Behaviour

On receiving a RegisterApplicationRSU primitive, the RSU I-Kernel shall insert the information given in the primitive into the ApplicationList for mandatory or non-mandatory applications, as applicable. It may use the information given in the parameters “MandatoryApplication” and “Priority”. If a B-Kernel is present, then the priority of the B-Kernel is decided by the I-Kernel. The profiles may be inserted into the ProfileList of the BST.

7.3.5 OBU: RegisterApplicationOBU

7.3.5.1 Reason

The reason for this is the invocation by an application.

7.3.5.2 Behaviour

On receiving a RegisterApplicationOBU, the OBU I-Kernel shall add this application to the list of registered applications.

Each element in the OBU shall have a unique number. The process shall guarantee that one unique EID is allocated for each element registered in the OBU (eid = 0 reserved for system).

7.3.6 OBU: DeregisterApplication

7.3.6.1 Reason

The reason for this is the invocation by an application.

7.3.6.2 Behaviour

On receiving a DeregisterApplication, the I-Kernel shall remove this application from the list of registered applications.

7.3.7 RSU: DeregisterApplication

7.3.7.1 Reason

The reason for this is the invocation by an application.

7.3.7.2 Behaviour

On receiving a DeregisterApplication, the I-Kernel shall remove the entry from the ApplicationList of the BST.

7.3.8 RSU: Release application

7.3.8.1 Reason

The reason for this is the invocation by an application.

7.3.8.2 Behaviour

On receiving an EndApplication, the I-Kernel shall delete the entry from the stored ApplicationList from the VST. If the application list is empty (i.e. all applications have ended), the I-Kernel shall transmit a release to the peer I-Kernel. It shall use the EVENT-REPORT.request service of the T-Kernel with the following parameter settings:

— Invoker identifier = (empty);

— Link identifier = LID;

— Element identifier = EID = 0;

— EventType = Release (0);

— EventParameter = (empty);

— Mode = FALSE;

— FlowControl = 1.

7.3.9 Reception of a Release

7.3.9.1 Reason

The I-Kernel receives a Release by means of an EVENT-REPORT.indication of the T-Kernel with the parameters:

— Invoker identifier = (empty);

— Link identifier = LID;

— EventType = Release (0);

— EventParameter = (empty);

— Mode = FALSE.

NOTE EID and FlowControl parameters are "mandatory" in Table 5. However, the EVENT-REPORT.indication does not need these parameters in the context of the RSU Release application process. Therefore, these are omitted.

7.3.9.2 Behaviour

The I-Kernel shall delete the VST related to the LID. The LID shall no longer be valid.

8 Broadcast kernel

8.1 General

The B-Kernel shall realize the collection, broadcast and distribution of information for different applications in the OBU and RSU by exchanging the broadcast pool.

The B-Kernel shall offer its services by means of service primitives defined in 8.2.

The B-Kernel shall realize the communication by sending a BroadcastPool defined in Annex A.

The B-Kernel shall realize the broadcast by a protocol with the behaviour defined in 8.3.

8.2 Services

8.2.1 Description of service primitives

The B-Kernel shall provide the following services.

— BroadcastData: The invocation of the BroadcastData service by the application on the RSU side shall result in the broadcast of information to the peer application on the OBU side or in the update of this information.

— GetBroadcastData: The invocation of the GetBroadcastData service by an application shall result in the retrieval of the broadcast data.

8.2.2 Format of service primitives

The broadcast ASDU for the service primitives shall have the formats defined in Tables 14 and 15.

Table 14 — BroadcastData

Parameter name	**ASN.1 type**
File	NamedFile

Table 15 — GetBroadcastData.request

Parameter name	**Request**	**Confirm**	**ASN.1 type**
Name	Mandatory	[a]	FileName
EID	Mandatory	[a]	Dsrc-EID
File	[a]	Mandatory	NamedFile
[a] Not applicable.			

8.2.3 Parameters

The parameters shall be set and interpreted as follows.

— The file shall be the NamedFile, which contains the information which shall be broadcast or retrieved, as applicable, from the broadcast pool.

— The name shall be the FileName of the file, which shall be retrieved from the broadcast pool.

— The EID shall be the Dsrc-EID of the element invoking the GetBroadcastData-service.

8.2.4 Location of services

On the RSU side, the B-Kernel shall provide the BroadcastData primitive. On the OBU side, the B-Kernel provides the GetBroadcastData.request and GetBroadcastData.confirm primitives.

8.3 Behaviour

8.3.1 RSU: Transmission of the BroadcastPool

8.3.1.1 Reason

System implementation requires transmission of the broadcast pool.

8.3.1.2 Behaviour

On the RSU side, the B-Kernel shall transmit the BroadcastPool defined in Annex A. It shall use the SET.request service of the T-Kernel periodically with the following parameter settings:

— Invoker identifier = (empty);

— Link identifier = 1111 1111_2;

— Element identifier = EID = 0;

— Attribute list = (0, BroadcastPool);

— Mode = FALSE;

— FlowControl = 1.

The determination of the length of the period between two transmissions is outside the scope of this International Standard.

8.3.2 OBU: Receiving the broadcast pool

8.3.2.1 Reason

The OBU B-Kernel receives the broadcast pool by means of a SET.indication of the T-Kernel with the following parameters:

— Invoker identifier = (empty);

— Link identifier = 1111 1111_2;

— Attribute list = (0, BroadcastPool);

— Mode = FALSE.

8.3.2.2 Behaviour

On the OBU side, the B-Kernel shall replace the current value of the BroadcastPool by the new value of the BroadcastPool.

8.3.3 RSU: BroadcastData

8.3.3.1 Reason

The reason for this is the invocation by an application.

8.3.3.2 Behaviour

The B-Kernel shall insert the file of the NamedFile into the content of the BroadcastPool and the FileName into the Directory. If there is a file with the same FileName in the BroadcastPool, this file is replaced by the new file.

8.3.4 OBU: GetBroadcastData.request

8.3.4.1 Reason

The reason for this is the invocation by an application.

8.3.4.2 Behaviour

The B-Kernel shall retrieve the file with the given FileName and give it to the application with Dsrc-EID (EID) by means of a GetBroadcastData.confirm.

9 Extensibility for different lower layer services and application interfaces

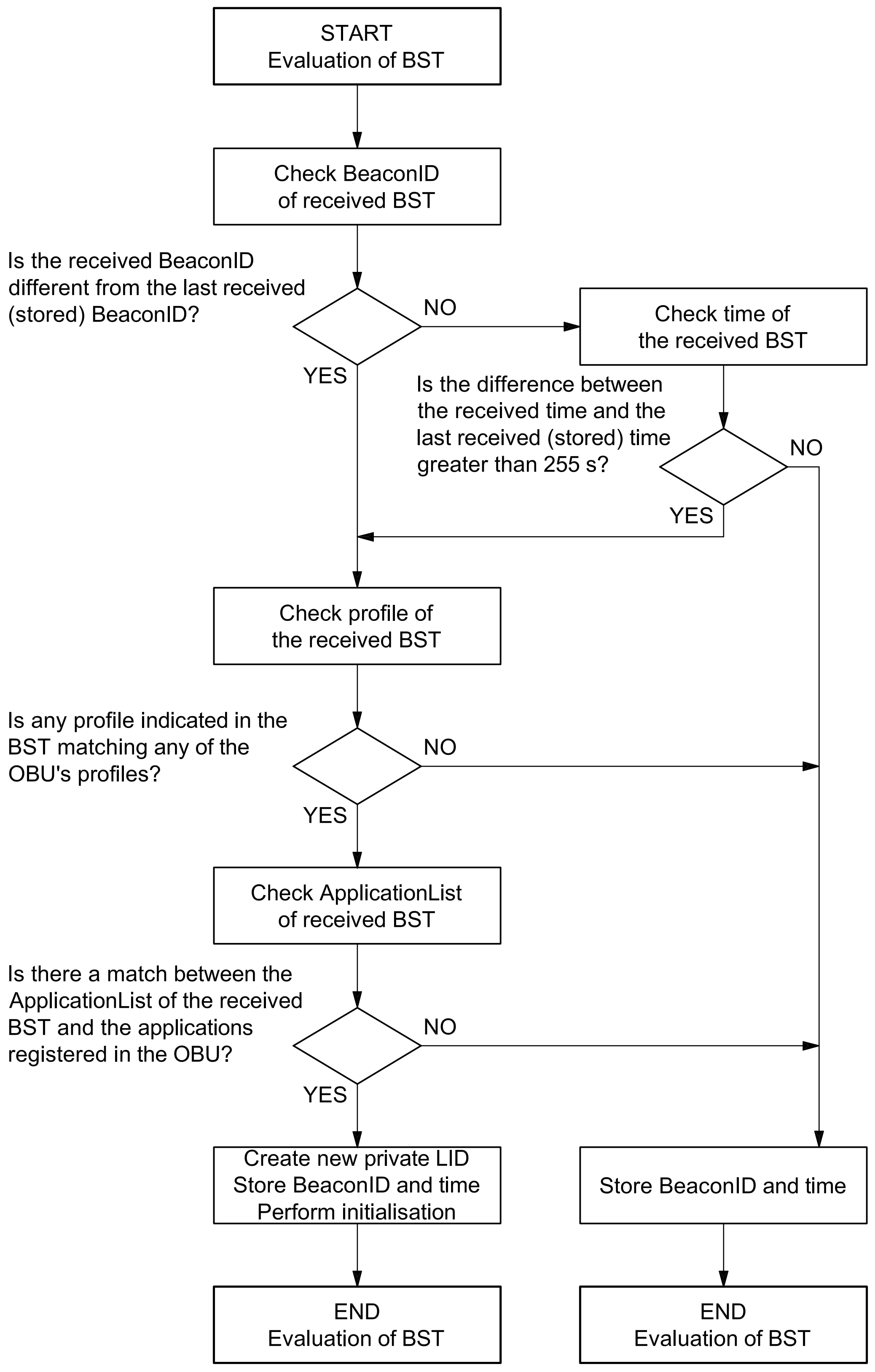
9.1 General

This clause specifies the extended definitions for this International Standard to conform to the different application interface (API) standards and/or the lower layer (data link layer) standards. A set of extended definitions could be identified as a “DSRC Profile”, which represents characteristics of the communication partners as described in Annex D, by applications. The profile may be able to be used for maintaining interoperability in applications.

9.2 Extended definitions

9.2.1 CEN

Extended definitions in CEN RTT DSRC standards (see related reference standards in 9.2.1.2) are described in this subclause. A data link layer link connection and an application layer initialisation are simultaneously executed in cooperation (see Figure 18).



NOTE The time may be calculated by the OBU from the time given in the BST.

Figure 18 — Evaluation of BST

9.2.1.1 Definitions

Details of extended definitions are as follows.

— Parameters of primitives defined in 6.2.3, described below, are mandatory.

— The FlowControl parameter of each GET.confirm, SET.confirm, ACTION.confirm and EVENT-REPORT.confirm is mandatory.

— The LID of INITIALZATION.request/indication defined in Table 6 shall be the broadcast address.

— Time between the last received BST and the current received BST defined in 7.3.2 shall be 255 seconds.

— The I-Kernel shall choose a random LID in VST transmission behaviour defined in 7.3.2.

— The EID of a SET.request service for the Broadcast Pool transmission defined in 8.3.1 shall be 0.

— Recovery/retransmission mechanisms are defined in the lower layer.

— ActionType definition and value assignments are available at http://www.nen.nl/cen278.

— Assignment of profile numbers according to Annex A in EN 12834 are based on the underlying definitions of DSRC profiles as defined in EN 13372.

NOTE The CEN (Comité Européen de Normalisation, European Committee for Standardisation) DSRC standard is a set of four standards; refer to http://www.cenorm.be.

9.2.1.2 Informative references

Informative references include CEN EN 12795:2003, CEN EN 13372:2004 and EN ISO 14906:2004.

9.2.2 IEEE/ASTM

Extended definitions in IEEE standards and ASTM standards (see related reference standards in 9.2.2.2) are described in this subclause.

NOTE 1 For IEEE (Institute of Electrical and Electronics Engineers), refer to http://www.ieee.org.

NOTE 2 For ASTM (American Society for Testing and Materials), refer to http://www.astm.org.

9.2.2.1 Definitions

Details of extended definitions are to be described.

9.2.2.2 Informative reference

This includes IEEE 1455:1999.

9.2.3 ARIB

Extended definitions in ARIB STD-T75:2001 are described in this subclause. In this ARIB DSRC standard, the BST/VST exchange for the application layer initialisation is made using PDUs after the data link layer link connection establishment. Its LID is a private address randomly chosen by the I-Kernel and it passes to the data link layer link with management services.

9.2.3.1 Definitions

Details of extended definitions are as follows.

— The FlowControl parameter of each GET.confirm, SET.confirm, ACTION.confirm and EVENT-REPORT.confirm is not applicable.

— The LID of INITIALISATION.request/indication defined in Table 6 shall be a private address (a randomly chosen LID).

— Adds FlowControl 12 defined in Table 16, DL-UNITDATA.request wait response request to Table 7.

Table 16 — Additional FlowControl parameter

Flow-Control	Application layer	LLC service
12	no flow control, answer with wait	DL-UNITDATA.request wait response request

— Adds Norm_end parameter defined in Table 17 to EndApplication of Table 13 in subclause 7.2.2, as optional. Norm_end indicates the EndApplication condition after completion of the application function. The I-Kernel may use this parameter to determine the value of the Timer T defined in 7.3.2.

Table 17 — Optional parameter of EndApplication

Parameter name	ASN.1 type	Condition
EID	Dsrc-EID	
LID	BIT STRING	
Norm_end	BOOLEAN	optional

T between the last received BST and the current received BST defined in 7.3.2 should be 0 through 255 seconds.

The I-Kernel shall use a private (randomly chosen) LID in VST transmission behaviour defined in 7.3.2.

The EID of a SET.request service for the broadcast pool transmission defined in 8.3.1 may be any number.

NOTE 1 The ARIB (Association of Radio Industries and Businesses, Japan) DSRC standard is one standard. Reference standards for this subclause are defined in 9.2.3.2. Refer to http://www.arib.or.jp or http://www.arib.or.jp/english/index.html.

NOTE 2 The timer value is an implementation parameter. If a timer value of more than 255 seconds is required, the maximum timer value might be changed.

NOTE 3 No implementation of the broadcast kernel functionality has become known according to this ARIB standard. Broadcast kernel functionality is to be considered further regarding the EID definition.

9.2.3.2 Informative references

These include ARIB STD-T75:2001 and ISO 14906:2004.

Annex A
(normative)

Data structures

A.1 Use of modules

This annex specifies a reference data structures defined in Clauses 6 to 8. Referring to this annex, a user of this International Standard shall specify the detail of data structures for a user system, and the T-Kernel in such a system shall use the DSRCData module and the DSRCTransferData module according to data structures.

Considering interoperability, the user should also register the data structures using an unambiguous identifier to the registration authority.

NOTE The IMPORT and EXPORT mechanisms are standardized in ISO 8824-1.

A.2 ASN.1 modules

```
DSRCData {iso(1) standard(0) dsrc(15628) dsrcData(0) version (1)}
DEFINITIONS AUTOMATIC TAGS::= BEGIN
              -- IMPORTS
              -- New type definitions shall be imported from other ASN.1 modules as
follows:
              -- Type1, Type2 FROM ModuleA;
              -- where:
              -- - Type1 and Type2 shall be replaced with the names of the types to be
imported.
              -- - ModuleA shall be replaced by the name of the exporting ASN.1 module
   -- EXPORTS everything;
Action-Request::=SEQUENCE{
              mode                    BOOLEAN,
              eid                     Dsrc-EID,
              actionType              ActionType,
              accessCredentials       OCTET STRING (SIZE (0..127,...))    OPTIONAL,
              actionParameter         Container   OPTIONAL,
              iid                     Dsrc-EID    OPTIONAL
              }
Action-Response::=SEQUENCE{
              fill                    BIT STRING (SIZE(1)),
              eid                     Dsrc-EID,
              iid                     Dsrc-EID         OPTIONAL,
              responseParameter       Container        OPTIONAL,
              ret                     ReturnStatus     OPTIONAL
              }
ActionType::=INTEGER(0..127,...)
```

```
             -- (0..118) Reserved for ISO/CEN use.
             -- ActionTypes below defined in ISO 14906
             --  0 : getStamped
             --  1 : setStamped
             --  2 : getSecure
             --  3 : setSecure
             --  4 : getInstance
             --  5 : setInstance
             --  6 : getNonce
             --  7 : setNonce
             --  8 : transferChannel
             --  9 : copy
             -- 10 : setMMI
             -- 11 : substract
             -- 12 : add
             -- 13 : debit
             -- 14 : credit
             -- 15 : echo
             -- (119-127) Reserved for private use
ApplicationContextMark::=Container          -- OCTET STRING (SIZE(0..127,...))
             -- Illustration of an ApplicationContextMark example
             -- can be found in ISO 14906, referred to as an EFC-ContextMark
ApplicationList::=SEQUENCE (SIZE (0..127,...)) OF
             SEQUENCE {
             aid         DSRCApplicationEntityID,
             eid         Dsrc-EID               OPTIONAL,
             parameter   ApplicationContextMark OPTIONAL
             }
AttributeIdList::=SEQUENCE (SIZE(0.. 127,...)) OF INTEGER(0..127,...)
AttributeList::=SEQUENCE (SIZE(0..127,...)) OF Attributes
Attributes::=SEQUENCE{
             attributeId       INTEGER (0..127,...),
             attributeValue    Container
             }
BeaconID::=SEQUENCE{
             manufacturerid    INTEGER(0..65535),
             individualid      INTEGER(0..134217727)
             } -- for registration of manufacturerid see www.nen.nl/cen278
BroadcastPool::=SEQUENCE{
             directoryvalue    Directory,
             content           SEQUENCE (SIZE(0..127,...)) OF File
             }
BST::=SEQUENCE{
             rsu                     BeaconID,
             time                    Time,
             profile                 Profile,
             mandApplications        ApplicationList,
             nonmandApplications     ApplicationList  OPTIONAL,
             profileList             SEQUENCE (SIZE(0..127,...)) OF Profile
             }
```

Container::=CHOICE{
```
        integer                 [0]   INTEGER,
        bitstring               [1]   BIT STRING,
        octetstring             [2]   OCTET STRING (SIZE (0..127, ...)),
        universalString         [3]   UniversalString,
        beaconId                [4]   BeaconID,
        t-apdu                  [5]   T-APDUs,
        dsrcApplicationEntityId[6]   DSRCApplicationEntityID,
        dsrc-Ase-Id             [7]   Dsrc-EID,
        attrIdList              [8]   AttributeIdList,
        attrList                [9]   AttributeList,
        broadcastPool           [10]  BroadcastPool,
        directory               [11]  Directory,
        file                    [12]  File,
        fileType                [13]  FileType,
        record                  [14]  Record,
        time                    [15]  Time,
        vector                  [16]  SEQUENCE (SIZE(0..255)) OF INTEGER(0..127,...),
                -- tags [17..69] are defined in ISO 14906 for CEN DSRC application
    use
                -- tags [70..86] are reserved for ISO/CEN DSRC application use
                -- tags [87..127] are reserved for private use and intended for the
                -- addressing of the corresponding private attribute identifiers.
                ...                 -- extension marker
                -- New attributes shall be inserted as:
                -- componentName1 [i] ModuleA.Type1
                -- where
                -- - componentName1 is a name unique within the Container definition
                -- - "i" is the registered tag chosen from the ranges as specified
    above.
                -- - Type1 is the name of an imported type and
                -- - ModuleA is the name of the module from which the type Type1 is
    imported.
                -- The prefix "ModuleA" is only required in case of a name conflict,
                -- if the name "Type1" is not also defined in the DSRCData module and
    not
                -- imported from another module, the prefix "ModuleA" should be
    omitted.
  }
```
Directory::=SEQUENCE (SIZE(0..127,...)) OF FileName
Dsrc-EID::=INTEGER(0..127, ...)

```
DSRCApplicationEntityID::=INTEGER{
            system                           (0),
            electronic-fee-collection        (1),
            freight-fleet-management         (2),
            public-transport                 (3),
            traffic-traveller-information    (4),
            traffic-control                  (5),
            parking-management               (6),
            geographic-road-database         (7),
            medium-range-preinformation      (8),
            man-machine-interface            (9),
            intersystem-interface            (10),
            automatic-vehicle-identification (11),
            emergency-warning                (12),
            private                          (13),
            multi-purpose-payment            (14),
            dsrc-resource-manager            (15),
            after-theft-systems              (16),
            cruise-assist-highway-system     (17),
            multi-purpose-information-system (18),
            multi-mobile-information-system  (19)
            -- (20..28) are reserved for ISO/CEN-dsrc-applications
            -- (29..30) are reserved for private use
            -- 31 is reserved for ISO/CEN-dsrc-applications
            }(0..31,...)
            -- For the latest standard use of application definitions,
            -- refer to Clause 9
            -- As an example, the application "electronic-fee-collection (1)"
            -- is standardized by ISO 14906
Event-Report-Request::=SEQUENCE{
            mode                   BOOLEAN,
            eid                    Dsrc-EID,
            eventType              EventType,
            accessCredentials      OCTET STRING (SIZE(0..127,...)) OPTIONAL,
            eventParameter         Container  OPTIONAL,
            iid                    Dsrc-EID   OPTIONAL
            }
Event-Report-Response::=SEQUENCE{
            fill       BIT STRING (SIZE(2)),
            eid        Dsrc-EID,
            iid        Dsrc-EID   OPTIONAL,
            ret        ReturnStatus     OPTIONAL
            }
EventType::=INTEGER{
            release     (0)
            -- (1..118) are reserved for ISO/CEN use
            -- (119..127) are reserved for private use
            }(0..127,...)
File::=SEQUENCE (SIZE(0..127,...)) OF Record
FileName::=SEQUENCE{
            aseID      Dsrc-EID,
            fileID     INTEGER(0..127,...)
            }
FileType::=NULL
                     -- Not defined. This might be defined in a future version.
Get-Request::=SEQUENCE{
            fill                   BIT STRING (SIZE(1)),
            eid                    Dsrc-EID,
            accessCredentials      OCTET STRING (SIZE(0..127,...)) OPTIONAL,
            iid                    Dsrc-EID         OPTIONAL,
            attrIdList             AttributeIdList  OPTIONAL
            }
Get-Response::=SEQUENCE{
```

```
             fill             BIT STRING (SIZE(1)),
             eid              Dsrc-EID,
             iid              Dsrc-EID         OPTIONAL,
             attributelist    AttributeList    OPTIONAL,
             ret              ReturnStatus     OPTIONAL
             }
Initialisation-Request::=BST
Initialisation-Response::=VST
NamedFile::=SEQUENCE{
             name       FileName,
             file       File
             }
                     -- NamedFile will be used in T-Kernel with GetBroadcastData-Request,
                     -- that might be specified in T-APDU in a future version.

ObeConfiguration::=SEQUENCE{
             equipmentClass   INTEGER(0..32767),
             manufacturerID   INTEGER(0..65535),
             obeStatus        INTEGER(0..65535) OPTIONAL
             }
Profile::=INTEGER (0..127,...)
             -- (0..118) are reserved for ISO/CEN use,
             -- (119..127) are reserved for private use
Record::=CHOICE{ simple VisibleString,
             ...
             }
ReturnStatus::=INTEGER{
             noError                (0),
             accessDenied           (1),
             argumentError          (2),
             complexityLimitation   (3),
             processingFailure      (4),
             processing             (5),
             chainingError          (6)
             -- (7..99) are reserved for future ISO/CEN use,
             -- (100..127) are reserved for private use
             }(0..127,...)
Set-Request::=SEQUENCE{
             fill                   BIT STRING (SIZE(1)),
             mode                   BOOLEAN,
             eid                    Dsrc-EID,
             accessCredentials      OCTET STRING (SIZE(0..127,...)) OPTIONAL,
             attrList               AttributeList,
             iid                    Dsrc-EID OPTIONAL
             }
Set-Response::=SEQUENCE{
             fill             BIT STRING (SIZE(2)),
             eid              Dsrc-EID,
             iid              Dsrc-EID OPTIONAL,
             ret              ReturnStatus OPTIONAL
             }
Time::=INTEGER(0..4294967295)
             -- The number of seconds passed since
             -- 1st January 1970, 00:00 (UTC)
T-APDUs::=CHOICE{
             action-request               [0]   Action-Request,
             action-response              [1]   Action-Response,
             event-report-request         [2]   Event-Report-Request,
             event-report-response        [3]   Event-Report-Response,
             set-request                  [4]   Set-Request,
             set-response                 [5]   Set-Response,
             get-request                  [6]   Get-Request,
             get-response                 [7]   Get-Response,
             initialisation-request       [8]   Initialisation-Request,
             initialisation-response      [9]   Initialisation-Response
             }
```

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

```
DSRCtransferData {iso(1) standard(0) dsrc(15628) dsrctransferData(1) version (1)}
DEFINITIONS::= BEGIN
   IMPORTS   T-APDUs
   FROM      DSRCData {iso(1) standard(0) dsrc(15628) dsrcData(0) version (1)};
   -- EXPORTS everything;
   Message::=              T-APDUs          -- Message is transferred over the DSRC link;
END
```

Annex B
(normative)

Naming and registration

B.1 Introduction

Several components of the DSRC system are identified by names. In order to ensure interoperability, it is essential that these names are used in an unambiguous way, and therefore registration of these names is required.

There are two different ways to control the unambiguity of names:

— registration of identifiers by the registration authority, and

— definition in application standards.

B.2 Items for registration

The following DSRC application layer component should be registered with a unique identifier according to registration procedures in ISO 14816:

— **manufacturerID:** manufacturerID is a worldwide unique identifier of manufacturers of DSRC equipment.

B.3 Items defined by application standards

It is up to the ISO/regional application standards described in Clause 9 to define and manage unambiguity for the following items. However, it is outside the scope of this International Standard to define items used by applications.

The following items are defined by DSRC application standards.

— **ActionType**: The ActionType shall be a universally unique identifier for an action.

— **AttributeID**: AttributeIDs shall be unique within the context of each element. Application standards may assign attributeIDs for applications covered by that standard.

— **EventType**: The EventType shall be a universally unique identifier for an event.

NOTE For reference information, NEN maintains a list (see http://www.nen.nl/cen278) of ActionType and EventType values assigned by DSRC application standards.

Annex C
(informative)

Example of coding

The following example shows application layer content of a BST/VST exchange, which is defined in Clause 7, for an electronic fee collection (EFC) application.

NOTE For EFC applications, ISO 14906 adds further detail to this International Standard.

Only DSRC application layer content is given. DSRC data link layer fields according to ISO/regional standards (LID, MAC, LLC, FCS, etc.) shall be added. Note that for the on-board unit to be able to transmit the VST, the unit must ask the RSU for a medium allocation. This medium access control functionality is part of the DSRC data link layer. The associated private window request (up link) and private window allocation (down link) messages are not shown here.

Table C.1 — BST example — One mandatory application (AID = 1 stands for EFC)

Octet #	Attribute/Field		Bits in Octet b_7 b_0	Description
0	Fragmentation header		1fff f001	No fragmentation. ffff: PDU number. Shall never be set to 0000_2 or 0001_2
1	BST	SEQUENCE	1000	INITIALISATION.request
	{			
	OPTION indicator		0	Nonmand applications not present
	BeaconID.ManufacturerId	INTEGER (0..65535),	mmm	(MSB) See list of manufacturers as formalised in ISO 14816 (CS2) and as managed by NEN
2			mmmm mmmm	
3			mmmm m	
	BeaconID.IndividualId	INTEGER(0..2^{27}-1),	iii	(MSB) 27 bit ID available for manufacturer
4			iiii iiii	
5			iiii iiii	
6			iiii iiii	
7	Time	INTEGER(0..2^{32}-1),	tttt tttt	(MSB) 32 bit real time
8			tttt tttt	The number of seconds passed since 1 January 1970, 00:00 (UTC)
9			tttt tttt	
10			tttt tttt	

11	Profile	INTEGER(0..127,...)	0ppp pppp	No extension, profile p
				p=0_{10}: 1,5 MHz subcarrier
				p=1_{10}: 2,0 MHz subcarrier

12	MandApplications	SEQUENCE(0..127,..) OF	0nnn nnnn	No extension, number of MandApplications. Only EFC application: n=1
	{			
13	OPTION indicator		0	EID not present
	OPTION indicator		0	Parameter not present
	AID	DSRCApplicationEntityID	00 0001	No extension, AID = 1 (EFC)
	}			
14	ProfileList	SEQUENCE(0..127,...) OF Profile	0000 0000	No extension, number of profiles in list = 0
	}			

Table C.2 — VST example – listing of the applications available on the mobile equipment

Octet #	Attribute/Field		Bits in Octet b_7 b_0	Description
0	Fragmentation header		`1fff f001`	No fragmentation. ffff: PDU number. Shall never be set to 0000_2 or 0001_2
1	VST	SEQUENCE	`1001`	INITIALISATION.response
	{			
	Fill	BIT STRING (SIZE(4))	`0000`	Set to 0
2	Profile	INTEGER(0..127,...)	`0ppp pppp`	No extension, profile p
3	Applications	SEQUENCE(0..127,...) OF	`0000 0001`	No extension, one application
4	{			
	OPTION indicator		`1`	EID present
	OPTION indicator		`1`	Parameter present
	AID	DSRCApplicationEntityID	`00 0001`	No extension, AID = 1 (EFC)
5	EID	Dsrc-EID	`eeee eeee`	Unique within the OBU and related to a context mark
6	Parameter	ApplicationContextMark	`0000 0010`	OCTET STRING = 2
7			`0000 0110`	No extension, octet string length = 6_{10}
8	EFC-ContextMark	SEQUENCE		EFC-Context-Mark is the EFC application specific content of the ApplicationContextMark, see ISO 14906
	{			
	ContractProvider	SEQUENCE		For value assignment see http://www.nen.nl/cen278
	{			
	CountryCode	BIT STRING(SIZE(10))	`0111 0001`	(MSB) 10 bit country code, according to ISO 3166 with ITA2 binary encoding based on ISO 14816 (CS1). The example given encodes Switzerland (CH).
9			`01`	
	IssuerIdentifier	INTEGER(0..16383)	`dd dddd`	(MSB) 14 bit issuer identifier d
10			`dddd dddd`	See ISO 14816 (CS1)
11	}			
	TypeOfContract	OCTET STRING (SIZE(2))	`tttt tttt`	(MSB) Type of contract
12			`tttt tttt`	
13	ContextVersion	INTEGER(0..127,...)	`0vvv vvvv`	No extension, context version v
	}			
	}			
14	ObeConfiguration	SEQUENCE		
	{			
	OPTION indicator		`1`	obeStatus present
	equipmentClass	INTEGER(0..32767,...)	`xxx xxxx`	(MSB)
15			`xxxx xxxx`	
16	manufacturerId	INTEGER(0..65535,...)	`mmmm mmmm`	(MSB) See ISO 14816 (CS2).
17			`mmmm mmmm`	
18	obeStatus	INTEGER(0..65535,...)	`ssss ssss`	Defined by manufacturer
19			`ssss ssss`	
	} }			

NOTE ISO 14906 provides a more detailed example in its Annex B.

Annex D
(normative)

Declaration of application layer features supported

For each item of DSRC equipment for which conformance to this International Standard is claimed, it shall be declared which features according to Table D.1 are implemented.

The profile represents features (characteristics) of the communication partners and should be contained in other layer features (parameters), since each layer protocol has interactions in relation to another layer feature. The profiles defined in Annex A may be handled by the I-Kernel, as described in Clause 8, and are transmitted inside the BST. When instances of an OBU enter in the DSRC communication zone of an RSU, the BST and VST are exchanged between these units. The RSU and OBU will notify profile(s) that the OBU and the RSU have done so, using VST and BST to negotiate for communication profile.

Table D.1 — Form for declaration of application layer features

Feature(s)			Status	Option implemented
T-Kernel	Fragmentation/defragmentation		Optional/mandatory	Yes ❑ No ❑ n.a. ❑
	Concatenation/deconcatenation		Optional/mandatory	Yes ❑ No ❑ n.a. ❑
	Multiplexing/demultiplexing		Optional/mandatory	Yes ❑ No ❑ n.a. ❑
	Fragmentation header	1 octet	Mandatory	Yes ❑ No ❑
		2 octet	Optional/mandatory	Yes ❑ No ❑ n.a. ❑
		3 octet	Optional/mandatory	Yes ❑ No ❑ n.a. ❑
	Service primitives	GET	Optional	Yes ❑ No ❑ n.a. ❑
		SET	Optional	Yes ❑ No ❑ n.a. ❑
		ACTION	Optional	Yes ❑ No ❑ n.a. ❑
		EVENT-REPORT	Mandatory	Yes ❑ No ❑
		INITIALISATION	Mandatory	Yes ❑ No ❑
I-Kernel			Optional/mandatory	Yes ❑ No ❑ n.a. ❑
B-Kernel			Optional	Yes ❑ No ❑
Timer T (seconds)			255/0…255	
LID for INITIALISATION.request			$1111\ 1111_2$ / Private LID	
If a value of “Timer T” is more than 255 seconds or not specified, fill the column with an exact value (time) or “n.a.” (not applicable).				

Annex E
(informative)

Lower layer services

E.1 Service definitions

E.1.1 General

The application layer requires a service interface to the lower layer services. The specific syntax and semantics of the implementation are outside the scope of this International Standard. For this reason, the service definitions are in generic terms rather than specific service primitives. However, any conformant lower layer service must provide services that correspond to each of the generic functional services defined in E.1.2. Additional lower layer services may also be provided, such as the periodic retransmit function. Implementation of each of these services does not make any implication regarding the relationship between the initiating equipment making the initial request and the responding equipment. These functions may be implemented in a peer-to-peer or a master-slave relationship.

E.1.2 Lower-layer-provided functions

E.1.2.1 General

To a service user, two forms of services are provided: unacknowledged connectionless data transfer services and acknowledged connectionless data transfer services. A flow control parameter may be used to select the appropriate lower layer service as follows.

a) Unacknowledged connectionless data transfer services: This set of data transfer services provides the means by which the service user (application layer) can exchange data unit(s) without establishment of a lower layer level connection on an unacknowledged basis. The data transfer can be point-to-point, multi-point, or broadcast.

b) Acknowledged connectionless data transfer services: The acknowledged connectionless data unit exchange services provide the means by which the service user (application layer) can exchange data unit(s) that are acknowledged at the lower layer without establishment of a lower layer level connection. The services provide a means by which an application layer at one item of equipment can send a data unit to another item of equipment, request a previously prepared data unit from responding equipment, or exchange data unit(s) with another station.

The following functional services should be provided to the application layer from the lower layer in order to perform the above services.

1) Data transfer — Lower layer sends data unit(s) containing information data from the local application layer to remote responding equipment. The unacknowledged connectionless data transfer service does not expect a response from the remote responding equipment. The acknowledged connectionless data transfer service expects a response from the remote responding equipment. The data unit may or may not require fragmentation.

2) Data reception — Responding equipment lower layer sends data unit(s) containing response data from the associated application layer to the initiating equipment. This applies to acknowledge connectionless data transfer services.

3) Response data transfer — Responding equipment lower layer sends data unit(s) containing response data from the associated application layer to the initiating equipment. This applies to acknowledge connectionless data transfer services.

4) Status response — Remote peer responding equipment responds with a status indication of the associated data unit(s). A status request may be made without an accompanying information data field. The response data unit may not have an information data field. The status information should contain the following status:

   — whether or not a response is sent in response to a related request; and

   — whether or not the requested response is available.

5) Layer 2 acknowledgement (ACK) — Layer 2 (data link layer) level acknowledgement. This represents a request that the data link layer performs an acknowledgement function when the data unit to which it applies is transmitted. A Layer 2 acknowledgement may consist of a transmitted response data unit generated by the data link layer in response to a transmission data unit that includes a Layer 2 acknowledgement request. The acknowledgement response data unit may indicate whether or not a data unit, with a valid FCS (frame check sequence), was received correctly. If the data unit was received correctly, the response may be a positive acknowledgement, or “ACK” (see Figure E.3).

The following optional function is allowed:

6) Periodic retransmit — This is a special form of the transferring data unit(s) service in which the data link periodically repeats the data unit transmission. The repetition rate must be provided to the lower layer. A repetition rate of zero serves to terminate transmissions. When a response is requested, such a response will occur only when provided by receiving peer equipment, which may not occur with every transmission.

7) Address determination — The lower layer responds to differences between public and private addressing.

E.1.2.2 Generic services

Generic services are the capabilities offered to a service user (application layer L7) from a service provider [the lower layer (data link layer L2)] to provide functional services as information data to transfer from the initiating equipment to the remote equipment or back (see Table E.1).

Table E.1 — Generic services

Service name	**Functional services**
request	A primitive is invoked by the service user (L7) and passed to the service provider (L2) to request services, such as transferring data unit(s).
indication	A primitive is issued by the service provider (L2) and passed to the service user (L7) to indicate that an internal lower layer action may be logically related to a remote service provider request, or may be caused by an event internal to the lower layer, i.e. notification from the service provider (L2) to the local service user (L7) that a request has been received from a peer remote service user (L7) and/or pass the received data unit and/or a status.
response	A primitive is invoked by the service user (L7) and passed to the service provider (L2) to acknowledge or complete some procedure invoked by an indication to a service user, i.e. transferring a response data unit back to the initiating peer service user (L7) from the local service user (L7).
confirm	A primitive is issued by the service provider (L2) and passed to the service user (L7) to convey the results of one or more associated previous service request(s) invoked by the initiating service user (L7), i.e. notification from the initiating service provider (L2) to the initiating service user (L7) that a response data unit and/or a status has been received.

Figure E.1 illustrates the relationship (sequence diagram) between these logical services.

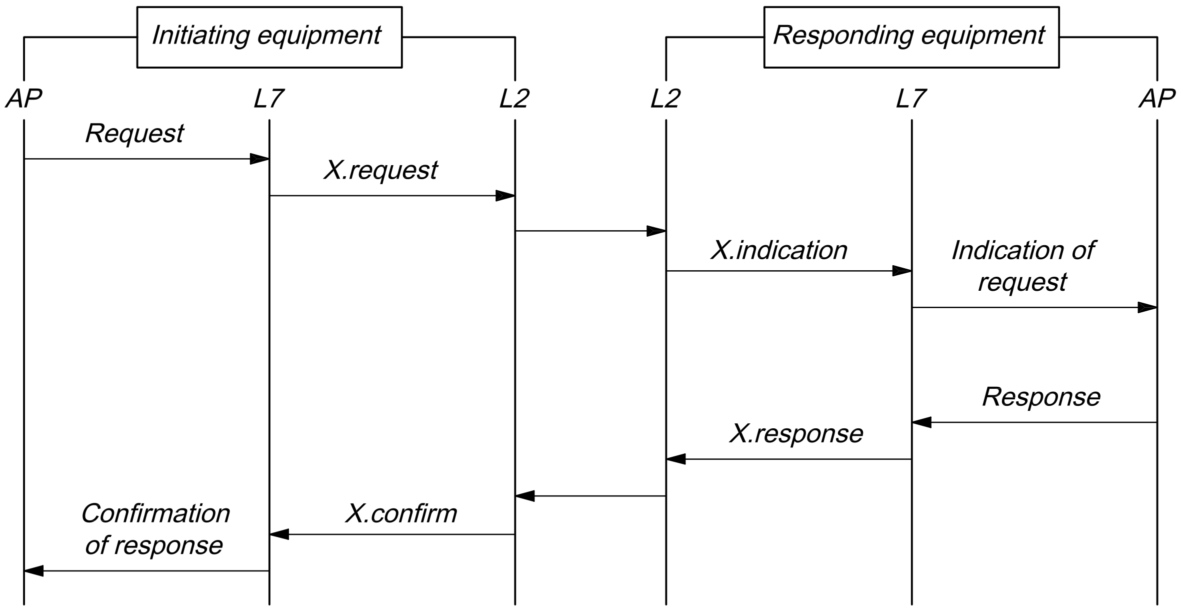


Figure E.1 — Relationship between generic services

E.2 Lower layer service primitives

The previously described services may be provided using specific service primitives. Examples of such primitives include the following.

a) The primitives have two types of services, request and/or indication. These include

— DL-UNITDATA.request without response request,

— DL-UNITDATA.request with response request,

— DL-UNITDATA.request wait response request,

— DL-UNITDATA.indication,

— DL-DATA-ACK.request,

— DL-DATA-ACK.indication,

— DL-DATA-ACK-STATUS.indication,

— DL-REPLY.request,

— DL-REPLY.indication,

— DL-REPLY-STATUS.indication,

— DL-REPLY-UPDATE.request, and

— DL-REPLY-UPDATE-STATUS.indication.

b) The primitives have the three types of services: request, indication, and confirm. These include

— DATASEND_NORESPOND, and

— DATASEND_NORESPOND_REPEAT.

c) The primitives have the full set of service types; request, indication, response, and confirm. These include

— DATASEND_RESPOND,

— DATASEND_RESPOND_REPEAT,

— SEND_BST_RESPOND, and

— SEND_BST_RESPOND_REPEAT.

NOTE In principle, primitives have specific FlowControl parameters defined in 6.2.4 and Clause 9.

E.3 Examples of lower layer service interactions

Examples of using the lower layer services are shown in Figures E.2 and E.3.

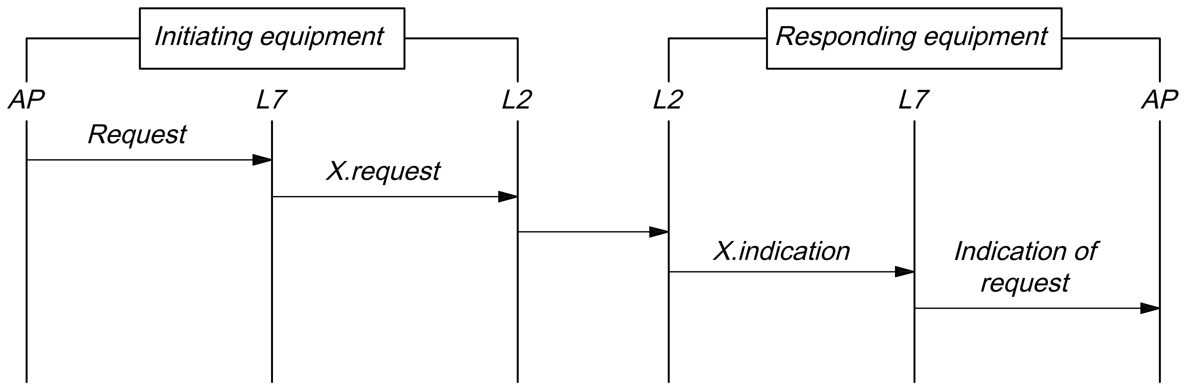


Figure E.2 — An unacknowledged data transfer service

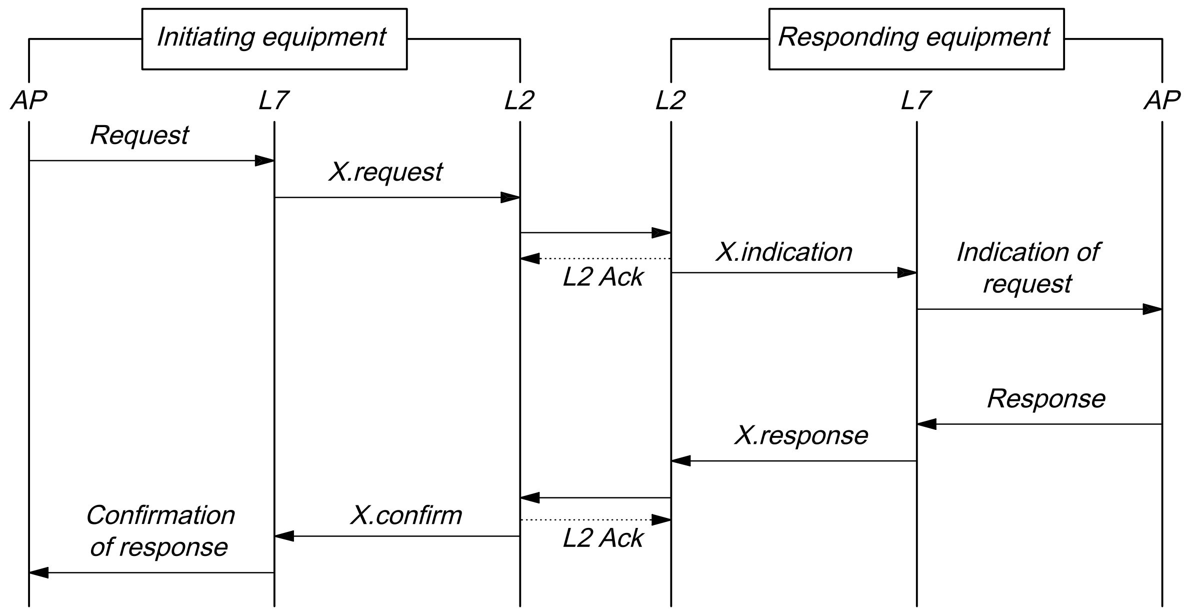


Figure E.3 — An acknowledged data transfer service

Bibliography

[1] ISO 14906, *Road transport and traffic telematics — Electronic fee collection — Application interface definition for dedicated short-range communication*

[2] CEN EN 12795:2003, *Road transport and traffic telematics — Dedicated short range communication (DSRC) — DSRC data link layer: medium access and logical link control*

[3] CEN EN 13372:2004, *Road transport and traffic telematics (RTTT) — Dedicated short-range communication — Profiles for RTTT applications*

[4] EN ISO 14906:2004, *Road transport and traffic telematics — Electronic fee collection — Application interface definition for dedicated short-range communication*

[5] IEEE 1455:1999, *IEEE Standard for message sets for vehicle/roadside communications*

[6] ARIB STD-T75:2001, *Dedicated short-range communication systems*

ICS 03.220.01; 35.240.60

Price based on 49 pages