IEEE Standard for Message Sets for Vehicle/Roadside Communications

Sponsor

Rail Transit Vehicle Interface Standards Committee
of the
IEEE Vehicular Technology Society

Reaffirmed 7 June 2006
Approved 26 June 1999

IEEE-SA Standards Board

Abstract: Those characteristics of a dedicated short-range communications (DSRC) system that are independent of the Physical and Data Link Layers (ISO model Layers 1 and 2) are specified. The required and optional features of the roadside equipment (RSE) and the onboard equipment (OBE) are specified. In addition, the Applications Layer (ISO model Layer 7) services and protocols, the RSE resource manager, the corresponding OBE command interpreter, and the application-specific messages are all specified. Standard supports and guidelines are provided for implementing secure DSRC systems.
Keywords: access control, application layer, automatic vehicle identification, AVI, beacon, Commercial Vehicle Operations, CVO, data authentication, data privacy, data security, dedicated short-range communications, DSRC, electronic toll collection, ETC, Layer 7, Mailbox, reader, resource manager, smart card, tag, transponder, vehicle to roadside communications, VRC

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Print: ISBN 0-7381-1769-2 SH94768
PDF: ISBN 0-7381-1770-6 SS94768

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Introduction

(This introduction is not part of IEEE Std 1455-1999, IEEE Standard for Message Sets for Vehicle/Roadside Communications.)

The electronic toll and traffic management (ETTM) field is rapidly evolving, and ETTM systems across North America use a variety of incompatible dedicated short-range communications (DSRC) products. Several mainline screening systems for commercial vehicle operations (CVOs) are now in operation and likewise use incompatible data definitions. In addition, the acceptance of the ASTM Standard for DSRC Layers 1 and 2 will create the potential for significant interference between radio frequency (RF) compatible systems.

The lack of standard message sets, lower layer protocols, and transponder resource definition threatens to postpone the deployment of DSRC systems, a technology critical to the intelligent transportation systems (ITS) marketplace. This standard consolidates and builds upon previous application-specific data definitions to support CVO, ETTM, and private DSRC usage. The open nature of this standard ensures that it is extensible to other DSRC areas, such as CVO credential information exchange, preclearance screening, fleet management, and vehicle condition monitoring.

While this standard has been developed to ensure efficient operation when used in conjunction with the ASTM Layer 1 and Layer 2 standards, it is not limited to that application. This standard may be use in conjunction with a variety of lower layer protocols, including both RF and other communications media.

At the time this standard was developed, the Working Group on Dedicated Short-Range Communications Applications for Intelligent Transportation Systems had the following membership:

Peter B. Houser, *Chair*

Frederico Alvarez	Joe Crabtree	David Newman
Lee Armstrong	Ronald F. Cunningham	Roger J. O'Conner
Charles Gary Belda	Ron DeLeon	Mike Onder
Hamed Benouar	Neil Gray	Satoshi Oyama
Broady Cash	Ed Grose	Tom Rieflin
Alex Castro	Martha Harrell	Lewis Sabounghi
Amos Chenoweth	George Herndon	Ove Salomousson
Stan Ciszewski	Chuck Johnson	Richard Schnacke
Clayton Collier	David Martin	Bo Strickland
Ken Cook	Russell McCarty	Robert Tiernay
Roy Courtney	Irwin Morse	Ray Yuan

The following members of the balloting committee voted on this standard:

Robert Barrett	Clayton Collier	Russell McCarty
James Bartol	Roy Courtney	Roger J. O'Connor
Charles Gary Belda	Joe Crabtree	Lewis Sabounghi
Hamed Benouar	Ronald F. Cunningham	Richard Schnacke
Amos Chenoweth	Richard W. Doering	Ray Yuan
	Peter B. Houser	

When the IEEE-SA Standards Board approved this standard on 26 June 1999, it had the following membership:

Richard J. Holleman, *Chair* **Donald N. Heirman,** *Vice Chair*
Judith Gorman, *Secretary*

Satish K. Aggarwal
Dennis Bodson
Mark D. Bowman
James T. Carlo
Gary R. Engmann
Harold E. Epstein
Jay Forster*
Ruben D. Garzon
James H. Gurney
Lowell G. Johnson
Robert J. Kennelly
E. G. "Al" Kiener
Joseph L. Koepfinger*
L. Bruce McClung
Daleep C. Mohla
Robert F. Munzner
Louis-François Pau
Ronald C. Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Akio Tojo
Hans E. Weinrich
Donald W. Zipse

*Member Emeritus

Also included is the following nonvoting IEEE-SA Standards Board liaison:

Robert E. Hebner

Contents

IEEE Standard for Message Sets for Vehicle/Roadside Communications

1. Overview

1.1 Scope

This standard is applicable to dedicated short-range communications (DSRC). Within the overall context of DSRC operations (illustrated in Figure 1), this standard governs the communications protocols above the open systems interconnection (OSI) data link layer for the DSRC wireless interface. This standard does not specify the interface between the transponder and other pieces of onboard equipment (OBE) (identified as "5" in Figure 1), nor does it specify the interface between the roadside equipment (RSE) and the back office equipment (BOE) that hosts an intelligent transportation systems (ITS) back office application (BOA) (identified as "1" in Figure 1). However, to obtain desirable levels of DSRC performance, it is essential that those interfaces be defined in a manner that is consistent with this standard.

Interface 2 in many systems may be the interface between the application layer and lower layer service. Nothing precludes combining a vehicle-to-roadside communication (VRC) controller, a reader, and an antenna into a single piece of equipment.

These communications protocols have been specified by defining the message set and data dictionary, the low-level commands used to control transponder resources and thereby access such messages, and the OSI application layer services used to connect the upper interface elements with the OSI data link layer elements. This standard also specifies the resources that may be present on a transponder and the means by which the RSE shall control those resources.

Product compliance with this standard shall be verified based upon tests of Interfaces 1 and 4 shown in Figure 1. Internal interfaces such as 2 are at the discretion of the equipment vendor and shall not be considered when determining product compliance.

1.2 Purpose

This standard is intended to be used in several ways by various DSRC equipment stakeholders:

— The standard specifies the internal resources that shall be provided within and commands that shall be recognized by compliant transponders. This information will be used by transponder manufacturers when developing compliant transponders. This information will also be used by RSE manufacturers when developing equipment that communicates with such transponders.

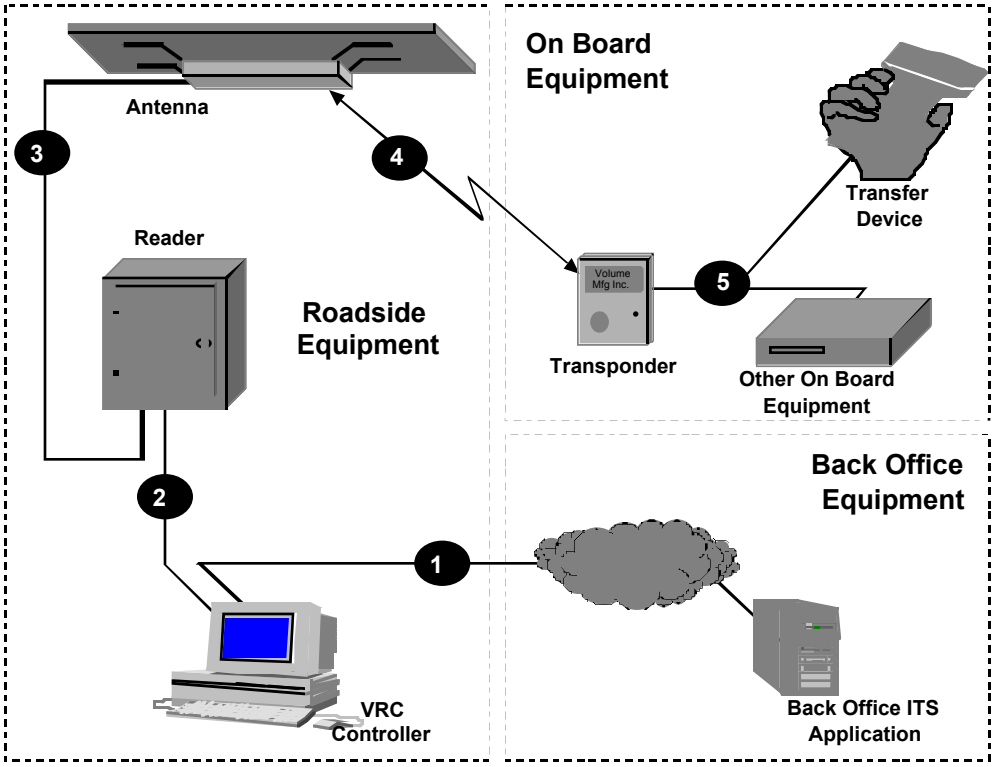

Figure 1—DSRC system interfaces

— This standard specifies the Layer 7 protocols that shall be used to complete the communications stack between the data link layer and higher level processing. This information will be used by transponder and beacon manufacturers when developing compliant equipment.

— This standard specifies the manner in which roadside DSRC equipment shall manage transponder resources. This information will be used by manufacturers of such RSE when developing software that utilizes transponder resources.

— This standard specifies the messages that shall be used to communicate information between ITS applications using DSRC. This information will be used by ITS application developers when they design and implement systems that use DSRC.

— This standard specifies the manner in which new systems achieve backward compatibility or noninterference with existing systems.

This standard specifies an overall structure for DSRC. System components may be selected within the compliant range allowed within this standard. For example, the user agencies may select specific messages from Clause 8. To achieve interoperability between agencies, it is essential that the user agencies enter into institutional agreements that coordinate their designs.

1.3 Standards maintenance

This standard provides three categories of data elements that have values with specific meanings within the context of the standard.

— *Available for uncontrolled use*. Values assigned to data elements in this category are not controlled by this standard and may be defined by the vendor or user.

— *Available for registration*. Values assigned to data elements in this category have been preallocated within this standard for future assignment to a specific agency or vendor.

  Currently, ranges of available values are listed in this standard for specific uses. An agency or vendor must submit a request to the IEEE to have a specific value, from a listed range of available values, designated as uniquely registered to that agency or vendor.

— *Reserved*. Values assigned to data elements in this category have been predefined within this standard to have specific meanings. These predefined values shall not be assigned to those data elements as having any other meaning.

A maintenance process associated with this standard will be required to control data element registration and coordinate clarifications, corrections, or modifications to this standard. This maintenance process will be defined by the IEEE.

1.4 Interoperability and noninterference

Four possible relationships between equipment and this standard exist:

— *Compliant equipment*. Transponders and RSE that meet all requirements of this standard are defined as compliant. Such equipment may omit some or all of the optional features defined in this standard.

— *Interoperable transponder.* Transponders that meet all requirements of this standard and provide the set of optional features identified in 5.3 and commands identified in 6.6 are defined as interoperable. This class of equipment is intended to specify transponders that may be used by multiple BOAs.

— *Interoperable RSE*. RSE that meets all requirements of this standard and includes a resource manager that services multiple BOAs is defined as interoperable.

— *Noninterfering equipment*. Transponders and RSE that substantially meet all requirements of this standard, but omit required capabilities in a manner that is transparent to compliant equipment, are defined to be noninterfering. For example, a noninterfering transponder may utilize a page of memory in a manner that does not comply with the Clause 8 message definitions as long as that memory page is protected against access by other compliant RSE. This class of equipment is intended to facilitate backward compatibility during system migration towards this standard.

1.5 Precedence of data representation

This standard contains cases where a specified data element includes both an Abstract Syntax Notation One (ASN.1) representation and a binary representation of that data element. If any conflict arises between the ASN.1 and binary representation of a given data element, the binary representation shall be considered normative and shall take precedence over the ASN.1 definition.

2. References

The following documents shall be used, when applicable, in the process of developing equipment and systems that will be compliant with the DSRC standard. When the following documents are superseded by an approved revision, then that revision shall apply.

ANSI X3.38-1988 (R1994), Codes—Identification of States, the District of Columbia, and the Outlying and Associated Areas of the United States for Information Interchange.[1]

[1]ANSI publications are available from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (http://www.ansi.org/).

ASTM Draft Standard, Dedicated Short-Range Communication (DSRC) Data Link Layer, E xxx-99, Version 5.0.[2]

CEN Draft Document: prENV278/9/#65 Dedicated Short Range Communication (DSRC)—Application Layer (Layer 7).[3]

IEEE Std 1489-1999, IEEE Standard for Data Dictionaries for Intelligent Transportation Systems—Part 1: Functional Area Data Dictionaries.[4]

GSS Global Specification For Short Range Communication. The platform for Interoperable Electronic Toll Collection and Access Control.[5]

ISO 3166-1:1997, Codes for the representation of names of countries and their subdivisions—Part 1: Country codes.[6]

ISO 3779:1983, Road vehicles—Vehicle identification numbering (VIN)—Content and structure.

ISO/IEC 7498-1:1994 Information technology—Open Systems Interconnection—Basic Reference Model: The Basic Model

ISO 7498-2:1989 Information processing systems—Open Systems Interconnection—Basic Reference Model—Part 2: Security Architecture

ISO/IEC 7498-3:1997 Information technology—Open Systems Interconnection—Basic Reference Model: Naming and addressing

ISO/IEC 7498-4:1989 Information processing systems—Open Systems Interconnection—Basic Reference Model—Part 4: Management framework

ISO 3780:1983, Road vehicles—World manufacturer identifier (WMI) code.

ISO/IEC 8824-1:1995, Information technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation.

ISO/IEC 8825-2:1996, Information technology—ASN.1 encoding rules: Specification of Packed Encoding Rules (PER).

ISO TC204 WG15 Committee Of Japan TICS/DSRC—DSRC Application Layer High Data Rate.

3. Definitions, abbreviations, and acronyms

For the purposes of this standard, the following terms, definitions, abbreviations, and acronyms apply. For terms not defined in this clause, English words are used in accordance with their definitions in the latest

[2]ASTM publications are available from the American Society for Testing and Materials, 100 Barr Harbor Drive, West Conshohocken, PA 19428-2959, USA (http://www.astm.org/).

[3]CEN publications are available from Nederlands Normalisatie-Instituut (NNI), Attn: Mr. J. A. Dijkstra, Kalfjeslaan 2, P.O. Box 5059, 2600 GB Delft, The Netherlands; telephone +31 15 2 690127; telefax. +31 15 2 690 242; jelte.dijkstra@nni.nl.

[4]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (http://www.standards.ieee.org/).

[5]GSS publications are available from Global Specification for Short-Range Communication.

[6]ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (http://www.ansi.org/).

editions of *Webster's New Collegiate Dictionary.* Electrical and electronic terms not defined in this clause or in *Webster's New Collegiate Dictionary* are used in accordance with their definitions in IEEE Std 100-1996.

3.1 Definitions

3.1.1 access credentials: Data that are transferred to establish the claimed identity of a roadside equipment (RSE) application.

3.1.2 antenna: A device used to send or receive radio waves.

3.1.3 application identifier (AID): An identifier that defines the category of dedicated short-range communications (DSRC) applications to which a specific application belongs. The only category currently defined by this standard is Mailbox.

3.1.4 back office application (BOA): Intelligent transportation system (ITS) or other application that resides and executes on the back office equipment (BOE). BOAs exchange messages with the onboard equipment (OBE) via the resource manager.

3.1.5 back office equipment (BOE): Computer equipment that hosts the applications and data required for some intelligent transportation system (ITS) function. Data is exchanged with the OBE via the vehicle-to-roadside communications (VRC) controller.

3.1.6 beacon: A roadside system at which dedicated short-range communications (DSRC) can be accomplished. A beacon typically consists of a reader and an antenna.

3.1.7 beacon service table (BST): A data structure created and transmitted by a beacon. It contains data such as the application identifier (AID) relevant for initiating communication with an onboard equipment (OBE) transponder. The reception of the BST by an OBE transponder results in a vehicle service table (VST) being sent back to the beacon.

3.1.8 broadcast mode: Beacon-initiated transmissions that are intended for all onboard equipment (OBE) in the communications zone.

3.1.9 command: A package of information transmitted from the roadside to the vehicle that requests that the transponder on the vehicle perform a specific action.

3.1.10 communications zone: The area of space within which a beacon can communicate with transponders in or on passing vehicles.

3.1.11 credentials: Information supplied to authenticate a communication.

3.1.12 digital signature: Information supplied with a transmission that allows the receiver to verify that the transmission has not been modified after initial generation.

3.1.13 element identifier (EID): A numeric identifier generated by the application on the onboard equipment (OBE) transponder. An EID is unambiguous within the context of a complete exchange of messages with an application on the roadside equipment (RSE). Multiple EIDs can be maintained between the RSE and OBE over a single data link session (LID).

3.1.14 invoker identifier (IID): The specific element identifier (EID) of a responding transponder.

3.1.15 kernel element (KE): An abstraction of data processing and data communication resources. An application KE represents a logical component of application layer functionality.

3.1.16 link identifier (LID): A string identifier generated by the lower layer service on the onboard equipment (OBE) transponder each time the transponder initiates a new session with a lower layer service on the roadside equipment (RSE).

3.1.17 lower layer service: A generic service required by this standard that provides the minimum subset of the functionality defined by Layers 4 through 1 of the open systems interconnection (OSI) model.

3.1.18 mailbox: The mechanism for storing and retrieving intelligent transportation systems (ITS) messages between the back office equipment (BOE) and onboard equipment (OBE). The specific processing mechanisms are defined by the resource manager.

3.1.19 memory image: A series of bits that can be stored within a contiguous portion of transponder memory and that may be passed as a parameter within commands initiated by the roadside equipment (RSE).

3.1.20 memory page: A segment of transponder memory that is assigned a unique location by which it may be referenced.

3.1.21 message: A package of information meeting a standard format that is sent to or from a transponder's memory.

3.1.22 nonce: A random or pseudo-random value used within an authentication system.

3.1.23 onboard equipment (OBE): Equipment located within a vehicle that supports the information exchange with roadside equipment (RSE).

3.1.24 private page: A memory page that has been reserved for the exclusive use of a particular agency or function. Access credentials may be required to reference the page.

3.1.25 profile: A unique numeric identifier that indicates a set of communications parameter values to be used in establishing a physical session.

3.1.26 public page: A memory page that may be used by any agency or application. No access credentials are associated with such a page.

3.1.27 reader: A component of a roadside beacon that provides the capabilities for radio wave communications with a transponder.

3.1.28 roadside equipment (RSE): Equipment located at a fixed position along the road transport network, providing communication and data exchange with the onboard equipment (OBE).

3.1.29 service data unit (SDU): The data associated with a service primitive.

3.1.30 service primitive: A function in the external interface of a kernel element (KE) that may be invoked to access services provided by the KE.

3.1.31 session: A series of transactions exchanged between the roadside and the vehicle while the vehicle is within a beacon's communications zone.

3.1.32 transaction: A functionally continuous and complete exchange of information between the roadside equipment (RSE) and the vehicle transponder.

3.1.33 transponder: The onboard component of a dedicated short-range communications (DSRC) system. Transponder elements are defined in Clause 5.

3.1.34 vehicle service table (VST): The answer of the onboard equipment (OBE) application layer in response to a received beacon service table (BST). The VST contains the identifier of applications supported by the OBE and profiles used for further point-to-point communication.

3.2 Abbreviations and acronyms

AID	application identifier
ASDU	application service data unit
ASN.1	Abstract Syntax Notation One
ASTM Draft 7	ASTM Draft Standard Dedicated Short Range Communications (DSRC) Data Link Layer: Medium Access and Logical Link Control Exxx-97.
B-KE	broadcast kernel element
BCD	binary coded decimal
BLOB	binary large object
BOA	back office application
BOE	back office equipment
BSDU	broadcast service data unit
BST	beacon service table
CEN	Comité Européen de Normalisation
CMV	commercial motor vehicle
CV	commercial vehicle
CVO	commercial vehicle operations
DSRC	dedicated short-range communications
DUNS	Dun & Bradstreet number
EID	element identifier
ETC	electronic toll collection
ETTM	electronic toll and traffic management
FIFO	first in first out
GSS	Global Specification for Short Range Communication
Hex	hexadecimal
ICC	Interstate Commerce Commission
IID	invoker identifier
I-KE	initialization kernel element
ISDU	initialization service data unit
ITS	intelligent transportation systems
KE	kernel element
LED	light emitting diode
LID	link identifier
LLC	logical link control
OBE	onboard equipment
OSI	open systems interconnection
PDA	personal digital assistant
PDU	protocol data unit
PER	Packed Encoding Rules

RF	radio frequency
RSE	roadside equipment
SDU	service data unit
T-KE	transfer kernel element
TSDU	transfer service data unit
UI	user interface
VRC	vehicle-to-roadside communications
VST	vehicle service table

4. Architecture and communications flow summary

Figure 2 illustrates this standard's DSRC architecture and the relationship between this standard and related standards. Within this architecture this standard governs

— The configuration of transponder resources (primarily specified in Clause 5).

— The commands to which the transponder responds (primarily specified in Clause 6).

— The Layer 7 protocols used to connect the transponder resources (on the vehicle) and the resource manager (on the roadside) with the lower layer protocols that physically accomplish the information transfer (primarily specified in Clause 9).

— The resource manager that manages and controls transponder resources in response to information received from BOAs (primarily specified in Clause 7).

— The messages used by BOAs to exchange information using DSRC (primarily specified in Clause 8).

4.1 DSRC context and processing flow

In the context of the architecture shown in Figure 2, and using the typical DSRC equipment configuration shown in Figure 1, the DSRC processing flow is typically as follows [use of the beacon service table (BST) and vehicle service table (VST) are fully defined in Clause 9]:

a) Compliant RSE and OBE, including beacons and transponders, are installed as appropriate. As illustrated in Figure 2, the complementary Layer 1, Layer 2, and Layer 7 processing in the RSE and OBE provides the basic structure for higher level information transfers.

b) At the time of installation, information such as the identifier registered to that service agency, may be written into the transponder's memory. It is anticipated that this transfer will typically be done by a dedicated beacon at a central facility, using the transponder commands defined in Clause 6.

c) Handheld data transfer devices or onboard computers may also store messages in areas of the transponder's memory. The interface for accomplishing this transfer is not controlled by this standard.

d) In transponders that support the optional memory-partitioning feature, memory may be partitioned by the vendor or the user agency. Subsequent memory page reservations may reference a defined partition.

e) In transponders that support access control, a memory page may be reserved by an agency. Such pages are referred to as "private pages," and all other pages are referred to as "public pages." This reservation may require that the requesting agency provide access controls and will result in a region of memory being permanently allocated for use exclusively by the agency. Access controls may be used to enforce this exclusivity. It is anticipated that this reservation will typically be done by a dedicated beacon at a central facility, using the transponder commands defined in Clause 6.

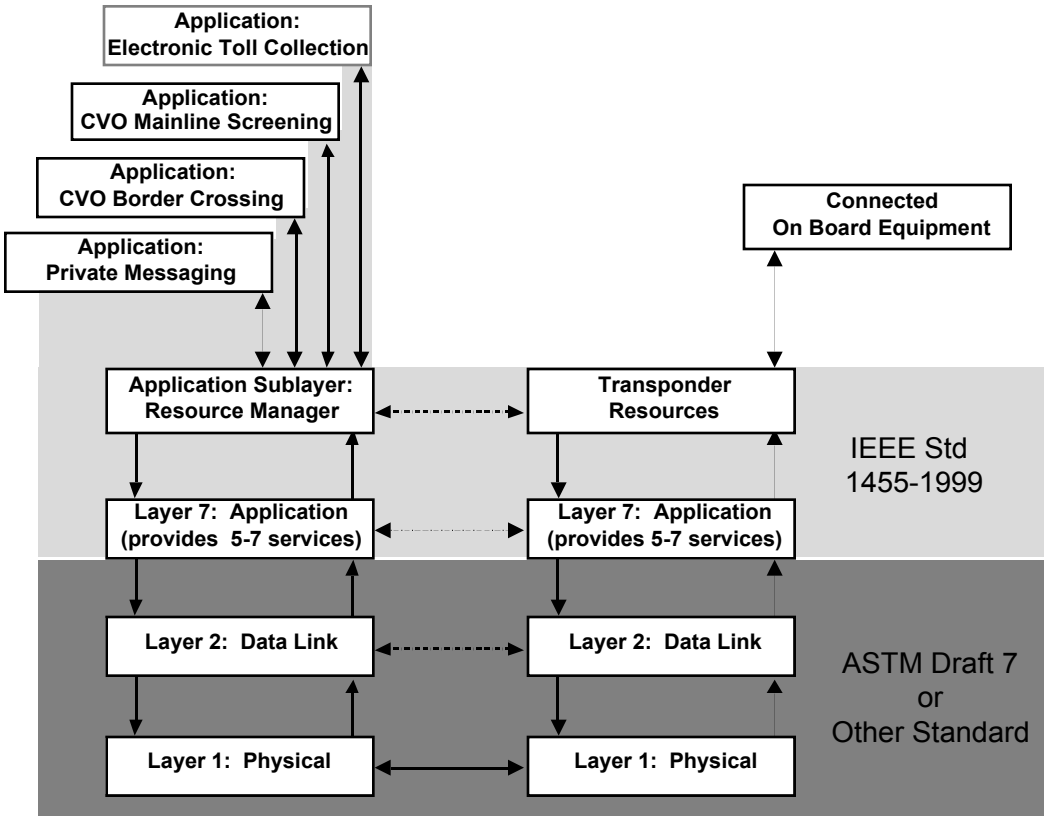

Figure 2—DSRC architecture and related standards

f) It is expected that the VRC controller will host the resource manager process. The BOAs that are connected with the VRC controller will register their data requirements with the resource manager. This registration may specify virtual page identifiers that shall be used to filter undesired transponders. The registration may also specify the information, including specific messages, that is desired from each relevant transponder that passes through the beacon's capture zone. However, this interface is not controlled by this standard.

g) As the transponder passes within the communication zone of the beacon, the beacon will transmit a BST. In response, the transponder will transmit a VST. Within this standard, the transponder's read-only, short read/write, and long read/write memory regions have been formatted to meet the VST definition as established by the Comité Européen de Normalisation (CEN).

h) The resource manager will process the received VST and transmit the information within it to the BOAs that have registered for that information.

i) Based upon previously received registrations, the resource manager may also use the transponder commands listed in Clause 6 to fetch images of additional memory pages. Those pages will also be processed and the information within them transmitted to the BOAs that have registered for that information.

j) Based upon the received information, the connected BOAs may request that the resource manager delete some existing messages or add new messages to a memory page. The resource manager shall respond to these requests and shall then transfer the updated memory image(s) to the transponder.

4.2 Access control flow summary

This standard provides for communications control fields and access control commands that may be used to restrict the set of commands to which the transponder will respond. These access control features are in all cases optional and may be omitted or only partially implemented within compliant equipment. It is anticipated that regulatory agencies, user agencies, or associations of user agencies may require that transponders used within their DSRC systems incorporate specific access control features as defined within this standard. However, such requirements do not preclude the manufacture and use of transponders that do not meet those agency's requirements, though such transponders could not be used to obtain services within such a user agency's system.

Access controls are typically used to provide data protection, data privacy, or data authentication. However, data privacy and authentication may also be accomplished by compliant equipment by using methods other than those defined by this standard. For example, Clause 8 states that message bodies may be encrypted using user-specific algorithms.

It is anticipated that, when implemented by the manufacturer and incorporated into a user agency's processing, these access control capabilities will typically be utilized as follows:

a) The transponder manufacturer will implement one or more security-related algorithms within the transponder's basic firmware. These algorithms will be suitable for confirming access control credentials that may be later received in conjunction with commands from the roadside.

b) Either as part of the initial fabrication or as a subsequent configuration process, the transponder's manufacturer may modify the firmware so that specific transponder commands or memory page references are governed by a set of access credentials. The selection of controlled commands and the access credentials used to control them may be requested by an end user agency. The method used by the manufacturer to perform this initial transponder configuration is not controlled by this standard. However, the manufacturer is encouraged to use the Reserve Memory Page and Release Memory Page commands defined in Clause 6.

c) In some cases the manufacturer may also configure the transponder with master credentials that allow the release of any previously reserved memory pages without presenting the access credentials specified at the time of the page reservation. This option is not governed by this standard.

d) In many cases a user agency will wish to utilize a reserved page with access controls to store essential or sensitive information, such as a financial account balance. In this case, assuming that the user agency has obtained a transponder that supports its selected security algorithms, the user agency may then reserve additional pages of memory using the Reserve Memory Page command and may specify the access credentials that will be used to control access to those pages as part of the Reserve Page command. This may typically be done at a central site using a controlled beacon installation.

e) Once configured by the user agency, the transponder may be installed in a consumer vehicle. When the vehicle passes through a beacon's communication zone, the beacon may issue a command to the transponder. Based upon the previous configurations imposed by the manufacturer or the user agency, the transponder may require that access controls be provided prior to executing the received command. Each of the commands specified in Clause 6 provides for the transmission of such access credentials with the command.

f) Many different security algorithms are currently available, and this standard does not govern the selection of such algorithms for DSRC. In general, the processing flow associated with such security algorithms requires a series of transactions between the roadside and the transponder, such as those illustrated in Figure 3. The transponder commands and responses specified in Clause 6 provide for the information transmission required by this figure.

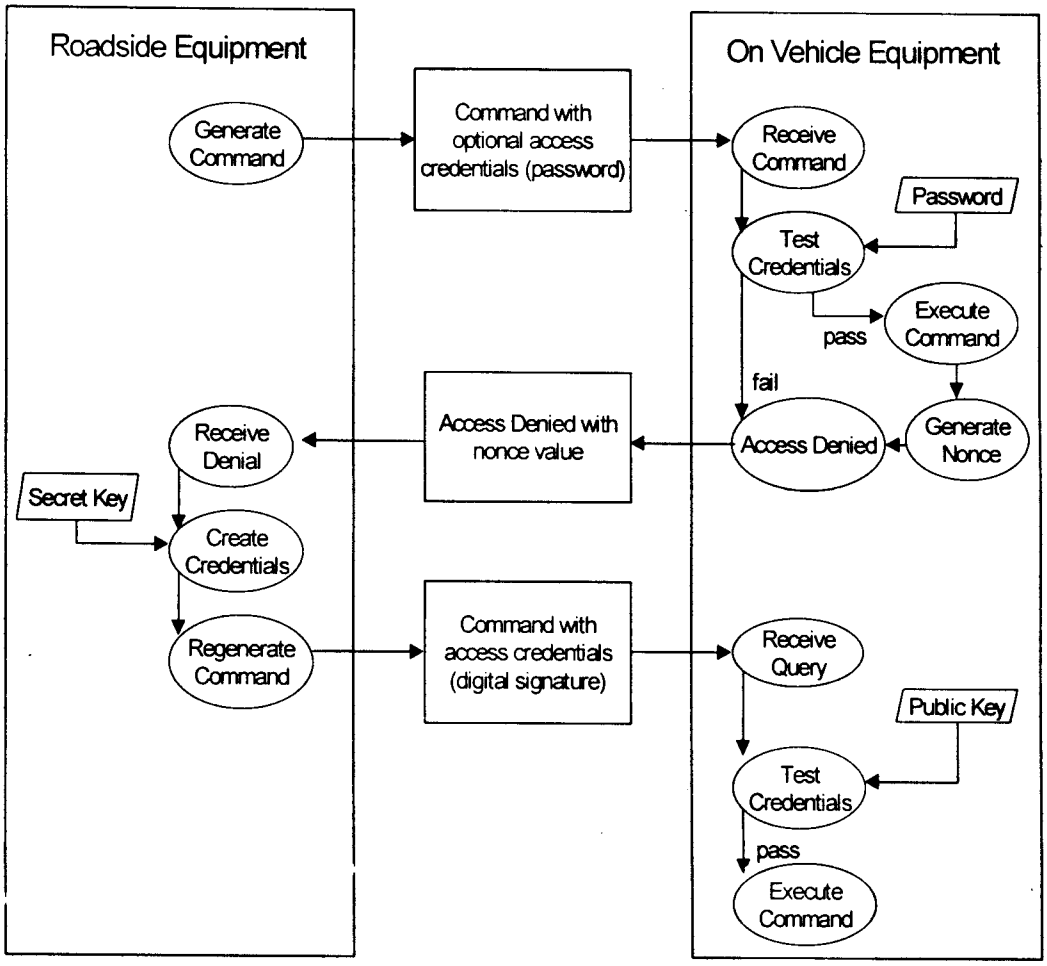

Figure 3—Access control processing flow

5. Transponder resources

Transponders that are compliant with this standard shall provide internal resources in accordance with the specifications described in this clause. While a range of transponders having various capabilities may be defined in a manner compliant with those specifications, the basic structure and the capabilities definitions shall be adhered to in all cases. Not all resources defined in this clause are mandatory in compliant transponders.

Many transponder identifiers provide for values that are available for registration. The registration process is controlled by the IEEE and is not defined within this document. See 1.3 for additional information on this topic.

5.1 Transponder resources definition

This clause functionally identifies and specifies the transponder resources requirements. These requirements shall in no way constrain the actual hardware implementation of transponders as long as the associated resources are partitioned in a manner compliant with this standard. A transponder compliant with this standard shall provide resources as defined in 5.1.1 through 5.1.12. These resources are illustrated in Figure 4.

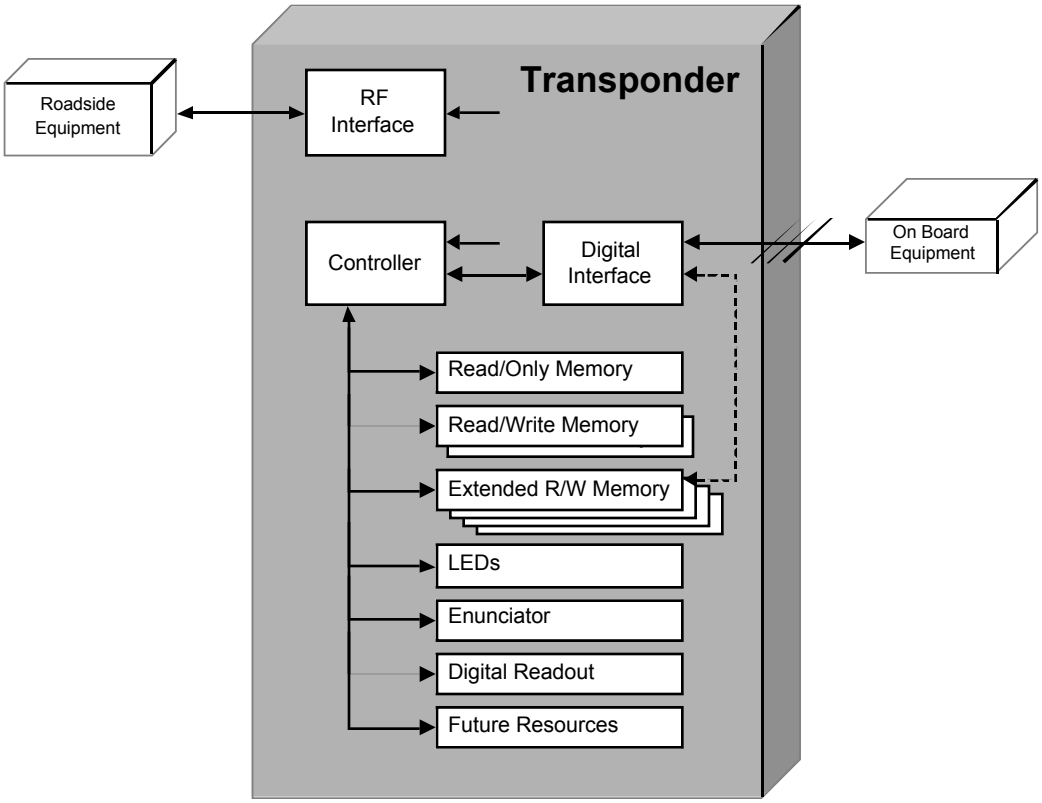

Figure 4—Transponder resources

5.1.1 Wireless interface

The transponder shall provide a wireless interface with the RSE. The characteristics of this interface are outside the scope of this standard. It is expected that this standard may be implemented in conjunction with a wide variety of radio frequency (RF) interfaces. However, the wireless interface must support the data transfers specified within this standard. This standard has been developed in coordination with ASTM Draft Standard for Dedicated Short-Range Communications (DSRC) Data Link Layer, Exxx-99[7] and the ISO 7498 series of standards to ensure compatibility among these standards.

5.1.2 Controller

The controller shall interpret and implement commands (listed in Clause 6) when received across the wireless interface. Implementation of those commands will typically require access to or control of the other transponder resources listed in this clause. The controller may also implement the interface with other OBE.

5.1.3 External interface

Compliant transponders may provide an external interface. The availability and characteristics of this interface shall be indicated in the read-only memory (as defined in 5.2). If implemented, this interface shall provide other pieces of OBE with access to the transponder's resources and through those resources provide communications with the roadside. The characteristics of this interface and the command set used across the external interface are outside the scope of this standard. However, it is anticipated that the command set will be comparable to that implemented across the wireless interface with the roadside.

[7]Information on references can be found in Clause 2.

5.1.4 Memory management and page identification

Memory within the transponder is formatted into partitions and pages. A partition is an area of memory that may be controlled by access credentials within which pages may be allocated. A page is an area of memory within a partition, which may also be protected with access credentials and from which data may be read or written.

As defined in 5.1.5 through 5.1.7, transponders may provide read-only memory, read/write memory, and/or extended read/write memory. While read-only memory and read/write memory pages are predefined, commands defined in Clause 6 allow dynamic configuration of the extended read/write memory. This dynamic configuration may be accomplished by allocating partitions within the extended read/write memory or by reserving pages.

Pages may be reserved either within an existing partition or within the overall extended read/write memory area. A partition identifier is specified when a partition is allocated, and it is referenced when a page is reserved within the partition. A page identifier is specified when a page is reserved, and it is referenced when a page is accessed. A sample of partition identifiers is provided in Table 1. A complete listing of defined partition identifiers is provided in Table E.1.

Table 1—Sample partition identifiers

Partition number	Partition designation
hex(0000)	Reserved
hex(0001 – FFFF)	Available for registration

Compliant transponders shall comply with the following requirements:

— The minimum memory page size shall be 128 bits.

— The maximum memory page size shall be 64 Kbytes. This limitation is based upon the maximum length of a read/write memory page command.

— The first memory page shall always exist and shall be a read-only memory page as defined in 5.1.5.

— All memory pages shall have an associated 16-bit page identifier that can be used by the transponder commands described in Clause 6.

— The first three transponder memory pages shall always be assigned the Page Identifiers hex(1) through hex(3).

— Page Identifiers hex(4) through hex(7) refer to predefined combinations of the first three memory pages.

— The page identifiers associated with the transponder user interface (UI) shall be assigned from values above hex(FEFF). These default values may be overridden by aliasing other page identifiers to the default identifiers using the commands defined in Clause 6.

— Unreserved page identifiers may be assigned to agencies on an implementation-specific basis.

A sample of defined page identifiers is provided in Table 2. Page numbers shall be unique within a transponder, i.e., duplicate page numbers shall not be used in different partitions.

Table 2—Sample page identifiers

Page number	Page designation
hex(0)	Reserved
hex(0001 .. FFFF)	Specific values are defined in Table E.2

5.1.5 Read-only memory

Compliant transponders shall provide 16 bytes (128 bits) of read-only memory. The information within this region shall be formatted as specified in 5.2.

The read-only memory shall be transmitted to the RSE within the VST (as defined in 9.5). The VST is returned by the OBE in response to a BST received from the RSE. The read-only memory may also be accessed using other memory access commands listed in Clause 6.

This region of memory is “read only” from the roadside.

5.1.6 Read/write memory

Compliant transponders may provide zero, one, or two read/write memory regions. The availability of these memory regions shall be as indicated in the read-only memory as defined in 5.2.

When present, the read/write memory regions shall have the following characteristics:

— The short read/write memory region shall provide 16 bytes (128 bits) of storage.

— The long read/write memory region shall provide 32 bytes (256 bits) of storage.

The read/write memory images may be transmitted to the RSE within the VST command response, as defined in Clause 6. The read/write memory regions may also be accessed using other memory access commands listed in Clause 6.

5.1.7 Extended read/write memory

Compliant transponders may provide one or more extended read/write memory regions. The availability of these memory regions shall be as indicated in the read-only memory, as defined in 5.2. The extended memory may be configured into logical pages by the manufacturer and/or by issuance of the Reserve Memory Page command, as defined in 6.4.9. The size of a dynamically created page is specified as an operand of the Reservation command. Associated with each page is a 16-bit page number. Permanent page numbers shall be reserved for specific purposes or agencies by registering the number. Table E.2 provides a list of preassigned page numbers and their associated use. These pages may, optionally, have access credentials to protect the page for write or read/write access.

The list of reserved pages for a given transponder may be requested by issuing the Query Memory Configuration command, as defined in 6.4.11.

5.1.8 Lamps

Compliant transponders may provide a red, a green, and a yellow lamp; it is not necessary for all lamps to be present. The availability of these lamps shall be as indicated in the read-only memory, as defined in 5.2. Lamps are controlled using the Set User Interface command specified in 6.4.6.

5.1.9 Enunciators

Compliant transponders may provide one or more enunciators. The availability of these enunciators shall be as indicated in the read-only memory, as defined in 5.2. Enunciators are controlled using the Set User Interface command specified in 6.4.6.

5.1.10 Character readout

Compliant transponders may provide a digital readout. The availability of a digital readout shall be as indicated in the read-only memory, as defined in 5.2. The digital readout is controlled using the Set User Interface command specified in 6.4.6 and the Map User Interface command specified in 6.4.7.

The digital readout shall display Text String messages (defined in 8.7.1) that are stored in the memory page to which the digital readout is mapped. Controls may be provided that enable the user to scroll from one message to another within the mapped memory page.

5.1.11 Keypad

Compliant transponders may provide a keypad. The availability of a keypad shall be as indicated in the read-only memory, as defined in 5.2.

The data that are entered using the keypad shall be stored as a Text String message in the memory page to which the keypad is mapped. The memory-related commands specified in Clause 6 shall be used to retrieve data entered at the keypad and to clear previously entered data.

5.1.12 Future resources

Future revisions of this standard may provide for additional UI resources. The availability of these resources is defined in 5.2; they shall be controlled using the Set User Interface command specified in 6.4.6 and the Map User Interface command specified in 6.4.7.

5.2 Read-only memory definition

Information within the read-only memory region shall be formatted as defined in Table 3 and described in 5.2.1 through 5.2.18. The read-only memory region corresponds exactly to the fixed length portion of the VST (the VST “header”) defined in 9.5.2.

Table 3—Read-only memory fields

Field name	Location (bits)	Length (bits)	Specification and description
T-APDU Tag	0-3	4	ASN.1 tag for an INITIALISATION.response (i.e., a VST) = hex(9)
Fill	4-7	4	Nonfunctional bits used to maintain byte boundaries
Profile	8-15	8	Profile field of VST
Number of Applications	16-23	8	Number of applications in the applications list = hex(1)
Application Identifier (AID)	24-31	8	Mailbox AID = hex(D)

Table 3—Read-only memory fields (continued)

Field name	Location (bits)	Length (bits)	Specification and description
EID/Revision Level	32-39	8	EID; shall be used for the IEEE Std 1455-1999 revision level
Container Tag	40-47	8	Octet string tag = hex(4); used to encapsulate the VST parameter
Octet String Length	48-55	8	The actual length (in octets) of the subsequent data in the octet string
Octet String Data		Includes following fields	An octet string used for ASN.1 compliance, which comprises the subsequent fields defined in this table
Returned Pages Flag	56-57	2	Bits that correspond to the two memory images returned, as specified in the BST
Reserved 1	58-60	3	Reserved by IEEE for future use
Memory Configuration	61-63	3	Defines the availability of various memory regions
Transponder Configuration	64-71	8	Defines the availability of various transponder peripherals, such as lamps and enunciators
Service Agency	72-87	16	Identifies the unique agency that is primarily responsible for issuing statements corresponding to services received by the transponder's user
Serial Number Type	88-91	4	Indicates how the serial number and manufacturer identifier should be interpreted
Manufacturer Identifier	92-107	16	Identifies the manufacturer of the transponder
Serial Number	108-127	20	Uniquely identifies the transponders produced under a single Manufacturer Identifier value

5.2.1 T-APDU Tag

The T-APDU Tag field is required to provide ASN.1 compliance (see the ASN.1 definition of T-APDUs in Annex A). This field shall be set to hex(9) to indicate an INITIALISATION.response.

5.2.2 Fill

The Fill field is required to maintain byte boundaries. This field shall be set to hex(0).

5.2.3 Profile

The Profile field contains communications profiles as defined by the specific lower layer service. The values for this field are defined in Table E.3, and a sample is shown in Table 4.

5.2.4 Number of Applications

This field contains the number of applications in the subsequent application list. The value for this field shall always be set to hex(1).

Table 4—Sample profile field values

Value	Definition
hex(0)	Reserved
hex(1)	Unspecified profile
hex(0 .. FF)	Available for registration

5.2.5 AID

The AID field is required to provide compatibility with the CEN VST definition. This field shall be set to hex(D), which is the AID for the Mailbox application.

5.2.6 EID/Revision Level

The EID field is required to provide compatibility with the CEN VST definition. Within this standard, the EID/Revision Level field indicates the revision level of this standard with which the transponder complies. Values shall be interpreted as defined in Table 5.

Table 5—EID/Revision Level field values

Value	Interpretation
hex(0)	Reserved
hex(1)	Prerelease (field testing; current value)
hex(2)	Initial release
hex(3 .. FF)	Reserved

5.2.7 Container Tag

The Container Tag field is required to provide ASN.1 compliance. This field shall be set to hex(4), which indicates an octet string.

5.2.8 Octet String Length

The Octet String Length field, which is required to provide ASN.1 compliance, indicates the length of the subsequent octet string data. The low order bit of octet string length must always be set to zero for ASN.1 compliance (meaning that this octet is used for actual length designation). The remaining 7 bits of the Octet String Length field contain the actual length (in octets) of the subsequent data in the octet string. Therefore, the maximum length for the data is 127 bytes.

The Octet String Length field shall be overwritten dynamically by the OBE transponder application during VST transmission. It is overwritten with a value that represents the sum of the size of the memory images being returned in the VST, which includes the read-only memory.

5.2.9 Octet String Data

The Octet String Data field is descriptive only and has no actual bit representation in and of itself. The purpose of this descriptive field is to indicate that the octet string data that follow the Octet String Length field include the subsequent fields defined for read-only memory and represent the balance of the VST structure.

5.2.10 Returned Pages Flag

The Returned Pages Flag field indicates which memory pages requested in the BST are present in the transponder and, therefore, which memory pages are being returned as part of the VST. This field shall be interpreted as defined in Table 6.

Table 6—Returned Pages field interpretation

Location (bits)	Interpretation
0	First page flag; a value of 1 indicates that the first page is returned within the VST
1	Second page flag; a value of 1 indicates that the second page is returned within the VST

5.2.11 Reserved 1

The Reserved 1 field is required to maintain byte boundaries. This field shall be set to hex(0).

5.2.12 Memory Configuration

The Memory Configuration field indicates which read/write memory regions are present. Values shall be interpreted as defined in Table 7.

Table 7—Read/write Memory Configuration field values

Value	Interpretation
hex(0)	No read/write memory present
hex(1)	Short read/write memory present
hex(2)	Long read/write memory present
hex(3)	Short read/write and long read/write memory present
hex(4)	Extended memory present
hex(5)	Short read/write and extended memory present
hex(6)	Long read/write and extended memory present
hex(7)	Short read/write, long read/write, and extended memory present

5.2.13 Transponder Configuration

The Transponder Configuration field indicates the configuration of installed transponder peripherals. The method of interpreting the field values is dependent upon the status of Bit 7. If Bit 7 is 1, then the remaining seven bits shall be individually interpreted to determine the peripherals configuration as defined in Table 8. If

Bit 7 is 0, then the remaining seven bits shall be interpreted as an enumerated value using Table E.4 (sample shown in Table 9).

Table 8—Transponder Configuration field interpretation, Bit 7 Set to 1

Location (bits)	Interpretation
7 (msb)	Always 1 when field is interpreted as binary flags rather than an enumerated value; if 0, then Table 9 applies
6	Red, yellow, and green lamps all present
5	Enunciator present
4	External network interface present
3	Character readout present
2	Keypad present
1	Reserved
0 (lsb)	Reserved

Table 9—Transponder Configuration enumerated field values, Bit 7 Set to 0

Value	Interpretation
hex(0)	Reserved
hex(1 .. 7)	Available for registration. Specific values are defined in Table E.4.

5.2.14 Service Agency

The Service Agency field indicates the service agency that is responsible for collecting fees incurred by the person using this transponder. Values shall be interpreted as defined in Table E.5. Sample table is shown below in Table 10.

Table 10—Sample Service Agency field values

Value	Interpretation
hex(0)	Reserved
hex(0001 .. FFFF)	Available for registration

5.2.15 Serial Number Type

The Serial Number Type field indicates the nature of the Manufacturer Identifier and the Serial Number fields when they are transmitted to the RSE. Those fields may be protected by encryption or masking, as indicated by the Serial Number Type field. The Serial Number Type field shall not be encrypted or masked. Values shall be interpreted as defined in Table 11.

Table 11—Serial Number Type field values

Value	Interpretation
hex(0)	Reserved
hex(1)	Clear; Manufacturer Identifier and Serial Number fields are not altered
hex(2)	Encrypted; Manufacturer Identifier and Serial Number fields are encrypted
hex(3)	Masked; Manufacturer Identifier and Serial Number fields are masked
hex(4 .. F)	Reserved

5.2.16 Manufacturer Identifier

The Manufacturer Identifier field indicates the manufacturer that produced the transponder. Values shall be interpreted as defined in Table E.6. Sample table is shown below in Table 12. The Manufacturer Identifier field may be masked or encrypted when transmitted to the RSE.

Table 12—Manufacturer Identifier field values

Value	Interpretation
hex(0)	Reserved
hex(0001 .. FFFF)	Available for registration

5.2.17 Serial Number

The Serial Number field shall uniquely identify a transponder within a set of devices having the same Manufacturer Identifier field value. The method of assigning values to this field shall be entirely controlled by the manufacturer. However, uniqueness shall be preserved. The Serial Number field may be masked or encrypted when transmitted to the RSE.

5.2.18 Unique identifier

The 40-bit sequence of data consisting of the Serial Number Type, Manufacturer Identifier, and Serial Number fields may be referred to as the transponder's unique identifier. The characteristics of the constituent fields shall be preserved as defined in 5.2.15 through 5.2.17.

5.3 Interoperability requirements

As defined in 1.4, compliant transponders meet all the requirements specified in this standard. Interoperable transponders additionally provide optional features. The following features defined in this subclause shall be provided in interoperable transponders:

— Read-only memory (128 bits), which shall not be protected by access credentials.

— Short read/write memory (128 bits), which shall not be protected by access credentials.

— Long read/write memory (256 bits), which shall not be protected by access credentials.

— Extended read/write memory (at least 512 bits).

— Red, yellow, and green lamps.

— An enunciator.

Interoperable transponders may also provide additional optional features such as using access credential protection.

Additional interoperability requirements are specified in 6.6.

6. Transponder commands and memory access

6.1 Basic concepts

Transponders that are compliant with this standard shall provide the capability to process the commands specified in this clause. These commands reference the memory, processing, and UI resources that may be present on the transponder. The commands are independent of the BOA that may be utilizing those resources. The availability of the resources that may be referenced by commands is indicated by bits allocated in read-only memory that are defined in 5.2 and by the results of the Query Memory Configuration command.

The RSE shall not intentionally generate commands to the transponder that reference resources known to be absent from the addressed transponder. However, each of the commands defined in this clause specifies the behavior that shall be exhibited when such absent resources are referenced.

The OBE is not in all cases required to provide the full set of behaviors specified for each of the commands specified in this clause. For each command, abnormal behaviors are specified that include the method (if any) by which the OBE shall notify the RSE if a received command is optional and has not been fully implemented in the receiving OBE.

Some of the commands defined in this clause, such as Read Memory Page and Write Memory Page, require transmission of an entire memory page image. The End Of Data message (defined in 8.7.4) may be used to terminate the region of a memory page that contains valid messages. When an End Of Data message is present, the RSE and OBE may transmit only that initial portion of a memory page that contains valid messages. If this optional feature is not implemented, then the area of a memory image that does not contain valid messages shall be transmitted and shall be set to zeroes.

6.2 Command set template

Each command shall consist of the fields shown in Table 13 and described in 6.2.1 through 6.2.6.

Table 13—Command set fields

Command Identifier	Command Transaction Identifier	Command Length	Access Control Length	Access Control	Command Parameters
1 byte	1 byte	2 bytes	1 byte	1 to 32 bytes	Variable

6.2.1 Command Identifier

6.2.1.1 Length

The length of the Command Identifier field shall be 1 byte.

6.2.1.2 Usage

The Command Identifier field shall identify the command to be performed and shall take on the values shown in Table 14. The high order bit (bit 7) of this field indicates the presence or absence of access credentials in the command. If bit 7 is 1, then the Access Control Length and Access Control fields shall be present after the Command Length field. If bit 7 is 0, then the Access Control Length and Access Control fields shall be omitted and the Command Length field shall be followed by the Command Parameter field.

Table 14—Command Identifiers

Codes	Codes with credentials	Meaning	OBE command support
hex(0 .. F)	hex(80 .. 8F)	Reserved	
hex(10)	hex(90)	Read Memory Page	Required to access memory other than through memory images returned in the VST
hex(11)	hex(91)	Write Memory Page	Optional* (required if read/write memory is present)
hex(12)	hex(92)	Append Message	Optional* (required if read/write memory is present)
hex(13)	hex(93)	Initialize Circular Queue	Optional (required for circular queues)
hex(14)	hex(94)	Write Circular Queue	Optional* (required for circular queues)
hex(15) .. (1F)	hex(95) .. (9F)	Reserved	
hex(20)	hex(A0)	Set User Interface	Optional* (required if OBE has UI)
hex(21)	hex(A1)	Map User Interface	Optional
hex(22 .. 2F)	hex(A2 .. AF)	Reserved	
hex(30)	hex(B0)	Sleep Transponder	Optional
hex(31.. 3F)	hex(B1.. BF)	Reserved	
hex(40)	hex(C0)	Reserve Memory Page	Optional (required if extended memory is present)
hex(41)	hex(C1)	Release Memory Page	Optional (required if Reserve Memory Page command is implemented)
hex(42)	hex(C2)	Query Memory Configuration	Optional (required if Reserve Memory Page command is implemented)

Table 14—Command Identifiers (continued)

Codes	Codes with credentials	Meaning	OBE command support
hex(43)	hex(C3)	Reserve Memory Partition	Optional
hex(44)	hex(C4)	Release Memory Partition	Optional (required if Reserve Memory Partition command is implemented)
hex(45 .. 6F)	hex(C5 .. EF)	Reserved	
hex(70 .. 7F)	hex(F0 .. FF)	Available for manufacturer-specific testing	Optional—shall not be used in production units deployed in the field

*These commands are supported in broadcast mode.

6.2.2 Command Transaction Identifier

6.2.2.1 Length

The length of the Command Transaction Identifier field shall be 1 byte.

6.2.2.2 Usage

The Command Transaction Identifier field shall be an identifier that is uniquely calculated for each instance of a command. This identifier is returned in the command response and allows the resource manager to match a received response to a specific sent command.

6.2.3 Command Length

6.2.3.1 Length

The length of the Command Length field shall be 2 bytes.

6.2.3.2 Usage

The Command Length field shall specify the total length in bytes of the command instance, including all fields except the Command Identifier field, the Command Transaction Identifier field, and this Command Length field. The maximum value of this field effectively constrains the maximum size of a transferred memory image.

6.2.4 Access Control Length

6.2.4.1 Length

The length of the Access Control Length field shall be 1 byte.

6.2.4.2 Usage

The Access Control Length field shall specify the length of the Access Control field in bytes. This field, if present, will never be zero.

6.2.5 Access Control

6.2.5.1 Length

The length of the Access Control field shall vary up to 32 bytes, as specified by the Access Control Length field.

6.2.5.2 Usage

The Access Control field shall be used to provide access controls for command instances. The actual value of the Access Control field is implementation-specific and would typically follow some type of encryption and/or authentication scheme.

6.2.6 Command Parameters

6.2.6.1 Length

The length of the Command Parameters field shall be fixed for each command except for the Write Memory, Append Message, Write Circular Queue, and Set User Interface commands, which have variable parameter lengths.

6.2.6.2 Usage

The Command Parameters field shall be specific to each command set.

6.3 Command information flow

This standard provides for two communication modes. In the “connected” mode, all communications are prefaced by a BST/VST exchange that connects the RSE to a specific transponder. In the “broadcast” mode, transponder commands are broadcast from the RSE to all passing transponders without first establishing an RSE-to-OBE connection using a BST/VST. Transponders shall remain ready to receive a communication at all times, subject to vendor-specific power consumption optimization.

The connected mode is recommended because it allows the RSE to verify the transponder configuration before additional commands are transmitted to the transponder. However, the broadcast mode may be appropriate in certain applications where the communication opportunity is constrained.

Clause 9 discusses all application layer services in detail. Annex C provides illustrations for the flow of commands from the resource manager to the OBE transponder application via the application layer.

6.3.1 Connected mode information flow

In the connected mode, the following RSE-to-OBE information flow shall be observed:

a) The resource manager shall register itself as part of its startup sequence by using the RegisterApplicationBeacon service in the application layer. This registration causes the RSE application layer to construct a BST and initiate communication with potential OBE transponders.

b) The connection is established when the OBE application layer returns a VST to the resource manager.

c) The resource manager shall determine appropriate commands, based upon registration requests from the connected BOAs.

d) The resource manager shall formulate the command instance using supplied information.

e) The resource manager shall use the ACTION.request service in the application layer to transmit the command to the OBE. The OBE shall provide a response if required by the Mode field in the Action.request.

f) The OBE shall process the command received from the resource manager as an ACTION.indication service and shall respond, if required, using the ACTION.response service of the OBE application layer.

g) The resource manager shall process the received ACTION.confirm, potentially resending the command with appropriate access controls.

6.3.2 Broadcast mode information flow

The broadcast mode is used to transmit a command or a fixed set of commands to every transponder that passes through the RSE communication zone. In the broadcast mode, the following RSE-to-OBE information flow shall be observed:

a) The resource manager may use the BroadcastData.request service of the RSE application layer to broadcast to the OBE.

b) In that case, the OBE transponder application shall use the GetBroadcastData.request service of the OBE application layer to access a command or command set that was broadcast from the resource manager.

c) The resource manager may also use the ACTION.request service of the RSE application layer to broadcast a command or command set in a broadcast mode. This may be accomplished by setting the LID field of the ACTION.request to a global LID; in this case a response from the OBE may also be requested by setting the Mode field of the ACTION.request.

d) This ACTION.request will fail if the lower layer service associated with the application layer does not support the optional DATASEND_RESPOND_REPEAT or DATASEND_NORESPOND_ REPEAT messages defined in 9.3.2. In this case, the RSE application layer may choose to repeat the command.

   Also, subject to the capabilities of the lower layer media, the RSE must provide for the case that multiple transponders are within the communications zone at the time of transmission. The OBE transponder application shall use the ACTION.response service of the OBE application layer to send command responses back to the resource manager in response to a command received as a result of the resource manager sending that command using a broadcast ACTION.request.

6.4 Command definitions

The commands shall be created by the RSE and processed by the OBE as specified in 6.4.1 through 6.4.13. The command identifier values are shown with the Access Credential flag set to 0. The Command Parameter field locations are shown as if the Access Control Length and Access Control fields were not present. Details regarding command responses are provided in 6.5.

6.4.1 Read Memory Page (mandatory)

The Read Memory Page command shall initiate transmission of the specified OBE memory pages to the RSE.

6.4.1.1 Command set definition

The Read Memory Page command shall consist of the fields shown in Table 15.

Table 15—Read Memory Page command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(10) / hex(90)
Command Transaction Identifier	1	1	Identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credential bytes. A nonzero value indicates that the Page Identifier field will be offset by the indicated number of bytes (max. 32)
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: Page Identifier	End of Access Control field + 1 End of Access Control field + 2	2	Identifies referenced memory page (as per specification in Table E.2)

6.4.1.2 OBE normal behaviors

The OBE shall access the memory page specified by the Read Memory Page command.

If the OBE successfully executes the Read Memory Page command, the OBE shall send a response with the Response Identifier field set to Command Success, the Response Data Length field set to the length of the read memory image, and the memory image itself in the Response Data field.

6.4.1.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions. See Table 29 for definitions.

— Command Not Recognized

— Access Control Error

— Page Not Defined

— Memory Access Error

— Command Failed

6.4.2 Write Memory Page (optional)

The Write Memory Page command is suffixed by a memory image that shall be stored in the specified OBE memory page.

6.4.2.1 Command set definition

The Write Memory Page command shall consist of the fields shown in Table 16.

Table 16—Write Memory Page command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(11) / hex(91)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credential bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: Page Identifier	End of Access Control field + 1 End of Access Control field + 2	2	Identifies referenced memory page (as per specification in Table E.2)
Memory Image	End of Access Control field + 3 .. n		The information that shall consist of sequenced messages followed by zero-fill bytes or an End Of Data message identifier. The length of this image is only constrained by the maximum command length defined in 6.2.3.

6.4.2.2 OBE normal behaviors

The OBE shall store the memory image within the received command by completely overwriting the referenced agency memory page.

If the OBE successfully executes the Write Memory Page command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

6.4.2.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized
— Access Control Error
— Page Not Defined
— Page Length Mismatch
— Memory Access Error
— Command Failed

6.4.3 Append Message (optional)

The Append Message command is suffixed by a message image that shall be appended to the end of the previously used memory within the specified OBE memory page.

6.4.3.1 Command set definition

The Append Message command shall consist of the fields shown in Table 17.

Table 17—Append Message command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(12) / hex(92)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional). Number of access credential bytes
Access Control	5 .. n	0/1 .. 32	(Optional). Access credentials
Command Parameter: Page Identifier	End of Access Control field + 1 End of Access Control field + 2	2	Identifies referenced memory page (as per specification in Table E.2)
Message Image	End of Access Control field + 3 .. n		The information that shall be appended to the specified memory page

6.4.3.2 OBE normal behaviors

The OBE shall append the message image within the command to the end of existing messages in the specified page. Positioning within the page shall be determined by chaining through the stored messages until the first occurrence of either an End Of Data message or hex(00) following a message, where the next message header would begin. The message image shall be inserted at this point, overwriting the End Of Data message, if present. The new data shall be suffixed by either an End Of Data message or by zero filling the remainder of page.

If the OBE successfully executes the Append Message command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

6.4.3.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized

— Access Control Error

— Page Not Defined

— Insufficient Memory

— Memory Access Error

— Command Failed

6.4.4 Initialize Circular Queue (optional)

The Initialize Circular Queue command shall cause all of the memory within the specified OBE extended memory page to be cleared to zeros and shall set any control data to indicate that the queue is empty.

6.4.4.1 Command set definition

The Initialize Circular Queue command shall consist of the fields shown in Table 18.

Table 18—Initialize Circular Queue command fields

Field name	Location (bytes)	Length (bytes)	Specification and description
Command Identifier	0	1	hex(13) / hex(93)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credential bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: Page Identifier	End of Access Control field + 1 End of Access Control field + 2	2	Identifies referenced memory page (as per specification in Table E.2)

6.4.4.2 OBE normal behaviors

The OBE shall clear the specified page to zeros and shall set any control data to indicate that the queue is empty.

If the OBE successfully executes the Initialize Circular Queue command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

6.4.4.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized

— Page Not Defined

— Access Control Error
— Memory Access Error
— Command Failed

6.4.5 Write Circular Queue (optional)

The Write Circular Queue command is suffixed by a message image that shall be written to the end of the circular queue within the specified OBE memory page.

6.4.5.1 Command set definition

The Write Circular Queue command shall consist of the fields shown in Table 19.

Table 19—Write Circular Queue command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(14) / hex(94)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credential bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: Page Identifier	End of Access Control field + 1 End of Access Control field + 2	2	Identifies referenced memory page (as per specification in Table E.2)
Message Image	End of Access Control field 3 .. n		The information that shall be written to the end of the circular queue in the specified memory page

6.4.5.2 OBE normal behaviors

The OBE shall write the message image within the received command by locating the end of the current circular queue contained in the referenced memory page, placing the message image at the end of the queue, and updating any queue control information. Existing messages shall be deleted on a first-in-first-out (FIFO) basis if required to create sufficient available memory for the insertion of the message image. All memory in the page that is not used for message storage shall be set to zero.

If the OBE successfully executes the Write Circular Queue command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

6.4.5.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized

— Page Not Defined

— Access Control Error

— Insufficient Memory

— Memory Access Error

— Command Failed

6.4.6 Set User Interface (optional)

The Set User Interface command is suffixed by data specifying UI behaviors that shall be implemented by the OBE. The RSE may determine the UI elements that are available for a transponder by interpreting the Transponder Configuration field that is stored in the OBE read-only memory and transmitted to the RSE within the VST. The Transponder Configuration field is defined in Table 3. The RSE shall not address UI elements that are absent for a given OBE.

6.4.6.1 Command set definition

The Set User Interface command shall consist of the fields shown in Table 20.

Table 20—Set User Interface command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(20) / hex(A0)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credential bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: User Interface Element	End of Access Control field + 1 End of Access Control field + 2	2	Each command will affect only the addressed elements defined as follows: Bit 0: Red lamp Bit 1: Yellow lamp Bit 2: Green lamp Bit 3: Enunciator Bit 4: Character display Bits 5-15: Additional UI elements
Type (ObeUICmdType)	End of Access Control field + 3	1	AbsoluteOff (0), AbsoluteOn (1), TimedCommand (2), Flashing Command (3), Reserved (4 .. 255)
Attributes: Absolute Command (0 bytes) Timed Command (2 bytes) Flashing Command (5 bytes)	End of Access Control field + 4 .. n	0 2 4 1	(Unused for Absolute Command) Time period in 125 ms increments Cycle state bitmap, 125 ms per bit Repetition count (1 .. 255)

6.4.6.2 OBE normal behaviors

The OBE shall alter the UI element specified within the command parameter in the following fashion, depending upon the value of the type parameter:

— *Absolute Off command*. Turns the addressed UI element off. State is maintained until changed by a subsequent command.

— *Absolute On command*. Turns the addressed UI element on. State is maintained until changed by a subsequent command.

— *Timed command*. Turns the addressed UI element on for a specified period of time. The time period is specified in 125 ms increments.

— *Flashing command*. Cycles the state of the addressed UI element based upon a 4-byte bit map. Each bit in the map represents an interval of 125 ms (the total bit map represents 4 s). A repetition byte indicates the number of times the bit map cycle pattern should be performed (1 to 255 times).

If the OBE successfully executes the Set User Interface command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

The OBE may arbitrarily turn off any element after a preset period of time to preserve battery life.

The completion of a UI command shall not be affected by the reception of any other transmissions from the RSE except for an overriding UI command.

Table E.2 defines a memory page associated with an enunciator. This is intended to support systems that provide a synthetic speech interface or other sophisticated auditory cues. The Set User Interface command shall always be used to initiate changes to the UI, but the data in the enunciator memory page may be used to control the specific action.

Also, Table E.2 defines a memory page associated with the character readout. The Set User Interface command shall always be used to initiate changes to the character display, but the specific information shown may be retrieved from the character display memory page.

6.4.6.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized

— Access Control Error

— Device Error

— Command Failed

6.4.7 Map User Interface (optional)

The Map User Interface command shall cause the OBE to map a page of memory to the specified UI component. This reservation shall include the establishment of access control procedures. Mapping a page of memory to a specified UI element affects the behavior of the transponder when a Set User Interface command is subsequently received by indicating the information that is enunciated or displayed.

6.4.7.1 Command set definition

The Map User Interface command shall consist of the fields shown in Table 21.

Table 21—Map User Interface command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(21) / hex(A1)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credentials bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: User Interface Element	End of Access Control field + 1	1	Defines the UI element to be mapped 0: Keypad 1: Character Display 2: Enunciator voice 1 3-255: Reserved for additional UI elements
Page Identifier	End of Access Control field + 2	2	Identifies referenced memory page (as per specification in Table E.2)

6.4.7.2 OBE normal behaviors

The OBE shall first determine whether the specified page identifier has already been reserved. If the page identifier exists, then that page shall be used for all subsequent UI actions that reference the specified UI element. The predefined UI page identifiers listed in Table E.2 may always be used to reference the specific pages that have been currently selected using the Map User Interface command. See 5.1.8 through 5.1.11 for how the data within the UI memory pages shall be utilized.

If the OBE successfully executes the Map User Interface command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

6.4.7.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized

— Access Control Error

— Previously Reserved

— Device Error

— Command Failed

6.4.8 Sleep Transponder (optional)

The Sleep Transponder command shall cause the receiving transponder to sleep (disable RF reception and transmission) for a period of time specified in the command instance.

6.4.8.1 Command set definition

The Sleep Transponder command shall consist of the fields shown in Table 22.

Table 22—Sleep Transponder command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(30) / hex(B0)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credential bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: Sleep Duration	End of Access Control field + 1 End of Access Control field + 2	2	Sleep time duration in 125 ms increments

6.4.8.2 OBE normal behaviors

The OBE shall cause the transponder to cease responding to RF signaling for the specified period.

If the OBE successfully executes the Sleep Transponder command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field. Upon completion, the link shall be considered terminated.

When an RSE wishes an OBE to reinitialize, a sleep duration of 0 will result in the OBE reinitializing with the next Frame Control Frame that it receives.

6.4.8.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized

— Access Control Error

— Device Error

— Command Failed

6.4.9 Reserve Memory Page (optional)

The Reserve Memory Page command shall cause the OBE to reserve a page of memory for the specified agency. This reservation shall include the establishment of access control procedures.

6.4.9.1 Command set definition

The Reserve Memory Page command shall consist of the fields shown in Table 23.

Table 23—Reserve Memory Page command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(40) / hex(C0)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credential bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Partition Identifier	End of Access Control field + 1 End of Access Control field + 2	2	Specifies the partition identifier in which the memory shall be allocated (as per specification in Table E.1). A value of 0 implies that the page shall be allocated within unpartitioned memory.
Command Parameter: Page Size	End of Access Control field + 3 End of Access Control field + 4	2	Length (in bytes) of the memory page that is requested
Page Identifier	End of Access Control field + 5 End of Access Control field + 6	2	Specifies the page identifier that will be associated with the allocated memory (as per specification in Table E.2)
Page Access Credential Type and Length	(End of Page Identifier field + 1)	0/1	(Optional.) Bits 0 .. 1 indicate the access credential scope: Bit 0 = Access Credentials applied to Read access Bit 1 = Access Credentials applied to Write access Bits 2 .. 7 : Number of access credentials bytes that shall be applied to reserved page; max value = 32
Page Access Credentials	(End of Access Credential Type and Length + 1 .. n)	0/1 .. 32	(Optional.) Access credentials that shall be applied to reserved page

6.4.9.2 OBE normal behaviors

The OBE shall determine whether sufficient unallocated memory exists in the specified partition to satisfy the request and that the page identifier is available. If so, the memory page shall be defined from the available memory in the specified partition. This reservation shall then be honored when subsequent page-related commands are received.

If the OBE successfully executes the Reserve Memory Page command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

6.4.9.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized
— Access Control Error
— Partition Not Defined
— Page Not Defined
— Previously Reserved
— Device Error
— Command Failed
— Insufficient Memory

6.4.10 Release Memory Page (optional)

The Release Memory Page command shall cause the OBE to release a page of memory that had previously been reserved by the specified agency. This release shall require the same access controls (if any) that were specified at the time of page reservation.

6.4.10.1 Command set definition

The Release Memory Page command shall consist of the fields shown in Table 24.

Table 24—Release Memory Page command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(41) / hex(C1)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credentials bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: Page Identifier	End of Access Control field + 1 End of Access Control field + 2	2	Identifies referenced memory page (as per specification in Table E.2)

6.4.10.2 OBE normal behaviors

The OBE shall release the previously reserved page, allowing it to be reserved by other agencies and returning the associated extended memory to the pool available for new reservation requests.

If the OBE successfully executes the Release Memory Page command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

6.4.10.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized
— Access Control Error
— Page Not Defined
— Device Error
— Command Failed

6.4.11 Query Memory Configuration (optional)

The Query Memory Configuration command shall cause the OBE to return to the RSE the information that describes the organization of memory including reserved memory partitions, reserved memory pages, and free (unreserved) memory.

Execution of this command may be controlled by access credentials. When this command is successfully executed, it shall return a complete description of the transponder memory organization. The data returned in response to this command shall not be affected by credentials required for specific partition page or memory access.

6.4.11.1 Command set definition

The Query Memory Configuration command shall consist of the fields shown in Table 25.

Table 25—Query Memory Configuration command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(42) / hex(C2)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	(4)	0/1	(Optional.) Number of access credentials bytes
Access Control	(5 .. n)	0/1 .. 32	(Optional.) Access credentials
Command Parameter			None

6.4.11.2 OBE normal behaviors

The OBE shall return the roadside information that describes the memory configuration.

If the OBE successfully executes the Query Memory Configuration command, the OBE shall send a response with the Response Identifier field set to Command Success, the Response Data Length field set to the length of the returned memory configuration data, and the memory configuration data itself shall be returned in the Response Data field (see 6.5.5).

6.4.11.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized

— Access Control Error

— Device Error

— Command Failed

6.4.12 Reserve Memory Partition (optional)

The Reserve Memory Partition command shall cause the OBE to reserve a partition of extended memory for the specified agency. This reservation shall include the establishment of access control procedures.

6.4.12.1 Command set definition

The Reserve Memory Partition command shall consist of the fields shown in Table 26.

Table 26—Reserve Memory Partition command fields

Field Name	Location (bytes)	Length	Specification and Description
Command Identifier	0	1	hex(43) / hex(C3)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credential bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: Partition Size	End of Access Control field + 1 End of Access Control field + 2	2	Length (in bytes) of the memory partition that is requested

Table 26—Reserve Memory Partition command fields (continued)

Field Name	Location (bytes)	Length	Specification and Description
Partition Identifier	End of Access Control field + 3 End of Access Control field + 4	2	Identifies the partition identifier that will be associated with this request; Table 1 defines the valid range of values for partition identifiers
Partition Access Credential Length	(End of Partition Identifier field + 3)	0/1	(Optional.) Number of access credential bytes that shall be applied to this partition; max value = 32
Partition Access Credentials	(End of Partition Access Credential Type field + 1.. n)	0/1 .. 32	(Optional.) Access credentials that shall be required when reserving memory pages within this partition

6.4.12.2 OBE normal behaviors

The OBE shall determine whether sufficient nonpartitioned memory exists to satisfy the request and whether the partition identifier is available. If so, the partition shall be defined from the available memory. This partition shall then be honored when subsequent partition-related commands are received.

If the OBE successfully executes the Reserve Memory Partition command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

6.4.12.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized
— Access Control Error
— Previously Reserved
— Device Error
— Command Failed
— Insufficient Memory

6.4.13 Release Memory Partition (optional)

The Release Memory Partition command shall cause the OBE to release a partition of memory that had previously been reserved by the specified agency. This release shall require the same access controls (if any) that were specified at the time of partition reservation.

6.4.13.1 Command set definition

The Release Memory Partition command shall consist of the fields shown in Table 27.

Table 27—Release Memory Partition command fields

Field name	Location (bytes)	Length	Specification and description
Command Identifier	0	1	hex(44) / hex(C4)
Command Transaction Identifier	1	1	Uniquely identifies an instance of a command
Command Length	2 .. 3	2	Defines the total length (bytes) of all fields in this command excluding the length of the Command Identifier field, Command Transaction Identifier field, and this Command Length field
Access Control Length	4	0/1	(Optional.) Number of access credentials bytes
Access Control	5 .. n	0/1 .. 32	(Optional.) Access credentials
Command Parameter: Partition Identifier	End of Access Control field + 1 End of Access Control field + 2	2	Identifies the partition identifier that will be associated with this request; Table E.1 defines the valid range of values for partition identifiers.

6.4.13.2 OBE normal behaviors

The OBE shall release the previously reserved partition and return the memory to the pool available for partitioning.

If the OBE successfully executes the Release Memory Partition command, the OBE shall send a response with the Response Identifier field set to Command Success. The response shall contain no data in the Response Data field.

6.4.13.3 OBE abnormal responses

The following Response Identifier field values defined in Table 29 may be returned for abnormal conditions:

— Command Not Recognized
— Access Control Error
— Partition Not Defined
— Device Error
— Command Failed

6.5 Standard command responses

Each command response shall consist of the fields shown in Table 28 and described in 6.5.1 through 6.5.5.

Table 28—Command set fields

Response Command Identifier	Response Transaction Identifier	Response Identifier	Response Data Length	Response Data
1 byte	1 byte	1 byte	2 bytes	Variable

6.5.1 Response Command Identifier

6.5.1.1 Length

The length of the Response Command Identifier field shall be 1 byte.

6.5.1.2 Usage

The Response Command Identifier field shall contain the command identifier of the original command that was sent to the OBE and effected this response. Command Identifier field values are listed in Table 14.

6.5.1.3 Default value

The Response Command Identifier field has no default value.

6.5.2 Response Transaction Identifier

6.5.2.1 Length

The length of the Response Transaction Identifier field shall be 1 byte.

6.5.2.2 Usage

The Response Transaction Identifier field shall contain the transaction identifier of the received command to which the response is addressed.

6.5.2.3 Default value

The Response Transaction Identifier field has no default value.

6.5.3 Response Identifier

6.5.3.1 Length

The length of the Response Identifier field shall be 1 byte.

6.5.3.2 Usage

The Response Identifier field shall contain a value indicating the status of command execution in the OBE.

6.5.3.3 Default value

The Response Identifier field has no default value.

6.5.3.4 Response definitions

Table 29 lists the valid values and their interpretation. (See ObeResponse in A.2 for ASN.1 definitions.) A command response will only contain valid response data when the Response Identifier field is set to Command Success. All other values indicate failure conditions.

6.5.4 Response Data Length

6.5.4.1 Length

The length of the Response Data Length field shall be 2 bytes.

Table 29—Response Identifier values

Response name	Value	Specification and description
Reserved	hex(0)	
Command Success	hex(1)	Completion is normal.
Command Failed	hex(2)	Completion is abnormal due to some unspecified condition.
Command Not Recognized	hex(3)	The command was invalid or unsupported by the OBE. This response is used if the RSE references a UI element that is not present on a transponder.
Access Control Error	hex(4)	Access credentials may not have been supplied in a required situation or the supplied credentials may be invalid; a nonce value is returned from the OBE.
Page Not Defined	hex(5)	The page identifier does not match a reserved page in the OBE.
Partition Not Defined	hex(6)	The partition identifier does not match an existing partition in the OBE.
Device Error	hex(7)	A malfunction has occurred in the OBE hardware or software.
Memory Access Error	hex(8)	The requested memory is faulty.
Page Length Mismatch	hex(9)	The length of the memory image is greater than the length of the referenced memory page, or command execution would require crossing a page boundary.
Insufficient Memory	hex(A)	Available free memory is insufficient in the referenced page to perform the command.
Previously Reserved	hex(B)	The specified page or partition identifier has already been reserved by a previous Reserve command.
Reserved	hex(C .. EF)	Reserved.
Vendor Area	hex(F0 .. FF)	Available for vendor-specific failure conditions.

6.5.4.2 Usage

The Response Data Length field shall specify the total length (in bytes) of the data contained in the Response Data field. Only the Read Memory Page and Query Memory Configuration commands return response data. Response data may also be created for a nonce if required access credentials are incorrect. The Response Data Length field shall always be present even if the response contains no response data.

6.5.4.3 Default value

The default value of the Response Data Length field shall be zero (0) if the response contains no data.

6.5.5 Response Data

6.5.5.1 Length

The length of the Response Data field shall be variable.

6.5.5.2 Usage

Three cases exist for which response data are returned. These cases are described in 6.5.5.2.1 through 6.5.5.2.3.

6.5.5.2.1 Case 1

— Command = Read Memory Page

— Response Identifier = Command Success

In this case, the Response Data field contains the requested OBE memory image.

6.5.5.2.2 Case 2

a) Command = Query Memory Configuration

b) Response Identifier = Command Success

In this case the Response Data field contains a contiguous snapshot of the transponder memory configuration represented as a set of triplets, where each triplet in the set consists of three 16-bit values defined as follows:

1) Block Size: The size in bytes of this memory block.

2) Page Identifier: The page identifier associated with this block of memory. A value of hex(0) means the memory is unallocated.

3) Partition Identifier: The partition to which this block of memory belongs. A value of hex(0) means no associated partition exists.

6.5.5.2.3 Case 3

— Command = Any Command

— Response Identifier = Access Control Error

In this case, the Response Data field contains a nonce value.

6.5.5.3 Default value

The Response Data field does not exist except for the three cases specified in 6.5.5.2.

6.5.6 Response definitions

The OBE normal behaviors, which are described for each command definition in 6.4, specifically state the values that shall be contained in the Response Identifier field. The response data, if any, that shall be returned in the response is also included in these descriptions.

The only abnormal response (any value for the Response Identifier field other than Command Success) that shall return a non-null Response Data field is the Access Control Error, which returns a nonce value.

The Received Transaction Identifier field of any response shall always be set to the value of the Transaction Identifier field of the command to which the OBE is responding.

6.6 Interoperability requirements

As defined in 1.4, compliant transponders shall meet all the requirements specified in this standard. Interoperable transponders shall additionally provide specified features. The following commands defined in this subclause shall be implemented in interoperable transponders:

— Read Memory Page

— Write Memory Page

— Set User Interface

— Sleep Transponder

— Reserve Memory Page (If Reserve Memory Page is not supported, then available extended memory shall be preallocated into pages by the manufacturer.)

— Release Memory Page (Required when Reserve Memory Page is present.)

— Query Memory Configuration

Interoperable transponders may also implement additional optional commands.

Additional interoperability requirements are specified in 5.3.

6.7 Error detection and processing

The following methods shall be applied on the OBE and RSE to detect and process errors that may be present in commands and command responses.

6.7.1 OBE command error detection processing

The OBE shall check commands received from the RSE for the following error conditions prior to execution:

a) Verify that the command identifier is defined.

b) Verify that the command length matches the length of the received information.

c) For commands having fixed parameters, verify that the command length matches the value defined in this standard.

d) For command parameters that have a limited value domain, verify that all command parameters have values defined within this standard.

Additional, vendor-specific error checks may be provided.

If any of these conditions is detected, the OBE shall reject the command and shall issue a command response with the appropriate response identifier.

6.7.2 RSE command response error detection processing

The RSE shall check command responses received from the OBE for the following error conditions prior to execution:

a) Verify that the response command identifier is defined.

b) Verify that the response identifier is defined.

c) Verify that the response command identifier is the same as the command identifier used in the previously transmitted command having a matching response transaction identifier.

d) Verify that the response data length matches the length of the received information.

Additional, vendor-specific error checks may be provided.

If any of these error conditions is detected, the RSE shall reject the command response. The RSE may retransmit the command that resulted in the erroneous command response, using the identical information. If the OBE receives such a duplicated command, it shall regenerate and transmit the appropriate command response, but shall not reexecute the defined command processing. Reception of an erroneous command response may indicate a flaw in the overall processing, and the command that generated the condition should generally not be retransmitted.

7. Resource manager

The resource manager shall provide the roadside "operating system" that accepts, arbitrates, implements, and responds to requests for DSRC services that are received from one or more BOAs. The resource manager shall be the initiator of all commands to the OBE controller, acting as the master in a master-slave relationship. The functional relationships are illustrated in Figure 5.

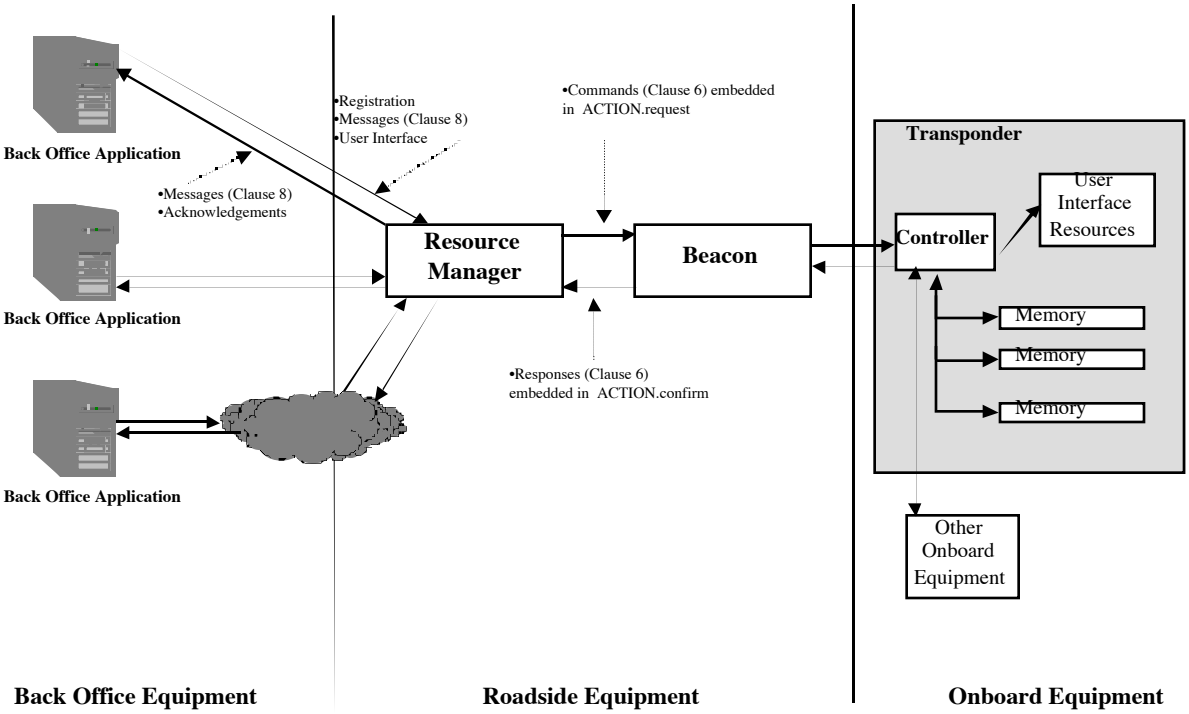

Figure 5—Resource manager relationships

As discussed in 1.1 and illustrated in Figure 1, an illustrative DSRC roadside system shall consist of a single VRC controller connected to one or more readers with each reader connected to one or more antennas. Compliant DSRC roadside installations shall provide a resource manager function as specified in this clause, and it is anticipated that the resource manager will in most cases be hosted within the VRC controller so that it may manage the transponder resources within an entire field of DSRC communications. However, this standard does not require any specific mapping of the resource manager to specific hardware.

Other equipment to accomplish functions such as automatic vehicle classification, weigh-in-motion, vehicle detection, etc., may exist at the roadside or in the roadway; however, this standard does not govern such equipment.

This clause specifies the characteristics of the resource manager function. A ITS application may include RSE other than that required for DSRC to perform functions such as vehicle classification or weigh-in-motion. This equipment is not governed by this standard.

7.1 Resource manager processing summary

The resource manager shall implement the following processing flow:

a) The resource manager shall initially accept registrations from BOAs that specify the set of DSRC resources to which each BOA requires access. This registration process is further specified in 7.2.2.

b) The resource manager shall then communicate with the connected beacons to configure each beacon's initial transactions with transponders that may pass within the beacon's communications zone. This communication process is further specified in 7.3.

c) When a transponder enters a beacon's communications zone, the beacon will notify the resource manager and may also transmit one or more memory images that have been retrieved from the transponder. The resource manager may then request the retrieval of additional memory pages and may provide access credentials required for the retrieval.

d) The resource manager shall then parse any received memory images and shall transmit the information contained within the memory images to the BOAs that have registered for it.

e) In response, the BOAs may request that specific messages be deleted or that additional messages be stored within the specified transponder memory region.

f) The resource manager shall then create a new memory image in response to the requests from the BOA and shall transmit that memory image to the beacon for storage within the transponder's memory region.

[Steps d), e), and f) are further specified in 7.4.]

g) The resource manager shall also accept requests from the BOAs that the transponder's UI be manipulated. The resource manager shall arbitrate those requests when received from multiple BOAs and shall then communicate appropriate commands to the beacon for transmission to the transponder. This process is specified further in 7.5.

7.2 BOA interface

Since the BOA is the ultimate user of the DSRC information, it is essential that this standard allow for such information transmission. However, the actual specification of the interface between the resource manager and the BOA is beyond the scope of this standard. This subclause, therefore, specifies the capabilities that are required within all implementations of that interface and also provides guidance on how the interface might be implemented.

7.2.1 Physical media

BOAs shall be provided with a communications link via which they will

— Be able to register themselves with the resource manager
— Be able to specify messages of interest it wants to receive
— Get the messages of interest as they are received
— Have a means of updating these messages upon receipt
— Have a means of specifying special actions that the resource manager must automatically perform upon receipt of these messages of interest

This standard does not govern the BOA interface; it is anticipated that the interface may be implemented using a variety of physical media. It is solely the responsibility of the system integrator and the user agency to select and implement the BOA interface using the media or combination of media appropriate for the cost, bandwidth, and physical limitations applicable to the DSRC installation.

7.2.2 Registration

Prior to receiving any transponder-derived information from the resource manager, the BOA shall register with the resource manager to define the types of information required and the conditions under which it should be transmitted. The classes of registration that are available and may be supported by the resource manager are listed in Table 30.

Table 30—Registration parameters

Registration information type	Description and usage
Unique Identifiers of Interest	Specifies a set of transponders that are of interest to the registering application; data reports shall be made to the BOA only for transponders with included unique identifiers.
Unique Identifiers Not of Interest	Specifies a set of transponders that are not of interest to the registering application; data reports shall never be made to the BOA for transponders with included unique identifiers.
Service Agencies of Interest	Specifies a set of service agencies that are of interest to the registering application; data reports shall be made to the BOA only for transponders with included service agencies.
Service Agencies Not of Interest	Specifies a set of service agencies that are not of interest to the registering application; data reports shall never be made to the BOA for transponders with included service agencies.
Beacon Identifiers	Specifies a set of beacon identifiers that are of interest to the registering application; data reports shall be made to the BOA only for transponders that are in communication with an included beacon.
Message Identifiers	Specifies a set of message identifiers that are of interest to the registering application; only those messages identifiers registered by the BOA shall be reported to the BOA when a transponder communicates with a beacon.
Transponder Resources	Specifies a set of transponder resources, such as memory or peripheral configurations, that must be present in the transponder; the registering BOA shall be notified of messages only from transponders with these resources identified in the read-only memory.
Memory Page Identifiers	Specifies a set of transponder memory pages that are of interest to the registering BOA; only information that is stored within the memory pages registered by the BOA shall be reported to the BOA when a transponder communicates with a beacon.

These registration parameters essentially restrict the volume of information passed using the methods described in 7.2.3. Specific resource manager implementations need not implement all the registration parameters. However, the full set of unfiltered information defined in 7.2.3 may be transmitted to the BOA.

7.2.3 Information transmission to the BOA

In response to the registration information received from the BOA, the resource manager shall utilize the beacon interface specified in 7.3 to elicit transfer of information from the transponders that communicate with connected beacons. As a result of this communication, the resource manager will obtain one or more memory images. The resource manager shall then determine whether the communicating transponder matches the previously received registration filters.

If the transponder meets the registration constraints, then the resource manager shall process the memory images for which the BOA has registered to extract that information. For the read-only memory region, this processing shall consist solely of formatting the binary information contained within the memory region into appropriate data structures for transmission. For all other memory regions, the resource manager shall parse the memory region to extract individual messages. Each message shall then be compared to the list of message identifiers that have been registered with the BOA. If the BOA has registered for a message that is present in the memory image, then the message shall be transmitted to the BOA. Messages may be reformatted for transmission to the BOA.

Once the resource manager has extracted all information within the transponders for which the BOA has registered, the resource manager shall assign an identifier to each message that will be transmitted to the BOA. The extracted information, including all messages and their identifiers, shall then be transmitted to the BOA using the selected communication channel.

7.2.4 Message reception from the BOA

After the BOA receives information from the resource manager that has been extracted from memory regions within a communicating transponder, the BOA may choose to alter the information stored within a specific memory region of a transponder. Two methods of alteration shall be provided:

a) The BOA may request that a message be deleted from a specific memory region. This alteration is accomplished by specifying the corresponding message sequence number.

b) The BOA may request that an additional message be stored within the transponder's memory region. This alteration is accomplished by creating the desired message and then passing it to the resource manager.

These memory image alteration requests shall be processed as specified in 7.4.

Due to vehicle movement and other factors, it is possible that the resource manager will be unable to accomplish the requested memory image alterations. If this condition is detected by the resource manager, it shall be reported to the requesting BOA.

7.2.5 UI requests from the BOA

After the BOA receives information from the resource manager that has been extracted from a communicating transponder, the BOA may choose to manipulate the transponder's UI. The resource manager shall provide service methods that allow the BOA to individually manipulate each of the UI resources defined in Clause 5. These service requests shall be processed as defined in 7.5.

Due to vehicle movement and other factors, it is possible that the resource manager will be unable to accomplish the requested UI actions. If this condition is detected by the resource manager, it shall be reported to the requesting BOA.

7.2.6 Predefined transponder sessions

In some cases, the BOA may require the capability to predefine sequences of actions (which can be treated as a single logical action) that should be taken by the resource manager upon arrival of a transponder. Such a sequence is only executed when the transponder is within the beacon's communications field. This is likely to occur when the BOA is connected to the resource manager via a low-speed interface. In such a case, it is unlikely that the communication of individual sequential commands can be successfully accomplished during the period of time in which the vehicle is within the beacon's communications field.

The resource manager may provide for this requirement by implementing registration messages, configuration files, or custom software that implements the required sequences of actions. If this capability is provided, the resource manager shall still report all pertinent information received from or transmitted to the transponder using the processes defined in 7.2.4 and 7.2.5.

7.3 Beacon interface (nonmandatory)

It is anticipated that in many cases the resource manager will be implemented within a VRC controller that is physically distinct from, but connected to, the beacon. In this case, it will be necessary for the VRC controller to communicate with the beacon to control the beacon configuration, elicit information from each tran-

sponder that passes through the beacon's communication region, and transfer information to the beacon for storage on the transponders. However, this standard recognizes that in some cases the VRC controller and the beacon will actually be a single piece of equipment, hosting both the resource manager and the beacon functionality.

This standard anticipates that future efforts may be undertaken to specify a standard interface between the resource manager and the beacon. Absent such a standard, the following guidelines are recommended:

— The interface will typically correspond to the interface between the application layer and lower layer service as described in Clause 9.

— The interface should allow the resource manager to initiate, terminate, monitor, and otherwise control the communications channel with the beacon.

— The interface should allow the resource manager to query the configuration and health of the beacon and any connected equipment.

— The interface should allow the resource manager to mute the beacon, i.e., cause the beacon to cease RF transmission and reception.

— The interface should allow the resource manager to specify that lower layer actions target a specific beacon.

— The interface should allow the beacon to transmit to the resource manager a transponder command response (specified in Clause 6).

— In some DSRC systems, the vehicle speed and size of the beacon communications region will constrain the period of time during which the beacon may communicate with a transponder. The physical capabilities of the interface between the resource manager and the beacon should accommodate the correspondingly required transmission speeds.

7.4 Memory page management

The resource manager shall arbitrate, manage, and control all transponder memory regions that may be modified using the commands listed in Clause 6. This capability shall be provided as follows, and in such a way as to meet the requirements specified in 7.2.3 and 7.2.4:

— The resource manager shall retrieve all memory regions that have been previously requested as part of BOA registrations. The resource manager shall not retrieve or process in any way memory regions that have not been requested as part of a BOA registration. The resource manager shall not write to pages that are unchanged.

— The resource manager shall parse all retrieved memory regions to isolate the messages stored within them.

— The resource manager shall transmit the messages to the BOAs that have previously registered for the messages. Messages that are stored within pages that have been reserved to a specific agency shall only be transmitted to BOAs that specify that page identifier as part of their registration parameters.

— If a BOA requests that messages be deleted from or added to a page, then the resource manager shall perform the memory consolidation process defined in 7.4.1. The resource manager shall only accept requests for changes to reserved memory pages from BOAs that specified the reserving page identifier as part of their registration parameters. If a page is not reserved, i.e., is a public page, then the resource manager shall accept requests for changes to that page from any (and potentially multiple) BOAs that request access to that page as part of their registration parameters.

— After performing the memory consolidation process, the resource manager shall transmit the modified memory image to the beacon for storage on the transponder. The resource manager may use the Write Memory Page, Append Message, or Write Circular Queue command as appropriate and as supported by the transponder.

7.4.1 Memory consolidation

After the resource manager has retrieved a memory region from a transponder, parsed the messages from that region, passed the messages to BOAs that have registered for them, and received requests for memory image updates from the BOAs, the resource manager will have three sets of information corresponding to the memory region:

— Existing messages. A list of messages that were stored within the memory region when it was received.

— Obsolete messages. A list of message sequence numbers corresponding to messages that one or more BOAs have requested be deleted.

— Requested messages. A list of messages that BOAs have requested be added.

The resource manager shall then create a list of new messages by performing the following steps:

a) Each existing message shall be analyzed to determine whether it has expired. This shall be determined by comparing the message expiration date stored within the specified message to the date at which the analysis is performed.

b) Each existing message that has not expired shall then be placed on the new message list if it is not present on the obsolete message list (in the resource manager).

c) If room is available within the designated transponder memory region, all requested messages shall be added to the new message list. If insufficient room exists for all requested messages, the resource manager may add a subset of the requested message list to the new message list using a site-specific prioritization algorithm. The BOAs shall be notified if insufficient memory space exists for a message.

Once the new message list has been created, it shall be used to generate a new memory image of the designated transponder memory region.

7.5 UI management

The resource manager shall accept requests for UI services from the communicating BOAs as specified in 7.2.5. When the resource manager is servicing only a single BOA or when no conflicts exist in the UI requests received from multiple communicating BOAs, then the resource manager shall directly translate the UI service requests into the transponder UI commands specified in Clause 6.

In some cases, conflicts may arise between BOAs that request access to the same UI resources. For example, an Electronic Toll Collection application might request illumination of the green lamp, while a Border Crossing application requests illumination of a red lamp. Site-specific rules shall be established for the arbitration of conflicting UI directives.

8. ITS application messages

8.1 Message concepts

Application messages are the data constructs that provide for communication between applications and positive identification of vehicles, containers, chassis, etc. Each application area, such as Electronic Toll Collection or Border Clearance, has messages that are unique to the application. In addition, there are utility messages that may be utilized in multiple applications. This clause defines the general format of messages, specific message sets for each application area, and data element definitions for all messages.

8.1.1 Message format

Each application message shall consist of a header and a body. The header component is defined across all applications. The message body consists of the application data fields. Message body content is unique to each message type within each application. The specific data elements that are used in message headers are formally defined in 8.2.

8.1.2 Message encoding

The encoding and decoding of message fields into transfer syntax shall be performed by the application. All messages shall be encoded according to ASN.1 Packed Encoding Rules (PER), unaligned, as specified in ISO/IEC 8825-2:1996. The application may also encrypt the message body using an application-specific technique.

The specification of each application message includes an ASN.1 value specification of the message body with a standard header. Following the sample value assignments is a bit-level layout of the resulting encoding. Within the bit-level layouts, a period (.) is used to indicate octet alignment and an “x” is used as a bit placeholder with no specific value.

NOTE—Annex B provides considerations used for the ASN.1 PER encodings as they have been applied within this standard.

8.2 Message headers

Two forms of the message header exist. The standard, or “long form,” header, which is 5 bytes long, is used to prefix messages that are stored in transponder memory pages that are at least 512 bits in length. The “short form” header, which is 3 bytes long, is used to prefix messages that are stored in transponder memory pages that are less then 512 bits in length. Message headers shall not be encrypted. Tables 31 and 32 provide a layout of each message type.

Table 31—Standard message format

Application Identifier	Message Identifier	Message Expiration Date	Message Length	Error Detect Code	Message Body
Bits 0 .. 5	Bits 6 .. 11	Bits 12 .. 23	Bits 24 .. 31	Bits 32 .. 39	Variable

Table 32—Short message format

Message Identifier	Message Expiration Date	Message Length	Error Detect Code	Message Body
Bits 0 .. 4	Bits 5 .. 11	Bits 12 .. 15	Bits 16 .. 23	Variable

8.2.1 Standard message header

Table 33 specifies the fields and the permissible field values for the standard header. Messages using long headers are constrained to 255 bytes in length (excluding the header) by the definition of the Message Length field.

Table 33—Standard message header fields

Field	Field name	Type	Length	Values
1	Application Identifier	Integer	6 bits	See Table 34
2	Message Identifier	Integer	6 bits	See Tables 35 through 38
3	Message Expiration Date	Integer	12 bits	(0 .. 4095); Days since last decade; a message expiration date equal to hex(FFF) indicates that the message never expires.
4	Message Length	Integer	8 bits	(0 .. 255); Length of message body, in bytes (does not include header)
5	Error Detect Code	Bit string	8 bits	XOR Checksum of the message body

Standard message headers shall have an application identifier and a message identifier that uniquely identify the message type. Table 34 lists values for the application identifier. Table 35 through Table 38 list values for message identifiers within each application.

The Message Expiration Date field shall specify the point in time after which the message may be deleted. This field supports a message lifetime of up to 180 days. The field shall be interpreted as follows:

— If the message expiration date is less than or equal to 3652 and the current date within the decade is greater than the message expiration date, then the message shall be deleted.

— If the message expiration date is greater than 3652 and the current date within the decade is greater than 180 and less than 3472, then the message shall be deleted.

— If the message expiration date is greater than 3652 and the current date within the decade is less than 180, then the message shall be deleted if the message expiration date is less than the current date within the decade plus 3652.

Table 34—Application identifiers

Code	Application
0	Reserved
1	Electronic Toll and Traffic Management (ETTM)
2	Commercial Vehicle (CV) Management
3	Common Utility Messages
4 .. 59	Reserved
60	Private (Uncontrolled); Message identifiers associated with this application identifier are available for uncontrolled use
61	Private (Controlled); Message identifiers associated with this application identifier are available for registration
62 .. 63	Reserved

Table 35—Message identifiers: ETTM

Code	Message description
0	Reserved
1	Toll System Entry
2	Toll Vehicle Classification
3	Toll Variable Pricing
4	Toll System Enroll
5 .. 63	Reserved

Table 36—Message identifiers: CV Management

Code	Message description
0	Reserved
1	Border Trip Identification
2	Border Clearance Event
3	Border Lock Notification
4	Border Lock Status
5	Border Itinerary Identification
6	Commercial Motor Vehicle (CMV) Screening Identification
7	CMV Screening Event
8	CMV Screening Identification—Expanded
9	CMV Screening Event—Expanded
10 .. 63	Reserved

Table 37—Message identifiers: Common Utility Messages

Code	Message description
0	Reserved
1	Text String
2	RSE to Other OBE—Generic Data
3	Other OBE to RSE—Generic Data
4	End Of Data
5 .. 63	Reserved

Table 38—Message identifiers: Private Controlled

Code	Message description
0	Reserved
01 .. 63	Available for registration of private reserved messages

8.2.1.1 ASN.1 specification

```
Dsrcmsg-Header      ::=     SEQUENCE
{
        application-ID          INTEGER (0..63),        -- Dsrcmsg-ApplicationIdentity
        message-ID              INTEGER (0..63),        -- Dsrcmsg-MessageIdentifier
        message-date            INTEGER (0..4095),      -- Dsrcmsg-Date
        message-length          INTEGER (0..255),       -- Dsrcmsg-Length
        message-checksum        BIT STRING (SIZE(8))    -- Dsrcmsg-ErrorDetect
}
```

8.2.1.2 ASN.1 sample values

```
 Dsrcmsg-Header      ::=     SEQUENCE
 {                                                   -- Begin Standard Header
        application-ID          1,                   -- ETTM Application Identifier
        message-ID              1,                   - Toll Entry Message Identifier
        message-date            0,                   -- 1/1/1990
        message-length          0,                   -- 0 byte message body
        message-checksum        '00'H                -- XOR checksum (not calculated)
                                                     -- End Header / Begin Body
 }
```

8.2.1.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000001	application-ID
6	00.0001	message-ID
12	0000.00000000.	message-date
24	00000000	message-length
32	xxxxxxxx.	message-checksum
40		----- [end of Header]

8.2.2 Short message header

Short message headers shall use the short message identifier instead of the application identifier and message identifier. Table 39 lists the fields and the permissible field values for the short message header. Messages using short headers are constrained to 30 bytes in length (excluding the header) by the definition of the Message Length field.

Table 39—Short message header fields

Field	Field name	Type	Length	Values
1	Message Identifier	Integer	5 bits	See Table 40
2	Message Expiration Month	Integer	7 bits	(0 .. 127); Months since last decade. A value of 127 indicates that a message never expires.
3	Message Length	Integer	4 bits	(1 .. 15); Length of message body in byte pairs (does not include header)
4	Error Detect Code	Bit string	8 bits	XOR checksum of the message body

Short message headers shall use the short message identifier instead of the application identifier and message identifier. Table 40 lists the permissible values for short message identifiers.

Table 40—Short message identifiers

Code	Message description
0	Reserved
1	Toll Entry
2	Toll Vehicle Classification
3	Toll Variable Pricing
4	Toll System Enroll
5	Border Trip Identification
6	Border Clearance Event
7	Border Lock Notification
8	Border Lock Status
9	Border Itinerary Identification

Table 40—Short message identifiers (continued)

Code	Message description
10	Reserved by IEEE for future use
11	CMV Screening Clearance Event
12	Reserved by IEEE for future use
13	CMV Screening Clearance Event—Expanded
14	Utility Text String
15	Utility RSE to Other OBE
16	Utility Other OBE to RSE
17	Utility End Of Data
18..31	Reserved by IEEE for future use

The Message Expiration Month field shall specify the point in time after which the message may be deleted. This field supports a message lifetime of up to 6 mo. The field shall be interpreted as follows:

— If the message expiration month is less than or equal to 120 and the current month within the decade is greater than the message expiration month, then the message shall be deleted.

— If the message expiration month is greater than 120 and the current month within the decade is greater than 6 and less than 114, then the message shall be deleted.

— If the message expiration month is greater than 120 and the current date within the decade is less than 7, then the message shall be deleted if the message expiration month is less than the current month within the decade plus 120.

8.2.2.1 ASN.1 specification

```
Dsrcmsg-Header      ::=         SEQUENCE
{
      short-message-ID          INTEGER (0..31),          -- Dsrcmsg-ShortMessageIdentity
      message-month             INTEGER (0..4095),        -- Dsrcmsg-ShortDate
      message-length            INTEGER (1..16),          -- Dsrcmsg-ShortLength
      message-checksum          BIT STRING (SIZE(8))      -- Dsrcmsg-ErrorDetect
}
```

8.2.2.2 ASN.1 sample values

```
Dsrcmsg-Header      ::=         SEQUENCE
{                                                       -- Begin Short Header
      short-message-ID                1,                -- Toll Entry Message Identifier
      short-message-month             0,                -- 1/1/1990
      short-message-length            4,                -- 5 byte pair message body
      message-checksum                ‘00’H             -- XOR checksum (not calculated)
                                                        -- End Header / Begin Body
}
```

8.2.2.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	00001	short-message-ID
7	000.0000	short-message-date
12	0100.	short-message-length
16	xxxxxxxx.	message-checksum
24		----- [end of Header]

8.3 Message data elements

This subclause describes the data elements used in the body of the application message. Each data element is accompanied by the corresponding ASN.1 name. A list of all the data elements and their ASN.1 attributes is provided in Annex A.

8.3.1 Common application data elements

The following elements are defined by this standard and were designed for use across DSRC applications.

8.3.1.1 Timestamp (Dsrc-Time)

DSRC date/time values shall be expressed as a 4 byte integer indicating the number of seconds since January 1, 1970 GMT.

8.3.1.2 Beacon Identifier (Beacon-Identity)

The roadside beacon shall have a unique identifier consisting of a 16 bit identifier registered to that agency followed by a 16 bit agency-unique serial number.

8.3.1.3 Transponder Identifier (Transponder-Identity)

The transponder shall have a unique identifier (see 5.2.18) consisting of 40 bits, which represent the Manufacturer Identifier and Serial Number fields (and associated subfields) defined for read-only memory (see 5.2)

8.3.2 Data elements—application-specific

Application data elements are specified using ASN.1 syntax in Annex A.

8.4 Electronic Toll Collection (ETC) message set

Table 41 summarizes the messages that have been defined for the ETTM ETC application. This subclause details the specific formats, conditions, and uses for each message.

8.4.1 Toll System Entry message

The Toll System Entry message contains information to identify where and when a vehicle entered the tolling system. It is generated by the entry beacon, stored in the transponder memory, and received by the exit beacon. The timestamp shall not be modified by the transponder after the message is written. See Table 42.

Table 41—ETC message summary

Message name	Description
Toll System Entry	Identifies where and when a vehicle entered the system; this message may also be used for traffic probes
Vehicle Classification	Describes the vehicle characteristics related to tolling
Variable Pricing	Provides toll information to charge a user under variable pricing
System Enroll	Establishes enrollment with a service provider

Table 42—Toll System Entry message

Field	Data element name	Type	Constraint
1	Tollevent-Timestamp	Integer	Dsrc-Time
2	Beacon-Identity	Bit string	16 bit Agency + 16 bit Serial

8.4.1.1 ASN.1 specification

```
Toll-Entry-Message ::=      SEQUENCE
{
      header                Dsrcmsg-Header,
      timestamp             Dsrc-Time,                -- Tollevent-Timestamp
      beaconID              BIT STRING (SIZE(32))     -- Beacon-Identity
}
```

8.4.1.2 ASN.1 sample values

```
Toll-Entry-Message ::=      SEQUENCE
{                                                     -- Begin Standard Header
      application-ID        1,                        -- ETTM Application Identifier
      message-ID            1,                        -- Toll Entry Message Identifier
      message-date          0,                        -- 1/1/1990
      message-length        8,                        -- 8 byte message body
      message-checksum      '00'H,                    -- XOR checksum (not calculated)
                                                      -- End Header / Begin Body
      timestamp             0,                        -- 00:00:00 1/1/1970 GMT
      beaconID              '00020100'H               -- Agency=2; Serial=256
}
```

8.4.1.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000001	application-ID
6	00.0001	message-ID
12	0000.00000000.	message-date
24	00001000	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00000000.00000000.	timestamp
	00000000.00000000.	
72	00000000.00000010.	beaconID - Agency component
	00000001.00000000.	beaconID - Serial component
104		----- [end of Body]

8.4.2 Vehicle Classification message

The Vehicle Classification message contains information that describes vehicle characteristics (e.g., vehicle profile and vehicle axles) related to toll processing. See Table 43.

Table 43—Vehicle Classification message

Field	Data element name	Type	Constraint
1	Vehicle-Characteristics	Bit string	Size (16)
2	Agency-Specifics	Octet string	Size (4)

8.4.2.1 ASN.1 specification

```
Vehicle-Classification-Message          ::=         SEQUENCE
{
      header                 Dsrcmsg-Header,
      characteristics        INTEGER (0..255),              -- Vehicle-Characteristics
      agency-data            OCTET STRING (SIZE (4))        -- Agency-Specifics
}
```

8.4.2.2 ASN.1 sample values

```
Sample-Message Vehicle-Classification-Message          ::=       SEQUENCE
{                                                                -- Begin Standard Header
      application-ID          1,                                 -- ETTM Application Identifier
      message-ID              2,                                 -- Vehicle Classification Message Identifier
      message-date            0,                                 -- 1/1/1990
      message-length          5,                                 -- 5 byte message body
      message-checksum        ‘00’H,                             -- XOR checksum (not calculated)
                                                                 -- End Header / Begin Body
      characteristics         0,                                 --
      agency-data                                                --
}
```

8.4.2.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000001	application-ID
6	00.0010	message-ID
12	0000.00000000.	message-date
24	00000101	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00000000.	characteristics
48	xxxxxxxx.xxxxxxxx	agency-data
80	xxxxxxxx.xxxxxxxx	----- [end of Body]

8.4.3 Variable Pricing message

The Variable Pricing message contains toll information needed to charge a user under variable pricing. It is generated by the entry beacon, stored in the transponder read/write memory, and received by the exit beacon. See Table 44.

Table 44—Variable Pricing message

Field	Data element name	Type	Constraint
1	Tollevent-Timestamp	Integer	Dsrc-Time
2	Beacon-Identity	Bit string	16 bit Agency + 16 bit Serial
3	Toll-Amount	Integer	(0 .. 65535)

8.4.3.1 ASN.1 specification

```
Pricing-Message     ::=     SEQUENCE
{
      header
      timestamp             Dsrc-Time,                  -- Tollevent-Timestamp
      beaconID              BIT STRING (SIZE(32))       -- Beacon-Identity
      toll-amount           INTEGER                     -- Toll-Amount
}
```

8.4.3.2 ASN.1 sample values

```
Pricing-Message     ::=     SEQUENCE
{                                                     -- Begin Standard Header
      application-ID        1,                        -- ETTM Application Identifier
      message-ID            3,                        -- Variable Pricing Message Identifier
      message-date          0,                        -- 1/1/1990
      message-length        10,                       -- 10 byte message body
      message-checksum      '00'H,                    -- XOR checksum (not calculated)
                                                      -- End Header / Begin Body
```

```
        timestamp           0,                  -- 00:00:00 1/1/1970 GMT
        beaconID            '00020100'H,        -- Agency=2; Serial=256
        toll-amount         100                 -- Toll amount = $1.00
}
```

8.4.3.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000001	application-ID
6	00.0011	message-ID
12	0000.00000000.	message-date
24	0000.1010	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00000000.00000000	timestamp
	00000000.00000000	
72	00000000.00000010.	beaconID - Agency component
	00000001.00000000.	beaconID - Serial component
104	xxxxxxxx.xxxxxxxx	toll-amount
120		----- [end of Body]

8.4.4 System Enroll message

The System Enroll message contains the information needed to enroll a transponder in a service provider's system. It is generated by the RSE, stored in the transponder read/write memory, and received by subsequent ETC BOAs. See Table 45.

Table 45—System Enroll message

Field	Data element name	Type	Constraint
1	Tollenroll-Timestamp	Integer	Dsrc-Time
2	Service-Agency	Bit String	16 bit Agency
3	Digital-Signature	Bit String	Size (64)

8.4.4.1 ASN.1 specification

```
System-Enroll-Message       ::=         SEQUENCE
{
        header
        timestamp           Dsrc-Time,                  -- Tollenroll-Timestamp
        service-agency      BIT STRING (SIZE(16)),      -- Enrolling service agency
        digital-signature   BIT STRING (SIZE(64))       -- Digital-Signature
}
```

8.4.4.2 ASN.1 sample values

```
Pricing-Message     ::=     SEQUENCE
{                                           -- Begin Standard Header
      application-ID        1,              -- ETTM Application Identifier
      message-ID            4,              -- System Enroll Message Identifier
      message-date          0,              -- 1/1/1990
      message-length        14,             -- 16 byte message body
      message-checksum      '00'H,          -- XOR checksum (not calculated)
                                            -- End Header / Begin Body
      timestamp             0,              -- 00:00:00 1/1/1970 GMT
      service-agency        '0002'H,        -- Agency=2
      digital-signature     0               -- Digital-Signature = 0
}
```

8.4.4.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000001	application-ID
6	00.0100	message-ID
12	0000.00000000.	message-date
24	0000.1110	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00000000.00000000	timestamp
	00000000.00000000	
72	00000000.00000010.	service agency
88	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	digital-signature
120	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	toll-amount
152		----- [end of Body]

8.5 Commercial Vehicle Operations (CVO) Border Clearance message set

Table 46 summarizes the messages that have been defined for the CVO Border Clearance application. This subclause details the specific formats, conditions, and uses for each message.

Table 46—CVO Border Clearance message summary

Message name	Description
Trip Identification Number	Transmits the unique trip load number
Border Clearance Event	Reports clearance event data to the vehicle
Electronic Lock Notification	Notifies roadside that the vehicle has electronic locks
Electronic Lock Status	Provides roadside with status of electronic lock
Itinerary Verification	Shows percent likelihood that vehicle maintained its itinerary
Warning/Notification	Indicates special attention for cargo or onboard sensor

8.5.1 Trip Identification Number message

The Trip Identification Number message contains the unique trip load number, consisting of the carrier's Dunn & Bradstreet number (DUNS) and a unique suffix. It is generated by a portable transfer device [e.g., a notebook computer or personal digital assistant (PDA)], stored in the transponder memory, and received by the beacon at a border clearance location. See Table 47.

Table 47—Trip Identification Number message

Field	Data element name	Type	Constraint
1	Triplоad-DunsNumber	NumericString	Size (9)
2	Triplоad-CarrierSerial	NumericString	Size (6)

8.5.1.1 ASN.1 specification

```
Trip-Identification-Message  ::=        SEQUENCE
{
      header                Dsrcmsg-Header,
      duns-number           NumericString (SIZE(9)),  -- Triplоad-DunsNumber
      carrier-serial        NumericString (SIZE(6))   -- Triplоad-CarrierSerial
}
```

8.5.1.2 ASN.1 sample values

```
Trip-Identification-Message ::=          SEQUENCE
{                                                    -- Begin Standard Header
      application-ID        2,                       -- CVO Application Identifier
      message-ID            1,                       -- Trip Identification Message Identifier
      message-date          0,                       -- 1/1/1990
      message-length        8,                       -- 8 byte message body
      message-checksum      '00'H,                   -- XOR checksum (not calculated)
                                                     -- End Header / Begin Body
      duns-number           123456789,
      carrier-serial        123456
}
```

8.5.1.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000001	application-ID
6	00.0001	message-ID
12	0000.00000000.	message-date
24	00001000.	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00010010.00110100.01010110.	duns-number
64	01111000.1001	
76	0001.00100011.01000101.0110	carrier-serial
100	xxxx.	octet alignment pad
104		----- [end of Body]

8.5.2 Border Clearance Event message

The Border Clearance Event message reports border clearance information to the vehicle. It is generated by the border crossing location DSRC controller, stored in the transponder memory, and received by the beacon at another border clearance location. See Table 48.

Table 48—Border Clearance Event message

Field	Data element name	Type	Constraint
1	Beacon-Identity	Bit string	Size (32)
2	Borderevent-Timestamp	Integer	Dsrc-Time
3	Borderevent-DriverClearance	Boolean	Go/True - NoGo/False
4	Borderevent-DriverClearanceFlag	Boolean	Valid/True - Invalid/False
5	Borderevent-CargoClearance	Boolean	Go/True - NoGo/False
6	Borderevent-CargoClearanceFlag	Boolean	Valid/True - Invalid/False
7	Borderevent-TractorClearance	Boolean	Go/True - NoGo/False
8	Borderevent-TractorClearanceFlag	Boolean	Valid/True - Invalid/False
9	Borderevent-ReserveClearance	Boolean	Reserved for future use
10	Borderevent-ReserveFlag	Boolean	Reserved for future use
11	Transponder-DigitalSignature	Bit string	Size (64)

8.5.2.1 ASN.1 specification

```
Border-Clearance-Event-Message      ::=          SEQUENCE
{
      header                  Dsrcmsg-Header,              -- Standard Header
      beacon-ID               BIT STRING SIZE((32)),       -- Beacon-Identity
      timestamp               Dsrc-Time,                   -- Borderevent-Timestamp
      driver-clearance        BOOLEAN,                     -- Borderevent-DriverClearance
      driver-clearance-flag   BOOLEAN,                     -- Borderevent-DriverClearanceFlag
      cargo-clearance         BOOLEAN,                     -- Borderevent-CargoClearance
      cargo-clearance-flag    BOOLEAN,                     -- Borderevent-CargoClearanceFlag
      tractor-clearance       BOOLEAN,                     -- Borderevent-TractorClearance
      tractor-clearance-flag  BOOLEAN,                     -- Borderevent-TractorClearanceFlag
      reserve-clearance       BOOLEAN,                     -- reserved field
      reserve-flag            BOOLEAN,                     -- reserved field
      digital-signature       BIT STRING (SIZE(64))        -- Transponder-DigitalSignature
}
```

8.5.2.2 ASN.1 sample values

```
Border-Clearance-Event-Message      ::=          SEQUENCE
{                                                         -- Begin Standard Header
      application-ID          2,                          -- CVO Application Identifier
      message-ID              2,                          -- Border Clearance Event Message ID
      message-date            0,                          -- 1/1/1990
      message-length          17,                         -- 17 byte message body
      message-checksum        '00'H,                      -- XOR checksum (not calculated)
                                                          -- End Header / Begin Body
      beaconID                '00020100'H,                -- Agency=2; Serial=256
      timestamp               0,                          -- 00:00:00 1/1/1970 GMT
      driver-clearance        TRUE,                       -- GO
      driver-clearance-flag   TRUE,                       -- VALID
      cargo-clearance         TRUE,                       -- GO
      cargo-clearance-flag    TRUE,                       -- VALID
      tractor-clearance       TRUE,                       -- GO
      tractor-clearance-flag  TRUE,                       -- VALID
      reserve-clearance       FALSE,                      -- Reserved - NOGO
      reserve-flag            FALSE,                      -- Reserved - INVALID
      digital-signature       0                           -- Transponder-DigitalSignature
}
```

8.5.2.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000010	application-ID
6	00.0010	message-ID
12	0000.00000000.	message-date
24	00010001	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00000000.00000010.	beaconID - Agency component
56	00000001.00000000.	beaconID - Serial component
72	00000000.00000000.00000000.00000000.	timestamp
104	1	driver-clearance
105	1	driver-clearance-flag
106	1	cargo-clearance
107	1	cargo-clearance-flag
108	1	tractor-clearance
109	1	tractor-clearance-flag
110	0	reserve-clearance
111	0.	reserve-flag
112	00000000.00000000.00000000.00000000.	digital-signature
144	00000000.00000000.00000000.00000000.	
176		----- [end of Body]

8.5.3 Electronic Lock Notification message

The Electronic Lock Notification message notifies the roadside that the vehicle contains electronic locks. It is generated by a portable transfer device (e.g., a notebook computer or PDA), stored in the transponder memory, and received by the beacon at a border clearance location. See Table 49.

Table 49—Electronic Lock Notification message

Field	Data element name	Type	Constraint
1	Lock-Quantity	Integer	(0 .. 15)
2	Lock-Identity	Bit string	Size (40); Transponder-Identity; the value of the preceding Lock-Quantity field indicates the number of occurrences of this field
3	Transponder-DigitalSignature	Bit string	Size (64)

8.5.3.1 ASN.1 specification

```
Border-Lock-Notification-Message   ::=      SEQUENCE
{
      header              Dsrcmsg-Header,              -- Standard Header
      lock-quantity       INTEGER (0..15)              -- Lock-Quantity
      lock-ID             BIT STRING (SIZE(40)),       -- Lock-Identity (Transponder ID)
      digital-signature   BIT STRING (SIZE(64))        -- Transponder-DigitalSignature
}
```

8.5.3.2 ASN.1 sample values

```
Border-Lock-Notification-Message   ::=          SEQUENCE
{                                                       -- Begin Standard Header
      application-ID      2,                            -- CVO Application Identifier
      message-ID          3,                            -- Electronic Lock Status Message ID
      message-date        0,                            -- 1/1/1990
      message-length      14,                           -- 14 byte message body
      message-checksum    '00'H,                        -- XOR checksum (not calculated)
                                                        -- End Header / Begin Body
      lock-quantity       1,                            -- Number of locks
      lock-ID             '0080000040'H,                -- Lock-Identity Man=2; Serial=4
      digital-signature   0                             -- Transponder-DigitalSignature
}
```

8.5.3.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000010	application-ID
6	00.0011	message-ID
12	0000.00000000.	message-date
24	00001110	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	0001	lock-quantity
44	0000.000010	lock-ID Manufacturer =2
54	00.00000000.00000000.00000100.	lock-ID Serial = 4
80	0000	lock-ID Reserved
84	0000.00000000.00000000.00000000.0000	digital-signature
116	0000.00000000.00000000.00000000.0000	
148	xxxx.	fill
152		----- [end of Body]

8.5.4 Border Lock Status message

The Electronic Lock Status message notifies the roadside regarding the status (e.g., Open, Close, Bad) of an electronic lock. It is generated by an electronic lock and received by the beacon at a border clearance location. See Table 50.

Table 50— Electronic Lock Status message

Field	Data element name	Type	Constraint
1	Lock-Identity	Bit string	Size (40); Transponder-Identity
2	Borderevent-Timestamp	Integer	Size (32); Dsrc-Time
3	Lock-CurrentStatus	Integer	(0 .. 7)
4	Lock-HistoryCount	Integer	(0 .. 15); the value of this field indicates the number occurrences of Fields 5 and 6
5	Lock-Status	Integer	(0 .. 7) 0 Open 1 Closed 2 Bad
6	Borderevent-Timestamp	Integer	Size (32); Dsrc-Time
7	Transponder-DigitalSignature	Bit string	Size (64)

8.5.4.1 ASN.1 specification

```
Border-Lock-Status-Message        ::=     SEQUENCE
{
      header              Dsrcmsg-Header,          -- Standard Header
      lock-ID             BIT STRING (SIZE(40)),   -- Lock-Identity (Transponder ID)
      border-time         Dsrc-Time                -- Borderevent-Timestamp
      lock-status         INTEGER (0..7)           -- Lock-Status
      lock-quantity       INTEGER (0..15)          -- Lock-HistoryCount
      lock-status-h1      INTEGER (0..7)           -- Lock-Status
      border-time-h1      Dsrc-Time                -- Borderevent-Timestamp
      digital-signature   BIT STRING (SIZE(64))    -- Transponder-DigitalSignature
}
```

8.5.4.2 ASN.1 sample values

```
Border-Lock-Status-Message::=              SEQUENCE
{                                                    -- Begin Standard Header
      application-ID      2,                         -- CVO Application Identifier
      message-ID          4,                         -- Electronic Lock Status Message ID
      message-date        0,                         -- 1/1/1990
      message-length      23,                        -- 23 byte message body
      message-checksum    '00'H,                     -- XOR checksum (not calculated)
                                                     -- End Header / Begin Body
      lock-ID             '0080000040'H,             -- Lock-Identity Man=2; Serial=4
      timestamp           0,                         -- 00:00:00 1/1/1970 GMT
      lock-status         0,                         -- Lock-Status = 0; Open
      lock-quantity       1                          -- Lock-HistoryCount = 1
      lock-status-h1      1                          -- Lock Status = 1; Close
      border-time-h1      0                          -- Borderevent-Timestamp
      digital-signature   0                          -- Transponder-DigitalSignature
}
```

8.5.4.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000010	application-ID
6	00.0100	message-ID
12	0000.00000000.	message-date
24	00001001	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	0000	lock-ID Reserved
44	0000.000010	lock-ID Manufacturer =2
54	00.00000000.00000000.00000100.	lock-ID Serial = 4
80	00000000.00000000.00000000.00000000.	timestamp
112	000	lock-status
115	0001	lock-count
119	0.01	lock-status-h1
122	000000.00000000.00000000.00000000.00	border-time-h1
154	000000.00000000.00000000.00000000.	digital-signature
184	00000000.00000000.00000000.0000000.00	
218	xxxxxx.	fill
224		----- [end of Body]

8.5.5 Itinerary Verification message

The Itinerary Verification message notifies the border clearance roadside on the percent likelihood that the vehicle maintained its preplanned itinerary. It is generated by an onboard computer and received by the beacon at a border clearance location. See Table 51.

Table 51—Itinerary Verification message

Field	Data element name	Type	Constraint
1	Vehicle-ItineraryQuality	Integer	(0 .. 100); 100 indicates the highest confidence that the vehicle has followed a specified itinerary. 0 indicates a high confidence that the vehicle has significantly deviated from a specified itinerary. Other values indicate intermediate levels of confidence.
2	Borderevent-Timestamp	Integer	Size (32); Dsrc-Time
3	Transponder-DigitalSignature	Bit string	Size (64)

8.5.5.1 ASN.1 specification

```
Border-Itinerary-Message    ::=        SEQUENCE
{
      header                Dsrcmsg-Header,          -- Standard Header
      itinerary-quality     INTEGER (0..255),        -- Vehicle-ItineraryQuality; Max=100
      border-time           0                        -- Borderevent-Timestamp
      digital-signature     BIT STRING (SIZE(64))    -- Transponder-DigitalSignature
}
```

8.5.5.2 ASN.1 sample values

```
Border-Itinerary-Message    ::=              SEQUENCE
{                                                      -- Begin Standard Header
      application-ID        2,                         -- CVO Application Identifier
      message-ID            5,                         -- Border Itinerary Message ID
      message-date          0,                         -- 1/1/1990
      message-length        13,                        -- 13 byte message body
      message-checksum      '00'H,                     -- XOR checksum (not calculated)
                                                       -- End Header / Begin Body
      itinerary-quality     64,                        -- Itinerary Quality = 64%
      border-time           0,                         -- Borderevent-Timestamp
      digital-signature     0,                         -- Digital Signature = 0
}
```

8.5.5.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000010	application-ID
6	00.0101	message-ID
12	0000.00000000.	message-date
24	00001101	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	01000000	itinerary-quality
48	00000000.00000000.00000000.00000000.	border-time
80	00000000.00000000.00000000.00000000.	digital-signature
112	00000000.00000000.00000000.00000000.	
144		----- [end of Body]

8.6 CVO Electronic Screening message set

Table 52 summarizes the messages that have been defined for the CVO Electronic Screening (also referred to as Mainline Screening) application. This subclause details the specific formats, conditions, and uses for each message.

Table 52—CVO Electronic Screening message summary

Message name	Description
CMV Screening Identification	Sets and sends vehicle and cargo data
CMV Screening Event	Reports clearance event data to the vehicle
CMV Screening Identification Expanded	Sets and sends vehicle and cargo data
CMV Screening Event Expanded	Reports clearance event data to the vehicle

8.6.1 CMV Screening Identification message

The CMV Screening Identification message provides the information necessary to conduct electronic screening of CVs at CV check stations in North America. It is generated by a portable transfer device (e.g., a notebook computer or PDA), stored in the transponder memory, and received by the beacon at a CV check station. It is transferred from the transponder to the beacon at mainline speeds. See Table 53.

Table 53—CMV Screening Identification message

Field	Data element name	Type	Constraint
1	Carrier-Identity	IA5string	Size (20); this field may be repeated up to 3 times
2	Vehicle-Identity	IA5string	Size (17); VIN
3	Vehicle-CargoType	IA5string	Size (6); Hazmat Code

8.6.1.1 ASN.1 specification

```
CMV -Clearance-Identification-Message      ::=      SEQUENCE
{
      header              Dsrcmsg-Header,        -- Standard Header
      carrier-ID          IA5String (SIZE(4)),   -- Carrier-Identity
      vin                 IA5String (SIZE(17)),  -- Vehicle-Identity
      cargo-code          IA5String (SIZE(6))    -- Vehicle-CargoType
}
```

8.6.1.2 ASN.1 sample values

```
CMV-Clearance-Identification-Message      ::=      SEQUENCE
{                                                   -- Begin Standard Header
      application-ID      2,                        -- CVO Application Identifier
      message-ID          6,                        -- Clearance ID Message Identifier
      message-date        0,                        -- 1/1/1990
      message-length      43,                       -- 43 byte message body
      message-check-      ‘00’H,                    -- XOR checksum (not calculated)
sum
                                                    -- End Header / Begin Body
      carrier-ID          64,                       --
      vin                 0,                        --
      cargo-code          0,                        --
}
```

8.6.1.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000010	application-ID
6	00.0110	message-ID
12	0000.00000000.	message-date
24	00101011	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	carrier-identity
72	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	border-time
104	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	
136	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	
168	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	
200	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	vehicle-identity
232	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	
264	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	
296	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	
328	xxxxxxxx.	
336	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	vehicle-cargo-type
368	xxxxxxxx.xxxxxxxx.	
384		----- [end of Body]

8.6.2 CMV Screening Event message

The CMV Screening Event message provides information documenting critical parameters of the last screening event. It is generated by the CV check station computer via a DSRC controller, stored in the transponder memory, and received by the beacon at a CV check station. See Table 54.

Table 54—CMV Screening Event message

Field	Data element name	Type	Constraint
1	Vehicle-GrossWeight	Integer	(0..16383); measured vehicle weight in 10 kg increments
2	Scale-Type	Integer	(1 .. 15); see Table 55
3	Vehicle-AxleNumber	Integer	(2 .. 17); measured number of vehicle axles
4	Beacon-Identity	Bit String	Size (32)
5	Mainlineevent-Timestamp	Integer	Size (32); Dsrc-Time
6	Mainlineevent-Bypass	Boolean	Go/True 1=Bypass/True, 0=Pullin/False

Table 55—Scale types

Values	Definitions
1	Jurisdictional weight
2	Mainline WIM
3	Ramp sorter WIM
4	Slow rollover WIM
5	Static scale weight
15	Operator-entered weight

8.6.2.1 ASN.1 specification

```
CMV-Screening-Event-Message     ::=         SEQUENCE
{
        header                  Dsrcmsg-Header,         -- Standard Header
        gross-weight            INTEGER                 -- Vehicle-GrossWeight
        scale-type              INTEGER                 -- Scale-Type
        axle-number             INTEGER                 -- Vehicle-AxleNumber
        beacon-ID               BIT STRING (SIZE(32)),  -- Beacon-Identity
        timestamp               Dsrc-Time               -- Mainlineevent-Timestamp
        pullin-clearance        BOOLEAN                 -- Mainlineevent-PullinClearance
}
```

8.6.2.2 ASN.1 sample values

```
CMV-Screening-Event-Message     ::=         SEQUENCE
{                                                       -- Begin Standard Header
        application-ID          2,                      -- CVO Application Identifier
        message-ID              7,                      -- Screening Event Message Identifier
        message-date            0,                      -- 1/1/1990
        message-length          12,                     -- 12 byte message body
        message-checksum        '00'H,                  -- XOR checksum (not calculated)
                                                        -- End Header / Begin Body
        gross-weight            500,                    -- 5000 Kg
        scale-type              1,                      -- Jurisdictional weight
        axle-number             4                       -- Vehicle-AxleNumber
        beacon-ID               '00020100'H,            -- Agency=2; Serial=256
        timestamp               0,                      -- 00:00:00 1/1/1970 GMT
        pullin-clearance        TRUE                    -- Go
}
```

8.6.2.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000010	application-ID
6	00.0111	message-ID
12	0000.00000000.	message-date
24	00001100	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00000111.110100	gross-weight
54	000100	scale-type
58	00100	axle-number
64	00000000.00000010.	beacon-ID - Agency component
80	00000001.00000000.	beacon-ID - Serial component
96	00000000.00000000.00000000.00000000.	timestamp
128	1	pullin-clearance
129		xxxxxxx -Fill
136		----- [end of Body]

8.6.3 CMV Screening Expanded Identification message

The CMV Screening Expanded Identification message provides information that may become necessary to conduct electronic screening of CVs at CV check stations in North America and is used in conjunction with the CMV Screening Identification message (see 8.6.1). It is generated by a portable transfer device (e.g., a notebook computer or PDA), stored in the transponder memory, and received by the beacon at a CV check station. It is transferred from the transponder to the beacon at mainline speeds. See Table 56.

Table 56—CMV Screening Expanded Identification message

Field	Data element name	Type	Constraint
1	Vehicle-ComponentIdentity	IA5string	Size (17); VIN
2	Driver-Identity	IA5string	Size (20)

8.6.3.1 ASN.1 specification

```
CMV-Screening-Expanded Identification-Message ::=     SEQUENCE
{
    header                  Dsrcmsg-Header,           -- Standard Header
    vehicle-component-ID    IA5String (SIZE(17)),     -- Vehicle-ComponentIdentity
    driver-ID               IA5String (SIZE(20))      -- Driver-Identity
}
```

8.6.3.2 ASN.1 sample values

```
CMV-Screening-Expanded Identification-Message    ::=      SEQUENCE
{                                                -- Begin Standard Header
      application-ID        2,                   -- CVO Application Identifier
      message-ID            8,                   -- Screening Event Message Identifier
      message-date          0,                   -- 1/1/1990
      message-length        37,                  -- 37 byte message body
      message-checksum      '00'H,               -- XOR checksum (not calculated)
                                                 -- End Header / Begin Body
      vehicle-component-ID                              --
      driver-ID                                  --
}
```

8.6.3.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000010	application-ID
6	00.1000	message-ID
12	0000.00000000.	message-date
24	00001100	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx. xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx. xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx. xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx. xxxxxxxx.	vehicle-component-ID
176	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx. xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx. xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx. xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx. xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.	driver-ID
336		----- [end of Body]

8.6.4 CMV Screening Expanded Event message

The CMV Screening Expanded Event message provides information documenting potentially critical parameters of the last clearance event and is used in conjunction with the CMV Screening Event message (see 8.6.2). It is generated by the CV check station computer via a DSRC controller, stored in the transponder memory, and received by the beacon at a CV check station. See Table 57.

Table 57—CMV Screening Expanded Event message

Field	Data element name	Type	Constraint
1	Vehicle-AxleNumber	Integer	(2 .. 17); measured number of vehicle axles
2	Vehicle-AxleWeight	Integer	(0 .. 4536); 10 kg steps; repeated for each axle
3	Vehicle-AxleSpacing	Integer	(0 .. 62); distance between axles in .5 m steps. Last value (for final axle) shall always be 0. Repeated for each axle.

8.6.4.1 ASN.1 specification

```
CMV-Screening-Expanded Event-Message      ::=          SEQUENCE
{
      header                  Dsrcmsg-Header,          -- Standard Header
      axle-number             INTEGER,                 -- Vehicle-AxleNumber
      axle-weight-1           INTEGER,                 -- Vehicle-AxleWeight
      axle-weight-2           INTEGER,                 -- Vehicle-AxleWeight
      axle-spacing-1          INTEGER,                 -- Vehicle-AxleSpacing
      axle-spacing-2          INTEGER,                 -- Vehicle-AxleSpacing
}
```

8.6.4.2 ASN.1 sample values

```
CMV-Screening-Expanded Event-Message      ::=          SEQUENCE
{                                                      -- Begin Standard Header
      application-ID          2,                       -- CVO Application Identifier
      message-ID              9,                       -- Screening Event Message Identifier
      message-date            0,                       -- 1/1/1990
      message-length          6,                       -- 6 byte message body
      message-checksum        '00'H,                   -- XOR checksum (not calculated)
                                                       -- End Header / Begin Body
      axle-number             2                        -- Vehicle-AxlesNumber
      axle-weight             100,                     -- 1000 kg
      axle-weight             100,                     -- 1000 kg
      axle-spacing            4                        -- 4 meters
      axle-spacing            0                        -- terminal axle
}
```

8.6.4.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000010	application-ID
6	00.1001	message-ID
12	0000.00000000.	message-date
24	00000110	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00010	axle-number
45	0011.11101000.0	axle-weight
58	0011111.010000	axle-weight
71	00.0100	axle-spacing
77	0000.00	axle-spacing
83	xxxxxx	octet alignment pad
88		----- [end of Body]

8.7 Common utility message set

Table 58 summarizes the messages that have been defined for common utility messages that may be used by any application. This subclause details the specific formats, conditions, and uses for each message.

Table 58—Common utility message summary

Message name	Description
Text String	Transmits a character string for storage or display
RSE to Other OBE	Data transfer from the RSE to equipment attached to the OBE network
Other OBE to RSE	Data transfer from network-attached OBE to the RSE
End Of Data	Marks end of messages within a memory page

8.7.1 Text String message

The Text String message contains a string of ASCII characters. It is generated by a BOA and stored in the transponder memory. See Table 59.

Table 59—Text String message

Field	Data element name	Type	Constraint
1	Utility-Text	IA5string	Size (1 .. 255) bytes

8.7.1.1 ASN.1 specification

```
Utility-Text-Message        ::=         SEQUENCE
{
    header                  Dsrcmsg-Header,             -- Standard Header
    utility-text            IA5String (SIZE(1..256))    -- Utility-Text
}
```

8.7.1.2 ASN.1 sample values

```
Utility-Text-Message        ::=         SEQUENCE
{                                                       -- Begin Standard Header
    application-ID          3,                          -- Common Utility Application Identifier
    message-ID              1,                          -- Text String Message Identifier
    message-date            0,                          -- 1/1/1990
    message-length          5,                          -- 5 byte message body
    message-checksum        '00'H,                      -- XOR checksum (not calculated)
                                                        -- End Header / Begin Body
    utility-text            "HELLO"                     -- Message Text
}
```

8.7.1.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000011	application-ID
6	00.0001	message-ID
12	0000.00000000.	message-date
24	00000101.	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	01001000.01001001.01001100.01001100.	utility-text
72	01001111.	
80		----- [end of Body]

8.7.2 RSE to Other OBE message

The RSE to Other OBE message contains an octet string of data from the RSE that is transferred to the OBE for delivery to the addressed attached equipment. See Table 60.

Table 60—RSE to Other OBE message

Field	Data element name	Type	Constraint
1	OBE-Address	BIT STRING	Size (32)
2	RSE-Data-Length	INTEGER	(0 .. 255)
3	RSE-Utility-Data	OCTET STRING	Size (1 .. 255)

8.7.2.1 ASN.1 specification

```
RSE-OBEBus-Message      ::=         SEQUENCE
{
    header              Dsrcmsg-Header,                 -- Standard Header
    OBE-address         BIT STRING (SIZE(32))           -- OBE network address
    RSE-data-length     INTEGER (1..255)                -- Length of the data in octets
    RSE-utility-data    OCTET STRING (SIZE(1..255)      -- Data string
}
```

8.7.2.2 ASN.1 sample values

```
RSE-OBEBus-Message      ::=         SEQUENCE
{                                                       -- Begin Standard Header
    application-ID      3,                              -- Common Utility Application Identifier
    message-ID          2,                              -- RSE to OBE Network Message Identifier
    message-date        0,                              -- 1/1/1990
    message-length      9,                              -- 9 byte message body
    message-checksum    '00'H,                          -- XOR checksum (not calculated)
                                                        -- End Header / Begin Body
    OBE-address         '00000010'H                     -- OBE Equipment Address = 16
    RSE-data-length     4                               -- Length of data component
    RSE-utility-data    '00FF00FF'                      -- Data component
}
```

8.7.2.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000011	application-ID
6	00.0010	message-ID
12	0000.00000000.	message-date
24	00001001.	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00000000.00000000.00000000.00010000.	OBE-address
72	00000100	RSE-data-length
80	00000000.11111111.00000000.11111111.	RSE-utility-data
112		----- [end of Body]

8.7.3 Other OBE to RSE message

The Other OBE to RSE message contains an octet string of data from the OBE-attached device identified by the OBE address that is transferred to the RSE via the transponder. See Table 61.

Table 61—Other OBE to RSE message

Field	Data element name	Type	Constraint
1	OBE-Address	Bit string	Size (32)
2	OBE-Data-Length	Integer	(0 .. 255)
3	OBE-Utility-Data	Octet string	Size (1 .. 255)

8.7.3.1 ASN.1 specification

```
OBEBus-RSE-Message      ::=         SEQUENCE
{
    header              Dsrcmsg-Header,                 -- Standard Header
    OBE-address         BIT STRING (SIZE(32))           -- OBE network address
    OBE-data-length     INTEGER (1..255)                -- Length of the data in octets
    OBE-utility-data    OCTET STRING (SIZE(1..255)      -- Data string
}
```

8.7.3.2 ASN.1 sample values

```
OBEBus-RSE-Message      ::=         SEQUENCE
{                                                       -- Begin Standard Header
    application-ID      3,                              -- Common Utility Application Identifier
    message-ID          3,                              -- OBE Network to RSE Message Identifier
    message-date        0,                              -- 1/1/1990
    message-length      12,                             -- 12 byte message body
    message-checksum    '00'H,                          -- XOR checksum (not calculated)
                                                        -- End Header / Begin Body
    OBE-address         '00000010'H                     -- OBE Equipment Address = 16
    OBE-data-length     4                               -- Length of data component
    OBE-utility-data    '00FF00FF'                      -- Data component
}
```

8.7.3.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000011	application-ID
6	00.0011	message-ID
12	0000.00000000.	message-date
24	00001001	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40	00000000.00000000.00000000.00010000.	OBE-address
72	00000100	OBE-data-length
80	00000000.11111111.00000000.11111111.	OBE-utility-data
112		----- [end of Body]

8.7.4 End Of Data message

The End Of Data message marks the end of messages within a memory page. It may be placed by either the RSE or by an intelligent transponder. This message has no fields.

8.7.4.1 ASN.1 specification

```
End-Data-Message ::=        SEQUENCE
{
        header              Dsrcmsg-Header,          -- Standard Header
}
```

8.7.4.2 ASN.1 sample values

```
End-Data-Message ::=        SEQUENCE
{                                                    -- Begin Standard Header
        application-ID      3,                       -- Common Utility Application Identifier
        message-ID          4,                       -- End of Data Message Identifier
        message-date        0,                       -- 1/1/1990
        message-length      0,                       -- no message body
        message-checksum    '00'H,                   -- XOR checksum (not calculated)
                                                     -- End Header / Begin Body
}
```

8.7.4.3 ASN.1 PER encoding

Bit	Bit value	Field definition
0	000011	application-ID
6	00.0100	message-ID
12	0000.00000000.	message-date
24	00000101.	message-length
32	xxxxxxxx.	message-checksum
		----- [end of Header]
40		----- [end of Body]

8.8 Private message set

Application identifiers have been set aside for private messages. The private uncontrolled application identifier is for unregistered private messages. It is anticipated that these messages will be used for testing. The private controlled application identifier is for messages that will be registered, having a unique message identifier for each registered message. The values for private uncontrolled and private controlled application identifiers are defined in Table 34. The values available for messages used for private controlled application identifiers are defined in Table 38. The values available for messages used for private uncontrolled application identifiers are by definition available for uncontrolled use.

8.9 Message error detection and processing

The following methods shall be applied on the OBE and RSE to detect and process errors detected in messages stored within transponder memory regions.

8.9.1 RSE error detection processing

The resource manager shall check received messages as follows prior to passing the messages to a BOA to verify that:

a) The application identifier is defined
b) The message identifier is defined for the application identifier
c) The message date is within the defined range of values
d) The message length matches the length of the received information
e) The error detect code is correct

If an error is detected, then the resource manager shall not pass the message to a BOA. The resource manager may execute site-specific error recovery processing, including deleting the defective message from the transponder memory region.

8.9.2 OBE error detection processing

The transponder shall not attempt to detect or otherwise process defective messages stored within transponder memory regions.

If the transponder is connected to other OBE, those devices may provide error detection and processing comparable to that defined above for the RSE.

9. Application layer

9.1 Scope

This standard applies to the application layer of a DSRC system. The purpose of the application layer is to provide communication services that allow the resource manager to communicate with the DSRC application on the OBE.

This standard provides a complete definition of the application layer architecture, services, and the interface to the service user that a DSRC vendor would require to implement a compliant application layer.

The services provided by the application layer are a subset of the application layer services defined by CEN. These services are interoperable in some cases with fully CEN-compliant application layers, when combined with a compliant lower layer service (described in 9.3.2).

Table 62 illustrates interoperability scenarios that incorporate the application layer defined in this standard (1455-compliant)[8]. These scenarios are

— *Scenario 1*. This scenario utilizes exclusively 1455-compliant elements within both the RSE and OBE. This scenario may be implemented in conjunction with any lower layer service that complies with specifications detailed in 9.3.2. In particular, when used in conjunction with an ASTM Data Link Layer and Physical Layer compliant lower layer service, this scenario is intended to provide the basis for North American DSRC interoperability.

— *Scenario 2*. This scenario is comparable to Scenario 1, but utilizes a fully CEN-compliant application layer in the RSE. This scenario illustrates that the 1455-compliant RSE application layer services represent an interoperable subset of those defined by CEN.

[8]Throughout this standard, the phrase “1455-compliant” shall signify compliance with IEEE Std 1455-1999.

— *Scenario 3*. This scenario is comparable to Scenario 2, but utilizes fully CEN-compliant application layers in both the RSE and the OBE. This scenario illustrates that the 1455-compliant RSE-and-OBE application layer services represent an interoperable subset of those defined by CEN. This scenario would allow the transponder to host CEN-compliant applications in addition to the 1455-compliant Mailbox application.

— *Scenario 4*. This hypothetical scenario, which could not actually be successfully implemented, illustrates that the 1455-compliant RSE application layer does not provide the services required by fully CEN-compliant RSE applications. A fully CEN-compliant RSE application layer must be used in this case.

— *Scenario 5*. This hypothetical scenario, which could not actually be successfully implemented, illustrates that the 1455-compliant OBE application layer does not provide the services required by fully CEN-compliant OBE applications. A fully CEN-compliant OBE application layer must be used in this case.

Table 62—Interoperability scenarios for the application layer

Scenario number	Inter-operable	RSE DSRC application	RSE application layer	RSE lower layer service	OBE lower layer service	OBE application layer	OBE DSRC application
1	Yes	1455* resource manager	1455 application layer	Any compliant service	Any compliant service	1455 application layer	1455 Mailbox application
2	Yes	1455 resource manager	CEN application layer	Any compliant service	Any compliant service	1455 application layer	1455 Mailbox application
3	Yes	1455 resource manager	CEN application layer	Any compliant service	Any compliant service	CEN application layer	1455 Mailbox application
4	No	Fully CEN compliant	1455 application layer	Don't care	Don't care	Don't care	Don't care
5	No	Don't care	Don't care	Don't care	Don't care	1455 application layer	Fully CEN compliant

*Refers to this standard, IEEE Std 1455-1999.

9.2 DSRC application domain assumptions

This standard of the application layer makes the following assumptions about the domain of DSRC applications for which the standard is intended:

— *Point-to-point communication*. Any session that includes the exchange of messages between a DSRC application on the RSE and the corresponding application on the OBE transponder is always through a single point-to-point communication between the two. The RSE may communicate simultaneously with multiple transponders by using broadcast services.

— *Master-slave*. In all RSE-to-OBE point-to-point connections, the resource manager acts as the master and the OBE transponder application is the slave.

The resource manager (and associated Mailbox application) is the only RSE application defined by this standard. However, nothing precludes a vendor from implementing a conformant application layer that includes additional functionality that might support additional applications.

9.3 Architecture

9.3.1 Scope

The application layer provides a subset of Layers 7 through 5 services as defined by the OSI model. In addition, the application layer requires an interface to the lower layer service that corresponds to OSI Layers 4 through 1 (see 9.3.2). Table 63 compares OSI model Layers 7 through 5 services to the services offered by the application layer. Some of the services are the responsibility of the BOA on the RSE.

Table 63—Comparison of OSI Layers 7 to 5 to the application layer

OSI service	Corresponding application layer service
Layer 7 (application)	
Provides access facilities for interconnecting with other applications	Service primitives provided by the initialization kernel element (I-KE) and transfer kernel element (T-KE)
Layer 6 (presentation)	
Negotiates a common syntax for transferring messages	The BOA encodes all application messages in ASN.1 PER, unaligned, format prior to invoking application layer services
Resolve differences in data representation	No differences allowed
Layer 5 (session)	
Organizes the dialog between two communicating programs and manages the data exchange; allowed modes are full duplex and half duplex	Initiated by BST/VST exchange
Establishes synchronization points in a dialog to allow dialogs to be interrupted then resumed. Session suspension and resumption are defined in 9.7.	No interruptions allowed

9.3.2 Lower layer service

The application layer assumes a generic lower layer service exists. This lower layer service provides the minimum subset of the functionality defined by Layers 4 through 1 of the OSI model. Table 64 summarizes the minimum subset of the services, defined by OSI Layers 4 through 1, that the application layer assumes are provided within the lower layer service.

Table 64—Required subset of OSI functionality for the lower layer service

OSI layer	Corresponding lower layer service
Layer 4 (transport)	— Fragmentation/Defragmentation — Message sequencing — Duplicate message handling
Layer 3 (network)	— Packet routing
Layer 2 (data link)	— Frame handling — Transmission error detection — Transmission error recovery
Layer 1 (physical)	— Physical information transmission

The application layer requires a service interface to the lower layer service. This service interface shall provide three basic classes of service for sending data from the RSE application layer and sending corresponding responses from the OBE application layer. Annex C illustrates the use of the lower layer service by the application layer.

The specific syntax and semantics of the generic lower layer service implementation may be vendor-specific. However, any conformant lower layer implementation must provide a lower layer service that corresponds to each of the required generic lower layer service classes defined in this subclause. The specific lower layer service implementation may also include additional services required for interoperability.

For the purposes of this standard, the lower layer service classes are defined using the following generic service identifiers:

a) DATASEND_RESPOND. This service class sends data from the RSE application layer and receives a confirmation that the data were received by the OBE and response data from the OBE. There shall always be an implementation of the DATASEND_RESPOND service class that shall provide the logical services defined in Table 65.

Table 65—DATASEND_RESPOND logical services

Logical service name	Functionality	Required logical parameters
request	Used by the RSE application layer to send data that will elicit the expected response	Link identifier (LID): identifies the OBE to which the request is directed; for a global LID, the RSE lower layer service shall broadcast the data to any OBE within range Data: along with data being sent, the byte count shall be specified
indication	Notification from the OBE lower layer service to the OBE application layer that data have been received from the RSE	Same as request except no count
response	Used by the OBE application layer to send the response back to the RSE	LID: identifies the OBE on which the data have been generated Status: the status of the request that is being confirmed Data: along with data being sent, the byte count shall be specified
confirm	Notification from the RSE lower layer service to the RSE application layer that response data from the OBE have been received	LID: identifies the OBE from which the data have been received Status: the status of the request that is being confirmed; if upper layers fail, it is provided by the RSE lower layer service to ensure that all requests result in a corresponding .confirm service Data: the data received from the OBE application if the transfer was successful

b) DATASEND_NORESPOND. This service class sends data from the RSE application layer with no subsequent OBE application confirmation that the data were received by the OBE. There shall always be an implementation of the DATASEND_NORESPOND service class that shall provide the logical services defined in Table 66.

Table 66—DATASEND_NORESPOND logical services

Logical service name	Functionality	Required logical parameters
request	Used by the RSE application layer to send the data that will elicit the expected response	LID: identifies the OBE to which the request is directed; for a global LID, the RSE lower layer service shall broadcast the data to any OBE within range Data: along with data being sent, the byte count shall be specified
indication	Notification from the OBE lower layer service to the OBE application layer that data have been received from the RSE	Same as request except no byte count
confirm	Allows RSE lower layers to confirm receipt of the data to the roadside application	LID: identifies the OBE to which the request was directed Status: the status of the request that is being confirmed; provided by the RSE lower layer service

c) DATASEND_RESPOND_REPEAT. This service class broadcasts the data on a periodic basis from the RSE to any OBE that is within range of the RSE. The RSE receives a confirmation each time the data are received by an OBE. DATASEND_RESPOND_REPEAT may be implemented to reduce communication overhead between the RSE application layer and the RSE lower layer service. Alternatively, the application layer may repetitively call DATASEND_RESPOND. If implemented, the DATASEND_RESPOND_REPEAT service class shall provide the logical services defined in Table 67.

Table 67—DATASEND_RESPOND_REPEAT logical services

Logical service name	Functionality	Required logical parameters
request	Used by the RSE application layer to initiate a process for periodically sending a single 1455-compliant command; the RSE lower layer service controls the rate of the send	Data: along with data being sent, the byte count shall be specified
indication	Notification from the OBE lower layer service to the OBE application layer that data have been received from the RSE	Same as request

Table 67—DATASEND_RESPOND_REPEAT logical services

Logical service name	Functionality	Required logical parameters
response	Used by the OBE application layer to send a response back to the RSE	LID: identifies the OBE that is generating the response Status: the status of the request that is being confirmed Data: the data making up the response
confirm	Notification from the RSE lower layer service to the RSE application layer that response data from the OBE have been received	LID: identifies the OBE from which the data have been received Status: the status of the request that is being confirmed; if upper layers fail, it is provided by the RSE lower layer service to ensure that all requests result in a corresponding .confirm service Data: the data received from the OBE application

d) DATASEND_NORESPOND_REPEAT. This service class broadcasts the data on a periodic basis from the RSE to any OBE that is within range of the RSE. There is no OBE application confirmation that an OBE has received the data. DATASEND_NORESPOND_REPEAT may be implemented to reduce communication overhead between the RSE application layer and the RSE lower layer service. Alternatively, the application layer may repetitively call DATASEND_NORESPOND. If implemented, the DATASEND_NORESPOND_REPEAT service class shall provide the logical services defined in Table 68.

Table 68—DATASEND_NORESPOND_REPEAT logical services

Logical service name	Functionality	Required logical parameters
request	Used by the RSE application layer to initiate a process for periodically sending a single source of data; the RSE lower layer service controls the rate of the send	Data: along with data being sent, the byte count shall be specified
indication	Notification from the OBE lower layer service to the OBE application layer that data have been received from the RSE	Same as request
confirm	Allows RSE lower layers to confirm receipt of the data to the roadside application	LID: identifies the OBE that received the data Status: the status of the request that is being confirmed; provided by the RSE lower layer service only upon receipt of a response from the OBE

e) SEND_BST_RESPOND. This service class sends a BST from the RSE application layer and receives a confirmation that the BST was received by the OBE. The confirmation shall include a returned VST. There shall always be an implementation of the SEND_BST_RESPOND service class that shall provide the logical services defined in Table 69.

Table 69—SEND_BST_RESPOND logical services

Logical service name	Functionality	Required logical parameters
request	Used by the RSE application to send a BST	Data: along with data being sent, the byte count shall be specified
indication	Notification from the OBE lower layer service to the OBE application layer that the RSE has sent a BST	Same as request
response	Used by the OBE application layer to send a VST back to the RSE	LID: identifies the OBE generating the VST Data: the response from the OBE
confirm	Notification from the RSE lower layer service to the RSE application layer that the VST has been received from the OBE	Same as response

f) SEND_BST_RESPOND_REPEAT. This service class broadcasts a BST on a periodic basis from the RSE to any OBE that is within range of the RSE. The RSE receives a confirmation each time the BST is received by an OBE. SEND_BST_RESPOND_REPEAT may be implemented to reduce communication overhead between the RSE application layer and the RSE lower layer service. If implemented, the SEND_BST_RESPOND_REPEAT service class shall provide the logical services defined in Table 70.

Table 70—SEND_BST_RESPOND_REPEAT logical services

Logical service name	Functionality	Required logical parameters
request	Used by the RSE application layer to initiate a process for periodically sending a BST; the RSE lower layer service controls the rate of the send	Data: along with data being sent, the byte count shall be specified
indication	Notification from the OBE lower layer service to the OBE application layer that the RSE has sent a BST	Same as request
response	Used by the OBE application layer to send a VST back to the RSE	LID: identifies the OBE generating the VST Data: the response from the OBE
confirm	Notification from the RSE lower layer service to the RSE application layer that the VST has been received from the OBE	Same as response

g) DATASEND*. This term is used to represent any of the service classes DATASEND_RESPOND, DATASEND_NORESPOND, DATASEND-RESPOND_REPEAT, and DATASEND_ NORESPOND_REPEAT.

h) SENDBST*. This term is used to represent either of the service classes SEND_BST_RESPOND and SEND_BST_RESPOND_REPEAT.

9.3.3 Application layer services

The application layer shall consist of an application layer kernel whose services are defined by a set of application kernel elements (KEs). An application KE represents a logical component of application layer functionality.

The application kernel shall consist of a T-KE, an I-KE, and a broadcast kernel element (B-KE).

Each available application layer kernel service shall be provided to service users by means of a service primitive defined in the form of an application service data unit (ASDU). Figure 6 shows the flow of an application layer service request, indication, response, and confirmation for a typical transaction between the RSE and OBE. Annex C illustrates in detail the service routines used to communicate between the resource manager and the application layer.

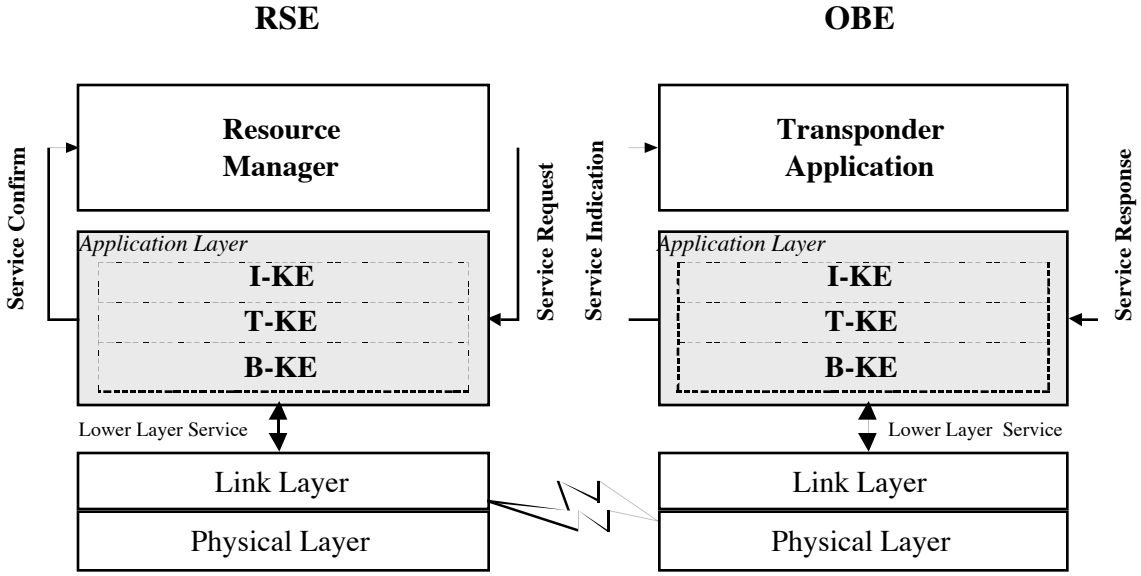

Figure 6—Application layer service flow

The application layer shall be implemented on both the RSE and the OBE. The application layer functionality may be combined with either the transponder application itself and/or lower layer service functions in a single physical program.

The application layer shall create LIDs in accordance with rules defined by the lower layers.

9.4 T-KE

The T-KE shall provide services to transfer information between the resource manager and the application running on the OBE transponder.

The T-KE shall offer its services by means of the service primitives defined in this subclause.

The T-KE shall provide its services by the processing protocol as described for each service primitive in this subclause.

9.4.1 Service primitive: ACTION

9.4.1.1 Definition

The T-KE shall provide the ACTION services defined by the following service primitives:

— The ACTION.request service shall be used by the resource manager to transmit any command to the OBE.

  For the Resource Manager commands that require a response from the OBE, the Mode field in the ACTION.request shall be set to 1. This setting shall be used for all the commands defined in Table 14, except for the broadcast mode versions of the Write Memory Page, Write Circular Queue, Append Message, and Set User Interface commands.

For the Resource Manager commands that do not require a response from the OBE, the Mode field in the ACTION.request shall be set to 0. This setting shall be used for the broadcast mode versions of the Write Memory Page, Write Circular Queue, Append Message, and Set User Interface commands defined in Table 14.

— The ACTION.indication service shall be used by the OBE application layer to notify the OBE application that an ACTION.request has been received from the resource manager.

— The ACTION.response service shall be used by the OBE application to send a command response back to the resource manager only when the Mode field in the corresponding ACTION.indication is set to 1.

— The ACTION.confirm service shall be used by the RSE application layer to notify the resource manager that an ACTION.response (containing a command response) has been received from an OBE application on the OBE.

T-KE service users shall communicate with ACTION services and event notifications by using the ASDU defined in Table 71. Figures C.2 and C.3 illustrate the use of ACTION services by the resource manager and by the OBE application.

Table 71—ACTION ASDU field definitions

Parameter name	Request value	Indication value	Response value	Confirm value	ASN.1 type
AID	1455-compliant Mailbox AID, hex(D)	Same as request			DSRCApplication EntityID
LID	LID returned by the OBE in the VST when the session was initialized or a global LID to make a broadcast request	Same as request	Same as request except for a global LID, when the LID shall be provided by OBE application layer service	Same as response	BIT STRING
EID	EID returned by OBE in the VST when the session was initialized; not used (set to 0) if the LID field contains a global LID		Same as request		Dsrc-EID
ActionType	hex(1); reserved value that defines the action of processing a Resource Manager command	Same as request			INTEGER
ActionParameter	Resource Manager command: ASN.1-BASIC-PER, UNALIGNED	Same as request			Container
Mode	hex(0) for a request where no response from the OBE is required hex(1) for a request where a response from the OBE is required	Same as request			Boolean

Table 71—ACTION ASDU field definitions (continued)

Parameter name	Request value	Indication value	Response value	Confirm value	ASN.1 type
FlowControl	See 9.4.3	Same as request	Same as request		INTEGER
ResponseParameter			OBE reply to Resource Manager command: ASN.1-BASIC-PER, UNALIGNED	Same as response	Container
ReturnCode			See 9.4.3	Same as response	ReturnStatus

9.4.1.2 Processing protocol

The T-KE shall process an ACTION service request as follows:

a) Encoding. The T-KE shall encode all ACTION.request/ACTION.response ASDUs (in ASN.1-BASIC-PER, UNALIGNED, format) prior to sending the ASDU to the lower layer service. The T-KE shall assume all data in either the ActionParameter or ResponseParameter field are a binary large object (BLOB) that represents either a Resource Manager command or command response that has already been encoded by the application.

b) Decoding. The T-KE shall decode all ACTION.indication/ACTION.confirm ASDUs (in ASN.1-BASIC-PER, UNALIGNED, format) that are contained within corresponding DATASEND*.indication/ DATASEND*.confirm messages received from the lower layer service. The T-KE shall assume all data in either the ActionParameter or ResponseParameter field are a BLOB that represents either an encoded Resource Manager command or encoded command response that the application itself will be responsible for decoding.

c) Fragmentation/Defragmentation. The T-KE shall not support fragmentation. For CEN compliance, a 1-octet fragmentation header with a value of hex(1) shall be prefixed to all ACTION.request/ ACTION.response service requests.

   All ACTION.indication/ACTION.confirm service requests that are contained within corresponding .indication/.confirm messages received from the lower layer service are assumed to contain a 1-octet fragmentation header with a value of hex(1). This fragmentation header shall be removed prior to passing that service request from the application layer up to the application itself. Any ACTION.indication/ACTION.confirm containing a fragmentation header with a value other than hex(1) shall represent an error condition, and the application layer shall not process the request any further.

d) Transmit using a lower layer confirmed service. The T-KE shall transmit all ACTION.request service requests, where the Mode field is set to 1, to the OBE by constructing and then initiating a DATASEND_RESPOND.request. The corresponding ACTION.response shall be transmitted by the T-KE on the OBE by constructing and then initiating a DATASEND_RESPOND.response.

e) Transmit using a lower layer unconfirmed service. The T-KE shall transmit all ACTION.request service requests, where the Mode field is set to 0, to the OBE by constructing and then initiating a DATASEND_NORESPOND.request.

f) Transmit using a lower layer service if the LID is global. The T-KE shall transmit all ACTION.request service requests with a global LID by constructing and then initiating either a DATASEND_RESPOND_REPEAT.request if the Mode field is set to 1, or a DATASEND_NORESPOND_REPEAT.request if the Mode field is set to 0.

g) Notify service user. The T-KE shall communicate all ACTION.indication and ACTION.confirm service requests back to the service user (either the resource manager or transponder application) for which the request is intended.

9.4.2 Service primitive: INITIALISATION

9.4.2.1 Definition

T-KE INITIALISATION services are invoked internally within the application layer by the I-KE. They are never directly accessed by either the resource manager or by the OBE application.

INITIALISATION services are used to send a BST from the RSE to the OBE and to return a corresponding VST from the OBE back to the RSE. The BST and VST are discussed in detail in 9.5.

The T-KE shall provide the INITIALISATION services defined by the following service primitives:

— The INITIALISATION.request service shall be used by the I-KE on the RSE to convert an I-KE RegisterApplicationBeacon service request into a BST that is transmitted to the OBE.

— The INITIALISATION.indication service shall be used internally by the OBE application layer T-KE to notify the application layer I-KE that a BST has been received from an RSE.

— The INITIALISATION.response service shall be used by the I-KE on the OBE to convert an I-KE RegisterApplicationVehicle service request into a VST that is transmitted back to the RSE.

— The INITIALISATION.confirm service shall be used internally by the RSE application layer T-KE to notify the application layer I-KE that a VST has been received from an OBE.

The I-KE shall communicate with INITIALISATION services and event notifications by using the ASDU defined in Table 72. Figure C.1 illustrates how the application layer on both the RSE and OBE use INITIALISATION services during the process of resource manager initialization.

Table 72—INITIALISATION ASDU field definitions

Parameter name	Request value	Indication value	Response value	Confirm value	ASN.1 type
LID			LID defined by the OBE lower layer service	Same as response	BIT STRING
InitialisationParameter	BST	BST	VST	VST	BST or VST

9.4.2.2 Processing protocol

The T-KE shall process an INITIALISATION request as follows:

a) Encoding. The T-KE shall encode all INITIALISATION.request/INITIALISATION.response ASDUs in ASN.1-BASIC-PER, UNALIGNED prior to sending the ASDU to the lower layer service. This shall include encoding the BST contained in the .request and the VST contained in the .response.

b) Decoding. The T-KE shall decode all INITIALISATION.indication/INITIALISATION.confirm ASDUs (in ASN.1-BASIC-PER, UNALIGNED, format) that are contained within corresponding SEND_BST*.indication/SEND_BST*.confirm messages received from the lower layer service. This shall include decoding of the BST contained in the .indication and the VST contained in the .confirm

c) Fragmentation/Defragmentation. The T-KE assumes fragmentation/defragmentation is handled by the lower layer service. For CEN compliance, a 1-octet fragmentation header with a value of hex(1) shall be prefixed to all INITIALISATION.request/INITIALISATION.response ASDUs prior to sending the ASDU via the lower layer service.

All INITIALISATION.indication/INITIALISATION.confirm ASDUs received from the lower layer service are assumed to contain a 1-octet fragmentation header with a value of zero. This fragmentation header shall be removed prior to passing that service request from the application layer up to the application itself. Any INITIALISATION.indication/INITIALISATION.confirm containing a fragmentation header with a value other than hex(1) shall represent an error condition, and the application layer shall not process the request any further.

d) Transmit using SEND_BST_RESPOND_REPEAT. If the lower layer service provides the SEND_BST_RESPOND_REPEAT service, the T-KE shall transmit all INITIALISATION.request service requests by constructing and then initiating a SEND_BST_RESPOND_REPEAT.request. The corresponding INITIALISATION.response shall be transmitted by the T-KE on the OBE by constructing and then initiating a SEND_BST_RESPOND_REPEAT.response.

e) Transmit using SEND_BST_RESPOND. If the lower layer service does not provide the optional SEND_BST_RESPOND_REPEAT service, the T-KE shall transmit all INITIALISATION.request service requests by constructing and then initiating a SEND_BST_RESPOND.request.

The T-KE shall also be responsible for continuing to send the SEND_BST_RESPOND.request on a periodic basis until another INITIALISATION.request is received. The rate of the send is implementation-specific.

f) Notify service user. The T-KE shall communicate all INITIALISATION.indication/ INITIALISATION.confirm service requests back to application layer I-KE for which the request is intended.

9.4.3 T-KE service request: parameter usage rules

a) The AID for the Mailbox application is the only AID defined by this standard. The value is hex(D).

b) The LID shall be provided by the OBE application layer services in accordance with rules provided by the lower layers, except for an RSE broadcast request that shall utilize a global LID.

c) EID is the EID of the OBE transponder to which the RSE currently has a session.

d) ActionType defines the requested action. The only value for ActionType defined by this standard is hex(1), which means process a Resource Manager command.

e) ActionParameter contains the definition of an action. This shall be one or more commands generated by the resource manager and encoded as an ASN.1 PER-encoded container.

f) FlowControl is a parameter that represents the requested behavior of the underlying data link communication service. FlowControl currently has no meaning and is reserved for future use. It shall always be set to a value of hex(0).

g) InitializationParameter is the information needed for the initialization of the communication sent via the I-KE. This information shall be the BST in the request from the RSE and the VST in the response from the OBE.

h) Mode defines whether a service confirm is required. A value of hex(1) shall mean a confirm is required. (i.e., connected). A value of hex(0) shall mean that a confirm is not provided (i.e., broadcast).

i) ResponseParameter contains the response to an action. This shall be one or more responses generated by the OBE, in response to Resource Manager commands, encoded as an ASN.1 PER-encoded container.

j) ReturnStatus is a return code issued in an application's service response. The code values are specified in the ReturnStatus data element defined in Annex A. The values shall correspond to the following meanings:

1) NoError. The requested operation was performed successfully.

2) AccessDenied. The requested operation was not performed for reasons pertinent to the security of the system.

3) ArgumentError. One or more attributes were not accessed because the identifier for the specified attribute was not recognized or was inappropriate for any reason or because the action invoked was not supported by the receiving entity.

4) ComplexityLimitation. The requested operation was not performed because a parameter was too complex.

5) ProcessingFailure. A general failure in processing the operation was encountered.

6) Processing. The requested operation is being processed, and the result is not yet available.

7) CommunicationsFailure. The requested operation was not performed, and a communications failure has been reported by the lower protocol layers.

9.4.4 Comparison to CEN T-KE processing

The internal T-KE processing protocol is streamlined to meet the needs of the application domain (see 9.2) and as such differs from the processing protocols as defined by CEN. Table 73 summarizes these differences.

Table 73—CEN T-KE processing protocol compared to IEEE Std 1455-1999

Functionality	CEN T-KE	IEEE Std 1455-1999 T-KE
Encoding	ASN.1-BASIC-PER, UNALIGNED	Does not encode application parameters received from the resource manager or the OBE application
Fragmentation	1-octet to 3-octet fragmentation header formats	Always inserts a 1-octet fragmentation header set to hex(80)
Octet alignment	Aligns with zero padding until the number of bits is a multiple of 8	Assumes all data are aligned on 8-bit boundaries
Multiplexing	Multiplexes service requests from multiple concurrent applications	Unsupported
Access to lower layer services	Uses the lower layer service assigned in the ASDU FlowControl parameter	Predefined by processing protocols defined for each service
Concatenation	Multiple fragments may be mapped onto a single logic link control (LLC) request	Unsupported
Demultiplexing	Will demultiplex multiple responses received in a single lower layer service protocol data unit (PDU)	Unsupported
Defragmentation	Will defragment a single logical response received across multiple lower layer service PDUs	Assumes a 1-octet fragmentation header set to hex(1)

Table 73—CEN T-KE processing protocol compared to IEEE Std 1455-1999 (continued)

Functionality	CEN T-KE	IEEE Std 1455-1999 T-KE
Decoding	ASN.1-BASIC-PER, UNALIGNED	Does not decode application parameters received from the resource manager or the OBE application
Construction of service data unit (SDU) for addressee	Constructed from intermediate PDU	Constructed directly from the ASDU of requesting resource manager

9.4.5 Optional T-KE asynchronous services

This standard does not include the definition of T-KE services to support the ability of the OBE to asynchronously notify the RSE of some event or status. However, vendors may implement these services; and it is recommended that any implementation follow the general model for application services that include the following four command types:

— SERVICE.request
— SERVICE.indication
— SERVICE.response
— SERVICE.confirm

Optional OBE T-KE services shall not impact communications with a compliant RSE application layer that did not support those optional services.

9.5 I-KE

The I-KE shall provide services to initialize a session between the resource manager and an application running on the OBE transponder.

The I-KE shall initialize the session by means of the BST. The size of one BST shall enable the transfer of the BST in one lower layer service primitive.

The I-KE shall offer its services by means of the service primitives defined in this subclause.

The I-KE shall use the INITIALISATION services of the application layer T-KE as defined in 9.4.2.

Figure C.1 illustrates the use of I-KE services by the resource manager and the OBE application, including the exchange of a BST and VST, during the process of resource manager initialization.

9.5.1 BST

As part of the initialization of a point-to-point connection between the RSE and the OBE and in response to a RegisterApplicationBeacon service call, the I-KE collects the resource manager AID, initial data, and protocol layer parameters relevant for the communication and assembles a BST. The BST is transmitted by the beacon to start the initialization process. Either the application layer or the lower layer service may control this cyclic transmission depending on the capability of the lower layer service (see 9.4.2.2).

The reception of the BST by an OBE transponder is the initiator of the point-to-point data transfer. The OBE transponder evaluates a received BST to determine whether a connection should be made, and if so, sends back a corresponding VST. Table 74 describes the individual fields and their values, which shall constitute the BST. Annex D contains a sample ASN.1 PER encoding of a BST.

An alternate form of the BST may be employed for high-performance situations. This compact form has a shorter beacon identifier (8 bits), no Time field, and a different T-APDU identifier. Table 75 describes the individual fields and their values contained in the compact BST.

The page identifiers contained within the BST shall control transponder activation and specify the memory pages that shall be returned. A transponder shall activate only if all nonzero page identifiers are available within the transponder. When a transponder activates, the memory images corresponding to the nonzero page identifiers shall be returned. Pages 1, 2, and 3 are formatted in a manner consistent with a VST, and the VST data structure will not be returned to the RSE if those pages are not specified. Compliant systems need not request those pages, i.e., both page identifiers may specify pages other than those corresponding to the VST region. If both page identifiers are set to 0, then no transponder will activate.

Table 74—BST field descriptions and values

Field name	ASN.1 type	Description	Size (bits)	Bit sequence = value
T-APDU	T-APDUs	A BST identifier required for compliance with the CEN ASN.1 definition of a BST	4	0-3 = ASN.1 T-APDUs value for a choice of INITIALISATION.request = hex(8)
Options Flag	BIT STRING (SIZE (1))	A bit indicating that the optional nonmandatory applications list field is missing; required for compliance with the CEN ASN.1 definition of a BST	1	4 = 0
beacon	BeaconID	An identifier composed of a manufacturer identifier (a unique identifier assigned by IEEE) and an individual identifier whose use is vendor-specific	43	5-20 = manufacturer identifier 21-47 = individual identifier
time	Time	The number of seconds from 01/01/1970 GMT	32	48-79 = time
profile	Profile	The profile that will be used to transmit the BST; profile definitions are specific to the lower layer service	8	80-87 = profile
mandApplications	ApplicationList	Formally defines the RSE applications within the CEN structure. In this sense, this standard supports only the Mailbox application, though additional ITS applications may be serviced by the resource manager. The ASN.1 encoding of the application list requires that the first octet define the number of elements in the list, which shall always be hex(1) The application list consists of the Mailbox AID, an EID, and a Parameter field; the Parameter field consists of a data type tag, a data length octet, and the data itself The Parameter field data define two page identifiers for which OBE memory images will be returned by the OBE A Page Identifier field shall be set to 0 if it is unused.	72	88-95 = number in list = hex(1) 96-103 = hex(D) (Mailbox AID) 104-111 = EID 112-119 = hex(4) = tag for octet string 120-127 = length of data = hex(C) 128-143 = first page identifier 144-159 = second page identifier
profileList	SEQUENCE OF Profile; only one profile in the sequence	Profile values, in addition to the profile value specified in the Profile field, that are supported by the RSE lower layer service; the ASN.1 encoding of profileList requires two octets	16	160-167 = number of profiles in list = hex(1) 168-175 = value of profile

Table 75—Compact BST field descriptions and values

Field name	ASN.1 type	Description	Size (bits)	Bit sequence = value
T-APDU	T-APDUs	A BST identifier required for compliance with the CEN ASN.1 definition of a BST	4	0-3 = ASN.1 T-APDUs value for a choice of compact INITIALISATION.request = hex(F)
Options Flag	BIT STRING (SIZE (1))	A bit indicating that the optional non-mandatory applications list field is missing; required for compliance with the CEN ASN.1 definition of a BST	1	4 = 0
beacon	BeaconID	A beacon identifier	8	5-12 = beacon identifier
profile	Profile	The profile that will be used to transmit the BST; profile definitions are specific to the lower layer service	8	13-20 = profile
mandApplications	ApplicationList	Formally defines the RSE applications within the CEN structure. In this sense, this standard supports only the Mailbox application, though additional ITS applications may be serviced by the resource manager. The ASN.1 encoding of the application list requires that the first octet define the number of elements in the list, which shall always be hex(1) The application list consists of the Mailbox AID, an EID, and a Parameter field; the Parameter field consists of a data type tag, a data length octet, and the data itself The Parameter field data define two page identifiers for which OBE memory images will be returned by the OBE in the VST A Page Identifier field shall be set to 0 if it is unused	72	21-28 = number in list = hex(1) 29-36 = hex(D) (Mailbox AID) 37-44 = EID 45-52 = hex(4) = tag for octet string 53-60 = length of data = hex(C) 61-68 = first return identifier 69-76 = second return identifier
profileList	SEQUENCE OF Profile; only one profile in the sequence	A profile, in addition to the profile specified in the Profile field, that is supported by the RSE lower layer service; the ASN.1 encoding of profileList requires two octets	16	77-84 = number of profiles in list = hex(1) 85-92 = value of profile

9.5.2 VST

The VST is constructed by the I-KE on the OBE transponder in response to a BST received from the RSE. The VST contains state information about the OBE that is required to create a session between the RSE and the OBE. Table 76 describes the individual fields and their values that shall constitute the VST. Annex D contains a sample ASN.1 PER encoding of a VST.

Table 76—VST field descriptions and values

Field name	Asn.1 type	Description	Size (bits)	Bit sequence = value
OBE Read Only Memory	See Table 3	The first portion of the VST corresponds exactly to the contents of OBE read-only memory, as defined in 5.2	128	0-127 = contents of OBE read-only memory
Returned Memory Images		The memory images, requested in the BST, shall be returned if present on the transponder. The length of this field is determined by the size of the returned memory image (either the entire page or that portion of the page up to the End Of Data marker).	128 to 128+ size of memory images	128-n = contiguous image of returned memory pages
ObeConfiguration	ObeConfiguration	Consists of Equipment Class, Manufacturer Identifier, and OBE Status fields The use of the Equipment Class field is specified by the manufacturer of the equipment The Manufacturer Identifier field is the identifier that the manufacturer has registered with IEEE, as identified in Table 12. The OBE Status field will use the first octet for transmitting OBE system status; the second octet is reserved for private use; the first octet is set by a logical OR of some combination of the following flags: hex(8) Absence of check data hex(10) Unrecognized check data hex(20) Battery failure hex(40) Peripheral interface error hex(80) Illegal tampering	end of memory images +1 to end of memory images + 49	Next 16 bits = equipment class Next 16 bits = manufacturer identifier Next 16 bits = OBE status

9.5.3 RSE service primitives: RegisterApplicationBeacon

9.5.3.1 Definition

The invocation of the RegisterApplicationBeacon service by the resource manager shall result in the notification to potential OBE transponders, with which a single point-to-point connection could be made, about the presence of the resource manager.

I-KE service users shall make a RegisterApplicationBeacon request in the form of an ASDU as defined in Table 77.

9.5.3.2 Processing protocol

The I-KE on the RSE shall process a RegisterApplicationBeacon service request from the resource manager as follows:

a) Construct a BST using the parameters contained in the RegisterApplicationBeacon service request.

Table 77—RegisterApplicationBeacon ASDU field definitions

Parameter name	Value	ASN.1 type
AID	1455-compliant Mailbox AID hex(D)	DSRCApplicationEntityID
Mandatory	Vendor-specific or hex(0) if not used	BOOLEAN
Priority	Vendor-specific or hex(0) if not used	INTEGER
EID	hex(0)	DSRC-EID
Profiles	The list of profiles supported by the RSE lower layer service	SEQUENCE OF Profile
Parameter	Vendor-specific or hex(0) if not used	Container whose contents specify memory pages that will be included in the BST to be returned in the VST

b) Construct and initiate a T-KE INITIALISATION.request service request.

c) Enable the ability for the I-KE to notify the resource manager if a VST is received back from the OBE (in the form of an INITIALISATION.confirm).

d) The RegisterApplicationBeacon service in turn invokes another application layer service (INITIAL-ISATION) that handles the direct communication with the lower layer service and all the associated responsibility for encoding/decoding and fragmentation/defragmentation. This means no encoding/decoding or fragmentation/defragmentation is necessary when processing a RegisterApplication-Beacon.

9.5.4 RSE service primitive: DeregisterApplication

9.5.4.1 Definition

The invocation of the DeregisterApplication service by the resource manager shall result in the fact that potential transponders on the OBE, with which a single point-to-point connection could be made, will no longer be notified about the presence of the resource manager.

I-KE service users shall make a DeregisterApplication request in the form of an ASDU as defined in Table 78.

Table 78—DeregisterApplication ASDU field definitions

Parameter name	Value	ASN.1 type
AID	1455-compliant Mailbox AID hex(D)	DSRCApplicationEntityID

9.5.4.2 Processing protocol

The I-KE shall process a DeregisterApplication service request from the resource manager as follows:

a) The I-KE ceases to notify the resource manager concerning the reception of a VST from an OBE transponder.

b) Since this service terminates in the RSE application layer, no encoding/decoding or fragmentation/defragmentation is required.

9.5.5 RSE service primitive: NotifyApplicationBeacon

9.5.5.1 Definition

The invocation of the NotifyApplicationBeacon service by the application layer I-KE on the RSE shall result in the notification of the resource manager about the presence of a potential OBE transponder with which a single point-to-point connection could be made. This notification shall include the EID of the associated transponder.

I-KE service users shall make a NotifyApplicationBeacon request in the form of an ASDU as defined in Table 79.

Table 79—NotifyApplicationBeacon ASDU field definitions

Parameter name	Value	ASN.1 type
Priority	Vendor-specific or hex(00) if not used	INTEGER
EID	EID returned by OBE in the VST when the session was initialized	DSRC-EID
LID	Vendor-specific or hex(00) if not used	BIT STRING
Parameter	Vendor-specific or hex(0) if not used	Container
ObeConfiguration	See 9.5.6	ObeConfiguration

9.5.5.2 Processing protocol

The I-KE shall generate a NotifyApplicationBeacon service request directed to the resource manager as follows:

a) A NotifyApplicationBeacon service request is generated when a VST is received from an OBE transponder in the form of an INITIALISATION.confirm request from the I-KE on the RSE.

b) The EID, LID, Parameter, and ObeConfiguration fields are copied from the received VST into the NotifyApplicationBeacon service request.

c) The NotifyApplicationBeacon request is sent to the resource manager. The interface used to send the request is implementation-specific.

d) The NotifyApplicationBeacon service is actually preceded by another application layer service (INITIALISATION) that handles the direct communication with the lower layer service and all the associated responsibility for encoding/decoding and fragmentation/defragmentation. This means no encoding/decoding or fragmentation/defragmentation is necessary when processing NotifyApplicationBeacon.

9.5.6 OBE service primitive: RegisterApplicationVehicle

9.5.6.1 Definition

The invocation of the RegisterApplicationVehicle service by the OBE application shall result in activating the I-KE to inspect each BST received from the RSE for a matching AID.

I-KE service users shall make a RegisterApplicationVehicle request in the form of an ASDU as defined in Table 80.

Table 80—RegisterApplicationVehicle ASDU field definitions

Parameter name	Value	ASN.1 type
AID	1455-compliant Mailbox AID hex(D)	DSRCApplicationEntityID
Priority	Vendor-specific or hex(0) if not used	INTEGER
EID	hex(0)	DSRC-EID
Profiles	The list of profiles supported by the OBE	SEQUENCE OF Profile
Parameter	hex(0)	Unused

9.5.6.2 Processing protocol

The I-KE on the OBE shall process a RegisterApplicationVehicle service request from the OBE application as follows:

a) Place the AID in a list of registered applications, indicating that the application is ready to be notified regarding potential RSE communications partners.

b) Initiate the inspection of each incoming BST for the registered AID.

c) If an AID match is made, send a NotifyApplicationVehicle after saving the BeaconID from the BST.

9.5.7 OBE service primitive: NotifyApplicationVehicle

9.5.7.1 Definition

The invocation of the NotifyApplicationVehicle service by the application layer I-KE on the OBE shall result in the notification of the OBE application about the presence of a potential communication partner and the LID generated by the OBE.

I-KE service shall make a NotifyApplicationVehicle request in the form of an ASDU as defined in Table 81.

Table 81—NotifyApplicationVehicle ASDU field definitions

Parameter name	Value	ASN.1 type
Beacon	Beacon identifier from BST	BeaconID
Priority	Vendor-specific or hex(00) if not used	INTEGER
EID	EID from the BST	DSRC-EID
LID	Generated in the OBE	BIT STRING
Parameter	Vendor-specific or hex(0) if not used	Container

9.5.7.2 Processing protocol

The I-KE shall generate a NotifyApplicationBeacon service request directed to the OBE application as follows:

a) A NotifyApplicationVehicle service request is generated when a BST is received in the form of an INITIALISATION.indication request from the I-KE on the RSE that contains an AID matching an AID provided in a prior RegisterApplicationVehicle.

b) The BeaconID, Priority, and EID fields are copied from the received BST into the NotifyApplicationVehicle service request along with the generated LID.

c) The NotifyApplicationVehicle request is sent to the OBE application. The interface used to send the request is implementation-specific.

9.5.8 RSE service primitive: ReadyApplication

9.5.8.1 Definition

The invocation of the ReadyApplication service by the resource manager on the RSE shall notify the RSE application layer that the LID is no longer needed by the resource manager, i.e., that the roadside will not communicate further with the specified transponder.

I-KE service shall make a ReadyApplication request in the form of an ASDU as defined in Table 82.

Table 82—ReadyApplication ASDU field definitions

Parameter name	Value	ASN.1 type
EID	EID from the BST	DSRC-EID
LID	Generated in the OBE	BIT STRING

9.5.8.2 Processing protocol

The resource manager shall generate a ReadyApplication service request directed to the RSE application layer when the resource manager does not require further communications with the specified transponder. The RSE application layer shall process the request as follows:

a) The application layer may notify the lower layer services that media are no longer required for communications with the specified transponder.

b) Since this service terminates in the RSE, no encoding/decoding or fragmentation/defragmentation is required.

9.5.9 OBE service primitive: DeregisterApplicationVehicle

9.5.9.1 Definition

The invocation of the DeregisterApplicationVehicle service by the OBE application shall result in the fact that potential beacons on the roadside will no longer be notified of the presence of the Mailbox application within the transponder. In transponders that do not provide CEN-compliant applications other than the 1455-compliant Mailbox application, this request shall disable communications with the roadside.

I-KE service shall make a DeregisterApplicationVehicle request in the form of an ASDU as defined in Table 83.

Table 83— DeregisterApplicationVehicle ASDU field definitions

Parameter name	Value	ASN.1 type
AID	1455-compliant Mailbox AID hex(D)	DSRCApplicationEntityID

9.5.9.2 Processing protocol

The I-KE shall process a DeregisteApplicationVehicle service request from the OBE application as follows:

a) The specified AID value shall be removed from the list of registered applications in the VST.

b) Since this service terminates in the OBE, no encoding/decoding or fragmentation/defragmentation is required.

9.5.10 RSE service request parameter usage rules

The I-KE service primitive parameters shall be used as follows:

— The AID for the Mailbox application is the only AID defined by this standard. The value is hex(D).

— EID shall be the EID of the OBE transponder to which the RSE currently has a session.

— The LID is generated by the OBE that is either the originator or recipient of the current service request.

— ObeConfiguration is the configuration of the OBE transponder. See 9.5.2, which describes the ObeConfiguration field (within the VST) in more detail.

— Parameter is currently vendor-specific, if used.

— Priority is the priority of the service request relative to other service requests and is currently vendor-specific, if used.

— Profiles is the list of profiles (inherent capabilities for sending and receiving data) supported by the lower layer service on the RSE.

9.6 B-KE

The B-KE shall provide services to broadcast unacknowledged information from the resource manager to a broadcast pool maintained by the OBE application layer as well as services for the OBE transponder application to access the broadcast pool.

The B-KE shall offer its services by means of the service primitives defined in this subclause.

The B-KE shall provide its services by the processing protocol as described for each service primitive in this subclause.

9.6.1 Service primitive: BroadcastData

9.6.1.1 Definition

The B-KE shall provide the BroadcastData services defined by the following service primitive: The Broadcast.request service shall be used by the resource manager to broadcast any command to any potential OBE within range of communication. Broadcast messages received by the OBE are stored in a broadcast pool within the OBE application layer.

The request requires that the Resource Manager command be encapsulated within a NamedFile. A NamedFile requires a file name to be assigned to it. The OBE application requires knowledge of file name in order to get the command from the broadcast pool on the OBE. The conventions for file name use and meaning are implementation-specific.

The resource manager shall invoke a BroadcastData service by using the ASDU defined in Table 84.

Table 84—BroadcastData ASDU field definitions

Parameter name	Request value	ASN.1 type
File	Resource Manager command	NamedFile

9.6.1.2 Processing protocol

The B-KE shall process a BroadcastData service request as follows:

a) Encoding. The B-KE shall assume all data in File field of a BroadcastData.request are a BLOB that represents the Resource Manager command that has already been encoded by the application.

b) Fragmentation. The B-KE does not support fragmentation. For CEN compliance, a 1-octet fragmentation header with a value of hex(1) shall be prefixed to all Broadcast.request service requests.

c) Transmit using DATASEND_NORESPOND. The B-KE shall transmit all BroadcastData.request service requests by constructing and then initiating a DATASEND_NORESPOND.request using a global LID. Alternatively DATASEND_NORESPOND_REPEAT may be used to initiate periodic data transmission.

   The B-KE shall also be responsible for continuing to send the DATASEND_NORESPOND.request on a periodic basis until another BroadcastData.request is received. The rate of the send is implementation-specific.

d) Add to broadcast pool on OBE. The B-KE on the OBE shall store a File (Resource Manager command), received from a BroadcastData.request sent from the RSE, in a broadcast pool. The rules for managing the broadcast pool are implementation-specific.

9.6.2 Service primitive: GetBroadcastData

9.6.2.1 Definition

The B-KE shall provide the GetBroadcastData services defined by the following service primitive: The GetBroadcastData service shall be used by the OBE transponder application to get any broadcast messages that are currently residing in the broadcast pool maintained by the OBE application layer.

The OBE application shall invoke a GetBroadcastData service by using the ASDU defined in Table 85.

9.6.2.2 Processing protocol

The B-KE shall process a GetBroadcastData service request as follows:

a) For a GetBroadcastData.request, the B-KE shall determine whether a File field exists in the OBE application layer broadcast pool whose file name matches the Name field in the GetBroadcastData.request.

Table 85—GetBroadcastData ASDU field definitions

Parameter name	Request value	Confirm value	ASN.1 type
Name	Name of file to retrieve from broadcast pool		FileName
EID	Currently has no meaning.; always set to hex(0)		Dsrc-EID
File		File (contains a Resource Manager command)	NamedFile

b) If a matching File field is found in the broadcast pool, the B-KE shall construct a GetBroadcast-Data.confirm, which contains an image of the matching file, and then send the GetBroadcast-Data.confirm back to the OBE application that generated the GetBroadcastData.request. The interface used to send the .confirm is implementation-specific.

c) The rules for managing the broadcast pool, such as whether to delete a retrieved File from the pool, are implementation-specific.

9.6.3 Service request parameter usage rules

— EID currently has no meaning.

— File contains the Resource Manager command being broadcast.

— Name is the name of the file to get from the broadcast pool.

9.7 Session suspend and resume

In general, once a session has been initiated through a BST/VST sequence, it will continue without interruption until all required information has been exchanged or the OBE exits the beacon's communications zone. However, in some cases the RSE may generate commands that cannot be completed promptly by the OBE. For example, use of access credentials may require OBE processing that delays the command exchange. Such delays result in a suspended session, which may be resumed using the methods described below.

As described in the previous subclauses, the normal communications processing flow that would lead to a suspended session is as follows:

a) OBE application calls RegisterApplicationVehicle to signal the OBE Layer 7 that it is ready to begin operation. 1455-compliant usage of RegisterApplicationVehicle requires that the Parameter field is not specified in the call.

b) A BST is received by the OBE Layer 1 and passed to OBE Layer 2, and OBE Layer 7 is signaled using SEND_BST_RESPOND.indication. OBE LAYER 7 then analyzes the BST to determine whether it was transmitted by a beacon with which a session is already ongoing. If a session is not ongoing, the OBE Layer 7 then constructs a VST and transmits it to the RSE. The VST contains a Parameter field that is filled by the OBE Layer 7 with the memory pages requested in the BST. If the BST was transmitted by a beacon with which a session is already ongoing, then the OBE Layer 7 discards the BST.

c) OBE Layer 7 also signals the OBE application that a BST has been received using NotifyApplica-tionVehicle. This signal carries the LID of the session, which has been assigned by the OBE Layer 7.

d) OBE Layer 7 then receives a command from the RSE and passes it to the OBE application using the signal Action.indication.

e) The OBE application determines that it cannot process the command in a timely manner. It responds by calling the OBE Layer 7 service Action.response with the ReturnCode set to Processing. The logic for determining that a command cannot be processed in a timely manner is not controlled by this standard and shall be defined by the manufacturer.

At this point the session is considered suspended. The OBE application shall continue to process the delaying command, and the OBE Layer 7 shall continue to process incoming BSTs, discarding those from the current beacon. The following subclauses describe the communications that may occur while the session is suspended and the methods by which the OBE application layer may resume the suspended session.

9.7.1 Command repetition

Once a session is suspended, the RSE may at any time retransmit the command that is being processed by the OBE application. The OBE Layer 7 shall process such commands normally, signaling the OBE application using Action.indication. The OBE application shall detect that this command is a retransmission of the previously received command by analyzing the command identifier and command transaction identifiers within the received command. If these values correspond to those of the command still being processed, then the OBE application shall respond by again calling the OBE Layer 7 service Action.response with the ReturnCode set to Processing. If the command identifier and command transaction identifiers do not correspond to the command currently being processed, then the OBE application shall abort the ongoing processing and shall respond instead to the newly received command. When the OBE application completes command processing, then it shall respond to the command repetition with the completed command response and the ReturnCode set to a value other than Processing.

9.7.2 Session reinitialization

Upon completion of the delaying command, the OBE application may asynchronously call the OBE Layer 7 service Action.response with the completed command response and the ReturnCode set to a value other than Processing. The OBE Layer 7 shall then resume the session by responding to the next BST received from the beacon with a VST, preserving the LID assigned to the original session. In this case the parameter contained within the Action.response may be transmitted to the RSE in either of two methods:

a) The OBE Layer 7 may create a standard VST, including a VST parameter that consists of the memory pages specified in the BST. In this case, the RSE must recognize the preserved LID and in response repeat the command that originally initiated the suspended session, maintaining the same command identifier and command transaction identifier. This command shall be processed normally by OBE Layer 7, but the OBE application shall detect the repetition of the command identifier and command transaction identifier and shall reply promptly by repeating the Action.response signal to the OBE Layer 7, with the completed command response and the ReturnCode set to a value other than Processing.

b) Alternatively, the OBE Layer 7 may create a modified VST, having all standard VST fields, but replacing the BST-specified memory pages in the Parameter field with the parameter contained in the Action.response received from the OBE application. The RSE shall detect that the LID received with the VST corresponds to the previous session and, based upon this detection, shall initiate session-specific processing of the Parameter field. Alternatively, the RSE may discard the VST parameter and obtain the command response by repeating the command as described in 9.7.2 a).

Annex A

(normative)

ASN.1 definitions

A.1 ASN.1 data definitions for the message set for this standard

```
DSRC-APPLICATION-SYSTEM DEFINITIONS IMPLICIT TAGS ::=

BEGIN

--=========================================================

-- This module defines 1455-compliant DSRC application messages

--=========================================================

--

-- Message Header Elements

- -

 DsrcmsgApplicationIdentifier    ::=         INTEGER
       {
             system-appl-id        (0),      -- Reserved for DSRC System
             ettm-appl-id          (1),      -- Electronic Toll & Traffic Mgmt
             cvo-appl-id           (2),      -- Commercial Vehicle Mgmt
             util-appl-id          (3),      -- Common Utilities
             private-appl-id       (60),     -- Private (Uncontrolled)
             privcon-appl-id       (61)      -- Private (Controlled)

       }     (0..63)

 DsrcmsgMessageIdentifier        ::=         INTEGER
         {
               system-msg                    (0)          -- DSRC System
               ettm-etc-msg-1                (1),         -- Toll Entry
               ettm-etc-msg-2                (2),         -- Toll Vehicle Classification
               ettm-etc-msg-3                (3),         -- Toll Variable Pricing
               cvo-border-msg1               (1),         -- Border Trip Identification Number
               cvo-border-msg2               (2),         -- Border Clearance Event
               cvo-border-msg-3              (3),         -- Border Lock Notification
               cvo-border-msg-4              (4),         -- Border Itinerary Verification
               cvo-border-msg-5              (5),         -- Border Warning/Notification
               cvo-mainline-msg-1            (6),         -- Mainline Clearance Identification
```

```
            cvo-mainline-msg-2          (7),        -- Mainline Clearance Event
            cvo-mainline-msg-3          (8),        -- Mainline Clearance ID
            cvo-mainline-msg-4          (9)         -- Mainline Clearance Event
        }   (0..127)

Dsrcmsg-ShortMessageIdentity    ::=     INTEGER
    {
        system-msg                      (0)         -- DSRC System
        ettm-etc-msg-1                  (1),        -- Toll Entry
        ettm-etc-msg-2                  (2),        -- Toll Vehicle Classification
        ettm-etc-msg-3                  (3),        -- Toll Variable Pricing
        ettm-etc-smsg-4                 (4),        -- Toll System Enroll
        cvo-border-smsg1                (5),        -- Border Trip Identification Number
        cvo-border-smsg2                (6),        -- Border Clearance Event
        cvo-border-smsg-3               (7),        -- Border Lock Notification
        cvo-border-smsg-4               (8),        -- Border Itinerary Verification
        cvo-border-smsg-5               (9),        -- Border Warning/Notification
        cvo-mainline-smsg-1             (10),       -- Mainline Clearance Identification
        cvo-mainline-smsg-2             (11),       -- Mainline Clearance Event
        cvo-mainline-smsg-3             (12),       -- Mainline Clearance ID
        cvo-mainline-smsg-4             (13)        -- Mainline Clearance Event
        util-smsg-1                     (14)        -- Utility Text String
        util-smsg-2                     (15)        -- Utility RSE to Other OBE
        util-smsg-3                     (16)        -- Utility Other OBE to RSE
        util-smsg-4                     (17)        -- Utility End Of Data
    }   (0..31)

Dsrcmsg-Date            ::=     INTEGER (0..4095)       -- Days since last decade
Dsrcmsg-Date            ::=     BIT STRING ((8))        -- Message Checksum
Dsrcmsg-Length          ::=     INTEGER (0..255)        -- Bytes in message body
Dsrcmsg-ShortLength     ::=     INTEGER (0..15)         -- Bytes in short message body
--
-- Local Elements
--
Dsrc-Time       ::=     INTEGER (0..4294967295)         -- Seconds since 1/1/70 GMT
                                                        -- max value: 2**32-1


--
-- Message Headers
--
```

```
Dsrcmsg--Header ::=         SEQUENCE
        {
            application-ID          Dsrcmsg-ApplicationIdentifier,      -- Application Identifier
            message-ID              Dsrcmsg-MessageIdentifier,          -- Standard Message Identifier
            message-date            Dsrcmsg-Date INTEGER (0..4095),     -- Days since last decade
            message-length          Dsrcmsg-Length INTEGER (0..255),    -- Bytes in message body
            message-checksum        BIT STRING (SIZE(8))                -- Message body XOR


        }

Dsrcmsg-ShortHeader     ::=     SEQUENCE
        {
            message-ID              Dsrcmsg-ShortMessageIdentity,       -- Short message identifier
            message-date            INTEGER (0..120),                   -- Months since last decade
            message-length          Dsrcmsg-ShortLength,                -- In byte pairs
            message-checksum        BIT STRING (SIZE(8))                -- Message body XOR
        }
--

-- Common Application Items

--

Agency-Specifics        ::=     BIT STRING  (SIZE(16))      -- For agency use

Beacon-Identity         ::=     SEQUENCE
        {
            agency-ID               BIT STRING  (SIZE(16)),     -- Agency identifier
            agency-serial           BIT STRING  (SIZE(16))      -- Agency specified serial number
        }

Borderevent-CargoClearance      ::=     BOOLEAN                 -- Go/True - No-Go/False
Borderevent-CargoClearanceFlag  ::=     BOOLEAN                 -- Valid/True - Invalid/False
Borderevent-DriverClearance     ::=     BOOLEAN                 -- Go/True - No-Go/False
Borderevent-DriverClearanceFlag ::=     BOOLEAN                 -- Valid/True - Invalid/False
Borderevent-Message             ::=     IA5String               -- variable size
                                        (SIZE(1..32))
Borderevent-Timestamp           ::=     Dsrc-Time               -- Seconds since 1/1/70 GMT
Borderevent-TractorClearance    ::=     BOOLEAN                 -- Go/True - No-Go/False
Borderevent-TractorClearanceFlag ::=    BOOLEAN                 -- Valid/True - Invalid/False
```

```
Carrier-Identity            ::=     SEQUENCE                    -- 2 Alpha + 2 Numeric
        {
                carrier-region      IA5String (SIZE(4)),        -- Geographical region
                carrier-unique      IA5String (SIZE(12))        -- Unique carrier ID
                carrier-terminal    IA5String (SIZE(4))         -- Unique terminal
        }

Digital-Signature           ::=     BIT STRING (SIZE(64))       -- Digital Signtaure

Driver-Identity             ::=     SEQUENCE
        {
                issuing-agency      IA5String (SIZE(4)),        -- Agency
                fhwa-cdl            IA5String (SIZE(16))        -- Commercial license number
        {

Lock-Identity               ::-     Transponder-Identity        --

Lock-CurrentStatus          ::=     Lock-Status                 --

Lock-HistoryCount           ::=     INTEGER (0..15)             --

Lock-Status                 ::=     INTEGER                     --
        {
                Open                (0),
                Close               (1),
                Bad                 (2)

        }       (0..7)

Lock-Quantity               ::=     INTERGER (0..15)            -- Number of locks

Mainlineevent-PullinClearance ::=   BOOLEAN                     -- Go/True - No/Go-False

Mainelineevent-Timestamp ::=        Dsrc-Time                   -- Seconds since 1/1/70 GMT

Obe-Address                 ::=     BIT STRING (SIZE(32))       --

Rse-Data-Length             ::=     INTERGER (0..255)           --

Rse-Utility-Data            ::=     OCTET STRING (SIZE(1..255)) --

Scale-Type ::=                      INTEGER                     -- Scale classifications
        {
                st-jurisdict-weight             (1),            -- Jurisdictional weight
                st-mainline-wim                 (2),            -- Mainline WIM
                st-rampsort-wim                 (3),            -- Ramp sorter WIM
```

```
        st-slowroll-wim                 (4),        -- Slow rollover WIM
        st-static-weight                (5),        -- Static scale weight
        st-operator-weight              (15)        -- Operator-entered weight
    }   (0..15)

Tollenroll-Timestamp        ::=  Dsrc-Time              -- Seconds since 1/1/70 GMT

Tollevent-Timestamp         ::=  Dsrc-Time              -- Seconds since 1/1/70 GMT

Toll-Amount                 ::=  INTEGER (0..65535)     -- Toll amount in pennies;

Transponder-DigitalSignature ::= BIT STRING (SIZE(64))  --

Transponder-Identity        ::=  SEQUENCE

        serial-type         BIT STRING (SIZE(4))        -- Serial number type
        manufacturer-ID     BIT STRING (SIZE(10)),      -- Manufacturer identifier
        unit-serial         BIT STRING (SIZE(26)),      -- Unique manufacturer serial

    }

Triploat-CarrierSerial      ::=  NumericString (0..9) (SIZE(6))  --

Triploat-DunsNumber         ::=  NumericString (0..9) (SIZE(9))  --

Utility-Text                ::=  IA5Sring (Size(1..256))         -- Utility-Text

Vehicle-AxleNumber          ::=  INTEGER (2..17)                 -- Max of 17 axles

Vehicle-AxleSpacing         ::=  INTEGER (0..62)                 -- .5 m steps

Vehicle-AxleWeight          ::=  INTEGER                         -- 10 kg steps

Vehicle-Characteristics     ::=  BIT STRING (SIZE(16))           -- Agency description of vehicle

Vehicle-ComponentIdentity   ::=  IA5String (SIZE(17))            -- Pulled Unit VIN

Vehicle-CargoType           ::=  IA5String (SIZE(6))             -- CFR 49 part 172

Vehicle-GrossWeight         ::=  INTEGER (0..16383)              -- 10-kg steps

Vehicle-Identity            ::=  IA5String (SIZE(17))            -- VIN

Vehicle-ItineraryQuality    ::=  INTEGER (0..100)                -- 0%-100%

Vehicle-OccupancyCount      ::=  INTEGER (1..4)                  -- Number of persons

END
```

A.2 ASN.1 data definitions for the commands in this standard

```
DSRC-COMMAND-SYSTEM DEFINITIONS IMPLICIT TAGS ::=

BEGIN

--=========================================================

-- This module defines 1455-compliant DSRC command data

--=========================================================

--


  ObeUICmdType ::=          INTEGER                    -- See Table 20
        {
        AbsoluteCmdOff              (0),
        AbsoluteCmdOn               (1)
        TimedCommand                (2),
        FlashingCommand             (3),

        ReservedForFutureP1455Use   (4..255)
        }(0..255)

  -- Command Response Values

  ObeResponse  ::=     INTEGER                       -- See Table 29
        {
              Reserved                    (0)
              CommandSuccess              (1),
              CommandFailed               (2),
              CommandNotRecognized        (3),
              AccessControlError          (4),
              Page Not Defined            (5)
              Partition Not Defined       (6)
              DeviceError                 (7),
              MemoryAccessError           (8),
              PageLengthMismatch          (9),
              InsufficientMemory          (10)
              Previously Reserved         (11)
              Reserved                    (12..239)
              Vendor Area                 (240-255)
        }(0..255)
```

A.3 ASN.1 data definitions for the application layer in this standard

```
Container::= CHOICE{
        integer                 [0]         INTEGER
        bitstring               [1]         BIT STRING,
        octetstring             [2]         OCTET STRING,
        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
        dummy 10                [10]        BIT STRING,
                                                --repeat this line up to
        dummy 127               [127]       BIT STRING,
        ...
        , contI.x [i]       ContainerI.x    --this line shall be given for each imported
                                            --ContainerI.x, where I.x is replaced by the related
                                            --postfix and i is the registered tag starting with 0.
                                            --Gaps shall be filled with contI.x [i] BIT STRING

Action-Request::=SEQUENCE{
            fill                    BIT STRING (SIZE(1)),
            mode                    BOOLEAN,
            eid                     Dsrc-EID,
            actionType              INTEGER (0..255),
            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
}

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

```
AttributeIDList::=SEQUENCE (0..255) OF INTEGER (0..255)

AttributeList::=SEQUENCE (0..255) OF Attributes
        Attribute::=SEQUENCE {
        attributeId                     INTEGER (0...127 ,...)
        attributeValue                  Container
}

BeaconID::=SEQUENCE {
                manufacturerid          INTEGER (0.. 262143),
                individualid            INTEGER (0..233-1)
}
BroadcastPool::=SEQUENCE {
                directoryvalue          Directory,
                content                 SEQUENCE (0.127 ,...) OF File
}
BST::=SEQUENCE {
                beacon                  BeaconID,
                profile                 Profile,
                mandApplications        ApplicationList,
                nonmandApplications     ApplicationList OPTIONAL
                time                    Time,
                profilelist             SEQUENCE (0..255) OF Profile
}

Directory::=SEQUENCE (0.255) OF FileName

Dsrc-EID::=INTEGER (0..255)

DSRCApplicationEntityID::=INTEGER {
        system                          (0)
        automatic-fee-collection        (1)
        freight-fleet-management        (2)
        public-transport                (3)
        traffic-traveler-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)
        mailbox                         (13)
} (0..31,...)
```

```
Initialisation-Request::=          BST

Initialisation-Response::=         VST

File::=                            SEQUENCE (0..255) of Record

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

ObeConfiguration::=SEQUENCE {
        manufacturerID             INTEGER (0.32767).
        equipmentClass             INTEGER (0.65535).
        obeStatus                  INTEGER (0.65535) OPTIONAL
}
Profile::=INTEGER (0..255)
Record::=CHOICE {...,
      recJ.y [i] RecordJ.y         --this line shall be given for each imported RecordJ.y.
                                   --where J.y is replaced by related postfix and j is the
                                   --registered tag

Return Status::=INTEGER {
        noError                    (0),
        access Denied              (1),
        argumentError              (2),
        complexityLimitation       (3),
        processingFailure          (4),
        processing                 (5)
} (0..255)

Time::= INTEGER (0..2^32–1)

T-APDUs::=CHOICE {
        action.request             [1]Action-Request,
        action.response            [2]Action-Response,
        event-report.request       [3]INTEGER,
        event-report.response      [4]INTEGER,
        set.request                [5]INTEGER,
        set.response               [6]INTEGER,
        get.request                [7]INTEGER,
        get.response               [8]INTEGER,
        initialisation.request     [9]Initialisation-Request,
        initialisation.response    [10]Initialisation-Response,
}
```

The INTEGER fields in T-APDU are placeholders, to keep PER encoding consistent with CEN, for application layer services not implemented by this standard. PER encoding starts at zero (0) for the first element of a CHOICE. For example, INITALISATION.request is encoded with a CHOICE value of 8.

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

Annex B

(normative)

ASN.1 considerations

B.1 ASN.1 language specifications

This standard uses ASN.1 specifications as defined by ISO/IEC 8824-1:1995.

B.1.1 Naming

Data names for those application data elements that are defined in IEEE Std 1489-1999 as "ENTITY.AttributeList" change to "Entity-AttributeList;" the capitalization changes and the period is replaced with a dash.

B.1.2 Coding style

Braces are coded on separate lines.

B.2 ASN.1 packed encoding specifications

This standard uses the ASN.1 PER as specified by ISO/IEC 8825-2:1996 for transfer syntax encodings. The unaligned variant is used, and is consistent with the CEN Layer 7 standard.

The PER encodings shall be of a canonical form in that every encoding of a message will have the same sub-field lengths and formats. Any field(s) with varying length will be positioned at the end of a message. This configuration will allow programmers to readily perform encoding and decoding of the transfer syntax without generalized PER routines or an ASN.1 compiler.

B.2.1 Type specifications

Typing recommendations used within this standard are described in B.2.1.1 through B.2.1.5.

B.2.1.1 Types: Integer

The Integer type specification will be used for the following:

a) Whole number measurements
b) Lists of enumerated or coded data items where the binary representation must be encoding independent
c) Indices for replication factors

Constraints for values and size are employed to ensure a fixed size in the transfer syntax.

B.2.1.2 Types: Boolean

The Boolean type is used to define binary data values such as “True/False” and “Go/No-Go.” The expected transfer syntax result is 1 bit.

B.2.1.3 Types: NumericString

The NumericString type is used to define numeric fields in a manner equivalent to Binary Coded Decimal (BCD). The representation of each digit occupies 4 bits in the transfer syntax. The value constraint of “(0..9)” is specified to exclude “blank” as the 0 value. Each 4 bit nibble represents a decimal digit. The number of bits in the transfer syntax is four times the number of digits.

B.2.1.4 Types: IA5String

The IA5String may be constrained by size to limit the number of characters.

B.2.1.5 Types: Bit String

Bit Strings are constrained by size to adhere to application field sizes.

B.2.2 PER encoding considerations

a) Extension markers are not used. This restriction precludes the use of the trailing ellipses in ASN.1 value range specifications [e.g., (0..127...)]. This restriction is specified because extension markers require the use of a leading bit in the field. The requirement for an extension modifier is satisfied at the application level through the use of a predefined value code that indicates the presence of an application-defined extension.

b) Field definitions should be constrained in a manner that forces a fixed length, either through value or size. When this constraint is not possible, the field should be placed at the end of the message.

c) Since the unaligned form of PER is being used, pad bits will occur only at the end of a message, as required to fill out a byte alignment.

d) By definition, PER encoding never includes ASN.1 tags, omits lengths for values that do not vary in length, and omits values of types that have been constrained to carry a single value.

B.2.3 Object identifier specification

The ASN.1 object identifier specifies an “identity tree” to identify a collection of information objects. This identifier is often required as a message prolog to unambiguously identify the appropriate ASN.1 module to the receiver. The transmission of the object identifier is considered to be optional when the context of the message is unambiguous.

The object identifier will not be transmitted for DSRC messages, as there is only one possible ASN.1 module definition and, therefore, no ambiguity in the message context.

Annex C

(normative)

Communications data flows

Figures C.1, C.2, and C.3 illustrate the message flow, through the application layer and lower layer service, between the resource manager and the OBE application during initialization, acknowledged commands, and unacknowledged commands.

The command names shown in between the resource manager/transponder layer (box) and the application layer (box) represent the application layer service interface as defined in 9.4 and 9.5.

The command names shown between the application layer (box) and the lower layer service (box) represent the lower level service interface as described in 9.3.2.

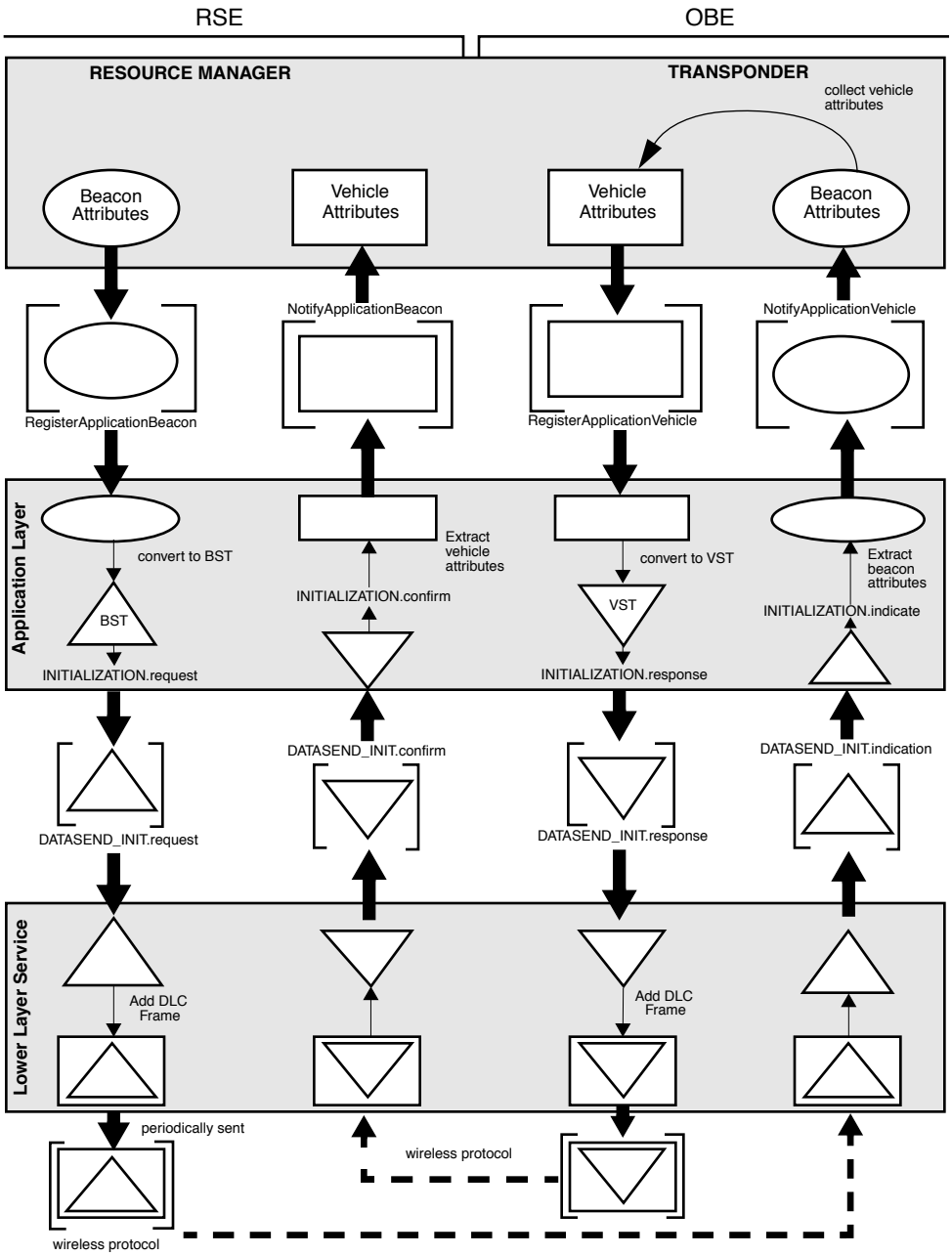

Figure C.1—Initialization sequence

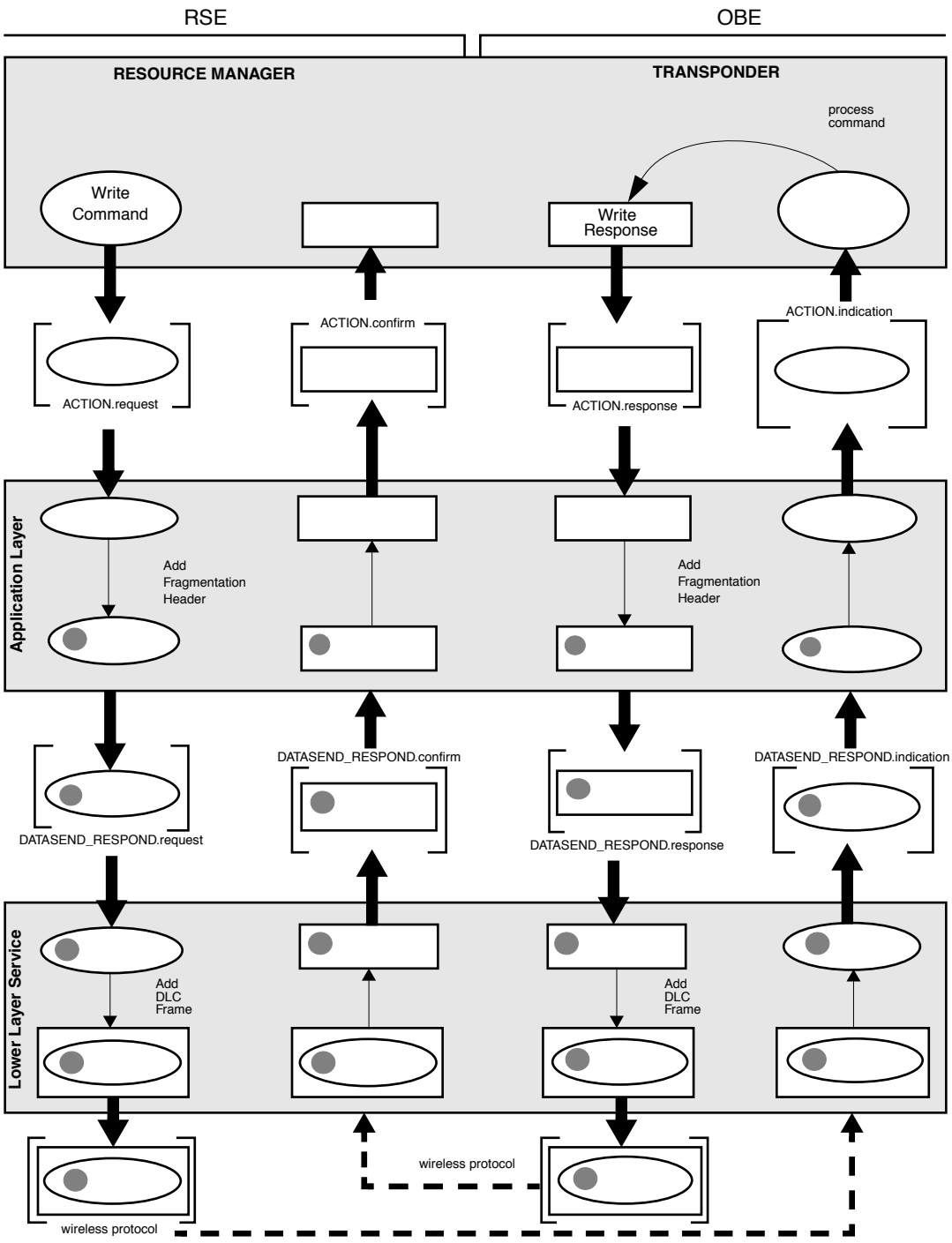

Figure C.2—Acknowledge Write Command

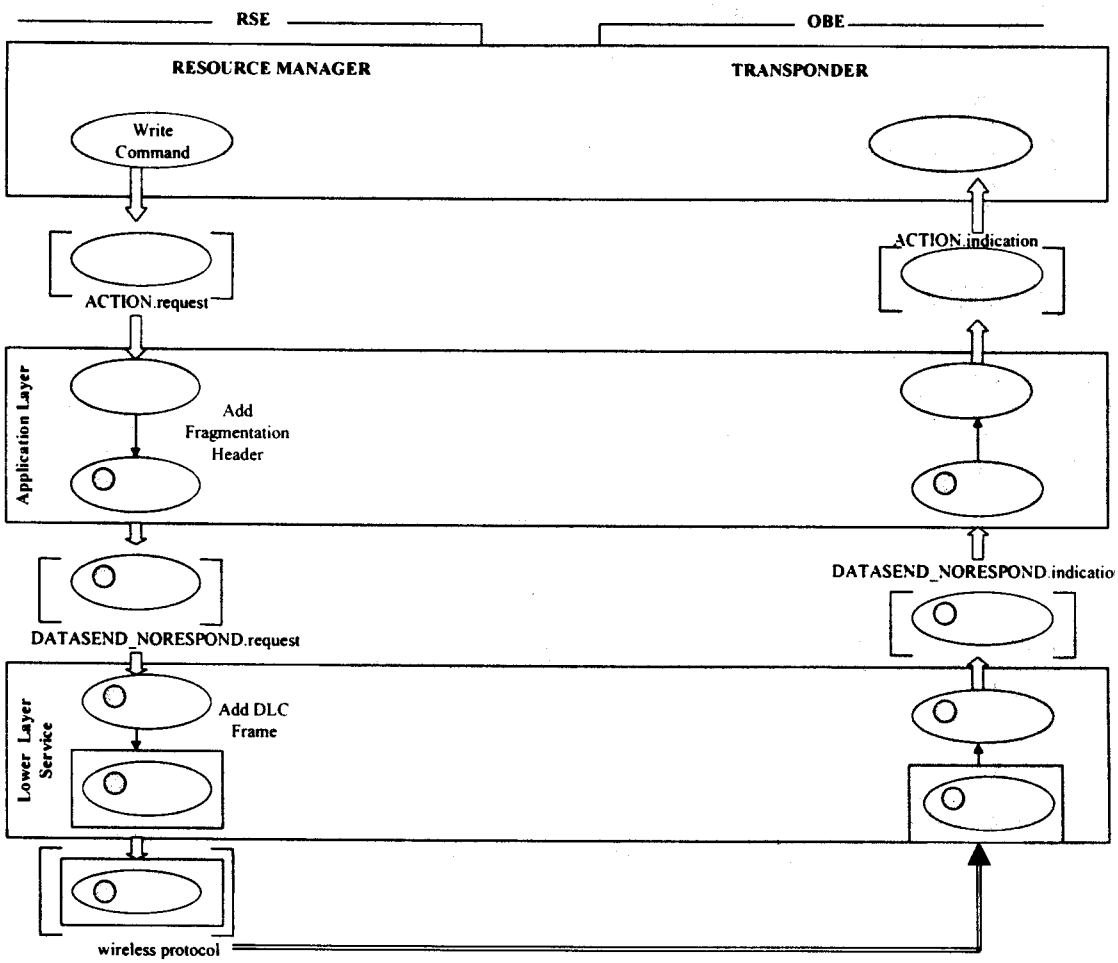

Figure C.3—Unacknowledged Write Command

Annex D

(normative)

ASN.1 PER encoding for a sample BST and VST

Table D.1 provides an example of ASN.1 PER encoding for a BST.

Table D.1—Example of ASN.1 PER encoding for a BST

Octet #	Settings for bits		Description
	7654	3210	
1	1000		T-APDU: INITIALISATION.request; for compliance with CEN ASN.1 BST definition
		0	Indicates optional nonmandatory applications field is not present in the BST
		000	Beginning of manufacturer identifier = Amtech = hex(3)
2	0000	0000	Continuation of manufacturer identifier = Amtech = hex(3)
3	0001	11	End of manufacturer identifier = Amtech = hex(3)
		00	Beginning of individual identifier = hex(00FE)
4	0000	0000	Continuation of individual identifier = hex(00FE)
5	1111	1110	End of individual identifier = hex(00FE)
6	0011	0010	Time = 8514720001 = hex(32C06E81)
7	1100	0000	
8	0110	1110	
9	1000	0001	
10	0000	0010	Profile = hex(2) (Profiles are defined by Layer 2 standards)
11	0000	0001	Number of elements in application list = hex(1)
12	1100	1101	Mailbox application AID = hex(D)
13	0000	0000	EID = IEEE revision level
14	0000	0100	Container tag = octet string = hex(4)
15	0000	1100	Octet string length (in octets) = total for subsequent Filter and Page identifiers = hex(C)
16	0000	0000	Filter Identifier 1 = Page hex(0008)
17	0000	1000	
18	0000	1111	Filter Identifier 2 = Page hex(0F00)
19	0000	0000	

Table D.1—Example of ASN.1 PER encoding for a BST (continued)

Octet #	Settings for bits		Description
	7654	3210	
20	0000	1111	Return Identifier 1 = Page hex(0F01)
21	0000	0001	
22	0000	1111	Return Identifier 2 = Page hex(0F02)
23	0000	0010	
24	0000	1111	Return Identifier 3 = Page hex(0F03)
25	0000	0011	
26	0000	0000	Return Identifier 4 = not used in this example = hex(0000)
27	0000	0000	
28	0000	0001	Number of elements in profile list = hex(1)
29	0000	0101	Profile value = hex(5)

Table D.2 provides an example of ASN.1 PER encoding for a VST.

Table D.2—Example of ASN.1 PER encoding for a VST

Octet #	Settings for bits		Description
	7654	3210	
1	1001		1T-APDU: INITIALISATION.response; for compliance with CEN ASN.1 VST definition
		0000	Fill bits for octet alignment
2	0000	0010	Profile = hex(2)
3	0000	0001	Number of elements in application list = hex(1)
4	1100	1101	Mailbox application AID = hex(D)
5	0000	0010	EID/revision level = initial release = hex(2)
6	0000	0100	Container tag = octet string = hex(4)
7	1001	1010	Octet string length (in octets) = number of octets in returned memory images + size of subsequent fixed fields in read-only memory (Table 3) = hex(9) + hex(A) for this example
8	0011		Bits indicating that the first two pages requested in the BST are returned in this VST
		00	Reserved
		11	Memory configuration = hex(3)
9	1111	1110	Transponder configuration = fully configured = hex(FE)
10	0000	0000	Service agency = California Department of Transportation

Table D.2—Example of ASN.1 PER encoding for a VST (continued)

Octet #	Settings for bits		Description
	7654	3210	
11	0000	0001	
12	0001		Serial number type = clear
		0011	Start of manufacturer identifier = Amtech
13	0000	0000	Continuation of manufacturer identifier
14	0000		End of manufacturer identifier
		1111	Start of serial number = F0F0F
15	0000	1111	Continuation of serial number = F0F0F
16	0000	1111	End of serial number = F0F0F
17 - x	Value of memory		hex(10) octets of returned memory image
x+1 – x+2	0000	0000	Equipment class field of ObeConfiguration = hex(7)
	0000	0011	
x+3 – x+4	0000	0000	Manufacturer Identifier field of ObeConfiguration = Amtech
	0000	0011	
x+5 – x+6	0000	0000	Status field of ObeConfiguration = no status set = hex(0)
	0000	0000	

Annex E

(normative)

Transponder identifier values

This annex contains transponder identifier values that are maintained by a registration process. Samples of these tables are located in the corresponding text clauses.

Table E.1—Registered partition identifiers

Partition number	Partition designation
hex(0000)	Reserved
hex(0001)	Lockheed Martin
hex(0002)	MFS
hex(0003)	Transcore
hex(0004)	Intermec Technologies, Amtech Systems Division
hex(0005)	Mark IV
hex(0006)	Raytheon
hex(0007)	Sirit
hex(0008 .. 00FF)	Available for registration
hex(0100)	California Department of Transportation
hex(0101)	On-board vehicle network - SAE J1708
hex(0102)	On-board vehicle network - SAE J1939
hex(F0103 .. FFFF)	Available for registration

Table E.2—Registered page identifiers

Page number	Page designation
hex(0)	Reserved
hex(1)	Read-only, 128 bits
hex(2)	Short read/write, 128 bits
hex(3)	Long read/write, 256 bits
hex(4)	Pages 1 + 2 as a single page
hex(5)	Pages 1 + 3 as a single page
hex(6)	Pages 1 + 2 + 3 as a single page
hex(7)	Pages 2 + 3 as a single page
hex(8 .. 00FF)	Available for unregistered use by equipment manufacturers to predefine physical pages

Table E.2—Registered page identifiers (continued)

Page number	Page designation
hex(0100)	California Department of Transportation
hex(0101)	On-board vehicle network - SAE J1708
hex(0102)	On-board vehicle network - SAS J1939
hex(0103 .. EFFF)	Available for registration
hex(F000 .. FEFF)	Available for unregistered use by public and private agencies
hex(FF00)	Keypad
hex(FF01)	Enunciator
hex(FF02)	Character readout
hex(FF03 .. FFFF)	Reserved by IEEE for future UI devices

Table E.3—Profile field values

Value	Definition
hex(0)	Reserved
hex(1)	Unspecified profile
hex(2 .. FF)	Available for registration

Table E.4—Transponder Configuration Enumerated field values, Bit 7 Set to 0

Value	Interpretation
hex(0)	Reserved
hex(1 .. 7F)	Available for registration

Table E.5—Service Agency field values

Value	Interpretation
hex(0)	Reserved
hex(1)	California Department of Transportation
hex(2 .. FFFF)	Available for registration

Table E.6—Manufacturer Identifier field values

Value	Interpretation
hex(0)	Reserved
hex(1)	Raytheon HTMS
hex(2)	Mark IV Industries
hex(3)	Amtech
hex(4)	Sirit
hex(5)	Micron
hex(6 .. FFFF)	Available for registration