INTERNATIONAL STANDARD

ISO 17356-4

First edition
2005-11-01

Road vehicles — Open interface for embedded automotive applications —

Part 4: OSEK/VDX Communication (COM)

Véhicules routiers — Interface ouverte pour applications automobiles embarquées —

Partie 4: Communications (COM) OSEK/VDX

Reference number
ISO 17356-4:2005(E)

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

Published in Switzerland

Contents

Page

Foreword ... iv

Introduction ... v

1 Scope ... 1

2 Normative references ... 1

3 Interaction Layer ... 1
3.1 Overview ... 1
3.2 Message reception ... 4
3.3 Message transmission ... 7
3.4 Byte order conversion and message interpretation ... 14
3.5 Deadline monitoring ... 16
3.6 Notification ... 21
3.7 Communication system management ... 22
3.8 Functional model of the Interaction Layer ... 26
3.9 Interfaces ... 28

4 Minimum requirements of lower communication layers ... 43

5 Conformance Classes ... 44

Annex A (informative) **Use of ISO 17356-4 (COM) with an OS not conforming to ISO 17356-3** ... 46

Annex B (informative) **Application notes** ... 47

Annex C (informative) **Callouts** ... 54

Foreword

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

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

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

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

ISO 17356-4 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 17356 consists of the following parts, under the general title *Road vehicles — Open interface for embedded automotive applications*:

— *Part 1: General structure and terms, definitions and abbreviated terms*

— *Part 2: OSEK/VDX specifications for binding OS, COM and NM*

— *Part 3: OSEK/VDX Operating System (OS)*

— *Part 4: OSEK/VDX Communication (COM)*

— *Part 5: OSEK/VDX Network Management (NM)*

— *Part 6: OSEK/VDX Implementation Language (OIL)*

Introduction

This part of ISO 17356 specifies a uniform communication environment for automotive control unit application software. It increases the portability of application software modules by defining common software communication interfaces and behaviour for internal communication [communication within an electronic control unit (ECU)] and external communication (communication between networked vehicle nodes), which is independent of the communication protocol used.

This part of ISO 17356 describes the behaviour within one ECU. It assumes that the communication environment described in this part of ISO 17356 is used together with an operating system that conforms to ISO 17356-3. For information on how to run the communication environment described in this part of ISO 17356 on operating systems that do not conform to ISO 17356-3, refer to Annex A.

Requirements

The following main requirements are fulfilled by this part of ISO 17356:

General communication functionality

This part of ISO 17356 offers services to transfer data between tasks and/or interrupt service routines. Different tasks may reside in one and the same ECU (internal communication) or in different ECUs (external communication). Access to ISO 17356-4 services is only possible via the specified Application Program Interface (API).

Portability, reusability and interoperability of application software

It is the aim of this part of ISO 17356 to support the portability, reusability and interoperability of application software. The API hides the differences between internal and external communication as well as different communication protocols, bus systems and networks.

Scalability

This part of ISO 17356 ensures that an ISO 17356-4 implementation can run on many hardware platforms. The implementation requires only a minimum of hardware resources, therefore different levels of functionality (conformance classes) are provided.

Support for ISO 17356-5 (Network Management-NM):

Services to support Indirect NM are provided. Direct NM has no requirements of this part of ISO 17356.

Communication concept

Figure 1 shows the conceptual model of this part of ISO 17356 and its positioning within the architecture defined by ISO 17356. This model is presented for better understanding, but does not imply a particular implementation of this part of ISO 17356.

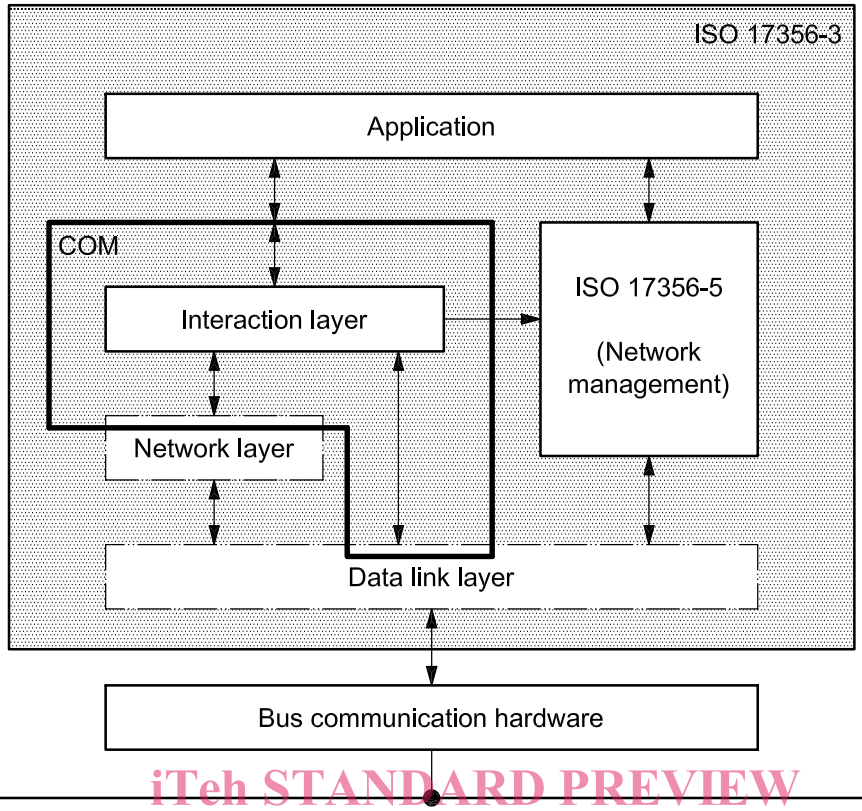

Figure 1 — COM's layer model

In this model, the scope of this part of ISO 17356 partly or entirely covers the following layers:

Interaction Layer

The Interaction Layer (IL) provides the ISO 17356-4 API which contains services for the transfer (send and receive operations) of messages. For external communication it uses services provided by the lower layers, whereas internal communication is handled entirely by the IL.

Network Layer

The Network Layer handles — depending on the communication protocol used — message segmentation/recombination and acknowledgement. It provides flow control mechanisms to enable the interfacing of communication peers featuring different levels of performance and capabilities. The Network Layer uses services provided by the Data Link Layer. This part of ISO 17356 does not specify the Network Layer; it merely defines minimum requirements for the Network Layer to support all features of the IL.

Data Link Layer

The Data Link Layer provides the upper layers with services for the unacknowledged transfer of individual data packets (frames) over a network. Additionally, it provides services for the NM. This part of ISO 17356 does not specify the Data Link Layer; it merely defines minimum requirements for the Data Link Layer to support all features of the IL.

Structure of this document

In the following text, the specification chapters are described briefly. Clauses 1 to 5 are normative, the appendices are descriptive.

Clause 1: Scope
This clause describes the motivation and requirements for this part of ISO 17356, the conceptual model used and the structure of the document.

Clause 2: Normative references

Clause 3: Interaction Layer

This clause describes the functionality of the IL of the ISO 17356-4 model and defines its API.

Clause 4: Minimum requirements of lower communication layers

This clause lists the requirements imposed by this part of ISO 17356 on the lower communication layers (Network Layer and Data Link Layer) to support all features of the IL.

Clause 5: Conformance Classes

This clause specifies the Communication Conformance Classes, which allow the adaptation of the feature content of ISO 17356-4 implementations to the target system's requirements.

Annex A: Use of this part of ISO 17356 (Com) with an OS not conforming to ISO 17356-3

Annex A gives hints on how to run this part of ISO 17356 on operating systems that do not conform to ISO 17356-3.

Annex B: Application notes

Annex B provides information on how to meet specific application requirements with the given ISO 17356-4 model.

Annex C: Callouts

Annex C supplies application examples for callouts.

Road vehicles — Open interface for embedded automotive applications —

Part 4:
OSEK/VDX Communication (COM)

1 Scope

This part of ISO 17356-4 (COM) specifies a uniform communication environment for automatic control unit application software.

It increases the portability of application software modules by defining common software communication interfaces and behaviours for internal communication (communication within an ECU) and external communication (communication between networked vehicle nodes), which is independent of the used communication protocol.

2 Normative references

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

ISO 17356-2, *Road vehicles — Open interface for embedded automotive applications — Part 2 OSEK/VDX specifications for binding OS, COM and NM*

ISO 17356-3, *Road vehicles — Open interface for embedded automotive applications — Part 3 OSEK/VDX Operating System (OS)*

ISO 17356-5, *Road vehicles — Open interface for embedded automotive applications — Part 5 OSEK/VDX Network Management (NM)*

ISO 17356-6, *Road vehicles — Open interface for embedded automotive applications — Part 6 OSEK/VDX Implementation Language (OIL)*

3 Interaction Layer

3.1 Overview

3.1.1 Presentation

The communication in this part of ISO 17356 is based on messages[1)]. A message contains application-specific data. Messages and message properties are configured statically via OIL (ISO 17356-6). The content and usage of messages is not relevant to this part of ISO 17356. Messages with a length of zero (see *zero-length messages*, Annex B) are allowed.

1) Messages are often called *signals*. Thus, COM offers a signal-based interface.

In the case of internal communication, the Interaction Layer (IL) makes the message data immediately available to the receiver (see Figure 2). In the case of external communication, the IL packs **one or more** messages into assigned *Interaction Layer Protocol Data Units* (I-PDU) and passes them to the underlying layer. The functionality of internal communication is a subset of the functionality of external communication. Internal-external communication occurs when the same message is sent internally as well as externally.

Administration of messages is done in the IL based on *message objects*. Message objects exist on the sending side (sending message object) and on the receiving side (receiving message object).

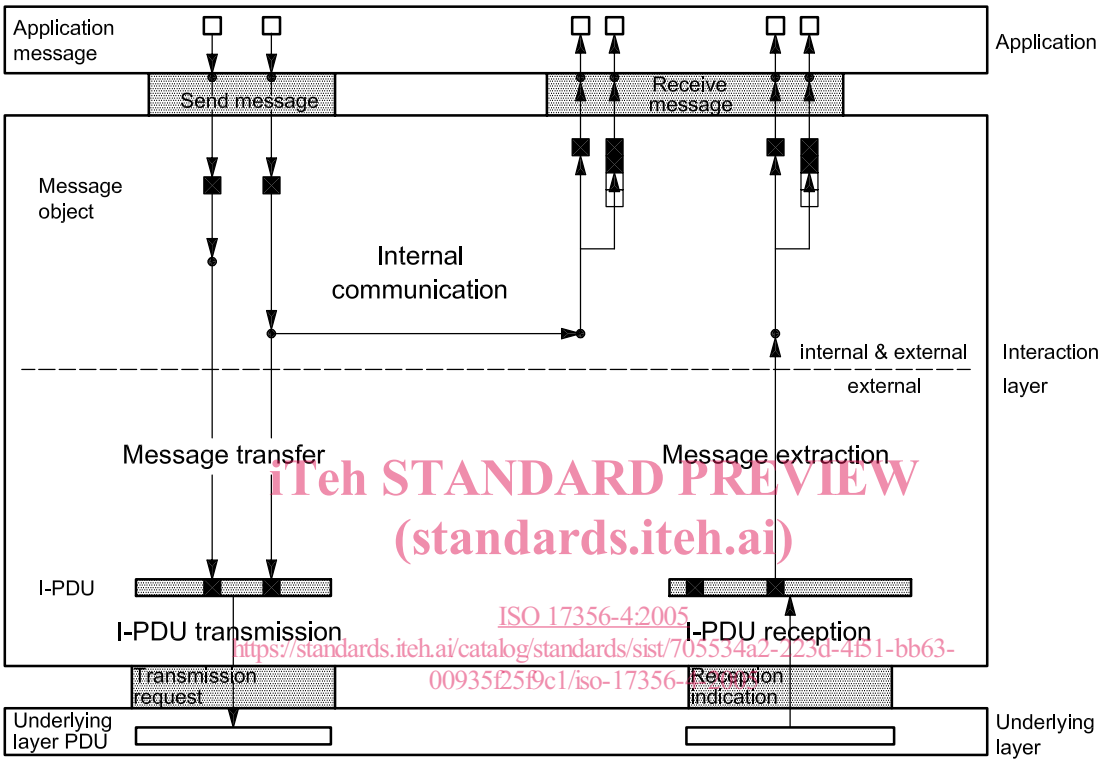

Figure 2 — Simplified model for message transmission and reception in ISO 17356-4

The data that is communicated between the IL and the underlying layer is organized into I-PDUs which contain one or more messages (see Figure 2). A message shall occupy contiguous bits within an I-PDU and shall not be split across I-PDUs. Within an I-PDU messages are bit-aligned. The size of a message is specified in bits.

The byte order (endianess) in a CPU can differ from the network representation or from other CPUs on the network. Therefore, to provide interoperability across the network, the IL provides a conversion from the network representation to the local CPU representation and vice versa, which is statically configured on a per-message basis.

The IL offers an Application Program Interface (API) to handle messages. The API provides services for initialization, data transfer and communication management. Services transmitting messages over network are non-blocking. This implies, for example, that a service that sends a message is unable to return a final transmission status because the transfer to the network is still in progress. This part of ISO 17356 provides *notification mechanisms* for an application to determine the status of a transmission or reception.

The functionality of the IL can be extended by *callouts.* (3.8 contains a description of where callouts can be inserted.)

3.1.2 Communication concept

Senders and receivers of messages are either tasks or interrupt service routines (ISRs) in an OS. Messages are sent to sending message objects and received from receiving message objects.

Message objects are identified using message identifiers. Message identifiers are assigned to message objects at system generation.

This part of ISO 17356 supports communication from “m” senders to “n” receivers (m:n communication). Zero or more senders can send messages to the same sending message object. Sending message objects are configured to store messages in zero or more receiving message objects for internal communication and in zero or one I-PDUs for external communication.

One or more sending message objects can be configured to store messages in the same I-PDU for external communication.

An I-PDU can be received by zero or more CPUs. In each CPU which receives the I-PDU, each message contained in the I-PDU is stored in zero or more receiving message objects. Zero or more receivers can receive messages from a receiving message object (see Annex B for additional information).

A receiving message object receives messages from either exactly one sending message object (internal communication) or exactly one I-PDU, or it receives no messages at all.

A receiving message object can be defined as either queued or unqueued. While a message received by a message object with the property “queued” (queued message) can only be read once (the read operation removes the oldest message from the queue), a message received from a message object with the property “unqueued” (unqueued message) can be read more than once; it returns the last received value each time it is read.

The queue size for message objects with the property “queued” is specified per message object and shall not be zero. If the queue of a receiving message object is full and a new message arrives, this message is lost.

This part of ISO 17356 is not responsible for allocating memory for the application messages, but it allows independent access to message objects for each sender and receiver. In the case of unqueued messages, an arbitrary number of receivers may receive the message. In the case of queued messages, only one receiver may receive the message. The IL guarantees that the data in the application’s message copies are consistent by the following means: the IL deals with messages automatically, and application message data is only read or written during a send or receive service call.

An external message can have one of two transfer properties:

— Triggered Transfer Property: the message in the assigned I-PDU is updated and a request for the I-PDU’s transmission is made.

— Pending Transfer Property: the message in the I-PDU is updated without a transmission request.

Internal messages do not have a transfer property. They are immediately routed to the receiver side.

There are three transmission modes for I-PDUs:

— Direct Transmission Mode: the transmission is explicitly initiated by sending a message with Triggered Transfer Property.

— Periodic Transmission Mode: the I-PDU is transmitted repeatedly with a pre-set period.

— Mixed Transmission Mode: the I-PDU is transmitted using a combination of both the Direct and the Periodic Transmission Modes.

This part of ISO 17356 supports only static message addressing. A statically addressed message has zero or more receivers defined at system generation time, each of which receives the message whenever it is sent. A message has either a static length or its length may vary up to some statically defined maximum. Messages with a maximum length are called *dynamic-length messages*.

This part of ISO 17356 provides a mechanism for monitoring the transmission and reception timing of messages, called *Deadline Monitoring*. Deadline Monitoring verifies on the sender side that the underlying layer confirms transmission requests within a defined time period and on the receiver side that periodic messages are received within a defined time period. The monitoring is performed based on I-PDUs.

The IL provides a fixed set of *filter* algorithms. On the sender side, a filter algorithm may be used which, depending on the message contents, discards the message. In this case, no external transmission is performed and the I-PDU is not updated. There is no filtering on the sender side for internal transmission. On the receiver side, a filter mechanism may be used per receiver in both internal and external transmission. For more details on filtering see 3.2.3 and 3.3.6.

3.1.3 Configuration

The configuration of messages and of their senders and receivers shall be defined at system generation time. Messages cannot be added or deleted at run-time, nor can the packing of messages to I-PDUs be changed. This applies to all configuration elements and their attributes unless otherwise stated.

Examples for configurable items include:

— Configuration of the transfer properties of messages and the transmission modes of I-PDUs,

— Packing of the messages to I-PDUs, and

— Usage of a queue by a receiver and the size of this queue.

The configuration of single CPUs is described in ISO-17356-6.

3.2 Message reception

3.2.1 General

This subclause states the services and functionality requirements of the message reception entity of the IL.

3.2.2 Message reception overview

The first few steps described in this section are applicable for external communication only.

Reception of a message starts with an *indication* of the delivery of its containing PDU from the underlying layer. If this indication does not yield an error, the reception was successful. In this case, an *I-PDU Callout* is called (if configured) and this PDU is copied into the I-PDU.

In the case of unsuccessful PDU reception *error indication* takes place and no data is delivered to the IL. Error indication can lead to *Message Reception Error* notification (Notification Class 3, described in 3.6.2).

After copying the data into the I-PDU further processing is performed separately for each contained message. If the I-PDU contains zero-length messages, these are processed last.

The *Reception Deadline Monitoring* takes place as described in Clause 3.5.1. Deadline Monitoring can invoke *Message Reception Error* notification (Notification Class 3, described in 3.6.2) when the message reception deadline is missed because the I-PDU that contains the message is not received in time.

The message data is then unpacked from the I-PDU and, if configured, a *Network-order Message Callout* is called for the message. Message *byte order conversion* is performed to convert from network representation

to the representation on the local CPU and, if configured, a *CPU-order Message Callout* is called for the message.

The following steps are applicable for both internal and external communication.

The *filtering* is applied to the message content. If the message is not filtered out, then the message data is copied into the receiver message object.

After filtering, *Message Reception* notification (Notification Class 1, described in 3.6.2) is invoked as appropriate. Notification is performed per message object.

Message data are copied from message object to application messages when the application calls the *ReceiveMessage* or *ReceiveDynamicMessage* API services.

3.2.3 Reception filtering

Filtering provides a means to discard the received message when certain conditions, set by message filter, are not met for the message value. The message filter is a configurable function that filters messages out according to specific algorithms. For each message, a different filtering condition can be defined through a dedicated algorithm.

Filtering is only used for messages that can be interpreted as C language unsigned integer types (characters, unsigned integers and enumeration).

For zero-length messages and dynamic-length messages, no filtering takes place.

While receiving messages, only the message values allowed by the filter algorithms pass to the application. If a value has been filtered out, the current instance of the message in the IL represents the last message value that passed through the filter.

Message filtering is performed per message object.

The following attributes are used by the set of filter algorithms (see Table 1):

— *new_value*: current value of the message;

— *old_value*: last value of the message (initialized with the initial value of the message, updated with *new_value* if the new message value is not filtered out);

— *mask, x, min, max, period, offset*: constant values; and

— *occurrence*: a count of the number of occurrences of this message.

If the message filter algorithm is *F_Always* for any particular message, no filter algorithm is included in the runtime system for the particular message.

Table 1 — Message filter algorithms

Algorithm reference	Algorithm	Description
F_Always	True	No filtering is performed so that the message always passes.
F_Never	False	The filter removes all messages.
F_MaskedNewEqualsX	*(new_value&mask) == x*	Pass messages whose masked value is equal to a specific value.
F_MaskedNewDiffersX	*(new_value&mask) != x*	Pass messages whose masked value is not equal to a specific value.
F_NewIsEqual	*new_value == old_value*	Pass messages which have not changed.
F_NewIsDifferent	*new_value != old_value*	Pass messages which have changed.
F_MaskedNewEqualsMaskedOld	*(new_value&mask) == (old_value&mask)*	Pass messages where the masked value has not changed.
F_ MaskedNewDiffersMaskedOld	*(new_value&mask) != (old_value&mask)*	Pass messages where the masked value has changed.
F_NewIsWithin	*min <= new_value <= max*	Pass a message if its value is within a predefined boundary.
F_NewIsOutside	*(min > new_value) || (new_value > max)*	Pass a message if its value is outside a predefined boundary.
F_NewIsGreater	*new_value > old_value*	Pass a message if its value has increased.
F_NewIsLessOrEqual	*new_value <= old_value*	Pass a message if its value has not increased.
F_NewIsLess	*new_value < old_value*	Pass a message if its value has decreased.
F_NewIsGreaterOrEqual	*new_value >= old_value*	Pass a message if its value has not decreased.
F_OneEveryN	*occurrence % period == offset*	Pass a message once every N message occurrences. Start: occurrence = 0. Each time the message is received or transmitted, occurrence is incremented by 1 after filtering. Length of occurrence is 8 bit (minimum).

3.2.4 Copying message data into message objects data area

Message data that are not filtered out are copied into the message object's data. One message may be delivered to one message object or more than one message object. In the latter case, the message objects may be a combination of any number of queued or/and unqueued messages.

Zero-length messages do not contain data. However, the notification mechanisms work in the same way as for non zero-length messages.

3.2.5 Copying data to application messages

The message object's data are copied to the application message by the API services *ReceiveMessage* or *ReceiveDynamicMessage*. The application provides the application message reference to the service.

This transfer of information between IL and application occurs for internal, external and internal-external communication.

For zero-length messages, no data is copied.

3.2.6 Unqueued and queued messages

3.2.6.1 Queued message

A queued message behaves like a FIFO (first-in first-out) queue. When the queue is empty, the IL does not provide any message data to the application. When the queue is not empty and the application receives the message, then the IL provides the application with the oldest message data and removes this message data from the queue.

If new message data arrives and the queue is not full, this new message is stored in the queue. If new message data arrives and the queue is full, this message is lost and the next *ReceiveMessage* call on this message object returns the information that a message has been lost.

NOTE For m:n communication, a separate queue is supported for each receiver and messages from these queues are consumed independently.

3.2.6.2 Unqueued message

Unqueued messages do not use the FIFO mechanism. The application does not consume the message during reception of message data – instead, a message may be read multiple times by an application once the IL has received it.

If no message has been received since the start of the IL, then the application receives the message value set at initialization.

Unqueued messages are overwritten by newly arrived messages.

3.3 Message transmission

3.3.1 Message transmission overview

Sending a message requires the transfer of the application message to the I-PDU (external communication) and/or the receiving message object(s) (internal communication).

A message that is transferred can be stored in zero or more message objects for internal receivers and in zero or one I-PDU for external communication.

The application message is transferred upon calling a specific API service (*SendMessage*, *SendDynamicMessage* or *SendZeroMessage*).

When the API service is called for internal communication, the message is directly handed to the receiving part of the IL (see 3.2) for further processing.

The following description is for external communication only.

For external communication, filtering on the sending side is performed. If the message is discarded, no further action takes place. No filtering takes place on zero-length messages or dynamic-length messages.

Thereafter, if configured, the *CPU-order Message Callout* is called, byte order conversion is performed, the *Network-order Message Callout* is called and the message is stored in the I-PDU.

The transfer of information between the application and IL may use any of the applicable transfer properties of messages: Triggered or Pending.

Transmission of messages via the underlying layers takes place based on I-PDUs. Transmission of I-PDUs may use any of the applicable transmission modes of I-PDUs: Direct, Periodic or Mixed.