

DIN EN 61158-6-8

ICS 35.100.70; 35.160

Ersatzvermerk
siehe unten**Industrielle Kommunikationsnetze –
Feldbusse –****Teil 6-8: Protokollspezifikation des Application Layer
(Anwendungsschicht) – Typ 8-Elemente (IEC 61158-6-8:2007);
Englische Fassung EN 61158-6-8:2008, nur auf CD-ROM**Industrial communication networks –
Fieldbus specifications –Part 6-8: Application layer protocol specification – Type 8 elements (IEC 61158-6-8:2007);
English version EN 61158-6-8:2008, only on CD-ROMRéseaux de communication industriels –
Spécifications des bus de terrain –Partie 6-8: Spécification des services des couches d'application – Éléments de type 8
(CEI 61158-6-8:2007);
Version anglaise EN 61158-6-8:2008, seulement en CD-ROM**Ersatzvermerk**

Mit DIN EN 61158-6-2:2008-09, DIN EN 61158-6-3:2008-09, DIN EN 61158-6-4:2008-09,
DIN EN 61158-6-5:2008-09, DIN EN 61158-6-7:2008-09, DIN EN 61158-6-9:2008-09,
DIN EN 61158-6-10:2008-09, DIN EN 61158-6-11:2008-09, DIN EN 61158-6-12:2008-09,
DIN EN 61158-6-13:2008-09, DIN EN 61158-6-14:2008-09, DIN EN 61158-6-15:2008-09,
DIN EN 61158-6-16:2008-09, DIN EN 61158-6-17:2008-09, DIN EN 61158-6-18:2008-09,
DIN EN 61158-6-19:2008-09 und DIN EN 61158-6-20:2008-09 Ersatz für DIN EN 61158-6:2005-05
Siehe jedoch Beginn der Gültigkeit

Gesamtumfang 106 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Beginn der Gültigkeit

Die von CENELEC am 2008-02-01 angenommene EN 61158-6-8 gilt als DIN-Norm ab 2008-09-01.

Daneben darf DIN EN 61158-6:2005-05 noch bis 2011-02-01 angewendet werden.

Nationales Vorwort

Mit DIN EN 61158-6-2:2008-09, DIN EN 61158-6-3:2008-09, DIN EN 61158-6-4:2008-09, DIN EN 61158-6-5:2008-09, DIN EN 61158-6-7:2008-09, DIN EN 61158-6-9:2008-09, DIN EN 61158-6-10:2008-09, DIN EN 61158-6-11:2008-09, DIN EN 61158-6-12:2008-09, DIN EN 61158-6-13:2008-09, DIN EN 61158-6-14:2008-09, DIN EN 61158-6-15:2008-09, DIN EN 61158-6-16:2008-09, DIN EN 61158-6-17:2008-09, DIN EN 61158-6-18:2008-09, DIN EN 61158-6-19:2008-09 und DIN EN 61158-6-20:2008-09 Ersatz für DIN EN 61158-6:2005-05.

Vorausgegangener Norm-Entwurf: E DIN EN 61158-600:2006-11.

Für diese Norm ist das nationale Arbeitsgremium K 956 „Feldbus“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

Die enthaltene IEC-Publikation wurde vom SC 65C „Industrial networks“ erarbeitet.

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis zu dem Datum (maintenance result date) unverändert bleiben soll, das auf der IEC-Website unter „<http://webstore.iec.ch>“ zu dieser Publikation angegeben ist. Zu diesem Zeitpunkt wird entsprechend der Entscheidung des Komitees die Publikation

- bestätigt,
- zurückgezogen,
- durch eine Folgeausgabe ersetzt oder
- geändert.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Einführung

Ein Feldbus ist ein digitaler serieller Datenbus für die Kommunikation zwischen Geräten der industriellen Automatisierungstechnik wie beispielsweise Messeinrichtungen, Stellgliedern, Reglern und speicherprogrammierbaren Steuerungen. Die Normen der Reihe DIN EN 61158 legen eine Anzahl von Feldbus-Protokolltypen fest.

Die vorliegende Norm legt das Protokoll des Application Layer für einen Feldbus des Typs 8 (INTERBUS) fest.

Erklärung zu Schutzrechten

Die Internationale Elektrotechnische Kommission macht darauf aufmerksam, dass ihr gegenüber erklärt wurde, dass die Übereinstimmung mit dem vorgesehenen Norm-Inhalt die Anwendung von Patenten erforderlich machen könnte. Nähere Informationen hierzu befinden sich im Vorwort der Europäischen Norm.

Es wird darauf aufmerksam gemacht, dass Bestandteile der vorliegenden Norm Gegenstand von weiteren Patentrechten sein können. Es liegt nicht im Verantwortungsbereich der DKE, einige oder alle dieser Patentrechte zu benennen.

Der Leser der vorliegenden Norm wird jedoch gebeten, ihm bekannte derartige Patentrechte der Geschäftsstelle mitzuteilen und entsprechende Informationen zur Verfügung zu stellen.

Änderungen

Gegenüber DIN EN 61158-6:2005-05 wurden folgende Änderungen vorgenommen:

- a) Streichung des Typ 6-Feldbusses wegen zu geringer Marktbedeutung;
- b) Hinzufügung von weiteren Feldbustypen;
- c) Aufteilung in mehrere Teile, jeweils als 6-2, 6-3, ..., 6-20 benummert.

Frühere Ausgaben

DIN V 19245-1: 1988-01
DIN 19245-1: 1991-04
DIN 19245-2: 1991-04
DIN EN 50170-1: 1997-07
DIN EN 50170-2: 1997-07
DIN EN 50170-3: 1997-07
DIN EN 50254: 1999-07
DIN EN 50170/A1: 2002-11
DIN EN 50170/A2: 2000-08
DIN EN 50170/A3: 2002-11
DIN EN 61158-6: 2005-05

– Leerseite –

Deutsche Fassung

Industrielle Kommunikationsnetze –
Feldbusse –
Teil 6-8: Protokollspezifikation des Application Layer (Anwendungsschicht) –
Typ 8-Elemente
(IEC 61158-6-8:2007)

Industrial communication networks –
Fieldbus specifications –
Part 6-8: Application layer protocol specification –
Type 8 elements
(IEC 61158-6-8:2007)

Réseaux de communication industriels –
Spécifications des bus de terrain –
Partie 6-8: Spécification des services des
couches d'application – Éléments de type 8
(CEI 61158-6-8:2007)

Diese Europäische Norm wurde von CENELEC am 2008-02-01 angenommen. Die CENELEC-Mitglieder sind gehalten, die CEN/CENELEC-Geschäftsordnung zu erfüllen, in der die Bedingungen festgelegt sind, unter denen dieser Europäischen Norm ohne jede Änderung der Status einer nationalen Norm zu geben ist.

Auf dem letzten Stand befindliche Listen dieser nationalen Normen mit ihren bibliographischen Angaben sind beim Zentralsekretariat oder bei jedem CENELEC-Mitglied auf Anfrage erhältlich.

Diese Europäische Norm besteht in drei offiziellen Fassungen (Deutsch, Englisch, Französisch). Eine Fassung in einer anderen Sprache, die von einem CENELEC-Mitglied in eigener Verantwortung durch Übersetzung in seine Landessprache gemacht und dem Zentralsekretariat mitgeteilt worden ist, hat den gleichen Status wie die offiziellen Fassungen.

CENELEC-Mitglieder sind die nationalen elektrotechnischen Komitees von Belgien, Bulgarien, Dänemark, Deutschland, Estland, Finnland, Frankreich, Griechenland, Irland, Island, Italien, Lettland, Litauen, Luxemburg, Malta, den Niederlanden, Norwegen, Österreich, Polen, Portugal, Rumänien, Schweden, der Schweiz, der Slowakei, Slowenien, Spanien, der Tschechischen Republik, Ungarn, dem Vereinigten Königreich und Zypern.

CENELEC

Europäisches Komitee für Elektrotechnische Normung
European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique

Zentralsekretariat: rue de Stassart 35, B-1050 Brüssel

© 2008 CENELEC – Alle Rechte der Verwertung, gleich in welcher Form und in welchem Verfahren, sind weltweit den Mitgliedern von CENELEC vorbehalten.

Ref. Nr. EN 61158-6-8:2008 D

Vorwort

Der Text des Schriftstücks 65C/476/FDIS, zukünftige 1. Ausgabe von IEC 61158-6-8, ausgearbeitet von dem SC 65C „Industrial networks“ des IEC/TC 65 „Industrial-process measurement, control and automation“, wurde der IEC-CENELEC Parallelen Abstimmung unterworfen und von CENELEC am 2008-02-01 als EN 61158-6-8 angenommen.

Dieser und die weiteren Teile der Normenreihe EN 61158-6 ersetzen EN 61158-6:2004.

Gegenüber EN 61158-6:2004 wurden die folgenden Änderungen durchgeführt:

- Streichung des Typ 6-Feldbusses wegen zu geringer Marktbedeutung;
- Hinzufügung von weiteren Feldbustypen;
- Aufteilung in mehrere Teile, jeweils als 6-2, 6-3, ..., 6-20 benummert.

Nachstehende Daten wurden festgelegt:

- spätestes Datum, zu dem die EN auf nationaler Ebene durch Veröffentlichung einer identischen nationalen Norm oder durch Anerkennung übernommen werden muss (dop): 2008-11-01
- spätestes Datum, zu dem nationale Normen, die der EN entgegenstehen, zurückgezogen werden müssen (dow): 2011-02-01

ANMERKUNG Die Anwendung einiger der aufgeführten Protokolltypen wird durch die Inhaber der Rechte an geistigem Eigentum hieran eingeschränkt. Die Zusage zur begrenzten Freigabe ihrer Rechte an geistigem Eigentum durch die Inhaber erlaubt in jedem Fall die Anwendung eines bestimmten Data Link Layer-Protokolls zusammen mit einem Physical Layer- und einem Application Layer-Protokoll in Typzusammenstellungen, die in der Normenreihe EN 61784 angegeben sind. Die Anwendung der verschiedenen Protokolltypen in anderen Zusammenstellungen erfordert möglicherweise die Erlaubnis des jeweiligen Inhabers der Rechte an dem entsprechenden geistigen Eigentum.

IEC und CENELEC weisen darauf hin, dass die Übereinstimmung mit dieser Norm die Verwendung der nachstehend aufgeführten Patente erfordern kann, wobei die Schreibweise [xx] auf deren Halter hinweist.

Typ 8:

DE 197 39 297 C2	[PxC] „Automatisierungssystem und Steuervorrichtung zur transparenten Kommunikation zwischen verschiedenen Netzwerken.“
US 2002/0042845 A1	[PxC] „Automation System and connecting Apparatus for the Transparent Communication between two Networks.“

Die IEC und CENELEC nehmen keine Stellung zur Rechtmäßigkeit, zur Gültigkeit und zum Anwendungsbereich dieser Patentrechte.

Der Halter dieser Patentrechte hat IEC zugesichert, dass er/sie bereit ist, über Lizenzen zu vernünftigen und nicht diskriminierenden Geschäftsbedingungen mit Antragstellern in der ganzen Welt zu verhandeln. In diesem Zusammenhang ist die Erklärung des Halters dieser Patentrechte bei IEC registriert. Informationen sind erhältlich bei:

[PxC]: Phoenix Contact GmbH & Co. KG
Intellectual Property Licenses & Standards
Flachsmarktstr. 8
D-32825 Blomberg,
Germany

Es wird auf die Möglichkeit hingewiesen, dass einige Texte dieser Norm Patentrechte berühren können, ohne dass diese vorstehend identifiziert wurden. IEC und CENELEC sind nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren.

Der Anhang ZA wurde von CENELEC hinzugefügt.

Anerkennungsnotiz

Der Text der Internationalen Norm IEC 61158-6-8:2007 wurde von CENELEC ohne irgendeine Abänderung als Europäische Norm angenommen.

In der offiziellen Fassung sind unter „Literaturhinweise“ zu den aufgelisteten Normen die nachstehenden Anmerkungen einzutragen:

IEC 61784-1 ANMERKUNG Harmonisiert als EN 61784-1:2008 (nicht modifiziert).

IEC 61784-2 ANMERKUNG Harmonisiert als EN 61784-2:2008 (nicht modifiziert).

ISO/IEC 9506-2 ANMERKUNG Harmonisiert als EN 29506-2:1993 (nicht modifiziert).

CONTENTS

INTRODUCTION	8
1 Scope	9
1.1 General	9
1.2 Specifications	9
1.3 Conformance	9
2 Normative references	10
3 Terms and definitions	10
3.1 ISO/IEC 7498-1 terms	10
3.9 ISO/IEC 8822 terms	11
3.11 ISO/IEC 9545 terms	11
3.16 ISO/IEC 8824 terms	11
3.37 ISO/IEC 8825 terms	12
3.40 Terms and definitions from IEC 61158-5-8	12
3.47 Other terms and definitions	12
4 FAL syntax description	13
4.1 FAL-AR PDU abstract syntax	13
4.2 Abstract syntax of PDUBody	16
4.3 Type definitions for ASEs	21
4.4 Object definitions	24
4.5 Abstract syntax of Data types	26
5 Transfer syntax	27
5.1 Peripherals encoding rules (PER)	27
5.2 Encoding of APDU types	27
5.3 Encoding of tagged type values	28
5.4 Encoding of simple values	30
6 Protocol machine overview	35
7 AP-context state machine	35
7.1 Primitive definitions	35
7.2 State machine description	36
7.3 AP to AP-context initiation state transitions	37
7.4 Functions	48
8 FAL service protocol machine (FSPM)	51
8.1 Summary	51
8.2 Primitive definitions	51
8.3 FSPM state tables	53
9 Application relationship protocol machines (ARPMs)	56
9.1 Queued user-triggered bidirectional-flow control (QUB-FC) ARPM	56
9.2 Buffered network-scheduled unidirectional (BNU) ARPM	78
9.3 Queued user-triggered bidirectional – transparent mode (QUB-TM) ARPM	86
10 DLL mapping protocol machine	90
10.1 Overview	90
10.2 Primitive definitions	91
10.3 DMPM state machine	94
Bibliography	101

Annex ZA (normative) Normative references to international publications with their corresponding European publications	102
Figure 1 – APDU overview	27
Figure 2 – APDU header	27
Figure 3 – PDU with type extension.....	27
Figure 4 – PDU with address extension	27
Figure 5 – PDU with type and length extension.....	28
Figure 6 – Example of an Establish-Request PDU	28
Figure 7 – Encoding of a PRIVATE tagged value	29
Figure 8 – Encoding of a context specific tagged value	29
Figure 9 – Identification information fields	29
Figure 10 – ID-info for tag 0 .. 14 , length entry 0 .. 6.....	29
Figure 11 – ID-info for tag 15 .. 255 , length entry 0 .. 6.....	30
Figure 12 – ID-info for tag 0 .. 14 , length entry 7 .. 255.....	30
Figure 13 – ID-info for tag 15 .. 255 , length entry 7 .. 255	30
Figure 14 – Encoding of Boolean value TRUE	30
Figure 15 – Encoding of Boolean value FALSE.....	30
Figure 16 – Encoding of Strings	30
Figure 17 – Encoding of BinaryDate value	31
Figure 18 – Encoding of BinaryDate2000 value	32
Figure 19 – Encoding of Time-of-day value.....	32
Figure 20 – Encoding of Time-difference value	33
Figure 21 – Encoding of Time value	33
Figure 22 – Example for an Object definition	35
Figure 23 – Primitives exchanged between protocol machines	35
Figure 24 – AP to AP-context initiation state machine	37
Figure 25 – State transition diagram of FSPM.....	53
Figure 26 – State transition diagram of QUB-FC ARPM	59
Figure 27 – State transition diagram of the BNU ARPM	81
Figure 28 – State transition diagram of QUB-TM AREP.....	88
Figure 29 – State transition diagram of DMPM.....	94
Table 1 – Primitives issued by FAL-user to AP-context	36
Table 2 – Primitives issued by AP-context to FAL-user	36
Table 3 – AP-context state machine sender transactions	37
Table 4 – AP-context state machine receiver transactions	41
Table 5 – Function ResetArep	49
Table 6 – Function ApContextTest	49
Table 7 – Function ServicesSupportedTest.....	49
Table 8 – Function ApExplicitConnection.....	49
Table 9 – Function ImmediateAcknowledge	49
Table 10 – Function ConfirmedServiceCheck	49

Table 11 – Function UnconfirmedServiceCheck	49
Table 12 – Function ArServiceCheck	50
Table 13 – Function ArFspmService	50
Table 14 – Function ArAcceeSupported	50
Table 15 – Function MaxFalPduLengthTest	50
Table 16 – Function NegotiateOutstandingServices	50
Table 17 – Function RequestedServicesSupportedTest	51
Table 18 – Function IndicatedServicesSupportedTest	51
Table 19 – Function InvokeldExistent	51
Table 20 – Function SameService	51
Table 21 – Primitives issued by AP-context to FSPM	52
Table 22 – Primitives issued by FSPM to AP-context	52
Table 23 – FSPM state table – sender transactions	53
Table 24 – FSPM state table – receiver transactions	55
Table 25 – Function SelectArep	55
Table 26 – Primitives issued by FSPM to ARPM	56
Table 27 – Primitives issued by ARPM to FSPM	56
Table 28 – Parameters used with primitives exchanged between FSPM and ARPM	57
Table 29 – QUB-FC ARPM states	59
Table 30 – QUB-FC ARPM state table – sender transactions	60
Table 31 – QUB-FC ARPM state table – receiver transactions	65
Table 32 – Function GetArepId ()	76
Table 33 – Function BuildFAL-PDU	76
Table 34 – Function FAL_Pdu_Type	77
Table 35 – Function AREPContextCheck()	77
Table 36 – Function AbortIdentifier	77
Table 37 – Function AbortReason	77
Table 38 – Function AbortDetail	77
Table 39 – Function StartTimer	78
Table 40 – Function StopTimer	78
Table 41 – Function ResetCounters	78
Table 42 – Function IncrementCounter	78
Table 43 – Function DecrementCounter	78
Table 44 – Function GetCounterValue	78
Table 45 – Primitives issued by FSPM to ARPM	79
Table 46 – Primitives issued by ARPM to FSPM	79
Table 47 – Parameters used with primitives exchanged between FSPM and ARPM	79
Table 48 – BNU ARPM states	81
Table 49 – BNU ARPM state table – sender transactions	81
Table 50 – BNU ARPM state table – receiver transactions	82
Table 51 – Function GetArepId ()	85
Table 52 – Function BuildFAL-PDU	85
Table 53 – Function FAL_Pdu_Type	85

Table 54 – Function AbortIdentifier	85
Table 55 – Function AbortReason	85
Table 56 – Function AbortDetail	85
Table 57 – Primitives issued by FAL to ARPM	86
Table 58 – Primitives issued by ARPM to FAL	86
Table 59 – Parameters used with primitives exchanged between FAL and ARPM	86
Table 60 – QUB-TM ARPM states	88
Table 61 – QUB-TM state table - sender transactions	88
Table 62 – QUB-TM state table - receiver transactions	89
Table 63 – Function GetArepId ()	89
Table 64 – Function BuildFAL-PDU	89
Table 65 – Function FAL_Pdu_Type	89
Table 66 – Function ResetCounters	90
Table 67 – Function IncrementCounter	90
Table 68 – Function DecrementCounter	90
Table 69 – Function GetCounterValue	90
Table 70 – Primitives issued by ARPM to DMPM	91
Table 71 – Primitives issued by DMPM to ARPM	92
Table 72 – Parameters used with primitives exchanged between ARPM and DMPM	93
Table 73 – Primitives exchanged between data-link layer and DMPM	93
Table 74 – DMPM state descriptions	94
Table 75 – DMPM state table – sender transactions	94
Table 76 – DMPM state table – receiver transactions	98
Table 77 – Function PickArep	99
Table 78 – Function FindAREP	99
Table 79 – Function SelectNextArep	100
Table 80 – Function ArepRole	100
Table 81 – Function FalArHeader	100
Table 82 – Function AddUcsPduHeader	100
Table 83 – Function RemoveUcsPduHeader	100
Table 84 – Function DILinkStatus	100

INTRODUCTION

This part of IEC 61158 is one of a series produced to facilitate the interconnection of automation system components. It is related to other standards in the set as defined by the “three-layer” fieldbus reference model described in IEC/TR 61158-1.

The application protocol provides the application service by making use of the services available from the data-link or other immediately lower layer. The primary aim of this standard is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer application entities (AEs) at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

- as a guide for implementors and designers;
- for use in the testing and procurement of equipment;
- as part of an agreement for the admittance of systems into the open systems environment;
- as a refinement to the understanding of time-critical communications within OSI.

This standard is concerned, in particular, with the communication and interworking of sensors, effectors and other automation devices. By using this standard together with other standards positioned within the OSI or fieldbus reference models, otherwise incompatible systems may work together in any combination.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-8: Application layer protocol specification – Type 8 elements

1 Scope

1.1 General

The fieldbus application layer (FAL) provides user programs with a means to access the fieldbus communication environment. In this respect, the FAL can be viewed as a “window between corresponding application programs.”

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 8 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard specifies interactions between remote applications and defines the externally visible behavior provided by the Type 8 fieldbus application layer in terms of

- a) the formal abstract syntax defining the application layer protocol data units conveyed between communicating application entities;
- b) the transfer syntax defining encoding rules that are applied to the application layer protocol data units;
- c) the application context state machine defining the application service behavior visible between communicating application entities;
- d) the application relationship state machines defining the communication behavior visible between communicating application entities.

The purpose of this standard is to define the protocol provided to

- 1) define the wire-representation of the service primitives defined in IEC 61158-5-8, and
- 2) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 8 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI application layer structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this standard is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-8.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of protocols standardized in the IEC 61158-6 series.

1.3 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems. Conformance is achieved through implementation of this application layer protocol specification.

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.

IEC 60559, *Binary floating-point arithmetic for microprocessor systems*

IEC 61158-3-8, *Industrial communication networks – Fieldbus specifications – Part 3-8: Data-link layer service definition – Type 8 elements*

IEC 61158-4-8, *Industrial communication networks – Fieldbus specifications – Part 4-8: Data-link layer protocol specification – Type 8 elements*

IEC 61158-5-8, *Industrial communication networks – Fieldbus specifications – Part 5-8: Application layer service definition – Type 8 elements*

ISO/IEC 7498 (all parts), *Information technology – Open Systems Interconnection – Basic Reference Model*

ISO/IEC 8822, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

ISO/IEC 8825, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 9545, *Information technology – Open Systems Interconnection – Application Layer structure*

3 Terms and definitions

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

3.1 ISO/IEC 7498-1 terms

This standard is partly based on the concepts developed in ISO/IEC 7498-1, and makes use of the following terms defined therein:

3.2
application entity

3.3
application process

3.4
application protocol data unit

3.5
application service element

3.6
application entity invocation

3.7
application transaction

3.8 transfer syntax

3.9 ISO/IEC 8822 terms

For the purposes of this document, the following term as defined in ISO/IEC 8822 applies:

3.10 abstract syntax

3.11 ISO/IEC 9545 terms

For the purposes of this document, the following terms as defined in ISO/IEC 9545 apply:

3.12 application-context

3.13 application-process-type

3.14 application-service-element

3.15 application control service element

3.16 ISO/IEC 8824 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824 apply:

3.17 any type

3.18 bitstring type

3.19 Boolean type

3.20 choice type

3.21 component type

3.22 false

3.23 integer type

3.24 module

3.25 null type

3.26 object identifier

3.27 octetstring type

3.28 production

3.29 simple type

3.30
sequence of type

3.31
sequence type

3.32
structured type

3.33
tag

3.34
tagged type

3.35
true

3.36
type

3.37 ISO/IEC 8825 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8825 apply:

3.38
encoding (of a data value)

3.39
data value

3.40 Terms and definitions from IEC 61158-5-8

3.41
application relationship

3.42
client

3.43
error class

3.44
publisher

3.45
server

3.46
subscriber

3.47 Other terms and definitions

The following terms and definitions are used in this standard.

3.48
called
service user or a service provider that receives an indication primitive or a request APDU

3.49
calling
service user or a service provider that initiates a request primitive or a request APDU

3.50
management information
network accessible information that supports the management of the Fieldbus environment

3.51

receiving

service user that receives a confirmed primitive or an unconfirmed primitive, or a service provider that receives a confirmed APDU or an unconfirmed APDU

3.52

resource

resource is a processing or information capability of a subsystem

3.53

sending

service user that sends a confirmed primitive or an unconfirmed primitive, or a service provider that sends a confirmed APDU or an unconfirmed APDU

4 FAL syntax description

4.1 FAL-AR PDU abstract syntax

4.1.1 Top level definition

The productions defined here shall be used with the Peripherals Encoding Rules (see 5.2) for APDU encoding.

```
APDU ::= CHOICE {
    [PRIVATE 0] ConfirmedSend-RequestPDU,
    [PRIVATE 1] ConfirmedSend-ResponsePDU,
    [PRIVATE 2] UnconfirmedSend-PDU,
    [PRIVATE 3] UnconfirmedAcknowledgedSend-CommandPDU,
    [PRIVATE 4] Establish-RequestPDU,
    [PRIVATE 5] Establish-ResponsePDU,
    [PRIVATE 6] Establish-ErrorPDU,
    [PRIVATE 7] Abort-PDU,
    [PRIVATE 8] DataSendAcknowledge-PDU
}
```

4.1.2 Confirmed send service

```
ConfirmedSend-RequestPDU ::= SEQUENCE {
    [APPLICATION 0] AddressAREP,
    InvokeID,
    ConfirmedServiceRequest
}

ConfirmedSend-ResponsePDU ::= SEQUENCE {
    [APPLICATION 1] AddressAREP,
    InvokeID,
    pduBody CHOICE {
        ConfirmedServiceResponse,
        ConfirmedServiceError
    }
}
```

4.1.3 Unconfirmed send service

```
UnconfirmedSend-PDU ::= SEQUENCE {
    [APPLICATION 2] AddressAREP,
    InvokeID,
    pduBody CHOICE {
        UnconfirmedServiceRequest,
        UnconfirmedSendPD-PDU
    }
}
```

4.1.4 Unconfirmed acknowledge send service

```
UnconfirmedAcknowledgeSend-CommandPDU ::= SEQUENCE {  
    [APPLICATION 3] AddressAREP,  
    InvokeID,  
    UnconfirmedServiceRequest  
}
```

4.1.5 Establish service

```
Establish-RequestPDU ::= SEQUENCE {  
    [APPLICATION 4] AddressAREP,  
    ConType,  
    MaxOSCC,  
    MaxOSCS,  
    MAXUCSC,  
    MAXUCSS,  
    CIU,  
    InvokeID,  
    initiateRequest  
} [PRIVATE 0] IMPLICIT Initiate-RequestPDU  
  
Establish-ResponsePDU ::= SEQUENCE {  
    [APPLICATION 5] AddressAREP,  
    InvokeID,  
    initiateResponse  
} [PRIVATE 0] IMPLICIT Initiate-ResponsePDU  
  
Establish-ErrorPDU ::= SEQUENCE {  
    [APPLICATION 6] AddressAREP,  
    InvokeID,  
    initiateError  
} [PRIVATE 0] IMPLICIT Initiate-ErrorPDU
```

4.1.6 MaxOSCC

MaxOSCC ::= Unsigned8

4.1.7 MaxOSCS

MaxOSCS ::= Unsigned8

4.1.8 MaxUCSC

MaxUCC ::= Unsigned8

4.1.9 MaxUCSS

MaxUCS ::= Unsigned8

4.1.10 CIU

CIU ::= Unsigned32

4.1.11 InvokeID

InvokeID ::= Unsigned8

4.1.12 ConType

```
ConType ::= ENUMERATED {  
    mmaz (0)  
}
```

4.1.13 Data send acknowledge service

```
DataSendAcknowledge-PDU ::= SEQUENCE {
    Protocol-Code,[APPLICATION 8],Address2ARP,    --- Protocol-Code in the higher nibble of the octet!!!
    Block-Number,
    Block-Length,
    Protocol-Data,
}
```

4.1.14 Protocol-Code

```
Protocol-Code ::= ENUMERATED {
    TCP/IP      (1)          --- TCP/IP protocol
}
```

4.1.15 Block-Number

```
Block-Number ::= Unsigned8
    --- 0 no blocking
    ---- 1...254 block number
    ---- 255 last block
```

4.1.16 Block-Length

```
Block-Length ::= Unsigned
```

4.1.17 Protocol-Data

```
Protocol-Data::= Any          --- transparent protocol transfer
```

4.1.18 Address2 ARP

```
Address2ARP ::= SEQUENCE {
    Destination-Address,
    Source-Address,
    Destination-Node,
    Destination-Subnode,
    Source-Node,
    Source-Subnode
}
```

4.1.19 Destination-Address

```
Destination-Address ::= OctetString          --- 3 Octets
```

4.1.20 Source-Address

```
Source-Address ::= OctetString          --- 3 Octets
```

4.1.21 Destination-Node

```
Destination-Node ::= Unsigned8
```

4.1.22 Source-Node

```
Source-Node ::= Unsigned8
```

4.1.23 Destination-Subnode

```
Destination-Subnode ::= Unsigned8
```

4.1.24 Source-Subnode

```
Source-Subnode ::= Unsigned8
```

4.2 Abstract syntax of PDUBody

4.2.1 Abort service

```
Abort-PDU ::= SEQUENCE {  
    [APPLICATION 7] AddressAREP,  
    Identifier,  
    ReasonCode,  
    AdditionalDetail  
}
```

4.2.2 ConfirmedServiceRequest

```
ConfirmedServiceRequest ::= CHOICE {  
    read-Request          [0] IMPLICIT Read-RequestPDU,  
    write-Request         [3] IMPLICIT Write-RequestPDU,  
    start-Request         [6] IMPLICIT Start-RequestPDU,  
    stop-Request          [9] IMPLICIT Stop-RequestPDU,  
    status-Request        [15] IMPLICIT Status-RequestPDU,  
    identify-Request       [18] IMPLICIT Identify-RequestPDU,  
    getAttributes1-Request [21] IMPLICIT GetAttributes-RequestPDU, -- short form  
    getAttributes2-Request [35] IMPLICIT GetAttributes-RequestPDU, -- long form  
    reset-Request         [36] IMPLICIT Reset-RequestPDU,  
    resume-Request        [39] IMPLICIT Resume-RequestPDU  
}
```

4.2.3 ConfirmedServiceResponse

```
ConfirmedServiceResponse ::= CHOICE {  
    read-Response         [1] IMPLICIT Read-ResponsePDU,  
    write-Response        [4] IMPLICIT Write-ResponsePDU,  
    start-Response        [7] IMPLICIT Start-ResponsePDU,  
    stop-Response         [10] IMPLICIT Stop-ResponsePDU,  
    status-Response       [16] IMPLICIT Status-ResponsePDU,  
    identify-Response     [19] IMPLICIT Identify-ResponsePDU,  
    getAttributes-Response [22] IMPLICIT GetAttributes-ResponsePDU,  
    reset-Response        [37] IMPLICIT Reset-ResponsePDU,  
    resume-Response       [40] IMPLICIT Resume-ResponsePDU  
}
```

4.2.4 ConfirmedServiceError

```
ConfirmedServiceError ::= CHOICE {  
    read-Error            [2] IMPLICIT ErrorType,  
    write-Error           [5] IMPLICIT ErrorType,  
    start-Error           [8] IMPLICIT ErrorFiType,  
    stop-Error            [11] IMPLICIT ErrorFiType,  
    status-Error          [17] IMPLICIT ErrorType,  
    identify-Error        [20] IMPLICIT ErrorType,  
    getAttributes-Error   [23] IMPLICIT ErrorType,  
    reset-Error           [38] IMPLICIT ErrorFiType,  
    resume-Error          [41] IMPLICIT ErrorFiType  
}
```

4.2.5 Error type

```
ErrorType ::= SEQUENCE {  
    errorClass            [0] IMPLICIT ErrorClass,  
    optionalParametersMap [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,  
    additionalCode        [1] IMPLICIT Integer16 OPTIONAL,  
    additionalDescription  [2] IMPLICIT VisibleString OPTIONAL,  
}
```

4.2.6 Error Fi type

```
ErrorFiType ::= SEQUENCE {  
    errorClass  
    additionalCode  
    fiState  
}
```

[0] IMPLICIT ErrorClass,
[1] IMPLICIT Integer16 OPTIONAL,
[3] IMPLICIT Integer8

4.2.7 Error class

```

ErrorClass ::= CHOICE
{
    vfdState          [1] IMPLICIT Integer8 {
                        other (0)
                    },
    applicationReference [2] IMPLICIT Integer8 {
                        other (0),
                        application-unreachable (1),
                        application-reference-invalid (2),
                        context-unsupported (3)
                    },
    definition         [3] IMPLICIT Integer8 {
                        other (0),
                        object-undefined (1),
                        object-attributes-inconsistent (2),
                        name-already-exists (3),
                        type-unsupported (4),
                        type-inconsistent (5)
                    },
    resource           [4] IMPLICIT Integer8 {
                        other (0),
                        memory-unavailable (1)
                    },
    service            [5] IMPLICIT Integer8 {
                        other (0),
                        object-state-conflict (1),
                        pdu-size (2),
                        object-constraint-conflict (3),
                        parameter-inconsistent (4),
                        illegal-parameter (5)
                    },
    access            [6] IMPLICIT Integer8 {
                        other (0),
                        object-invalidated (1),
                        hardware-fault (2),
                        object-access-denied (3),
                        invalid-address (4),
                        object-attribute-inconsistent (5),
                        object-access-unsupported (6),
                        object-non-existent (7),
                        type-conflict (8),
                        named-access-unsupported (9),
                        access-to-element-unsupported (10)
                    },
    objectDescription [7] IMPLICIT Integer8 {
                        other (0),
                        name-length-overflow (1),
                        od-overflow (2),
                        od-write-protected (3),
                        extension-length-overflow (4),
                        od-description-length-overflow (5),
                        operational-problem (6)
                    },
    conclude          [9] IMPLICIT Integer8 {
                        other (0),
                        further-communication-required (1)
                    },
    other             [8] IMPLICIT Integer8 {
                        other (0)
                    }
}

```

4.2.8 Unconfirmed PDUs

```

UnconfirmedServiceRequest ::= CHOICE {
    informationReport-Request [12] IMPLICIT InformationReport-RequestPDU,
    reject-Request            [34] IMPLICIT Reject-RequestPDU
}

```

UnconfirmedSendPD-PDU ::= BIT STRING

4.2.9 Management ASE

4.2.9.1 Get attributes service

```

GetAttributes-Request-PDU ::= SEQUENCE {
    listOfAttributes                               [PRIVATE 0] IMPLICIT Mn_SelectedAttributes,
    accessSpecification CHOICE {
        index                                     [1] IMPLICIT Gn_NumericID,
        variableName                             [2] IMPLICIT Gn_Name,
        fiName                                    [5] IMPLICIT Gn_Name,
        startIndex                               [7] IMPLICIT Gn_NumericID
    }
}

GetAttributes-ResponsePDU ::= SEQUENCE {
    more                                           [PRIVATE 0] IMPLICIT Gn_MoreFollows,
    listOfObjectDefinition                       [PRIVATE 1] IMPLICIT SEQUENCE OF Gn_ObjectDefinition
}

```

4.2.10 Application process ASE

4.2.10.1 Get status service

```

Status-RequestPDU ::= NULL

Status-ResponsePDU ::= SEQUENCE {
    logicalStatus                                [PRIVATE 0] IMPLICIT ENUMERATED {
        ready-for-communication                (0),
        limited-services-permitted             (2),
    },
    physicalStatus                              [PRIVATE 1] IMPLICIT ENUMERATED {
        operational                           (0),
        partially-operational                  (1),
        inoperable                            (2),
        needs-commissioning                    (3)
    },
    localDetail                                [PRIVATE 2] IMPLICIT BitString OPTIONAL
}

```

4.2.10.2 Identify service

```

Identify-RequestPDU ::= NULL

Identify-ResponsePDU ::= SEQUENCE {
    vendorName                                  [PRIVATE 0] IMPLICIT VisibleString,
    modelIdentifier                             [PRIVATE 1] IMPLICIT VisibleString,
    vendorRevision                             [PRIVATE 2] IMPLICIT VisibleString
}

```

4.2.10.3 Initiate service

```

Initiate-RequestPDU ::= SEQUENCE {
    versionObjectDefinitionsCalling              [PRIVATE 0] IMPLICIT Integer16,
    apDescriptorCalling                          [PRIVATE 1] IMPLICIT OctetString,
    accessProtectionSupportedCalling             [PRIVATE 2] IMPLICIT Ap_AccessProtectionSupported,
    passwordAndAccessGroupsCalling              [PRIVATE 3] IMPLICIT Ap_AccessControl,
    configuredMaxPduSizeSending                 [PRIVATE 4] IMPLICIT Unsigned8,
    configuredMaxPduSizeReceiving               [PRIVATE 5] IMPLICIT Unsigned8,
    listOfSupportedServicesCalling               [PRIVATE 6] IMPLICIT Mn_PduSupportedMap
}

Initiate-ResponsePDU ::= SEQUENCE {
    versionObjectDefinitionsCalled               [PRIVATE 0] IMPLICIT Integer16,
    apDescriptorCalled                          [PRIVATE 1] IMPLICIT OctetString,
    accessPrivilegeSupportedCalled               [PRIVATE 2] IMPLICIT Ap_AccessProtectionSupported,
    passwordAndAccessGroupsCalled               [PRIVATE 3] IMPLICIT Ap_AccessControl
}

```

```
Initiate-ErrorPDU ::= SEQUENCE {
    errorCode
        other
        max-fal-pdu-size-insufficient
        service-not-supported
        version-obj-def-incompatible
        user-initiate-denied
        password-error
        profile-number-incompatible
    },
    maxPduLengthSendingCalled
    maxPduLengthReceivingCalled
    listOfSupportedServicesCalled
}

[PRIVATE 0] IMPLICIT ENUMERATED {
    (0),
    (1),
    (2),
    (3),
    (4),
    (5),
    (6)
}

[PRIVATE 1] IMPLICIT Unsigned8,
[PRIVATE 2] IMPLICIT Unsigned8,
[PRIVATE 3] IMPLICIT Mn_PduSupportedMap
```

4.2.10.4 Reject service

```
Reject-RequestPDU ::= SEQUENCE {
    original-invokeID
    reject-code
    pdu-size
}

[PRIVATE 0] IMPLICIT Integer8,
[PRIVATE 1] IMPLICIT ENUMERATED {
    (5)
}
```

4.2.11 Function invocation ASE

4.2.11.1 Reset service

```
Reset-RequestPDU ::= SEQUENCE {
    keyAttribute
}

Gn_KeyAttribute

Reset-ResponsePDU ::= NULL
```

4.2.11.2 Resume service

```
Resume-RequestPDU ::= SEQUENCE {
    keyAttribute
}

Gn_KeyAttribute

Resume-ResponsePDU ::= NULL
```

4.2.11.3 Start service

```
Start-RequestPDU ::= SEQUENCE {
    keyAttribute
}

Gn_KeyAttribute

Start-ResponsePDU ::= NULL
```

4.2.11.4 Stop service

```
Stop-RequestPDU ::= SEQUENCE {
    keyAttribute
}

Gn_KeyAttribute

Stop-ResponsePDU ::= NULL
```

4.2.12 Variable ASE

4.2.12.1 Information report service

```
InformationReport-RequestPDU ::= SEQUENCE {
    index
    subIndex
    value
}

Gn_NumericID,
[PRIVATE 0] IMPLICIT Gn_SubIndex OPTIONAL,
[PRIVATE 1] IMPLICIT ANY
```

4.2.12.2 Read service

```
Read-RequestPDU ::= SEQUENCE {
    index
    subIndex
    Gn_NumericID,
    [PRIVATE 0] IMPLICIT Gn_SubIndex OPTIONAL
}

Read-ResponsePDU ::= SEQUENCE {
    value
    [PRIVATE 0] IMPLICIT ANY
}
```

4.2.12.3 Write service

```
Write-RequestPDU ::= SEQUENCE {
    index
    subIndex
    value
    Gn_NumericID,
    Gn_SubIndex OPTIONAL,
    [PRIVATE 0] IMPLICIT ANY
}

Write-ResponsePDU ::= NULL
```

4.3 Type definitions for ASEs

4.3.1 AP ASE types

4.3.1.1 Ap_AccessProtectionSupported

```
Ap_AccessProtectionSupported ::= Boolean
-- True means Access Protection is supported.
-- False means Access Protection is not supported.
```

```
Ap_AccessControl ::= BitString {
    password_Bit1
    password_Bit2
    password_Bit3
    password_Bit4
    password_Bit5
    password_Bit6
    password_Bit7
    password_Bit8
    access_Groups-1
    access_Groups-2
    access_Groups-3
    access_Groups-4
    access_Groups-5
    access_Groups-6
    access_Groups-7
    access_Groups-8
}
-- The Password (Unsigned8) is encoded as a bit string.
(8),
(7),
(6),
(5),
(4),
(3),
(2),
(1),
(16),
(15),
(14),
(13),
(12),
(11),
(10),
(9)
```

4.3.2 AR ASE types

4.3.2.1 Reason code

```
ReasonCode ::= Unsigned8
```

4.3.2.1.1 Additional detail

```
AdditionalDetaill ::= OctetString
```

4.3.2.2 AREP

```
AddressAREP ::= Unsigned8
-- Least significant octet of DLCEP address
```

4.3.3 Function Invocation ASE types

4.3.3.1 Fi_AccessPrivilege

```
Fi_AccessPrivilege ::= BitString {
    rightToStartPassword          (24),
    rightToStopPassword           (23),
    rightToDeletePassword         (22),
    rightToStartAccessGroup       (20),
    rightToStopAccessGroup        (19),
    rightToDeleteAccessGroup       (18),
    rightToStartAllPartner        (32),
    rightToStopAllPartner         (31),
    rightToDeleteAllPartner        (30),
    password_Bit1                 (8), -- The Password (Unsigned8) is encoded as a bit string.
    password_Bit2                 (7),
    password_Bit3                 (6),
    password_Bit4                 (5),
    password_Bit5                 (4),
    password_Bit6                 (3),
    password_Bit7                 (2),
    password_Bit8                 (1),
    access_Groups-1               (16),
    access_Groups-2               (15),
    access_Groups-3               (14),
    access_Groups-4               (13),
    access_Groups-5               (12),
    access_Groups-6               (11),
    access_Groups-7               (10),
    access_Groups-8               (9)
}
```

4.3.3.2 Fi_State

```
Fi_State ::= Unsigned8 {
    unrunnable                   (1),
    idle                         (2),
    running                      (3),
    stopped                      (4),
    starting                     (5),
    stopping                     (6),
    resuming                     (7),
    resetting                    (8)
}
```

4.3.4 Management ASE types

4.3.4.1 Mn_PduSupportedMap

```
Mn_PduSupportedMap ::= BIT STRING {
    getAttributes-RequestPDU      (1), -- Requester
    start-RequestPDU              (8),
    stop-RequestPDU               (9),
    resume-RequestPDU             (9),
    reset-RequestPDU              (9),
    read-RequestPDU               (11),
    write-RequestPDU              (12),
    informationReport-RequestPDU  (17),
    getAttributes-ResponsePDU     (25), -- Responder
    start-ResponsePDU             (33),
    stop-ResponsePDU              (33),
    resume-ResponsePDU            (33),
    reset-ResponsePDU             (33),
    read-ResponsePDU              (35),
    write-ResponsePDU             (36),
    informationReport-ResponsePDU (41)
}
```

4.3.5 Variable ASE types

4.3.5.1 General types

4.3.5.1.1 Gn_Deletable

Gn_Deletable ::= Boolean
-- True means deletable.
-- False means not deletable.

4.3.5.1.2 Gn_Reusable

Gn_Reusable ::= Boolean
-- True means reusable.
-- False means not reusable.

4.3.5.1.3 Gn_KeyAttribute

Gn_KeyAttribute ::= CHOICE {
-- When this type is specified, only the key attributes of the class referenced are valid.
 numericID [0] IMPLICIT Gn_NumericID,
 name [1] IMPLICIT Gn_Name,
 listName [2] IMPLICIT Gn_Name,
 numericAddress [4] IMPLICIT Gn_NumericAddress,
 symbolicAddress [5] IMPLICIT Gn_SymbolicAddress
}

4.3.5.1.4 Gn_Length

Gn_Length ::= Unsigned8

4.3.5.1.5 Gn_MoreFollows

Gn_MoreFollows ::= Boolean

4.3.5.1.6 Gn_Name

Gn_Name ::= OctetString

4.3.5.1.7 Gn_NumericID

Gn_NumericID ::= Unsigned16
-- The values of this parameter are unique within an AP.

4.3.5.1.8 Gn_ObjectDefinition

Gn_ObjectDefinition ::= OctetString
-- The semantics of this parameter are application specific.

4.3.5.1.9 Gn_SubIndex

Gn_SubIndex ::= Unsigned8

4.3.5.2 Gn_ObjectClass

Gn_ObjectClass ::= ENUMERATED {
 functionInvocation (3),
 fixedLengthStringDataType (5),
 structuredDataType (6),
 fixedLengthStringVariable (7),
 arrayVariable (8),
 dataStructureVariable (9),
}

4.3.5.3 Gn_TypeDescription

```
Gn_TypeDescription ::= CHOICE {  
    boolean [1] Gn_Length,  
    integer8 [2] Gn_Length,  
    integer16 [3] Gn_Length,  
    integer32 [4] Gn_Length,  
    unsigned8 [5] Gn_Length,  
    unsigned16 [6] Gn_Length,  
    unsigned32 [7] Gn_Length,  
    float [8] Gn_Length,  
    visiblestring [9] Gn_Length,  
    octetstring [10] Gn_Length,  
    binaryDate [11] Gn_Length,  
    timeOfDay [12] Gn_Length,  
    timeDifference [13] Gn_Length,  
    bitstring [14] Gn_Length  
}
```

4.4 Object definitions

4.4.1 Top level definition

```
Object-Definition ::= CHOICE {  
    [PRIVATE 0] ListHeader,  
    [PRIVATE 1] DataTypeList,  
    [PRIVATE 2] StaticList,  
    [PRIVATE 3] FunctionInvocationDefinition  
}
```

4.4.2 ListHeader

```
ListHeader ::= SEQUENCE {  
    numericId [PRIVATE 0] IMPLICIT Unsigned16,  
    romRamFlag [PRIVATE 1] IMPLICIT Boolean,  
    maxNameLength [PRIVATE 2] IMPLICIT Unsigned8,  
    accessProtectionSupported [PRIVATE 3] IMPLICIT Boolean,  
    versionOfObjectDefinition [PRIVATE 4] IMPLICIT Integer16,  
    localReferenceOfListHeader [PRIVATE 5] IMPLICIT Unsigned32 OPTIONAL,  
    numberOfEntriesInDataTypeList [PRIVATE 6] IMPLICIT Unsigned16,  
    localReferenceOfDataTypeList [PRIVATE 7] IMPLICIT Unsigned32 OPTIONAL,  
    firstNumericIdOfStaticList [PRIVATE 8] IMPLICIT Unsigned16,  
    numberOfEntriesInStaticList [PRIVATE 9] IMPLICIT Unsigned16,  
    localReferenceOfStaticList [PRIVATE 10] IMPLICIT Unsigned32 OPTIONAL,  
    firstNumericIdOfVariableListDefinition [PRIVATE 11] IMPLICIT Unsigned16,  
    numberOfEntriesInVariableListDefinition [PRIVATE 12] IMPLICIT Unsigned16,  
    localReferenceOfVariableListDefinition [PRIVATE 13] IMPLICIT Unsigned32 OPTIONAL,  
    firstNumericIdOfFunctionInvocationDefinition [PRIVATE 14] IMPLICIT Unsigned16,  
    numberOfEntriesInFunctionInvocationDefinition [PRIVATE 15] IMPLICIT Unsigned16,  
    localReferenceOfFunctionInvocationDefinition [PRIVATE 16] IMPLICIT Unsigned32 OPTIONAL  
}
```

4.4.3 DataTypeList

```

DataTypeList ::= CHOICE {
    [PRIVATE 0] DataTypeDefinition,
    [PRIVATE 1] StructuredDataTypeDefinition
}

DataTypeDefinition ::= SEQUENCE {
    numericId                      Gn_NumericID,
    objectClass                    Gn_ObjectClass,
    dataTypeNameLength             Gn_Length,
    dataTypeName                   [PRIVATE 0] IMPLICIT VisibleString OPTIONAL
}

StructuredDataTypeDefinition ::= SEQUENCE {
    numericId                      Gn_NumericID,
    objectClass                    Gn_ObjectClass,
    numberOfElements               [PRIVATE 0] IMPLICIT Integer8,
    recordList                     SEQUENCE OF SEQUENCE {
        numericIdOfDataTypeDefinition Gn_NumericID,
        dataLength                     Gn_Length
    }
}

```

4.4.4 StaticList

```

StaticList ::= CHOICE {
    [PRIVATE 0] VariableDefinition,
    [PRIVATE 1] ArrayDefinition,
    [PRIVATE 2] StructureDefinition
}

VariableDefinition ::= SEQUENCE {
    numericId                      Gn_NumericID,
    objectClass                    Gn_ObjectClass,
    numericIdOfDataTypeDefinition Gn_NumericID,
    dataLength                     Gn_Length,
    accessPrivilege                Vr_AccessPrivilege OPTIONAL,
    localReferenceOfVariable       [PRIVATE 0] IMPLICIT Unsigned32 OPTIONAL,
    variableName                   [PRIVATE 1] IMPLICIT VisibleString OPTIONAL,
    extension                       [PRIVATE 2] IMPLICIT OctetString OPTIONAL
}

ArrayDefinition ::= SEQUENCE {
    numericId                      Gn_NumericID,
    objectClass                    Gn_ObjectClass,
    numericIdOfDataTypeDefinition Gn_NumericID,
    dataLength                     Gn_Length,
    numberOfElements               [PRIVATE 0] IMPLICIT Unsigned8,
    accessPrivilege                Vr_AccessPrivilege OPTIONAL,
    localReferenceOfArray          [PRIVATE 1] IMPLICIT Unsigned32 OPTIONAL,
    arrayName                      [PRIVATE 2] IMPLICIT VisibleString OPTIONAL,
    extension                       [PRIVATE 3] IMPLICIT OctetString OPTIONAL
}

StructureDefinition ::= SEQUENCE {
    numericId                      Gn_NumericID,
    objectClass                    Gn_ObjectClass,
    numericIdOfDataTypeDefinition Gn_NumericID,
    accessPrivilege                Vr_AccessPrivilege OPTIONAL,
    structureName                  [PRIVATE 0] IMPLICIT VisibleString OPTIONAL,
    extension                       [PRIVATE 1] IMPLICIT OctetString OPTIONAL,
    localReferenceOfElement        [PRIVATE 2] IMPLICIT SEQUENCE OF Unsigned32 OPTIONAL
}

```

4.4.5 FunctionInvocationDefinition

```
FunctionInvocationDefinition ::= SEQUENCE {  
    numericId          Gn_NumericID,  
    objectClass        Gn_ObjectClass,  
    numberOfRelatedObjects [PRIVATE 0] IMPLICIT Unsigned8,  
    accessPrivilege    Fi_AccessPrivilege OPTIONAL,  
    deletable          Gn_Deletable,  
    reusable            Gn_Reusable,  
    functionInvocationState FI_State,  
    numericIdOfLoadRegion SEQUENCE OF Gn_NumericID,  
    functionInvocationName [PRIVATE 1] IMPLICIT VisibleString OPTIONAL,  
    extension           [PRIVATE 2] IMPLICIT OctetString OPTIONAL  
}
```

4.5 Abstract syntax of Data types

4.5.1 Notation for the Boolean type

```
Boolean ::= BOOLEAN  
-- TRUE if the value is non-zero.  
-- FALSE if the value is zero.
```

4.5.2 Notation for the Integer type

```
Integer ::= INTEGER -- any integer  
Integer8 ::= INTEGER (-128..+127) -- range  $-2^7 \leq i \leq 2^7-1$   
Integer16 ::= INTEGER (-32768..+32767) -- range  $-2^{15} \leq i \leq 2^{15}-1$   
Integer32 ::= INTEGER -- range  $-2^{31} \leq i \leq 2^{31}-1$ 
```

4.5.3 Notation for the Unsigned type

```
Unsigned ::= INTEGER -- any non-negative integer  
Unsigned8 ::= INTEGER (0..255) -- range  $0 \leq i \leq 2^8-1$   
Unsigned16 ::= INTEGER (0..65535) -- range  $0 \leq i \leq 2^{16}-1$   
Unsigned32 ::= INTEGER -- range  $0 \leq i \leq 2^{32}-1$ 
```

4.5.4 Notation for the Floating Point type

```
Floating32 ::= BIT STRING SIZE (4) -- IEC-60559Single precision
```

4.5.5 Notation for the BitString type

```
BitString ::= BIT STRING -- For generic use
```

4.5.6 Notation for the OctetString type

```
OctetString ::= OCTET STRING -- For generic use
```

4.5.7 Notation for VisibleString type

```
VisibleString ::= VISIBLE STRING -- For generic use
```

4.5.8 Notation for TimeOfDay type

```
TimeOfDay ::= OctetString6
```

4.5.9 Notation for TimeDifference type

```
TimeDifference ::= OctetString6
```

4.5.10 Notation for BinaryDate2000 type

BinaryDate2000 ::= OctetString8

5 Transfer syntax

5.1 Peripherals encoding rules (PER)

The Peripherals Encoding Rule is the encoding of the PDU types defined in 4.1.

5.2 Encoding of APDU types

5.2.1 Overview of APDU encoding

The APDUs encoded with the PER shall have a uniform format. The APDUs shall consist of two major parts, the “APDU Header” part and the “APDU Body” part as shown in Figure 1.

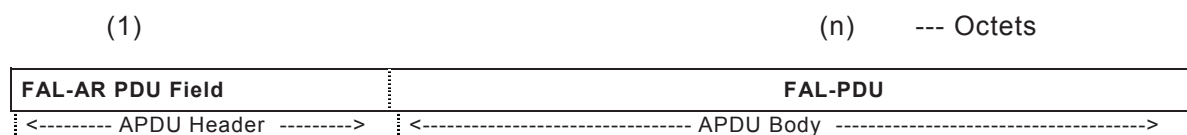


Figure 1 – APDU overview

5.2.2 APDU header encoding

In 4.1, an APDU header is defined as a tagged type of the tag class 'APPLICATION'.

The APDU Header (also called FAL-AR PDU Field) shall be encoded using one or more octets as described in this subclause.

Within the APDU header, the PDU Type tag (e.g. Establish-RequestPDU, ConfirmedSend-RequestPDU...) and the address information (AddressAREP) is encoded .

If the value of the PDU type is < 3, the APDU header shall be encoded using one octet as defined in Figure 2.

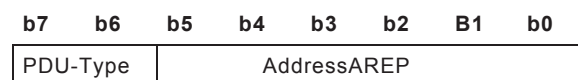


Figure 2 – APDU header

If the value of the PDU type is >= 3 an extension octet is used for the encoding of the tag. The use of a tag extension octet shall be indicated by setting both bits in the PDU type field of the first octet to 1. If the address information (AddressAREP) is greater than 62, an address extension octet shall be used for the encoding of the address value. In this case, the use of an address extension shall be indicated by setting all bits of the address field in the first octet to 1 (see Figure 3 and Figure 4).



Figure 3 – PDU with type extension

The S field is reserved for system management extensions. For FAL-PDUs this flag is set to 0.



Figure 4 – PDU with address extension

Figure 5 shows the encoding of an APDU header using the tag extension and the address extension octet.

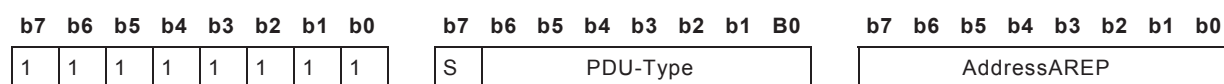


Figure 5 – PDU with type and length extension

Special encoding of the Establish-Request PDU type:

The encoding of the Establish-Request PDU differs from the rules described above, since in this PDU additional parameters, which contain the AR context, are passed. The parameters are the connection type (master-master connection for acyclic data transfer), the parameter CIU (Control Interval User-triggered) and the outstanding service counters. The connection type parameter is encoded in the most-significant three bits. The least-significant 5 bits contain the number of the additional establish parameters.

If the tag for the address requires an extension (the Establish PDU type has an extension), a minimum PDU length of 9 octets is used in an establish request (see Figure 6).

Example for Establish-RequestPDU:

Establish-RequestPDU tag : 4
AddressAREP : 64
ConnectionType : 0 for MMAZ
CIU : FFFFFFFF
S : 0
MaxOSCC : 1
MaxOSCS : 1
MaxUCSC : 1
MaxUCSS : 1

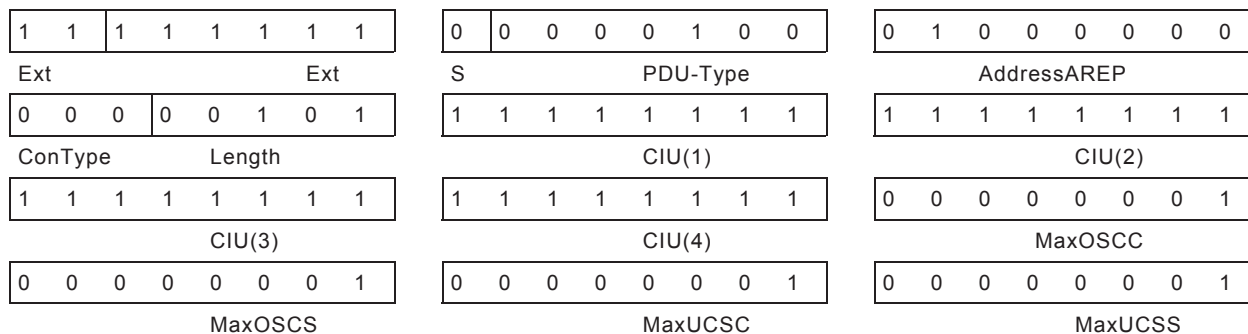


Figure 6 – Example of an Establish-Request PDU

5.2.3 APDU body encoding

The format of the APDU body is described in 4.1 using the ASN.1 syntax. In the following clauses the encoding of the used language elements is described.

5.3 Encoding of tagged type values

5.3.1 Encoding of tagged type values of tag class PRIVATE

Tagged type values with tags of class PRIVATE shall be encoded in the same way as their base types. Tags of class PRIVATE are not explicit encoded (see Figure 7). They are used only to reach syntactical correctness of the ASN.1 definitions.