

DIN EN 61158-6-9

ICS 35.100.70; 35.160

Ersatz für
DIN EN 61158-6-9:2013-03
Siehe Anwendungsbeginn

**Industrielle Kommunikationsnetze –
Feldbusse –
Teil 6-9: Protokollspezifikation des Application Layer
(Anwendungsschicht) – Typ 9-Elemente
(IEC 61158-6-9:2014);
Englische Fassung EN 61158-6-9:2014, nur auf CD-ROM**

Industrial communication networks –
Fieldbus specifications –
Part 6-9: Application layer protocol specification – Type 9 elements
(IEC 61158-6-9:2014);
English version EN 61158-6-9:2014, only on CD-ROM

Réseaux de communication industriels –
Spécifications des bus de terrain –
Partie 6-9: Spécification du protocole de la couche application – Éléments de type 9
(CEI 61158-6-9:2014);
Version anglaise EN 61158-6-9:2014, seulement en CD-ROM

Gesamtumfang 107 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

Anwendungsbeginn

Anwendungsbeginn für die von CENELEC am 2014-09-23 angenommene Europäische Norm als DIN-Norm ist 2015-09-01.

Für DIN EN 61158-6-9:2013-03 besteht eine Übergangsfrist bis 2017-09-23.

Nationales Vorwort

Vorausgegangener Norm-Entwurf: E DIN EN 61158-600:2013-08.

Für dieses Dokument ist das nationale Arbeitsgremium K 956 „Feldbus“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in 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 (stability 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 ein Dokument ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils aktuellste Ausgabe des in Bezug genommenen Dokuments.

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

Der Zusammenhang der zitierten Dokumente mit den entsprechenden Deutschen Dokumenten 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.

Das Präsidium des DIN hat mit Präsidialbeschluss 1/2004 festgelegt, dass DIN-Normen, deren Inhalt sich auf internationale Arbeitsergebnisse der Informationsverarbeitung gründet, unter bestimmten Bedingungen allein in englischer Sprache veröffentlicht werden dürfen. Diese Bedingungen sind für die vorliegende Norm erfüllt.

Da sich die Benutzer der vorliegenden Norm der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der EN 61158-6-9 veröffentlicht. Zu deren Abschnitt 1, der den Anwendungsbereich festlegt, und Abschnitt 3, der die Begriffe festlegt, wurde eine Übersetzung angefertigt und als Nationaler Anhang NA der vorliegenden Norm hinzugefügt. Für viele der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Diese Norm steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

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 Typ 9 fest.

Änderungen

Gegenüber DIN EN 61158-6-9:2013-03 wurden folgende Änderungen vorgenommen:

- a) Wertebereich für Zeitunterschiede berichtigt;
- b) Zustandsübergänge in Tabelle 3 berichtigt;
- c) im formalen Modell BNU AREP Klasse für transparent Timelines eingeführt.

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/A2, 2000-08
DIN EN 50170/A1: 2002-11
DIN EN 50170/A3: 2002-11
DIN EN 61158-6: 2005-05
DIN EN 61158-6-9: 2008-09, 2013-03

Nationaler Anhang NA (informativ)

1 Anwendungsbereich^{N1)}

1.1 Allgemeines

Der Feldbus-Application Layer (FAL) liefert Benutzerprogramme mit Hilfsmitteln für den Zugriff auf die Feldbuskommunikationsumgebung. In dieser Hinsicht kann der FAL als ein Fenster zwischen zugehörigen Anwendungsprogrammen betrachtet werden.

Diese Norm stellt für Typ 9-Feldbusse maßgebliche allgemeine Elemente für zeitkritischen und nicht zeitkritischen Datentransfer zwischen Anwendungsprogrammen in einer Automatisierungsumgebung bereit. Der Begriff „zeitkritisch“ (en: time-critical) wird verwendet, um die Präsenz eines Zeitfensters darzustellen, in dem eine oder mehrere festgelegte Aktivitäten mit einem vorbestimmten Grad an Gewissheit erledigt werden. Das Scheitern der Erledigung der festgelegten Aktivität innerhalb des Zeitfensters trägt das Risiko des Betriebsausfalls der Anwendung mit und somit das aufgeförderte Eingreifen mit dazugehörigem Risiko für Geräte, Anlage und möglicherweise Menschenleben.

Diese Norm definiert auf eine abstrakte Weise das von außen sichtbare Verhalten, das vom Typ 9 des Feldbus-Application Layer bereitgestellt wird, in Form von:

- a) einer abstrakten Syntax zur Festlegung der Application Layer-Protokoll-Dateneinheiten, die zwischen kommunizierenden Anwendungseinheiten übermittelt werden;
- b) der übertragenen Syntax zur Festlegung der Application Layer-Protokoll-Dateneinheiten, die zwischen kommunizierenden Anwendungseinheiten übermittelt werden,;
- c) des Anwendungskontext der Zustandsmaschine zur Festlegung des sichtbaren Dienstverhaltens zwischen kommunizierenden Anwendungseinheiten;
- d) der Anwendungsbeziehung der Zustandsmaschine zur Festlegung des sichtbaren Kommunikationsverhaltens zwischen kommunizierenden Anwendungseinheiten.

Der Zweck dieser Norm ist es, das Protokoll zu definieren, welches bereitgestellt wird zur

- Festlegung der Leitungsdarstellung der Dienstprimitiven, welche in IEC 61158-5-9 beschrieben sind, und
- zur Festlegung des von außen sichtbaren Verhaltens verbunden mit ihrer Übertragung.

Diese Norm beschreibt das Protokoll des Typ 9-Feldbus-Application Layer in Übereinstimmung mit dem OSI-Basis-Referenzmodell (ISO/IEC 7498-1) und der OSI-Application Layer-Struktur (ISO/IEC 9545).

1.2 Spezifikationen

Das hauptsächliche Ziel dieser Norm ist das Festlegen der Syntax und des Verhaltens des Application Layer-Protokolls, welches die in IEC 61158-5-9 beschriebenen Application Layer-Dienste vermittelt.

Ein zweites Ziel ist das Bereitstellen von Migrationspfaden von bisherigen bestehenden industriellen Kommunikationsprotokollen. Es ist dieses letztere Ziel, welches zur Steigerung von unterschiedlichen Protokollen, die in der Reihe IEC 61158-6 genormt werden, führt.

^{N1)} Die Nummerierung aus IEC 61158-6-9 wurde beibehalten.

1.3 Konformität

Diese Norm legt keine individuellen Implementierungen oder Produkte fest, auch beschränkt sie die Implementierungen von Application Layer-Einheiten innerhalb eines industriellen Automatisierungssystems nicht. Konformität wird erreicht durch das Implementieren dieser Application Layer-Protokoll-Spezifikation.

3.2 Begriffe aus IEC 61158-1

Für die Anwendung dieses Dokuments gelten die Begriffe nach IEC 61158-1.

3.2.1

Anwendung

(en: application)

Funktion oder Datenstruktur, für die Daten konsumiert oder produziert werden

3.2.2

Interoperabilität auf Application Layer

(en: application layer interoperability)

Fähigkeit von Anwendungsentitäten, mithilfe der Dienste des FAL koordinierte und kooperative Operationen auszuführen

3.2.3

Anwendungsobjekt

(en: application object)

Objektklasse, die zur Laufzeit für den Nachrichtenaustausch im Netzwerk und innerhalb des Netzwerkgerätes sorgt und diesen verwaltet

Anmerkung 1 zum Begriff: Es können mehrere Typen von Anwendungsobjektklassen definiert werden.

3.2.4

Anwendungsprozess

(en: application process)

Teil einer verteilten Anwendung in einem Netzwerk, der sich auf einem Gerät befindet und eindeutig adressiert wird

3.2.5

Anwendungsprozesskennung

(en: application process identifier)

unterscheidet mehrere in einem Gerät verwendete Anwendungsprozesse

3.2.6

Anwendungsprozessobjekt

(en: application process object)

Komponente eines Anwendungsprozesses, die über eine FAL-Anwendungsbeziehung identifizierbar und zugänglich ist

Anmerkung 1 zum Begriff: Definitionen von Anwendungsprozessobjekten setzen sich aus einem Satz von Werten für die Attribute deren Klasse zusammen (siehe Definition von „Anwendungsprozessobjektklasse“). Auf die Definitionen von Anwendungsprozessobjekten kann mithilfe der Dienste des Objektverwaltungs-ASE der FAL von einem entfernten Standort aus zugegriffen werden. Objektverwaltungsdienste der FAL können zum Laden und Aktualisieren von Objektdefinitionen, zum Lesen von Objektdefinitionen und zur dynamischen Erstellung und Löschung von Anwendungsobjekten und deren Definitionen genutzt werden.

3.2.7

Anwendungsprozessobjektklasse

(en: application process object class)

Klasse von Anwendungsprozessobjekten, die in Bezug auf den Satz der über das Netzwerk zugänglichen Attribute und Dienste der Anwendungsprozessobjekte definiert ist

3.2.8

Anwendungsbeziehung

(en: application relationship)

kooperative Verbindung zwischen zwei oder mehr Aufrufen von Anwendungsentitäten zum Zweck des Informationsaustausches und der Koordination des gemeinsamen Betriebs

Anmerkung 1 zum Begriff: Diese Beziehung wird entweder durch den Austausch von Protokolldateneinheiten der Anwendung oder als Ergebnis von Vorkonfigurationsaktivitäten aktiviert.

3.2.9

Anwendungsdienstelement für Anwendungsbeziehungen

(en: application relationship application service element)

Anwendungsdienstelement, das die alleinigen Mittel für den Aufbau und die Beendigung aller Anwendungsbeziehungen bereitstellt

3.2.10

Endpunkt der Anwendungsbeziehung

(en: application relationship endpoint)

Kontext und Verhalten einer Anwendungsbeziehung entsprechend der Sicht und Verwaltung durch einen der Anwendungsprozesse innerhalb der Anwendungsbeziehung

Anmerkung 1 zum Begriff: Jeder Anwendungsprozess, der in die Anwendungsbeziehung eingeschlossen ist, verwaltet seinen eigenen Endpunkt der Anwendungsbeziehung.

3.2.11

Attribut

(en: attribute)

Beschreibung eines von außen sichtbaren Merkmals oder einer von außen sichtbaren Funktion eines Objekts

Anmerkung 1 zum Begriff: Die Attribute eines Objekts enthalten Informationen zu veränderlichen Teilen eines Objekts. Attribute liefern üblicherweise Statusinformationen oder steuern den Betrieb eines Objekts. Sie können auch das Verhalten eines Objekts beeinflussen. Attribute werden in Klassenattribute und Instanzattribute unterteilt.

3.2.12

Verhalten

(en: behaviour)

Anzeige dafür, wie das Objekt auf bestimmte Ereignisse reagiert

Anmerkung 1 zum Begriff: Seine Beschreibung schließt die Beziehung zwischen Attributwerten und Diensten ein.

3.2.13

Klasse

(en: class)

Menge von Objekten, die allesamt den gleichen Typ von Systemkomponenten repräsentieren

Anmerkung 1 zum Begriff: Eine Klasse ist eine Verallgemeinerung des Objekts; eine Vorlage für die Festlegung von Variablen und Methoden. Alle Objekte in einer Klasse sind in ihrer Form und in ihrem Verhalten identisch, enthalten aber üblicherweise unterschiedliche Daten in ihren Attributen.

3.2.14

Klassenattribute

(en: class attributes)

Attribut, welches allen Objekten innerhalb derselben Klasse gemeinsam ist

3.2.15

Klassencode

(en: class code)

eindeutige Kennung, die jeder Objektklasse zugewiesen ist

3.2.16**klassenspezifischer Dienst**

(en: class specific service)

durch eine bestimmte Objektklasse definierter Dienst, um eine erforderliche Funktion auszuführen, welche nicht als allgemeiner Dienst ausgeführt wird

Anmerkung 1 zum Begriff: Ein klassenspezifisches Objekt ist für die Objektklasse, welche es definiert, eindeutig.

3.2.17**Client**

(en: client)

- (a) Objekt, das die Dienste eines anderen (Server-)Objekts zur Ausführung einer Aufgabe nutzt;
- (b) Initiator einer Nachricht, auf die ein Server reagiert, wie z. B. die Rolle eines AR-Endpunktes, in der dieser bestätigte Dienstanforderungs-APDUs an einen einzelnen AR-Endpunkt, welcher als Server agiert, ausgibt

3.2.18**Übertragungspfad**

(en: conveyance path)

unidirektionaler Strom von APDUs in einer Anwendungsbeziehung

3.2.19**zyklisch**

(en: cyclic)

Ereignisse, die sich regelmäßig wiederholen

3.2.20**zugeordnete AR**

(en: dedicated AR)

Anwendungsbeziehung, die direkt vom FAL-Benutzer verwendet wird

Anmerkung 1 zum Begriff: Bei zugeordneten ARs werden nur der FAL-Header und die Benutzerdaten übertragen.

3.2.21**Gerät**

(en: device)

physische Hardware-Verbindung zum Verbindungszweig

Anmerkung 1 zum Begriff: Ein Gerät kann mehr als einen Knoten umfassen.

3.2.22**Geräteprofil**

(en: device profile)

Zusammenstellung geräteabhängiger Informationen und Funktionalitäten, um Konsistenz zwischen ähnlichen Geräten desselben Gerätetyps herzustellen

3.2.23**dynamische AR**

(en: dynamic AR)

Anwendungsbeziehung, die nur unter Verwendung der Prozeduren zum Aufbau von ARs in einen aktiven Zustand versetzt werden kann

3.2.24**Endpunkt**

(en: endpoint)

eine der Daten übermittelnden Entitäten, die an einer Verbindung beteiligt ist

3.2.25

Fehler

(en: error)

Nichtübereinstimmung zwischen einem berechneten, beobachteten oder gemessenen Wert oder Zustand und dem festgelegten oder theoretisch richtigen Wert oder Zustand

3.2.26

Fehlerklasse

(en: error class)

allgemeine Gruppierung für Fehlerdefinitionen

Anmerkung 1 zum Begriff: Innerhalb einer Fehlerklasse sind Fehlercodes für spezifische Fehler festgelegt.

3.2.27

Fehlercode

(en: error code)

Kennzeichnung eines bestimmten Fehlertyps innerhalb einer Fehlerklasse

3.2.28

FAL-Subnetz

(en: FAL subnet)

Netzwerke, die aus einem oder mehreren Data Link Layer Segmenten bestehen

Anmerkung 1 zum Begriff: Sie dürfen Bridges, jedoch keine Router umfassen. FAL-Subnetze werden durch eine Untermenge der Netzwerkadresse identifiziert.

3.2.29

logisches Gerät

(en: logical device)

legt eine bestimmte FAL-Klasse fest, die eine Softwarekomponente oder eine Firmwarekomponente als eigenständige Einrichtung eines Automatisierungsgerätes abstrahiert

3.2.30

Verwaltungsinformationen

(en: management information)

über das Netzwerk zugängliche Informationen, die die Verwaltung des Betriebs des Feldbussystems, einschließlich des Application Layer, unterstützen

Anmerkung 1 zum Begriff: Zur Verwaltung gehören Funktionen wie die Steuerung, die Überwachung und die Diagnose.

3.2.31

Netzwerk

(en: network)

Reihe von Knoten, die über ein bestimmtes Kommunikationsmedium miteinander verbunden sind

Anmerkung 1 zum Begriff: Die Verbindungspfade zwischen beliebigen Paaren von Knoten können Repeater, Router und Gateways einschließen.

3.2.32

Peer

(en: peer)

Rolle eines AR-Endpunktes, in der dieser in der Lage ist, sowohl als Client als auch als Server zu agieren

3.2.33

vordefinierter AR-Endpunkt

(en: pre-defined AR endpoint)

AR-Endpunkt, der lokal innerhalb eines Gerätes ohne Nutzung des Erstellungsdienstes festgelegt ist

Anmerkung 1 zum Begriff: Vordefinierte ARs, die nicht bereits aktiviert sind, werden vor ihrer Nutzung hergestellt.

3.2.34

vorher aktivierter AR-Endpunkt

(en: pre-established AR endpoint)

AR-Endpunkt, der während der Konfiguration der AEs, die ihre Endpunkte steuern, in einen aktiven Zustand versetzt wird

3.2.35

Publisher

(en: publisher)

Rolle eines AR-Endpunktes, in der dieser APDUs auf den Feldbus für den Konsum durch einen oder mehrere Subscriber überträgt

Anmerkung 1 zum Begriff: Der Publisher muss die Identität oder Anzahl der Subscriber nicht kennen und kann seine APDUs mithilfe einer zugeordneten AR veröffentlichen. In dieser Norm sind zwei Typen von Publishern festgelegt, die jeweils einzeln definiert sind: Pull-Publisher und Push-Publisher.

3.2.36

Server

(en: server)

- a) Rolle eines AREP, in der dieser eine bestätigte Dienstantwort-APDU an den Client, der die Anforderung initiiert hat, zurückgibt;
- b) Objekt, das einem anderen (Client-)Objekt Dienste bereitstellt

3.2.37

Dienst

(en: service)

Operation oder Funktion, die ein Objekt und/oder eine Objektklasse auf Anforderung eines anderen Objekts und/oder einer anderen Objektklasse hin ausführt

Anmerkung 1 zum Begriff: Es sind eine Reihe von allgemeinen Diensten definiert und es sind Richtlinien für die Festlegung objektspezifischer Dienste vorgesehen. Objektspezifische Dienste sind Dienste, die durch eine bestimmte Objektklasse festgelegt sind, um eine erforderliche Funktion auszuführen, die nicht von einem allgemeinen Dienst ausgeführt wird.

3.2.38

Subscriber

(en: subscriber)

Rolle eines AREP, in der dieser von einem Publisher produzierte APDUs empfängt

Anmerkung 1 zum Begriff: In dieser Norm sind zwei Typen von Subscribern festgelegt, die jeweils einzeln definiert sind: Pull-Subscriber und Push-Subscriber.

– Leerseite –

English Version

**Industrial communication networks - Fieldbus specifications -
Part 6-9: Application layer protocol specification - Type 9
elements
(IEC 61158-6-9:2014)**

Réseaux de communication industriels - Spécifications des
bus de terrain - Partie 6-9: Spécification du protocole de la
couche application - Éléments de type 9
(CEI 61158-6-9:2014)

Industrielle Kommunikationsnetze - Feldbusse - Teil 6-9:
Protokollspezifikation des Application Layer
(Anwendungsschicht) - Typ 9-Elemente
(IEC 61158-6-9:2014)

This European Standard was approved by CENELEC on 2014-09-23. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

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

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



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

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Foreword

The text of document 65C/764/FDIS, future edition 3 of IEC 61158-6-9, prepared by SC 65C "Industrial networks" of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61158-6-9:2014.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2015-06-23
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2017-09-23

This document supersedes EN 61158-6-9:2012.

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

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

Endorsement notice

The text of the International Standard IEC 61158-6-9:2014 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

IEC 61784-1	NOTE	Harmonized as EN 61784-1.
IEC 61784-2	NOTE	Harmonized as EN 61784-2.

CONTENTS

INTRODUCTION.....	6
1 Scope	7
1.1 General.....	7
1.2 Specifications	7
1.3 Conformance.....	8
2 Normative references	8
3 Terms, definitions, symbols, abbreviations and conventions	8
3.1 Terms and definitions from other ISO/IEC standards	9
3.2 IEC 61158-1 terms.....	9
3.3 Abbreviations and symbols	13
3.4 Conventions	14
3.5 Conventions used in state machines	15
4 Abstract syntax.....	16
4.1 FAL-AR PDU abstract syntax	16
4.2 Abstract syntax of PDUBody	18
4.3 Type definitions for ASEs.....	21
4.4 Abstract syntax of data types	26
5 Transfer syntax	27
6 Structure of FAL protocol state machines	38
7 AP-Context state machines.....	40
7.1 VCR PM structure.....	40
7.2 VCR PM state machine.....	40
8 FAL service protocol machine (FSPM)	53
8.1 General.....	53
8.2 FSPM state tables	53
8.3 Functions used by FSPM	56
8.4 Parameters of FSPM/ARPM primitives	56
9 Application relationship protocol machines (ARPMs)	57
9.1 AREP mapping to data-link layer.....	57
9.2 Application relationship protocol machines (ARPMs)	66
9.3 AREP state machine primitive definitions	82
9.4 AREP state machine functions	83
10 DLL mapping protocol machine (DMPM).....	85
10.1 DMPM States	85
10.2 DMPM state table	85
10.3 Primitives exchanged between data-link layer and DMPM.....	91
10.4 Functions used by DMPM	93
Bibliography	95
Annex ZA (normative) Normative references to international publications with their corresponding European publications.....	96
Figure 1 – Insertion of identification information in the FMS PDU	28
Figure 2 – Identification	29
Figure 3 – Coding with identification.....	30

Figure 4 – Coding without identification.....	30
Figure 5 – Representation of the value true.....	30
Figure 6 – Representation of the value false	30
Figure 7 – Coding of data of data type Integer16	31
Figure 8 – Coding of data of data type Unsigned16	31
Figure 9 – Coding of data of data type Floating Point	32
Figure 10 – Coding of data of data type Visible String	32
Figure 11 – Coding of data of data type Octet String	33
Figure 12 – Coding of data of type Date	33
Figure 13 – Coding of data of data type Time-of-day	34
Figure 14 – Coding of data of data type Time-difference	35
Figure 15 – Coding of data of data type Bit String.....	35
Figure 16 – Coding of data of data type Time-value.....	36
Figure 17 – Coding of data of user data definitions with identifier.....	36
Figure 18 – Coding of data of user data definitions without identifier	36
Figure 19 – Coding of ID info for a SEQUENCE.....	37
Figure 20 – Relationships among protocol machines and adjacent layers.....	39
Figure 21 – Relationships among protocol machines and adjacent layers.....	40
Figure 22 – VCR state machine.....	41
Figure 23 – State transition diagram of FSPM	53
Figure 24 – State transition diagram of the QUU ARPM	67
Figure 25 – State transition diagram of QUB ARPM.....	69
Figure 26 – State transition diagram of the BNU ARPM	77
Figure 27 – State transition diagram of DMPM.....	85
Table 1 – Conventions used for state machines.....	15
Table 2 – Coding for Date type	33
Table 3 – AP-VCR state machine transactions	42
Table 4 – Primitives issued by FAL-User to VCR PM	51
Table 5 – Primitives issued by VCR PM to FAL-User	51
Table 6 – Primitives issued by VCR PM to FSPM	52
Table 7 – Primitives issued by FSPM to VCR PM	52
Table 8 – FSPM state table – sender transactions	54
Table 9 – FSPM state table – receiver transactions	55
Table 10 – Function SelectArep().....	56
Table 11 – Parameters used with primitives exchanged between FSPM and ARPM.....	56
Table 12 – QUU ARPM states.....	66
Table 13 – QUU ARPM state table – sender transactions	67
Table 14 – QUU ARPM state table – receiver transactions.....	68
Table 15 – QUB ARPM states	68
Table 16 – QUB ARPM state table – sender transactions.....	69
Table 17 – QUB ARPM state table – receiver transactions.....	71

Table 18 – BNU ARPM states	77
Table 19 – BNU ARPM state table – sender transactions	78
Table 20 – BNU ARPM state table – receiver transactions	79
Table 21 – Primitives issued from ARPM to DMPM	82
Table 22 – Primitives issued by DMPM to ARPM	82
Table 23 – Parameters used with primitives exchanged between ARPM and DMPM	83
Table 24 – Function GetArepld()	83
Table 25 – Function BuildFAS-PDU	84
Table 26 – Function FAS_Pdu_Type	84
Table 27 – Function AbortIdentifier	84
Table 28 – Function AbortReason	84
Table 29 – Function AbortDetail	84
Table 30 – DMPM state descriptions	85
Table 31 – DMPM state table – sender transactions	85
Table 32 – DMPM state table – receiver transactions	88
Table 33 – Primitives exchanged between data-link layer and DMPM	91
Table 34 – Function PickArep	93
Table 35 – Function FindAREP	93
Table 36 – Function LocateQubArep	94
Table 37 – Function SetIdentifier()	94

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 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 6-9: Application layer protocol specification – Type 9 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 9 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 defines in an abstract way the externally visible behavior provided by the Type 9 fieldbus Application Layer in terms of

- a) the abstract syntax defining the application layer protocol data units conveyed between communicating application entities,
- b) the transfer syntax defining the application layer protocol data units conveyed between communicating application entities,
- c) the application context state machine defining the application service behavior visible between communicating application entities; and
- d) the application relationship state machines defining the communication behavior visible between communicating application entities; and.

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

- 1) define the wire-representation of the service primitives defined in IEC 61158-5-9, and
- 2) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 9 IEC fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) 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 61158-5-9.

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 IEC 61158-6.

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 documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE All parts of the IEC 61158 series, as well as IEC 61784-1 and IEC 61784-2 are maintained simultaneously. Cross-references to these documents within the text therefore refer to the editions as dated in this list of normative references.

IEC 61158-1, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-3-1, *Industrial communication networks – Fieldbus specifications – Part 3-1: Data-link layer service definition – Type 1 elements*

IEC 61158-4-1, *Industrial communication networks – Fieldbus specifications – Part 4-1: Data-link layer protocol specification – Type 1 elements*

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

IEC 61158-5-9, *Industrial communication networks – Fieldbus specifications – Part 5-9: Application layer service definition – Type 9 elements*

ISO/IEC 646, *Information technology – ISO 7-bit coded character set for information interchange*

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

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

ISO/IEC 8825-1, *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*

ISO/IEC 10731, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Terms and definitions from other ISO/IEC standards

3.1.1 Terms and definitions from ISO/IEC 7498-1

- a) abstract syntax
- b) application entity
- c) application process
- d) application protocol data unit
- e) application service element
- f) application entity invocation
- g) application process invocation
- h) application transaction
- i) presentation context
- j) real open system
- k) transfer syntax

3.1.2 Terms and definitions from ISO/IEC 9545

- a) application-association
- b) application-context
- c) application context name
- d) application-entity-invocation
- e) application-entity-type
- f) application-process-invocation
- g) application-process-type
- h) application-service-element
- i) application control service element

3.1.3 Terms and definitions from ISO/IEC 8824-1

- a) object identifier
- b) type
- c) value
- d) simple type
- e) structured type
- f) component type
- g) tag
- h) Boolean type
- i) true
- j) false
- k) integer type
- l) bitstring type
- m) octetstring type
- n) null type
- o) sequence type
- p) sequence of type
- q) choice type
- r) tagged type
- s) any type
- t) module
- u) production

3.1.4 Terms and definitions from ISO/IEC 8825-1

- a) encoding (of a data value)
- b) data value
- c) identifier octets (the singular form is used in this standard)
- d) length octet(s) (both singular and plural forms are used in this standard)
- e) contents octets

3.2 IEC 61158-1 terms

For the purposes of the present document, the following IEC 61158-1 terms apply.

3.2.1

application

function or data structure for which data is consumed or produced

3.2.2

application layer interoperability

capability of application entities to perform coordinated and cooperative operations using the services of the FAL

3.2.3

application object

object class that manages and provides the run time exchange of messages across the network and within the network device

Note 1 to entry: Multiple types of application object classes may be defined.

3.2.4

application process

part of a distributed application on a network, which is located on one device and unambiguously addressed

3.2.5

application process identifier

distinguishes multiple application processes used in a device

3.2.6

application process object

component of an application process that is identifiable and accessible through an FAL application relationship

Note 1 to entry: Application process object definitions are composed of a set of values for the attributes of their class (see the definition for Application Process Object Class Definition). Application process object definitions may be accessed remotely using the services of the FAL Object Management ASE. FAL Object Management services can be used to load or update object definitions, to read object definitions, and to dynamically create and delete application objects and their corresponding definitions.

3.2.7

application process object class

class of application process objects defined in terms of the set of their network-accessible attributes and services

3.2.8

application relationship

cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation

Note 1 to entry: This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities.

3.2.9

application relationship application service element

application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.2.10

application relationship endpoint

context and behavior of an application relationship as seen and maintained by one of the application processes involved in the application relationship

Note 1 to entry: Each application process involved in the application relationship maintains its own application relationship endpoint.

3.2.11

attribute

description of an externally visible characteristic or feature of an object

Note 1 to entry: The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behaviour of an object. Attributes are divided into class attributes and instance attributes.

3.2.12

behaviour

indication of how the object responds to particular events

Note 1 to entry: Its description includes the relationship between attribute values and services.

3.2.13

class

set of objects, all of which represent the same kind of system component

Note 1 to entry: A class is a generalisation of the object; a template for defining variables and methods. All objects in a class are identical in form and behaviour, but usually contain different data in their attributes.

3.2.14

class attributes

attribute that is shared by all objects within the same class

3.2.15

class code

unique identifier assigned to each object class

3.2.16

class specific service

service defined by a particular object class to perform a required function which is not performed by a common service

Note 1 to entry: A class specific object is unique to the object class which defines it.

3.2.17

client

(a) object which uses the services of another (server) object to perform a task

(b) initiator of a message to which a server reacts, such as the role of an AR endpoint in which it issues confirmed service request APDUs to a single AR endpoint acting as a server

3.2.18

conveyance path

unidirectional flow of APDUs across an application relationship

3.2.19

cyclic

events which repeat in a regular and repetitive manner

3.2.20

dedicated AR

AR used directly by the FAL User

Note 1 to entry: On Dedicated ARs, only the FAL Header and the user data are transferred

3.2.21

device

physical hardware connection to the link

Note 1 to entry: A device may contain more than one node.

3.2.22

device profile

collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.2.23

dynamic AR

AR that requires the use of the AR establishment procedures to place it into an established state

3.2.24

endpoint

one of the communicating entities involved in a connection

3.2.25

error

discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.2.26

error class

general grouping for error definitions

Note 1 to entry: Error codes for specific errors are defined within an error class.

3.2.27

error code

identification of a specific type of error within an error class

3.2.28

FAL subnet

networks composed of one or more data link segments

Note 1 to entry: They are permitted to contain bridges, but not routers FAL subnets are identified by a subset of the network address.

3.2.29

logical device

specifies a certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

3.2.30

management information

network-accessible information that supports managing the operation of the fieldbus system, including the application layer

Note 1 to entry: Managing includes functions such as controlling, monitoring, and diagnosing.

3.2.31

network

series of nodes connected by some type of communication medium

Note 1 to entry: The connection paths between any pair of nodes can include repeaters, routers and gateways.

3.2.32

peer

role of an AR endpoint in which it is capable of acting as both client and server

3.2.33

pre-defined AR endpoint

AR endpoint that is defined locally within a device without use of the create service

Note 1 to entry: Pre-defined ARs that are not pre-established are established before being used.

3.2.34

pre-established AR endpoint

AR endpoint that is placed in an established state during configuration of the AEs that control its endpoints

3.2.35

publisher

role of an AR endpoint in which it transmits APDUs onto the fieldbus for consumption by one or more subscribers

Note 1 to entry: The publisher may not be aware of the identity or the number of subscribers and it may publish its APDUs using a dedicated AR. Two types of publishers are defined by this standard, Pull Publishers and Push Publishers, each of which is defined separately.

3.2.36

server

a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request

b) object which provides services to another (client) object

3.2.37

service

operation or function than an object and/or object class performs upon request from another object and/or object class

Note 1 to entry: A set of common services is defined and provisions for the definition of object-specific services are provided. Object-specific services are those which are defined by a particular object class to perform a required function which is not performed by a common service.

3.2.38

subscriber

role of an AREP in which it receives APDUs produced by a publisher

Note 1 to entry: Two types of subscribers are defined by this standard, pull subscribers and push subscribers, each of which is defined separately.

3.3 Abbreviations and symbols

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
APO	Application Object
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier

AR	Application Relationship
AREP	Application Relationship End Point
ASCII	American Standard Code for Information Interchange
ASE	Application service Element
Cnf	Confirmation
DL-	(as a prefix) data-link-
DLC	Data-link Connection
DLCEP	Data-link Connection End Point
DLL	Data-link layer
DLM	Data-link-management
DLSAP	Data-link service Access Point
DLSDU	DL-service-data-unit
FAL	Fieldbus Application Layer
ID	Identifier
IEC	International Electrotechnical Commission
Ind	Indication
LME	Layer Management Entity
OSI	Open Systems Interconnect
QoS	Quality of service
Req	Request
Rsp	Response
SAP	Service Access Point
SDU	Service Data Unit
SMIB	System Management Information Base
SMK	System Management Kernel
VFD	Virtual Field Device

3.4 Conventions

3.4.1 General concept

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of three parts: its class definitions, its services, and its protocol specification. The first two are contained in IEC 61158-5 subseries. The protocol specification for each of the ASEs is defined in this standard.

The class definitions define the attributes of the classes supported by each ASE. The attributes are accessible from instances of the class using the Management ASE services specified in IEC 61158-5 standard. The service specification defines the services that are provided by the ASE.

This standard uses the descriptive conventions given in ISO/IEC 10731.

3.4.2 Conventions for class definitions

The data-link layer mapping definitions are described using templates. Each template consists of a list of attributes for the class. The general form of the template is defined in IEC 61158-5-5.

3.4.3 Abstract syntax conventions

When the "optionalParametersMap" parameter is used, a bit number which corresponds to each OPTIONAL or DEFAULT production is given as a comment.

3.5 Conventions used in state machines

The state machines are described in Table 1:

Table 1 – Conventions used for state machines

#	Current state	Event / condition => action	Next state
Name of this transition.	The current state to which this state transition applies.	Events or conditions that trigger this state transaction. => The actions that are taken when the above events or conditions are met. The actions are always indented below events or conditions.	The next state after the actions in this transition is taken.

The conventions used in the state machines are as follows:

:= Value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.

xxx A parameter name.

Example:
Identifier := reason
means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'

"xxx" Indicates fixed value.

Example:
Identifier := "abc"
means value "abc" is assigned to a parameter named 'Identifier.'

= A logical condition to indicate an item on the left is equal to an item on the right.

< A logical condition to indicate an item on the left is less than the item on the right.

> A logical condition to indicate an item on the left is greater than the item on the right.

<> A logical condition to indicate an item on the left is not equal to an item on the right.

&& Logical "AND"

|| Logical "OR"

This construct allows the execution of a sequence of actions in a loop within one transition. The loop is executed for all values from start_value to end_value.

Example:
for (Identifier := start_value to end_value)
actions
endfor

This construct allows the execution of alternative actions depending on some condition (which might be the value of some identifier or the outcome of a previous action) within one transition.

```
Example:
  If (condition)
    actions
  else
    actions
  endif
```

Readers are strongly recommended to refer to the subclauses for the AREP attribute definitions, the local functions, and the FAL-PDU definitions to understand protocol machines. It is assumed that readers have sufficient knowledge of these definitions, and they are used without further explanations.

4 Abstract syntax

4.1 FAL-AR PDU abstract syntax

4.1.1 Top level definition

The productions defined here shall be used with the rules for APDU encoding (see 5.1.2).

```
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

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

4.1.5 InvokeID

InvokeID ::= Unsigned8

4.1.6 Establish service

MaxOSCC ::= Unsigned8

MaxOSCS ::= Unsigned8

MaxUCSC ::= Unsigned8

MaxUCSS ::= Unsigned8

CIU ::= Unsigned32

```
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

4.1.7 ConType

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

4.1.8 Data send acknowledge service

```
DataSendAcknowledge-PDU ::= SEQUENCE {
    Protocol-Code, [APPLICATION 8], Address2ARP,
    Block-Number,
    Block-Length,
    Protocol-Data,
}
```

--- Protocol-Code in the higher nibble of the octet!!!

4.1.9 Protocol-code

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

--- TCP/IP protocol

4.1.10 Block-number

Block-Number ::= Unsigned8

--- 0 no blocking
---- 1...254 block number
---- 255 last block

4.1.11 Block-length

Block-Length ::= Unsigned

4.1.12 Protocol-data

Protocol-Data ::= Any

--- transparent protocol transfer

4.1.13 Address2 ARP

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

4.1.14 Destination-address

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

4.1.15 Source-address

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

4.1.16 Destination-node

Destination-Node ::= Unsigned8

4.1.17 Source-node

Source-Node ::= Unsigned8

4.1.18 Destination-subnode

Destination-Subnode ::= Unsigned8

4.1.19 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

Error type as specified in 4.1.4.6.

4.2.6 Error Fi type

```
ErrorFiType ::= SEQUENCE {
    errorClass      [0] IMPLICIT ErrorClass,
    additionalCode  [1] IMPLICIT Integer16 OPTIONAL,
    fiState         [3] IMPLICIT Integer8
}
```

4.2.7 Error class

Error Class as specified in 4.1.4.7.

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  
        ready-for-communication          (0),  
        limited-services-permitted       (2),  
    },  
    physicalStatus  
        operational                     (0),  
        partially-operational           (1),  
        inoperable                     (2),  
        needs-commissioning            (3),  
    },  
    localDetail  
}
```

[PRIVATE 0] IMPLICIT ENUMERATED {
 (0),
 (2),
}

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

[PRIVATE 2] IMPLICIT BitString OPTIONAL

4.2.10.2 Identify service

Identify-RequestPDU ::= NULL

```
Identify-ResponsePDU ::= SEQUENCE {  
    vendorName  
    modelIdentifier  
    vendorRevision  
}
```

[PRIVATE 0] IMPLICIT VisibleString,
[PRIVATE 1] IMPLICIT VisibleString,
[PRIVATE 2] IMPLICIT VisibleString

4.2.10.3 Initiate service

```
Initiate-RequestPDU ::= SEQUENCE {  
    versionObjectDefinitionsCalling  
    apDescriptorCalling  
    accessProtectionSupportedCalling  
    passwordAndAccessGroupsCalling  
    configuredMaxPduSizeSending  
    configuredMaxPduSizeReceiving  
    listOfSupportedServicesCalling  
}
```

[PRIVATE 0] IMPLICIT Integer16,
[PRIVATE 1] IMPLICIT OctetString,
[PRIVATE 2] IMPLICIT Ap_AccessProtectionSupported,
[PRIVATE 3] IMPLICIT Ap_AccessControl,
[PRIVATE 4] IMPLICIT Unsigned8,
[PRIVATE 5] IMPLICIT Unsigned8,
[PRIVATE 6] IMPLICIT Mn_PduSupportedMap

```
Initiate-ResponsePDU ::= SEQUENCE {  
    versionObjectDefinitionsCalled  
    apDescriptorCalled  
    accessPrivilegeSupportedCalled  
    passwordAndAccessGroupsCalled  
}
```

[PRIVATE 0] IMPLICIT Integer16,
[PRIVATE 1] IMPLICIT OctetString,
[PRIVATE 2] IMPLICIT Ap_AccessProtectionSupported,
[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),
}

4.3.1.2 Ap_AccessControl

```
Ap_AccessControl ::= BitString {
    password_Bit1      (8),
    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)
}
```

-- The Password (Unsigned8) is encoded as a bit string.

4.3.2 AR ASE types

4.3.2.1 Reason code and additional detail

4.3.2.1.1 Reason code

ReasonCode ::= Unsigned8

4.3.2.1.2 Additional detail

AdditionalDetaill ::= OctetString

4.3.2.2 AREP

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

4.3.2.3 Abort type identifier

```
Identifier ::= ENUMERATED {
    fal-user      (0),
    fal-APO-ASE   (1),
    fal-AR-ASE    (2),
    dll           (3)
}
```

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),
    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)
}
```

-- The Password (Unsigned8) is encoded as a bit string.

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 Vr_AccessPrivilege

```
Vr_AccessPrivilege ::= BitString {
    rightToReadPassword          (24),
    rightToWritePassword         (23),
    rightToDeletePassword        (22),
    rightToReadAccessGroup       (20),
    rightToWriteAccessGroup       (19),
    rightToDeleteAccessGroup      (18),
    rightToReadAllPartners        (32),
    rightToWriteAllPartners       (31),
    rightToDeleteAllPartners      (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)
}
```

4.3.6 General types

4.3.6.1 Gn_Deletable

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