

INTERNATIONAL STANDARD

IEC
61158-6

Second edition
2000-01

**Digital data communications for
measurement and control –
Fieldbus for use in industrial control systems –**

**Part 6:
Application Layer protocol specification**



Reference number
IEC 61158-6:2000(E)

Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

Consolidated publications

Consolidated versions of some IEC publications including amendments are available. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available in the IEC catalogue.

Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
Published yearly with regular updates
(On-line catalogue)*
- **IEC Bulletin**
Available both at the IEC web site* and as a printed periodical

Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV).

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment*. *Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams*.

* See web site address on title page.

INTERNATIONAL STANDARD

IEC
61158-6

Second edition
2000-01

Digital data communications for measurement and control – Fieldbus for use in industrial control systems –

Part 6: Application Layer protocol specification

© IEC 2000 – Copyright - all rights reserved

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

International Electrotechnical Commission
Telefax: +41 22 919 0300

3, rue de Varembé Geneva, Switzerland
e-mail: inmail@iec.ch IEC web site <http://www.iec.ch>



Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE **XH**

For price, see current catalogue

CONTENTS

INTRODUCTION	22
1 Scope.....	23
2 Normative references	24
3 Definitions	25
3.1 Definitions from other ISO/IEC Standards	25
3.1.1 Definitions from ISO/IEC 7498-1	25
3.1.2 Definitions from ISO/IEC 8822	25
3.1.3 Definitions from ISO/IEC 9545	25
3.1.4 Definitions from ISO/IEC 8824	25
3.1.5 Definitions from ISO/IEC 8825	25
3.2 Definitions from IEC 61158-5	25
3.3 Other definitions	26
3.3.1 Type 1 Definitions	26
3.3.1.1 called.....	26
3.3.1.2 calling.....	26
3.3.1.3 interoperability	26
3.3.1.4 management information.....	26
3.3.1.5 receiving.....	26
3.3.1.6 resource	26
3.3.1.7 sending.....	26
3.3.2 Type 2 Definitions	26
3.3.2.1 Allocate	26
3.3.2.2 Application.....	26
3.3.2.3 Application objects.....	26
3.3.2.4 Attribute.....	26
3.3.2.5 Behaviour	27
3.3.2.6 Class.....	27
3.3.2.7 Class attributes.....	27
3.3.2.8 Class code.....	27
3.3.2.9 Class specific service	27
3.3.2.10 Client.....	27
3.3.2.11 Connection	27
3.3.2.12 Connection ID (CID).....	27
3.3.2.13 Connection path.....	27
3.3.2.14 Connection point.....	27
3.3.2.15 Consume	27
3.3.2.16 Consumer.....	27
3.3.2.17 Consuming application.....	27
3.3.2.18 Cyclic	27
3.3.2.19 Device	27
3.3.2.20 Device profile.....	28
3.3.2.21 End node	28
3.3.2.22 End point	28
3.3.2.23 Error.....	28

3.3.2.24	Frame.....	28
3.3.2.25	Instance.....	28
3.3.2.26	Instance attributes	28
3.3.2.27	Instantiated	28
3.3.2.28	Keeper.....	28
3.3.2.29	Little endian	28
3.3.2.30	Lpacket.....	28
3.3.2.31	Member	28
3.3.2.32	Message router	28
3.3.2.33	Multicast connection	28
3.3.2.34	Network	28
3.3.2.35	Object.....	28
3.3.2.36	Object specific service	29
3.3.2.37	Originator	29
3.3.2.38	Point-to-point connection	29
3.3.2.39	Produce.....	29
3.3.2.40	Producer.....	29
3.3.2.41	Server	29
3.3.2.42	Service	29
3.3.2.43	Target.....	29
3.3.2.44	Temporary node.....	29
3.3.2.45	Transaction id	29
3.3.2.46	Unconnected message manager (UCMM)	29
3.3.2.47	Unconnected service.....	29
3.3.3	Type 3 Definitions	29
3.3.3.1	Access Protection	29
3.3.3.2	Address-assignment-table.....	29
3.3.3.3	Channel.....	29
3.3.3.4	Channel related Diagnosis	30
3.3.3.5	Configuration check	30
3.3.3.6	Configuration fault	30
3.3.3.7	Configuration Identifier.....	30
3.3.3.8	Configuration Identifier related Diagnosis	30
3.3.3.9	Control Commands	30
3.3.3.10	Data consistency	30
3.3.3.11	Default address	30
3.3.3.12	Device related diagnosis	30
3.3.3.13	Diagnosis information	30
3.3.3.14	Diagnosis information collection	30
3.3.3.15	DP-Master (Class 1)	30
3.3.3.16	DP-Master (Class 2)	30
3.3.3.17	DP-Slave	30
3.3.3.18	Freeze	30
3.3.3.19	Group	30
3.3.3.20	I/O data	31
3.3.3.21	Ident Number	31
3.3.3.22	Index	31
3.3.3.23	Master parameter set.....	31

3.3.3.24	Module	31
3.3.3.25	Process data	31
3.3.3.26	Real Configuration	31
3.3.3.27	Slot	31
3.3.3.28	Sync	31
3.3.4	Type 4 Definitions	31
3.3.5	Type 5 Definitions	31
3.3.6	Type 6 Definitions	31
3.3.7	Type 7 Definitions	31
3.3.8	Type 8 Definitions	31
3.4	Abbreviations and symbols	31
3.4.1	Type 1 Abbreviations and symbols	31
3.4.2	Type 2 Abbreviations and symbols	32
3.4.3	Type 3 Abbreviations and symbols	32
3.4.4	Type 4 Abbreviations and symbols	33
3.4.5	Type 5 Abbreviations and symbols	33
3.4.6	Type 6 Abbreviations and symbols	33
3.4.7	Type 7 Abbreviations and symbols	33
3.4.8	Type 8 Abbreviations and symbols	34
3.5	Conventions	34
3.5.1	Conventions for Type 1	34
3.5.1.1	Conventions for Class Definitions	34
3.5.1.2	Abstract Syntax Conventions	34
3.5.2	Conventions for Type 2	34
3.5.2.1	Attribute specification	34
3.5.2.2	Common Services	35
3.5.3	Conventions for Type 3	38
3.5.3.1	Abstract Syntax Conventions	38
3.5.3.2	Convention for the Encoding of Reserved Bits and Octets	38
3.5.3.3	Conventions for the Common Coding Structure of specific Field Octets	38
3.5.4	Conventions for Type 4	39
3.5.5	Conventions for Type 5	39
3.5.6	Conventions for Type 6	39
3.5.7	Conventions for Type 7	39
3.5.8	Conventions for Type 8	39
3.6	Conventions used in State Machines	39
3.6.1	Conventions for Type 1	39
3.6.2	Conventions for Type 2	40
3.6.2.1	State Machine Conventions	40
3.6.3	Conventions for Type 3	42
3.6.4	Conventions for Type 4	44
3.6.5	Conventions for Type 5	44
3.6.6	Conventions for Type 6	44
3.6.7	Conventions for Type 7	44
3.6.7.1	Evaluation networks	45
3.6.7.2	Simple actions	45
3.6.7.3	Conditions	45
3.6.8	Conventions for Type 8	45

4	Type 1	46
4.1	FAL Syntax Description	46
4.1.1	FAL-AR PDU Abstract Syntax 1	46
4.1.1.1	Confirmed Send Service	46
4.1.1.2	Unconfirmed Send Service	46
4.1.1.3	Unconfirmed Acknowledged Send Service	46
4.1.1.4	Idle Send Service	47
4.1.1.5	AR-XON-OFF Send Service	47
4.1.1.6	Establish Service	47
4.1.2	FAL-AR PDU Abstract Syntax 2	47
4.1.2.1	Confirmed Send Service	47
4.1.2.2	Unconfirmed Send Service	48
4.1.2.3	Unconfirmed Acknowledged Send Service	48
4.1.2.4	Idle Send Service	48
4.1.2.5	AR-XON-OFF Send Service	48
4.1.2.6	Establish Service	48
4.1.2.7	MaxOSCC	49
4.1.2.8	MaxOSCS	49
4.1.2.9	MaxUCSC	49
4.1.2.10	MaxUCSS	49
4.1.2.11	XON_OFF	49
4.1.2.12	CIU	49
4.1.3	Abstract Syntax of PDUBody	49
4.1.3.1	Abort Service	49
4.1.3.2	InvokeID	49
4.1.3.3	ConfirmedServiceRequest	50
4.1.3.4	ConfirmedServiceResponse	51
4.1.3.5	ConfirmedServiceError	52
4.1.3.6	Error Type	52
4.1.3.7	Error Class	53
4.1.3.8	Unconfirmed PDUs	54
4.1.3.9	Management ASE	54
4.1.3.10	Application Process ASE	56
4.1.3.11	Load Region ASE	57
4.1.3.12	Function Invocation ASE	59
4.1.3.13	Variable Access ASE	60
4.1.3.14	Event Management ASE	63
4.1.3.15	Type Definitions	66
4.1.4	Data Types	82
4.1.4.1	Notation for the Boolean Type	82
4.1.4.2	Notation for the Integer Type	82
4.1.4.3	Notation for the Unsigned Type	82
4.1.4.4	Notation for the Floating Point Type	82
4.1.4.5	Notation for the BitString Type	83
4.1.4.6	Notation for the OctetString Type	83
4.1.4.7	Notation for VisibleString Type	83
4.1.4.8	Notation for the UNICODEString Type	83
4.1.4.9	Notation for the FieldbusTime Type	83

4.1.4.10	Notation for the Universal Time Type	83
4.1.4.11	Notation for Binary Time Type	83
4.1.4.12	Notation for BCD Type	83
4.1.4.13	Notation for Compact Boolean Array Type	83
4.1.4.14	Notation for Compact BCD Array Type	83
4.1.4.15	Notation for BinaryDate Type	83
4.1.4.16	Notation for TimeOfDay Type	83
4.1.4.17	Notation for TimeDifference Type	83
4.1.4.18	Notation for TimeValue Type	83
4.1.5	Reason Codes	84
4.1.5.1	Introduction	84
4.2	Transfer Syntaxes	85
4.2.1	Transfer Syntax 1	85
4.2.1.1	FAL Header	85
4.2.1.2	Traditional Encoding Rule (TER)	86
4.2.1.3	Compact Encoding Rule (CER)	94
4.3	FAL Protocol State Machines Structure	109
4.4	AP Context State Machine	110
4.4.1	Primitive Definitions	110
4.4.1.1	Primitives Exchanged between FAL-User and AP-Context	110
4.4.2	State Machine Description	110
4.4.3	AP-AP Context Initiation State Transitions	111
4.4.4	Functions	122
4.5	FAL Service Protocol Machine (FSPM)	125
4.5.1	Primitive Definitions	125
4.5.1.1	Primitives Exchanged between AP_Context and FSPM	125
4.5.1.2	Parameters of AP_Context/FSPM Primitives	126
4.5.2	FSPM State Tables	127
4.5.2.1	Functions	130
4.6	Application Relationship Protocol Machines (ARPMs)	131
4.6.1	Queued Usertriggered Unidirectional (QUU) ARPM	131
4.6.1.1	Primitive Definitions	131
4.6.1.2	Parameters of FSPM/ARPM Primitives	131
4.6.1.3	QUU ARPM State Machine	134
4.6.2	Queued Usertriggered Bidirectional-Connection Oriented (QUB-CO) ARPM	137
4.6.2.1	Primitive Definitions	137
4.6.2.2	DLL Mapping of QUB AREP Class	137
4.6.2.3	QUB AREP State Machine	141
4.6.3	Queued Usertriggered Bidirectional-Connectionless (QUB-CI) ARPM	150
4.6.3.1	Primitive Definitions	150
4.6.3.2	DLL Mapping of QUB-CL AREP Class	150
4.6.3.3	QUB-CL ARPM State Machine	152
4.6.4	Queued Usertriggered Bidirectional-Segmentation (QUB-Seg) ARPM	157
4.6.4.1	Primitive Definitions	157
4.6.4.2	Parameters of FSPM/ARPM Primitives	157
4.6.4.3	QUB-Seg ARPM State Machine	161
4.6.5	Queued Usertriggered Bidirectional-Flow Control (QUB-FC) ARPM	175
4.6.5.1	QUB-FC Primitive Definitions	175

4.6.5.2	Parameters of FSPM/ARPM Primitives	176
4.6.5.3	QUB-FC ARPM State Machine	179
4.6.6	Buffered Usertriggered Bidirectional (BUB) ARPM	201
4.6.6.1	Primitive Definitions	201
4.6.6.2	DLL Mapping of BUB AREP Class	202
4.6.6.3	BUB ARPM State Machine	205
4.6.7	Buffered Networkscheduled Bidirectional (BNB) ARPM	216
4.6.7.1	Primitive Definitions	216
4.6.7.2	Primitives issued by ARPM to FSPMParameters of FSPM/ARPM Primitives	216
4.6.7.3	DLL Mapping of BNB AREP Class	217
4.6.7.4	BNB ARPM state machine	220
4.6.8	Buffered Networkscheduled Unidirectional (BNU) ARPM	240
4.6.8.1	Primitive Definitions	240
4.6.8.2	DLL Mapping of BNU AREP Class	241
4.6.8.3	BNU AREP State Machine	244
4.6.9	Buffered Networkscheduled Unidirectional-MP (BNU-MP) ARPM	251
4.6.9.1	Primitive Definitions	251
4.6.9.2	DLL Mapping of BNU-MP AREP Class	252
4.6.9.3	BNU-MP ARPM State Machine	255
4.7	DLL Mapping Protocol Machine (DMPM)	263
4.7.1	Primitive Definitions	263
4.7.1.1	Primitives Exchanged between DMPM and ARPM	263
4.7.1.2	Parameters of ARPM/DMPM Primitives	265
4.7.1.3	Primitives Exchanged between Data Link Layer and DMPM	266
4.7.1.4	Parameters of DMPM/Data Link Layer Primitives	267
4.7.2	DMPM State Machine	268
4.7.2.1	DMPM States	268
4.7.2.2	DMPM State Table	268
4.7.2.3	Functions used by DMPM	274
4.7.3	Data Link Layer Service Selection	276
4.7.3.1	Introduction	276
4.8	Protocol Options	276
4.8.1	Protocol Option 1	276
4.8.1.1	Introduction	276
4.8.1.2	Abstract Syntax Selection	277
4.8.1.3	Protocol Machine Overview	277
4.8.1.4	Application Relationship Protocol Machine (ARPM) Selection	277
4.8.1.5	Encoding Rule Selection	277
4.8.2	Protocol Option 2	277
4.8.2.1	Introduction	277
4.8.2.2	Abstract Syntax	278
4.8.2.3	Protocol Machine Overview	278
4.8.3	Protocol Option 3	279
4.8.3.1	Introduction	279
4.8.3.2	Abstract Syntax	279
4.8.3.3	Protocol Machine Overview	279
5	Type 2	280
5.1	Type 2 Abstract Syntax	280

5.1.1	FAL PDU Abstract Syntax	280
5.1.1.1	PDU Structure	280
5.1.1.2	UCMM_PDUs	281
5.1.1.3	Transport_Headers	282
5.1.1.4	CM_PDUs	284
5.1.1.5	CM PDU Components	290
5.1.1.6	MR Headers	297
5.1.1.7	OM_Service_PDU	297
5.1.1.8	Message and Connection Paths	306
5.1.1.9	Class, Attribute And Service Codes	312
5.1.1.10	Error Codes	316
5.1.2	Data Abstract Syntax Specification	325
5.1.2.1	Transport Format Specification	325
5.1.2.2	Abstract Syntax Notation	326
5.1.2.3	Control Network Data Specification	326
5.1.2.4	Data Type Specification / Dictionaries	327
5.2	Type 2 Transfer Syntax	330
5.2.1	Compact Encoding	330
5.2.1.1	Encoding Rules	330
5.2.1.2	Encoding Constraints	330
5.2.1.3	Examples (Informative)	330
5.2.2	Data Type Reporting	336
5.2.2.1	Object Data Representation	336
5.2.2.2	Elementary Data Type Reporting	337
5.2.2.3	Constructed Data Type Reporting	337
5.3	Structure of Type 2 FAL Protocol State Machines	341
5.4	Type 2 AP Context State Machine	341
5.5	FAL Service Protocol Machine (FSPM)	341
5.5.1	Primitive Definitions	341
5.5.2	Parameters of Primitives	344
5.5.3	FSPM State Machines	345
5.6	Application Relationship Protocol Machines (ARPMs)	345
5.6.1	Connection-less ARPM (UCMM)	345
5.6.1.1	Primitive Definitions	346
5.6.1.2	Parameters of Primitives	347
5.6.1.3	UCMM State Machines	347
5.6.1.4	Examples of UCMM Sequences (Informative)	353
5.6.1.5	Management UCMM	354
5.6.2	Connection-oriented ARPMs (Transports)	355
5.6.2.1	Transport PDU Buffer	355
5.6.2.2	Transport Classes	355
5.6.2.3	Common Primitive Definitions	356
5.6.2.4	Parameters of Common Primitives	357
5.6.2.5	Transport State Machines – Class 0	357
5.6.2.6	Transport State Machines – Class 1	360
5.6.2.7	Transport State Machines – Class 2	367
5.6.2.8	Transport State Machines – Class 3	374
5.6.2.9	Transport State Machines – Classes 4, 5, 6	382

5.6.2.10	Transport State Machines – Class 4	391
5.6.2.11	Transport State Machines – Class 5	397
5.6.2.12	Transport State Machines – Class 6	410
5.7	DLL Mapping Protocol Machine (DMPM)	429
5.7.1	Introduction	429
5.7.1.1	Link Producer	429
5.7.1.2	Link Consumer	429
5.7.2	Primitive Definitions	429
5.7.2.1	Primitives Exchanged between DMPM and ARPM	429
5.7.2.2	Parameters of ARPM/DMPM Primitives	430
5.7.2.3	Primitives Exchanged between Data Link Layer and DMPM	430
5.7.2.4	Parameters of DMPM/Data Link Layer Primitives	430
5.7.3	DMPM State Machine	431
5.7.3.1	DMPM States	431
5.7.3.2	Functions used by DMPM	432
5.7.4	Data Link Layer Service Selection	432
5.8	Protocol Options	432
6	Type 3	433
6.1	FAL Syntax Description	433
6.1.1	APDU Abstract Syntax	433
6.1.2	Data Types	435
6.2	Transfer Syntax	435
6.2.1	Coding of Basic Data Types	435
6.2.2	Coding Section related to Data Exchange PDUs	436
6.2.2.1	Coding of the Field Outp_Data	436
6.2.2.2	Coding of the Field Inp_Data	436
6.2.3	Coding Section related to Slave Diagnosis PDUs	436
6.2.3.1	Coding of the Field Station_status_1	436
6.2.3.2	Coding of the Field Station_status_2	437
6.2.3.3	Coding of the Field Station_status_3	437
6.2.3.4	Coding of the Field Diag_Master_Add	437
6.2.3.5	Coding of the Field Ident_Number	437
6.2.3.6	Coding of the Field Header_Byte	437
6.2.3.7	Coding of the Field Alarm_Type	438
6.2.3.8	Coding of the Field Status_Type	438
6.2.3.9	Coding of the Field Slot_Number	439
6.2.3.10	Coding of the Field Alarm_Specifier	439
6.2.3.11	Coding of the Field Status_Specifier	439
6.2.3.12	Coding of the Field Diagnosis_User_Data	439
6.2.3.13	Coding of the Field Modul_Status_Array	439
6.2.3.14	Coding of the Field Identifier_Diagnosis_Data_Array	441
6.2.3.15	Coding of the Field Identifier_Number	442
6.2.3.16	Coding of the Field Channel_Number	442
6.2.3.17	Coding of the Field Type_of_Diagnosis	442
6.2.3.18	Coding of the Field Revision_Number	443
6.2.4	Coding Section related to Parameterisation PDU	443
6.2.4.1	Coding of the Field Station_status	443
6.2.4.2	Coding of the Field WD_Fact_1	444

6.2.4.3	Coding of the Field WD_Fact_2.....	444
6.2.4.4	Coding of the Field min_TSDR.....	444
6.2.4.5	Coding of the Field Group_Ident.....	444
6.2.4.6	Coding of the Field User_Prm_Data.....	444
6.2.4.7	Coding of the Field DPV1_Status_1.....	445
6.2.4.8	Coding of the Field DPV1_Status_2.....	445
6.2.4.9	Coding of the Field DPV1_Status_3.....	445
6.2.5	Coding Section related to Configuration PDUs.....	446
6.2.5.1	Coding of the Field Cfg_Identifier.....	446
6.2.5.2	Coding of the Field Special_Cfg_Identifier.....	446
6.2.5.3	Coding of the Fields Length_Byte.....	447
6.2.5.4	Coding of the Field Manufacturer_Specific_Data.....	447
6.2.5.5	Coding of the Field Extended_Length_Byte.....	447
6.2.5.6	Coding of the Field Data_Type.....	448
6.2.6	Coding Section related to Global Control PDUs.....	449
6.2.6.1	Coding of the Field Control_Command.....	449
6.2.6.2	Coding of the Field Group_Select.....	450
6.2.7	Coding Section related to Function Identification and Errors.....	450
6.2.7.1	Coding of the Field Function_Num.....	450
6.2.7.2	Coding of the Field Error_Decode.....	452
6.2.7.3	Coding of the Field Error_Code_1.....	453
6.2.7.4	Coding of the Field Error_Code_2.....	453
6.2.8	Coding Section related to Master Diagnosis PDU.....	454
6.2.8.1	Coding of the Field MDiag_Identifier.....	454
6.2.8.2	Coding of the Field System_Diagnosis.....	454
6.2.8.3	Coding of the Field USIF_State.....	454
6.2.8.4	Coding of the Field Hardware_Release_DP.....	454
6.2.8.5	Coding of the Field Firmware_Release_DP.....	455
6.2.8.6	Coding of the Field Hardware_Release_User.....	455
6.2.8.7	Coding of the Field Firmware_Release_User.....	455
6.2.8.8	Coding of the Field Data_Transfer_List.....	455
6.2.9	Coding Section related to Upload/Download/Act Para PDUs.....	455
6.2.9.1	Coding of the Field Area_Code_UpDownload.....	455
6.2.9.2	Coding of the Field Timeout.....	456
6.2.9.3	Coding of the Field Max_Len_Data_Unit.....	456
6.2.9.4	Coding of the Field Add_Offset.....	456
6.2.9.5	Coding of the Field Data.....	456
6.2.9.6	Coding of the Field Data_Len.....	456
6.2.9.7	Coding of the Field Area_CodeActBrct.....	456
6.2.9.8	Coding of the Field Area_CodeAct.....	456
6.2.9.9	Coding of the Field Activate.....	457
6.2.10	Coding Section related to the Bus Parameter Set.....	457
6.2.10.1	Coding of the Field Bus_Para_Len.....	457
6.2.10.2	Coding of the Field DL_Add.....	457
6.2.10.3	Coding of the Field Baud_rate.....	457
6.2.10.4	Coding of the Fields T _{SL} , min T _{SDR} , max T _{SDR}	458
6.2.10.5	Coding of the Fields T _{QUI} , T _{SET} , G, HSA, max_retry_limit.....	458

6.2.10.6	Coding of the Field T_{TR} (Target Token Rotation Time).....	458
6.2.10.7	Coding of the Field Bp_Flag (Busparameter Flag)	458
6.2.10.8	Coding of the Field Min_Slave_Interval.....	458
6.2.10.9	Coding of the Field Poll_Timeout.....	458
6.2.10.10	Coding of the Field Data_Control_Time	458
6.2.10.11	Coding of the Field Alarm_Max	458
6.2.10.12	Coding of the Field Max_User_Global_Control	458
6.2.10.13	Coding of the Field Master_User_Data_Len	459
6.2.10.14	Coding of the Field Master_Class2_Name	459
6.2.10.15	Coding of the Field Master_User_Data	459
6.2.11	Coding Section related to the Slave Parameter Set.....	459
6.2.11.1	Coding of the Field Slave_Para_Len.....	459
6.2.11.2	Coding of the Field SI_Flag (Slave Flag).....	459
6.2.11.3	Coding of the Field Slave_Type.....	460
6.2.11.4	Coding of the Field Max_Diag_Data_Len.....	460
6.2.11.5	Coding of the Field Max_Alarm_Len.....	460
6.2.11.6	Coding of the Field Max_Channel_Data_Length	460
6.2.11.7	Coding of the Field Diag_Upd_Delay	460
6.2.11.8	Coding of the Field Alarm_Mode	460
6.2.11.9	Coding of the Field Add_SI_Flag	460
6.2.11.10	Coding of the Field Prm_Data_Len.....	461
6.2.11.11	Coding of the Field Prm_Data	461
6.2.11.12	Coding of the Field Cfg_Data_Len.....	461
6.2.11.13	Coding of the Field Cfg_Data	461
6.2.11.14	Coding of the Field Add_Tab_Len.....	461
6.2.11.15	Coding of the Field Number_of_Entries	461
6.2.11.16	Coding of the Field Add_Tab_Entry_Header	461
6.2.11.17	Coding of the Field I/O_Data_Length.....	461
6.2.11.18	Coding of the Field I/O_Config_Address.....	461
6.2.11.19	Coding of the Field Host_Address	461
6.2.11.20	Coding of the Field Slave_User_Data_Len	462
6.2.11.21	Coding of the Field Slave_User_Data	462
6.2.12	Coding Section related to Statistic Counters	462
6.2.12.1	Coding of the Field Frame_sent_count and SD_count.....	462
6.2.12.2	Coding of the Field Error_count and SD_error_count.....	462
6.2.13	Coding Section related to Set Slave Address PDU.....	462
6.2.13.1	Coding of the Field New_Slave_Add.....	462
6.2.13.2	Coding of the Field No_Add_Change.....	462
6.2.13.3	Coding of the Field Rem_Slave_Data	462
6.2.14	Coding Section related to Initiate/Abort PDUs.....	462
6.2.14.1	Coding of the Field Features_Supported_1.....	462
6.2.14.2	Coding of the Field Features_Supported_2.....	462
6.2.14.3	Coding of the Field Profile_Features_Supported_1	462
6.2.14.4	Coding of the Field Profile_Features_Supported_2	463
6.2.14.5	Coding of the Field Profile_Ident_Number	463
6.2.14.6	Coding of the Field S_Type (Source Type)	463
6.2.14.7	Coding of the Field D_Type (Destination Type).....	463
6.2.14.8	Coding of the Field S_Len (Source Length)	463

6.2.14.9	Coding of the Field D_Len (Destination Length).....	463
6.2.14.10	Coding of the Field S_API (Source Application Identifier).....	463
6.2.14.11	Coding of the Field D_API (Destination Application Identifier).....	463
6.2.14.12	Coding of the Field S_SCL (Source Security Level).....	463
6.2.14.13	Coding of the Field D_SCL (Destination Security Level).....	463
6.2.14.14	Coding of the Field S_Network_Address.....	463
6.2.14.15	Coding of the Field D_Network_Address.....	463
6.2.14.16	Coding of the Field S_MAC_Address.....	464
6.2.14.17	Coding of the Field D_MAC_Address.....	464
6.2.14.18	Coding of the Field Send_Timeout.....	464
6.2.14.19	Coding of the Field Server_SAP.....	464
6.2.14.20	Coding of the Field Subnet.....	464
6.2.14.21	Coding of the Field Instance_Reason_Code.....	464
6.2.15	Coding Section related to Read/Write/Data Transport PDUs.....	465
6.2.15.1	Coding of the Field Index.....	465
6.2.15.2	Coding of the Field Length.....	465
6.2.16	Example of Diagnosis-RES-PDU.....	466
6.2.17	Example of Chk_Cfg-REQ-PDU.....	467
6.2.18	Example of Chk_Cfg-REQ-PDU with Extension.....	467
6.2.19	Example Structure of the Data_Unit for Data_Exchange.....	468
6.3	FAL Protocol State Machines.....	469
6.3.1	Overall Structure.....	469
6.3.1.1	Fieldbus Service Protocol Machines (FSPM).....	469
6.3.1.2	Master to Slave Cyclic (MS0).....	469
6.3.1.3	Master (Class 1) to Slave Acyclic (MS1).....	469
6.3.1.4	Master (Class 2) to Slave Acyclic (MS2).....	469
6.3.1.5	Master Master Acyclic (MM1/MM2).....	469
6.3.1.6	DLL Mapping Protocol Machines (DMPM).....	469
6.3.2	Assignment of State Machines to Devices.....	470
6.3.3	Overview DP-Slave.....	471
6.3.4	Overview DP-Master (Class 1).....	472
6.3.5	Overview DP-Master (Class 2).....	473
6.3.6	Cyclic Communication between DP-Master (Class 1) and DP-Slave.....	473
6.3.7	Acyclic Communication between DP-Master (Class 2) and DP-Master (Class 1).....	475
6.3.8	Application Relationship Monitoring.....	477
6.3.8.1	Monitoring of the MS0 - AR.....	477
6.3.8.2	Monitoring of the MS2 - AR.....	478
6.4	AP Context State Machine.....	480
6.5	FAL Service Protocol Machines (FSPMs).....	480
6.5.1	FSPMS.....	480
6.5.1.1	Primitive Definitions.....	480
6.5.1.2	State Machine Description.....	483
6.5.1.3	FSPMS State Table.....	486
6.5.1.4	Functions.....	495
6.5.2	FSPMM1.....	496
6.5.2.1	Primitive Definitions.....	496
6.5.2.2	State Machine Description.....	499
6.5.2.3	FSPMM1 State Table.....	501

6.5.2.4	Functions	514
6.5.3	FSPMM2	514
6.5.3.1	Primitive Definitions	514
6.5.3.2	State Machine Description	516
6.5.3.3	FSPMM2 State Table	517
6.5.3.4	Functions	521
6.6	Application Relationship Protocol Machines (ARPMs)	521
6.6.1	MSCY1S	521
6.6.1.1	Primitive Definitions	521
6.6.1.2	State Machine Description	522
6.6.1.3	MSCY1S State Table	526
6.6.1.4	Functions	539
6.6.2	MSAC1S	541
6.6.2.1	Primitive Definitions	541
6.6.2.2	State Machine Description	542
6.6.2.3	MSAC1S State Table	544
6.6.2.4	Functions	551
6.6.3	MSRM2S	551
6.6.3.1	Primitive Definitions	551
6.6.3.2	State Machine Description	551
6.6.3.3	MSRM2S State Table	554
6.6.4	MSAC2S	556
6.6.4.1	Primitive Definitions	556
6.6.4.2	State Machine Description	557
6.6.4.3	MSAC2S State Table	560
6.6.5	MSCY1M	569
6.6.5.1	Primitive Definitions	569
6.6.5.2	State Machine Description	570
6.6.5.3	MSCY1M State Table	573
6.6.6	MSAL1M	583
6.6.6.1	Primitive Definitions	583
6.6.6.2	State Machine Description	584
6.6.6.3	MSAL1M State Table	587
6.6.7	MSAC1M	591
6.6.7.1	Primitive Definitions	591
6.6.7.2	State Machine Description	592
6.6.7.3	MSAC1M State Table	597
6.6.8	MMAC1	601
6.6.8.1	Primitive Definitions	601
6.6.8.2	State Machine Description	601
6.6.8.3	MMAC1 State Table	602
6.6.9	MSAC2M	606
6.6.9.1	Primitive Definitions	606
6.6.9.2	State Machine Description	608
6.6.9.3	MSAC2M State Table	611
6.6.10	MMAC2	619
6.6.10.1	Primitive Definitions	619
6.6.10.2	State Machine Description	620

6.6.10.3	MMAC2 State Table	621
6.7	DLL Mapping Protocol Machines (DMPMs)	625
6.7.1	DMPMS	625
6.7.1.1	Primitive Definitions	625
6.7.1.2	State Machine Description	628
6.7.1.3	DMPMS State Table	629
6.7.1.4	Functions	632
6.7.2	DMPMM1	633
6.7.2.1	Primitive Definitions	633
6.7.2.2	State Machine Description	637
6.7.2.3	DMPMM1 State Table	638
6.7.2.4	Functions	642
6.7.3	DMPMM2	642
6.7.3.1	Primitive Definitions	642
6.7.3.2	State Machine Description	645
6.7.3.3	DMPMM2 State Table	646
6.7.3.4	Functions	648
6.8	Protocol Options	649
6.9	Parameters for a DP-Slave	649
7	Type 4	650
7.1	FAL Syntax Description	650
7.1.1	FAL-AR PDU Abstract Syntax	650
7.1.1.1	Abstract Syntax of APDU Header	650
7.1.1.2	Abstract Syntax of APDU Body	650
7.1.2	Data Types	651
7.2	Transfer Syntaxes	652
7.2.1	APDU Encoding	652
7.2.1.1	APDU Header Encoding	652
7.2.1.2	APDU Body Encoding	654
7.2.2	Variable Object Encoding and Packing	656
7.2.2.1	Encoding of Simple Variables	656
7.2.2.2	Encoding of Constructed Variables	656
7.2.2.3	Alignment	656
7.2.2.4	Variable Object Attributes	657
7.2.3	Error Codes	658
7.3	FAL Protocol State Machines	659
7.4	AP Context State Machine	660
7.5	FAL Service Protocol Machine (FSPM)	661
7.5.1	Primitives Exchanged between FAL User and FSPM	661
7.5.2	FSPM States	661
7.5.2.1	FSPM Proxy Object States	661
7.5.2.2	FSPM Real Object State Machine Description	665
7.6	Application Relationship Protocol Machine (ARPM)	666
7.6.1	Primitives Exchanged between ARPM and FSPM	666
7.6.2	ARPM States	667
7.6.2.1	Overview	667
7.6.2.2	Sender State Transitions	667
7.6.2.3	Receiver State Transitions	668

7.7	DLL Mapping Protocol Machine (DMPM)	669
7.7.1	Data Link Layer Service Selection	669
7.7.1.1	Introduction	669
7.7.2	Primitives Exchanged between ARPM and DLPM	669
7.7.3	Primitives Exchanged between DLPM and Data Link Layer	669
7.7.4	DLPM States	670
7.7.4.1	Overview	670
7.7.4.2	Sender State Transitions	670
7.7.4.3	Receiver State Transitions	671
7.8	Protocol Options	672
8	Type 5	673
8.1	FAL Syntax Description	673
8.1.1	PDU Abstract Syntax	673
8.1.2	Abstract Syntax of Data Types	673
8.2	Transfer Syntax	673
8.2.1	Data Type Encodings	673
8.2.2	Type 5 Header	674
8.2.2.1	Conveyed Reference Use	675
8.2.3	Type 5 Trailer	679
8.2.4	APDU Body	679
8.2.4.1	Type 5 AR ASE Services	680
8.2.4.2	SMK ASE Services	680
8.2.4.3	Type 1 ASE Services	684
8.2.4.4	LAN Redundancy Services	698
8.2.4.5	List of Type 1 Managed AP Attributes	698
8.2.5	Error APDU Bodies	703
8.2.5.1	Common Error Parameters	703
8.2.5.2	FI Error Parameters	703
8.2.5.3	OD Error Parameters	704
8.3	FAL Protocol State Machine Structure	704
8.4	AP Context State Machine	704
8.5	FAL Service Protocol Machine (FSPM)	704
8.6	Application Relationship Protocol Machines (ARPMs)	704
8.6.1	Client / Server Session ARPM	704
8.6.1.1	Primitive Definitions	704
8.6.1.2	DLL Mapping of Client / Server AREP Class	705
8.6.1.3	Client / Server AREP State Machine	706
8.6.2	Publisher / Subscriber Session ARPM	717
8.6.2.1	Primitive Definitions	717
8.6.2.2	DLL Mapping of Publisher / Subscriber AREP Class	717
8.6.2.3	Publisher / Subscriber ARPM State Machine	718
8.7	DLL Mapping Protocol Machine (DMPM)	723
8.7.1	Primitive Definitions	723
8.7.1.1	Primitives Exchanged between DMPM and ARPM	723
8.7.1.2	Parameters of ARPM/DMPM Primitives	723
8.7.1.3	Primitives Exchanged between the Socket model and DMPM	723
8.7.1.4	Parameters of DMPM/Socket model Primitives	724
8.7.2	DMPM State Machine	724

8.7.2.1	DMPM States.....	724
8.7.2.2	DMPM State Table.....	724
8.7.2.3	Functions used by DMPM.....	725
8.8	Protocol Options	726
9	Type 6.....	727
10	Type 7.....	728
10.1	Abstract syntax of data type for Type 7	728
10.1.1	Boolean	728
10.1.2	Integer.....	728
10.1.3	Bit string.....	728
10.1.4	Octet string.....	729
10.1.5	Sequence (sequence or sequence of)	729
10.1.6	Choice	730
10.1.7	Null.....	730
10.1.8	Object identifier	730
10.1.9	Default.....	731
10.1.10	Example of encoding (informative)	731
10.2	Transfer Syntaxes.....	732
10.2.1	MPS ASE FAL PDU data types.....	733
10.2.1.1	Syntax notation	733
10.2.1.2	Simple predefined types.....	734
10.2.1.3	PDU Types	734
10.2.2	MCS AR ASE AL PDU types.....	744
10.2.2.1	Coding rules	744
10.2.2.2	Structure of the PDUs.....	745
10.2.3	SUB MMS ASE FAL PDU type.....	756
10.2.3.1	SUB-MMS-PDU	756
10.2.4	VMD ASE Protocol.....	767
10.2.4.1	Environment management protocol	767
10.2.4.2	Protocol concerning the vmd management	771
10.2.5	Domain ASE protocol.....	772
10.2.5.1	Introduction.....	772
10.2.5.2	Definition of the confirmed delete domain service protocol.....	773
10.2.5.3	Definition of the confirmed initiate download sequence service protocol.....	773
10.2.5.4	Definition of the confirmed upload segment service protocol.....	775
10.2.5.5	Definition of the confirmed terminate upload sequence service protocol.....	775
10.2.5.6	Definition of the confirmed get domain attributes service protocol.....	775
10.2.6	Program Invocation ASE protocol.....	776
10.2.6.1	Introduction.....	776
10.2.6.2	Definition of the confirmed create program invocation service protocol	776
10.2.6.3	Definition of the confirmed delete program invocation service protocol.....	777
10.2.6.4	Definition of the confirmed start service protocol	778
10.2.6.5	Definition of the confirmed stop service protocol.....	778
10.2.6.6	Definition of the confirmed resume service protocol.....	778
10.2.6.7	Definition of the confirmed reset service protocol	778
10.2.6.8	Definition of the confirmed kill service protocol.....	778
10.2.6.9	definition of the confirmed service protocol get program invocation attributes	778

10.2.7	Variable ASE protocol	779
10.2.7.1	Common definitions	779
10.2.7.2	Definition of the confirmed read service protocol	780
10.2.7.3	Definition of the confirmed write service protocol	781
10.2.7.4	Definition of the unconfirmed information report service protocol	782
10.2.7.5	Definition of the confirmed define variable list service protocol	782
10.2.7.6	Definition of the confirmed delete variable list service protocol	783
10.2.7.7	Definition of the confirmed get variable access attributes service protocol	784
10.2.7.8	Definition of the confirmed get variable list attributes service protocol	784
10.2.8	Event ASE protocol	785
10.2.8.1	Introduction	785
10.2.8.2	Definition of the unconfirmed event notification service protocol	786
10.2.8.3	Definition of the confirmed acknowledge event notification service protocol	787
10.2.8.4	Definition of the confirmed alter event condition monitoring service protocol	787
10.2.8.5	Definition of the confirmed get alarm summary service protocol	788
10.2.8.6	Definition of the confirmed get event condition attribute service protocol	789
10.2.9	Directory ASE Protocol	790
10.3	Structure of Protocol Machines	790
10.4	AP-Context state machine	792
10.5	Sub-MMS FAL Service Protocol Machine (FSPM)	792
10.5.1	Projection of the SUB-MMS PDUs on the MCS services	792
10.5.2	Projection of the SUB-MMS abort service on the MCS services	792
10.5.3	Construction of a SUB-MMS-PDU from a service primitive	792
10.5.4	Extraction of a valid service primitive from a SUB-MMS-PDU	793
10.5.5	Negotiation of an abstract syntax and a transfer syntax commonly called presentation-context	793
10.5.6	Identification of the SUB-MMS core abstract syntax	795
10.5.7	Identification of the application context name	795
10.5.8	Identification of the ASE of the core abstract syntax and the transfer syntax	795
10.6	DLL Mapping Protocol Machine (DMPM) and Association Relationship Protocol Machine (ARPM)	796
10.6.1	MPS ARPM and DMPM	796
10.6.1.1	Service procedures	796
10.6.1.2	Service elements procedures	803
10.6.2	MCS ARPM and DMPM	805
10.6.2.1	Association establishment	806
10.6.2.2	Association termination	809
10.6.2.3	Association revocation	814
10.6.2.4	Associated mode data transfer	816
10.6.2.5	Non-associated mode data transfer	837
10.7	Protocol options	841
10.7.1	Conformances classes	841
10.7.1.1	Vertical conformance	842
10.7.1.2	Horizontal conformance	843
10.7.1.3	Type conformance	847
10.7.1.4	Protocol Implementation Conformance statement	849
11	Type 8	859
11.1	FAL Syntax Description	859

11.1.1	FAL-AR PDU Abstract Syntax.....	859
11.1.1.1	Confirmed Send Service	859
11.1.1.2	Unconfirmed Send Service	859
11.1.1.3	Unconfirmed Acknowledge Send Service.....	859
11.1.1.4	Establish Service	859
11.1.1.5	ConType.....	860
11.1.2	Abstract Syntax of PDUBody	860
11.1.2.1	Abort Service	860
11.1.2.2	ConfirmedServiceRequest.....	860
11.1.2.3	ConfirmedServiceResponse	860
11.1.2.4	ConfirmedServiceError.....	861
11.1.2.5	Error Type	861
11.1.2.6	Error Fi Type	861
11.1.2.7	Error Class	861
11.1.2.8	Unconfirmed PDUs	861
11.1.2.9	Management ASE	861
11.1.2.10	Application Process ASE.....	861
11.1.2.11	Function Invocation ASE	862
11.1.2.12	Variable ASE	863
11.1.3	Type definitions for ASEs.....	863
11.1.3.1	AP ASE Types	863
11.1.3.2	AR ASE Types	863
11.1.3.3	Function Invocation ASE Types.....	863
11.1.3.4	Management ASE Types.....	864
11.1.3.5	Variable ASE Types	864
11.1.3.6	General Types	864
11.1.3.7	Object Definitions	865
11.1.4	Abstract Syntax of Data Types	866
11.1.4.1	Notation for BinaryDate2000 Type.....	867
11.2	Transfer Syntax	867
11.2.1	Peripherals Encoding Rules (PER).....	867
11.2.2	Encoding of APDU types	867
11.2.2.1	Overview of APDU encoding	867
11.2.2.2	APDU Header encoding	867
11.2.2.3	APDU Body encoding.....	869
11.2.3	Encoding of tagged type values.....	869
11.2.3.1	Encoding of tagged type values of tag class PRIVATE.....	869
11.2.3.2	Encoding of context specific tagged type values	870
11.2.4	Encoding of simple values.....	871
11.2.4.1	Encoding of a Boolean value	871
11.2.4.2	Encoding of String values (visible string, octet string, bit string).....	871
11.2.4.3	Encoding of an Integer value.....	871
11.2.4.4	Encoding of an Unsigned value	872
11.2.4.5	Encoding of a floating-point value.....	872
11.2.4.6	Encoding of a BinaryDate value.....	872
11.2.4.7	Encoding of a BinaryDate2000 value	872
11.2.4.8	Encoding of a Time of Day value.....	873
11.2.4.9	Encoding of a Time Difference value	873

11.2.4.10	Encoding of a Time value	874
11.2.4.11	Encoding of a Null value	874
11.2.4.12	Encoding of an ANY value	874
11.2.4.13	Encoding of Structured Types	875
11.2.4.14	Encoding of a SEQUENCE value	875
11.2.4.15	Encoding of a SEQUENCE OF value	875
11.2.4.16	Encoding of a CHOICE value	875
11.2.4.17	Encoding of an ENUMERATED value	875
11.2.4.18	Encoding of an Object-Definition value	875
11.3	FAL Protocol State Machine Structure	875
11.4	AP Context State Machine	875
11.5	FAL Service Protocol Machine (FSPM)	875
11.6	Application Relationship Protocol Machines (ARPMs)	876
11.6.1	Queued Usertriggered Bidirectional-Flow Control (QUB-FC) ARPM	876
11.6.1.1	QUB-FC Primitive Definitions	876
11.6.1.2	DLL Mapping of QUB-FC AREP Class	876
11.6.1.3	Attributes	877
11.6.1.4	DLL services	878
11.6.1.5	QUB-FC ARPM State Machine	878
11.6.2	Buffered Networkscheduled Unidirectional (BNU) ARPM	878
11.6.2.1	Primitive definitions	878
11.6.2.2	DLL Mapping of BNU AREP Class	878
11.6.2.3	Attributes	879
11.6.2.4	BNU AREP state machine	879
11.7	DLL Mapping Protocol Machine	879
11.7.1	Overview	879
11.7.2	Primitive definitions	880
11.7.2.1	Primitives Exchanged between DMPM and ARPM	880
11.7.2.2	Parameters of ARPM/DMPM primitives	881
11.7.2.3	Primitives Exchanged between Data Link Layer and DMPM	881
11.7.2.4	Parameters of DMPM/Data Link Layer Primitives	882
11.7.3	DMPM State Machine	882
11.7.3.1	DMPM States	882
11.7.3.2	DMPM State Table	882
11.7.3.3	Functions used by Type-8 DMPM	888
11.8	Protocol Option	890
11.8.1	Introduction	890
11.8.2	Abstract Syntax	890
11.8.3	Protocol machine overview	890
11.8.3.1	Application Relationship Protocol Machine (ARPM) selection	890
11.8.3.2	Encoding Rule Selection	890
11.8.3.3	DLL Mapper selection	890

INTERNATIONAL ELECTROTECHNICAL COMMISSION

DIGITAL DATA COMMUNICATIONS FOR MEASUREMENT AND CONTROL – FIELDBUS FOR USE IN INDUSTRIAL CONTROL SYSTEMS –

Part 6: Application Layer protocol specification

FOREWORD

- 1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.
- 3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
- 4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
- 5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.
- 6) Attention is drawn to the possibility that some of the elements of this standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International standard IEC 61158-6 has been prepared by subcommittee 65C: Digital communications, of IEC technical committee 65: Industrial-process measurement and control.

This second edition cancels and replaced the first edition which was issued as a technical specification in 1999. It constitutes a technical revision and now has the status of an International Standard.

This second edition adds seven distinct sets of services, each with a corresponding protocol, to the set of services and protocols of the first edition.

The text of this standard is based on the following documents:

FDIS	Report on voting
65C/226/FDIS	65C/231/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 3.

IEC 61158 consists of the following parts, under the general title *Digital data communications for measurement and control — Fieldbus for use in industrial control systems*:

- Part 1: Introductory guide (under preparation)
- Part 2: Physical layer specification and service definition
- Part 3: Data Link Service definition
- Part 4: Data Link Protocol specification
- Part 5: Application Layer service definition
- Part 6: Application Layer protocol specification
- Part 7: System management (under consideration)
- Part 8: Conformance testing (under consideration)

The committee has decided the the contents of this publication will remain unchanged until 2006. At this date, the publication will be

- reconfirmed;
- withdrawn;
- replaced by a revised edition; or
- amended.

Withdrawn

INTRODUCTION

This standard describes the Fieldbus Application Layer (FAL) protocol that defines the information interchange and the interactions between Application Entity invocations (AE-Is) to support the services defined in IEC 61158-5.

An Application Process uses the Fieldbus Application Layer services to exchange information with other Application Processes. The services define the abstract interface between the application process and the Application Layer.

The Application Layer protocol is the set of rules that governs the format and meaning of the information exchange between the Application Layers in various devices. The Application Layer uses the protocol to implement the Application Layer services definitions.

The protocol machine defines the various states of an Application Layer and the valid transitions between the states. It may be considered as a finite state machine. The protocol machine is described using state tables. The information is exchanged between the application process and the protocol machine through application service data units. The protocol machine exchanges information with other protocol machines through application protocol data units (APDU).

This set of Application Layer standards does not specify individual implementations or products, nor does it constrain the implementations of Application Entities (AEs) and interfaces within the industrial automation system.

This set of Application Layer standards does not contain test procedures to ensure compliance with such requirements.

Withd
ra
w
n

DIGITAL DATA COMMUNICATIONS FOR MEASUREMENT AND CONTROL – FIELDBUS FOR USE IN INDUSTRIAL CONTROL SYSTEMS –

Part 6: Application Layer protocol specification

1 Scope

The Fieldbus Application Layer (FAL) is an Application Layer communication standard designed to support the conveyance of time-critical application requests and responses among devices in an automation environment. 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 in terms of

- the encoding rules that are applied to all the Application Layer Protocol Data Units (APDUs);
- the formal Abstract Syntax definitions of such APDUs;
- the protocol state machine descriptions that handle the APDUs and the primitives in the correct sequences;
- the mappings of the APDUs to and from the Data Link Layer services defined in IEC 61158-3.

The FAL encoding rules are designed assuming that both the encoder (sender) and the decoder (receiver) have the common knowledge of the abstract syntax. Wherever possible, data types identifiers are not encoded and transferred over the network.

NOTE This is why the Abstract Syntax Notation One / Basic Encoding Rule is not practical for the FAL.

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

- a) to the Fieldbus Data Link Layer at the boundary between the Application and Data Link Layers of the Fieldbus Reference Model, and
- b) to the System Management at the boundary between the System Management and Application Layers of the Fieldbus Reference Model.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 61158. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 61158 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards

IEC 61131-3:1993, *Programmable controllers – Part 3: Programming languages*

IEC 61158-2:1993, *Fieldbus Standard for use in industrial control systems – Part 2: Physical layer specification and service definition*

IEC 61158-3:2000, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 3: Data Link service definitions*

IEC 61158-4:2000, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 4: Data Link protocol specifications*

IEC 61158-5:2000, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 5: Application Layer service definitions*

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

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

ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

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

ISO/IEC 8825:1990, *Information technology – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*

ISO/IEC 9506-2:1990, *Industrial automation systems – Manufacturing message specification – Part 2: Protocol specification*

ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application Layer structure*

ISO/IEC 10646: all parts, *Information technology – Universal Multiple-Octet Coded Character Set (UCS)*

ISO/IEC 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

ISO 8649:1996, *Information technology – Open Systems Interconnection – Service definition for the Association Control Service Element Protocol specification*

ISO 8650: all parts, *Information technology – Open Systems Interconnection – Connection-oriented protocol for the Association Control Service Element Protocol specification*

IEEE 754:1985, *Binary Floating-Point Arithmetic*

3 Definitions

For the purpose of this standard, the following definitions apply:

3.1 Definitions from other ISO/IEC Standards

3.1.1 Definitions from ISO/IEC 7498-1

- a) application entity;
- b) application process;
- c) application protocol data unit;
- d) application service element;
- e) application entity invocation;
- f) application process invocation;
- g) application transaction;
- h) real open system;
- i) transfer syntax.

3.1.2 Definitions from ISO/IEC 8822

- a) abstract syntax;
- b) presentation context.

3.1.3 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.

3.1.4 Definitions from ISO/IEC 8824

- 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.

3.1.5 Definitions from ISO/IEC 8825

- 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 Definitions from IEC 61158-5

- a) application relationship;
- b) conveyance path;
- c) client;
- d) dedicated AR;

- e) dynamic AR;
- f) error class;
- g) error code;
- h) name;
- i) numeric identifier;
- j) peer;
- k) pre-defined AR endpoint;
- l) pre-established AR endpoint;
- m) publisher;
- n) server.

3.3 Other definitions

NOTE Type 1 definitions shall apply to all the types except if the same definition is used with other types, in that case the latter shall have precedence.

The following definitions are used in this standard:

3.3.1 Type 1 Definitions

3.3.1.1 called

Service user or a service provider that receives an indication primitive or a request APDU.

3.3.1.2 calling

Service user or a service provider that initiates a request primitive or a request APDU.

3.3.1.3 interoperability

Capability of User Layer entities to perform coordinated and cooperative operations using the services of the FAL.

3.3.1.4 management information

Network accessible information that supports the management of the Fieldbus environment.

3.3.1.5 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.

3.3.1.6 resource

Resource is a processing or information capability of a subsystem.

3.3.1.7 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.

3.3.2 Type 2 Definitions

3.3.2.1 Allocate

To take a resource from a common area and assign that resource for the exclusive use of a specific entity.

3.3.2.2 Application

Function or data structure for which data is consumed or produced.

3.3.2.3 Application objects

Multiple object classes that manage and provide the run time exchange of messages across the network and within the network device.

3.3.2.4 Attribute

A description of an externally visible characteristic or feature of an object. 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.

3.3.2.5 Behaviour

Indication of how the object responds to particular events. Its description includes the relationship between attribute values and services.

3.3.2.6 Class

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

3.3.2.7 Class attributes

An attribute that is shared by all objects within the same class.

3.3.2.8 Class code

A unique identifier assigned to each object class.

3.3.2.9 Class specific service

A service defined by a particular object class to perform a required function which is not performed by a common service. A class specific object is unique to the object class which defines it.

3.3.2.10 Client

- (1) An object which uses the services of another (server) object to perform a task.
- (2) An initiator of a message to which a server reacts.

communication objects Components that manage and provide run time exchange of messages across the network such as the Connection Manager object, the unconnected message manager (UCMM), and the Message Router object.

3.3.2.11 Connection

A logical binding between two application objects. These application objects may be within the same or different devices.

3.3.2.12 Connection ID (CID)

Identifier assigned to a transmission that is associated with a particular connection between producers and consumers that identifies a specific piece of application information.

3.3.2.13 Connection path

The attribute is made up of a byte stream which defines the application object to which a connection instance applies.

3.3.2.14 Connection point

A buffer that is part of another object. The buffer is represented as a subinstance of an assembly object.

3.3.2.15 Consume

The act of receiving data from a producer.

3.3.2.16 Consumer

A node that is receiving data from a producer.

3.3.2.17 Consuming application

The application that consumes data.

3.3.2.18 Cyclic

Term used to describe events which repeat in a regular and repetitive manner.

3.3.2.19 Device

A physical hardware connection to the link. A device may contain more than one node.

3.3.2.20 Device profile

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

3.3.2.21 End node

A producing or consuming node.

3.3.2.22 End point

One of the communicating entities involved in a connection.

3.3.2.23 Error

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

3.3.2.24 Frame

Single data transfer on a link.

3.3.2.25 Instance

The actual physical occurrence of an object within a class. Identifies one of many objects within the same object class. For example: California is an instance of the object class state. The terms object, instance, and object instance are used to refer to a specific instance.

3.3.2.26 Instance attributes

An attribute that is unique to an object instance and not shared by the object class.

3.3.2.27 Instantiated

An object that has been created in a device.

3.3.2.28 Keeper

Object responsible for distributing link configuration data to all nodes on the link.

3.3.2.29 Little endian

Describes a model of memory organisation which stores the least significant byte at the lowest address. On the network medium, the lowest order byte is transferred first.

3.3.2.30 Lpacket

The Lpacket (or link packet) is a piece of application information that contains a size, control byte, tag, and link data. Peer Data Link Layers use Lpackets to send and receive service data units from higher layers in the OSI stack.

3.3.2.31 Member

A piece of an attribute that is structured as an array.

3.3.2.32 Message router

The object within a node that distributes messaging requests to the appropriate application objects.

3.3.2.33 Multicast connection

A connection from one node to many. Multicast connections allow messages from a single producer to be received by many consumer nodes.

3.3.2.34 Network

A series of nodes connected by some type of communication medium. The connection paths between any pair of nodes can include repeaters, routers and gateways.

3.3.2.35 Object

An abstract representation of a particular component within a device, i.e. :

(1) An abstract representation of a computer's capabilities. Objects can be composed of any or all of the following components:

- a) data (information which changes with time);

- b) configuration (parameters for behaviour);
- c) methods (things that can be done using data and configuration).

(2) A collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behaviour.

3.3.2.36 Object specific service

A service defined by a particular object class to perform a required function which is not performed by a common service. An object specific service is unique to the object class which defines it.

3.3.2.37 Originator

The client responsible for establishing a connection path to the target.

3.3.2.38 Point-to-point connection

A connection that exists between two nodes only. Connections can be either point-to-point or multicast.

3.3.2.39 Produce

Act of sending data to be received by a consumer.

3.3.2.40 Producer

A node that is responsible for sending data.

3.3.2.41 Server

An object which provides services to another (client) object.

3.3.2.42 Service

Operation or function that an object and/or object class performs upon request from another object and/or object class. 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.

3.3.2.43 Target

The end-node to which a connection is established.

3.3.2.44 Temporary node

Same as transient node.

3.3.2.45 Transaction id

Field within the UCMM header that matches a response with the associated request. The server echoes this field in the response message.

3.3.2.46 Unconnected message manager (UCMM)

The component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object.

3.3.2.47 Unconnected service

The messaging service which does not rely on the set up of a connection between devices before allowing information exchanges.

3.3.3 Type 3 Definitions

3.3.3.1 Access Protection

The access to certain functions of a DP-Slave is restricted to one DP-Master.

3.3.3.2 Address-assignment-table

Mapping of the DP-Masters internal IO-addresses to the decentralised inputs and outputs.

3.3.3.3 Channel

A single physical or logical connection of inputs or outputs of a DP-Slave to the process.

3.3.3.4 Channel related Diagnosis

This diagnosis information refers to a single input or output channel of the DP-Slave.

3.3.3.5 Configuration check

In the start-up phase, the DP-Master transmits the expected configuration for comparison with the real configuration to the DP-Slave.

3.3.3.6 Configuration fault

If the DP-Slave detects a not acceptable difference between the expected configuration and the real configuration, then a configuration fault is indicated in the diagnosis information.

3.3.3.7 Configuration Identifier

For each input- and/or output-module of a DP-Slave at least one Configuration Identifier is defined in the configuration string.

3.3.3.8 Configuration Identifier related Diagnosis

This diagnosis information is related to the input- or output-range for which a Configuration Identifier is defined in the configuration string.

3.3.3.9 Control Commands

Control commands will be transferred from the DP-Master to the DP-Slave. With the control commands it is possible to clear the outputs, or to Freeze the inputs and/or synchronise the outputs of the DP-Slaves.

3.3.3.10 Data consistency

The input- or output-data-range that always shall be transmitted consistent between the DP-Master and the DP-Slave.

3.3.3.11 Default address

The default address of a DP-Slave is 126.

This address will be used as initial value if address setting will be done via PROFIBUS-DP.

3.3.3.12 Device related diagnosis

This diagnosis information is related to the whole Slave.

3.3.3.13 Diagnosis information

This term is used to identify the whole diagnosis data of a DP-Slave.

3.3.3.14 Diagnosis information collection

A system-diagnosis information that is collected at the DP-Master (Class 1).

3.3.3.15 DP-Master (Class 1)

A device that controls the assigned DP-Slaves; usually a programmable controller or distributed control system.

3.3.3.16 DP-Master (Class 2)

A device that interacts as a configuration or diagnosis tool; usually a programming device.

3.3.3.17 DP-Slave

A device that interacts to one DP-Master as a provider for cyclic I/O data exchange. In addition acyclic functions could be provided.

3.3.3.18 Freeze

By means of this function the input data of several DP-Slaves will be simultaneously frozen.

3.3.3.19 Group

A set of DP-Slaves which perform a Freeze or Sync operation.

3.3.3.20 I/O data

Data which have to be transferred cyclically for the purpose of processing

3.3.3.21 Ident Number

The Ident Number specifies a DP-Master (Class 1) or DP-Slave device type.

3.3.3.22 Index

Address of an object within an application process.

3.3.3.23 Master parameter set

The Master parameter set includes the configuration and parameterisation data of all DP-Slaves that are assigned to the corresponding DP-Master and the bus parameters.

3.3.3.24 Module

Addressable unit inside the DP-Slave.

3.3.3.25 Process data

Data which are already pre-processed and transferred acyclically for the purpose of information or further processing.

3.3.3.26 Real Configuration

The real configuration describes the input- and output-data-ranges of the DP-Slave. Additionally, the definition of the data consistency is included in the real configuration.

3.3.3.27 Slot

Address of a module within a DP-Slave.

3.3.3.28 Sync

By means of this function the outputs of several DP-Slaves will be simultaneously updated.

3.3.4 Type 4 Definitions

There are no Type 4 definitions.

3.3.5 Type 5 Definitions

There are no Type 5 definitions.

3.3.6 Type 6 Definitions

There are no Type 6 definitions.

3.3.7 Type 7 Definitions

There are no Type 7 definitions.

3.3.8 Type 8 Definitions

There are no Type 8 definitions.

3.4 Abbreviations and symbols

NOTE Type 1 abbreviations and symbols shall apply to all the types except the same abbreviation or symbols are used with other types, in that case the latter shall have precedence.

3.4.1 Type 1 Abbreviations and symbols

AE	Application Entity
AE-I	Application Entity Invocation
AL	Application Layer
AP	Application Process
Ap_	Prefix for Data types defined for AP ASE
Ar_	Prefix for Data types defined for AR ASE
APDU	Application Protocol Data Unit
AR	Application Relationship
AREP	Application Relationship End Point
ASE	Application Service Element

ASN.1	Abstract Syntax Notation One
BCD	Binary Coded Decimal
BER	Basic Encoding Rule
cnf	confirmation primitive
DI_	Prefix for Data types defined for Data Link Layer types
DL	Data Link
DLC	Data Link Connection
DLCEP	Data Link Connection End Point
DLSAP	Data Link Service Access Point
DLSDU	Data Link Service Data Unit
Dt_	Prefix for Data types defined for Data Type ASE
Err	Error (used to indicate an APDU type)
Er_	Prefix for Error types defined
Ev_	Prefix for Data types defined for Event ASE
FAL	Fieldbus Application Layer
Fi_	Prefix for Data types defined for Function Invocation ASE
FIFO	First In First Out
Gn_	Prefix for Data Types defined for general use
ID	Identifier
IEC	International Electrotechnical Commission
in	input primitive
ind	indication primitive
ISO	International Organization for Standardization
LAS	Link Active Scheduler
Lr_	Prefix for Data types defined for Load Region ASE
lsb	least significant bit
Mn_	Prefix for Data Types defined for Management ASE
msb	most significant bit
out	output primitive
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
QoS	Quality Of Service
Req	Request (used to indicate an APDU type)
req	request primitive
Rsp	Response (used to indicate an APDU type)
rsp	response primitive
SAP	Service Access Point
SDU	Service Data Unit
ToS	Type Of Service
Vr_	Prefix for Data types defined for Variable ASE

3.4.2 Type 2 Abbreviations and symbols

ASCII	American Standard Code for Information Interchange
CID	connection ID
CIP	The control and information protocol. CIP includes both connected and unconnected messaging.
DLL	Data Link Layer
PDU	protocol data unit
OSI	open systems interconnection (see ISO 7498)
Rcv	receive
Rx	receive
SDU	service data unit
SEM	state event matrix
STD	state transition diagram, used to describe object behaviour
Tx	transmit
Xmit	transmit
CM_API	actual packet interval
O2T or O⇒T	originator to target (connection parameters)
CM_RPI	requested packet interval
TUI	table unique identifier
T2O or T⇒O	target to originator (connection parameters)

3.4.3 Type 3 Abbreviations and symbols

Ack	Acknowledgement
Add	Address
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
AR	Application Relationship
AREP	Application Relationship End Point
ARL	Application Relationship List
ASE	Application Service Element
Cfg	Configuration
cnf	confirmation primitive

CREP	Communication Relationship End Point
CRL	Communication Relationship List
D_	Destination
DL	Data Link
DLM	Data Link-management
DLSAP	Data Link Service Access Point
DLSDU	Data Link Service Data Unit
DMPM	Data Link Mapping Protocol Machine
DMPMM1	Data Link Mapping Protocol Machine DP-Master (Class 1)
DMPMM2	Data Link Mapping Protocol Machine DP-Master (Class 2)
DMPMS	Data Link Mapping Protocol Machine DP-Slave
DSAP	Destination Service Access Point
FAL	Fieldbus Application Layer
FIFO	First In First Out
FSPMM1	FAL Service Protocol Machine DP-Master (Class 1)
FSPMM2	FAL Service Protocol Machine DP-Master (Class 2)
FSPMS	FAL Service Protocol Machine DP-Slave
HSA	Highest Station Address
ID	Identifier
IEC	International Electrotechnical Commission
ind	indication primitive
Inp	Input
ISO	International Organization For Standardization
L_sdu	Link Service Data Unit
lsb	least significant bit
msb	most significant bit
OSI	Open Systems Interconnection
Outp	Output
Param	Parameter
PDU	Protocol Data Unit
RD_	Read
Rem	Remote
Req	Request (used to indicate an APDU type)
req	request primitive
REQ	Request to indicate a Request PDU
RES	Response to indicate a Response PDU
RM	Resource Manager
RPL_UPD	Reply Update
Rsp	Response (used to indicate an APDU type)
rsp	response primitive
S_	Source
SAP	Service Access Point
SCL	Security Level
SD	Start Delimiter
SDN	Send Data with no Acknowledge
SDR	Station Delay Responder
Seq	Sequence
Src	Source
SRD	Send and Request Data with Reply
SSAP	Source Service Access Point
TR	Technical Report
USIF	User Interface
WD	Watchdog

3.4.4 Type 4 Abbreviations and symbols

There are no Type 4 abbreviations and symbols

3.4.5 Type 5 Abbreviations and symbols

There are no Type 5 abbreviations and symbols

3.4.6 Type 6 Abbreviations and symbols

There are no Type 6 abbreviations and symbols

3.4.7 Type 7 Abbreviations and symbols

A	Actuator
ACSE	Standard ISO 8649/ISO8650
AEI	Application Entity Implementation
API	Invocation of the Application Process
BER	ISO8825 Basic Encoding Rules
BT	Buffer Transfer - Layer 2
C	Controller
CC	Cell Controller

CADM	Computer Assisted Design & Manufacturing
CAPM	Computer Assisted Product Management
Cnf	Confirm
CS	Control System
CSD	Control System Device
DLSE	Data Link Service Element
Ind	Service Indication
MCS	Messaging Common Services
MMS	Standard ISO9506
MPS	Periodic/Aperiodic Manufacturing Services
MSG	Messaging Services - Layer 2
MAP3.0	ISO functional Profile
OS	Operator station
Rp	Service Response
Rq	Request
S	Sensor

3.4.8 Type 8 Abbreviations and symbols

There are no Type 8 abbreviations and symbols

3.5 Conventions

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate clause. 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. 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.

3.5.1 Conventions for Type 1

3.5.1.1 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.

3.5.1.2 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.

3.5.2 Conventions for Type 2

3.5.2.1 Attribute specification

Attributes are defined in an Attribute Table using the format and terms defined in Figure 1.

Attribute ID	Name	Data type	Semantics of values
--------------	------	-----------	---------------------

Figure 1: Attribute table format and terms

The **Attribute ID** shall be a unique integer identification value assigned to an attribute. The valid ranges for Attribute IDs shall be as specified in 5.1.1.9.1.3. The Attribute ID shall identify the particular attribute being accessed.

Name shall refer to the name assigned to the attribute. An attribute may contain sub-elements, which may also have names, as is the case with the **STRUCT** of data type. The attribute name shall be the name which appears first, or at the top row in the name column (the row that contains the Attribute ID).

Every attribute shall be assigned a **Data type** which shall be either an elementary or a derived data type. The data types specified for the defined attributes shall be used in all implementations.

NOTE The elementary data types are defined in IEC 61158-5.

Semantics of values shall specify the meaning of the value(s) of the attribute. If this information needs more room than can fit in the table it will immediately follow the Class Attribute table. Included in the Class Attribute table will be an appropriate reference to this information.

If a Class Attribute is *optional*, then a default value or a special case processing method shall be defined such that the Client (Requester) can process the error message that occurs when accessing those objects that choose not to implement the class attribute.

3.5.2.2 Common Services

3.5.2.2.1 Service_PDU definitions

Each service has unique parameters for request and response. The service request and response parameters shall be defined using service Request/Response parameter tables as defined in Figure 2.

Name	Type	Semantics of values
------	------	---------------------

Figure 2: Service request/response parameter

Name shall refer to the name given to the service request/response parameter.

Type shall specify the data type of the service request/response parameter.

Semantics of values shall specify the meaning of the values of the service request/response parameter, e.g. "the value is counts of microseconds."

3.5.2.2.2 Get_Attribute_All Response

3.5.2.2.2.1 General Definition

When the Get_Attribute_All common service is included in the list of supported System/Object Management services for an object class, then the **Get_Attribute_All** response shall be detailed for this class : the sequence or order of the data returned in the Service_ResponsePDU shall be specified. There are three ways in which the Get_Attribute_All Service_ResponsePDU may be specified:

- list the ordering of the attributes in the response message;
- specify the actual data array of the response message;
- combine the above two methods such that the actual data array also contains the attribute numbers.

Whichever method is used, the rules specified in Table 1 shall be adhered to when specifying the Get_Attribute_All Service_ResponsePDU of an object class for both the Class Attributes and the Instance Attributes.

Table 1: Get_Attribute_All response service rules

Rule number	Rule
1	If the definition of the Get_Attribute_All response includes optional attributes, then default values shall be specified in the response description. These defaults shall be used if an implementation does not support the optional attributes. If any of the following optional attributes are included in an object specification, but not supported in the implementation of the object, then the following shall be adhered to: - if the class attribute "Optional attribute list" is not supported, the default value of zero shall be inserted into the response array and no optional attribute numbers shall follow; - if the class attribute "optional service list" is not supported, the default value of zero shall be inserted into the response array and no optional service numbers shall follow.
2	If new attributes are added to an existing object, those attributes shall be added to the end of the response attribute list or data array to ensure compatibility with different object revisions.
3	Whichever method is used to specify the response, it shall be done in such a way as to be unambiguous, including rules to deal with variable length fields and padding.
4	The Get_Attribute_All response shall include only the open attributes; it shall not include any vendor specific attributes.

Table 2 is an example of the attribute ordering method of specifying the service data portion of a Get_Attribute_All response for class level attributes of an object which supports **optional gettable** class attributes 1,2,3 and 4.

Table 2: Example Class level object/service specific reply data of Get_Attribute_All

Class attribute ID	Attribute name and default value
1	Revision, default = 0x0001
2	Max Instance, default = 0x0000
3	Number of Instances, default = 0x0000
4	Attribute list, number of attributes, default = 0x0000

Table 3 is an example of the data array method of specifying the service data portion of a Get_Attribute_All response for class level attributes of an object which supports **optional gettable** class attributes 1,2,3 and 4.

Table 3: Example Get_Attribute_All data array method

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Revision (low byte) Default = 1							
1	Revision (high byte) Default = 0							
2	Max Instance (low byte) Default = 0							
3	Max Instance (high byte) Default = 0							
4	Number of Instances (low byte) Default = 0							
5	Number of Instances (high byte) Default = 0							
6	Optional Attribute List : number of attributes (low byte) Default = 0							
7	Optional Attribute List : number of attributes (high byte) Default = 0							
8	Optional Attribute List : optional attribute #1 (low byte)							
9	Optional Attribute List : optional attribute #1 (high byte)							
2n + 6	Optional Attribute List : optional attribute #n (low byte)							
2n + 7	Optional Attribute List : optional attribute #n (high byte)							

3.5.2.2.2 Revisions

The defined Get_Attribute_All response for an object may increase in size with each revision of the object; however, to insure interoperability, the format of the first part of the response shall remain parseable by a client of the older revision of the object. Clients (Requesters) need not make use of this compatibility requirement.

NOTE The Revision class attribute is not the revision of an implementation (which is reflected in the Identity Object, Minor/Major revision status bits), but the revision of the *class definition*.

3.5.2.2.3 Set_Attribute_All Request

3.5.2.2.3.1 General Definition

When the Set_Attribute_All common service is included in the list of supported common services, then the format of the **Set_Attribute_All request** shall be included in the object specification. The sequence or order of the data supplied in the Service Data portion of the request shall be specified. There are three ways in which the Set_Attribute_All request may be specified:

- list the order of the attributes in the request message;
- specify the actual data array of the request message;
- combine the above two methods such that the actual data array also contains the attribute numbers.

Whichever method is used, the rules specified in Table 4 shall be adhered to when specifying an object's Set_Attribute_All request in the object specification for both the Class Attributes and the Instance Attributes.

Table 4: Set_Attribute_All request service rules

Rule number	Rule
1	An object shall support the Set_Attribute_All service only if all settable attributes shown in the Set_Attribute_All request are implemented as settable.
2	If new settable attributes are added to an existing object, those attributes shall be added to the end of the request attribute list or data array.
3	Whichever method is used to specify the response, it shall be done in such a way as to be unambiguous, including rules to deal with variable length fields and padding.
4	The Set_Attribute_All request shall include only the open attributes. It shall not include any vendor specific attributes.

Table 5 is an example of the attribute ordering method of specifying the service data portion of a Set_Attribute_All request for instance level attributes of an object which supports **required settable** instance attributes 7,8,9,10,11 and 12.

Table 5: Example Set_Attribute_All attribute ordering method

Class Attribute ID	Attribute name
7	Output Range
8	Value Data Type
9	Fault State
10	Idle State
11	Fault Value
12	Idle Value

Table 6 is an example of the data array method of specifying the service data portion of a Set_Attribute_All request for instance level attributes of an object which supports **required settable** instance attributes 7,8,9,10,11 and 12.

Table 6: Example Set_Attribute_All data array method

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Output Range							
1	Value Data Type							
2	Fault State							
3	Idle State							
4	Fault Value (low byte)							
5	Fault Value (high byte)							
6	Idle Value (low byte)							
7	Idle Value (high byte)							

3.5.2.2.3.2 Revisions

A server processing a Set_Attribute_All request may need to process more data than is expected by the server if the client has implemented a newer revision of this object while the server an older revision. As a result, the server object may have to process a Set_Attribute_All request that contains more data because the additional attributes were not recognised. If the server receives more data in a Set_Attribute_All request than it expects the server shall respond with a general status code equal to *Too much data* (0x15).

NOTE The Revision class attribute is not the revision of an implementation (which is reflected in the Identity Object, Minor/Major revision status bits), but the revision of the *class definition*.

3.5.3 Conventions for Type 3

3.5.3.1 Abstract Syntax Conventions

The following conventions and rules are used:

1. PDUs are described as octets or groups of octets
2. Groups of octets separated by a comma appear in the order they are transferred
4. If optional octets are not present the following octets appear without a gap
5. If octets or groups of octets are grouped within "{ }" the order is arbitrary
6. If octets or groups of octets are marked with "*" they may appear more than once. If it is used within a "{ }" section the may appear mixed with other octets or group of octets of this section.
7. Octets can be grouped or values can be assigned within "()"
8. Complex APDUs may be built by means of substitutions (sub-structures)
9. Exclusive selections of octets or groups of octets are separated by "^"

NOTE1 The formal PDU example

APOE_PDU=Octet1,OctetGroup1,[Octet2],[Octet3],[OctGroup2*],OctetGroup3^Octet4)

According to this the following variants are valid on the wire (non exhaustive):

Variant 1: Octet1, OctetGroup1, Octet2, Octet3, OctetGroup2,OctetGroup3

Variant 2: Octet1, OctetGroup1, Octet2, Octet3, OctetGroup2, OctetGroup2, OctetGroup2,OctetGroup3

Variant 3: Octet1, OctetGroup1, OctetGroup2, OctetGroup2, OctetGroup2,OctetGroup3, OctetGroup2

Variant 4: Octet1, OctetGroup1, OctetGroup2, OctetGroup3, OctetGroup2, OctetGroup2, OctetGroup2, OctetGroup2

Variant 5: Octet1, OctetGroup1, Octet3, Octet4

NOTE2 The arbitrary order implies that groups of octets are characterised by a special header that is described within the coding rules.

NOTE3 The APDU syntax implies that according to the maximum DLSDU a APDU shall not exceed 244 octets in total.

3.5.3.2 Convention for the Encoding of Reserved Bits and Octets

The term "reserved" may be used to describe bits in octets or whole octets. All bits or octets that are reserved should be set to zero.

3.5.3.3 Conventions for the Common Coding Structure of specific Field Octets

APDUs may contain specific fields that carry information in a primitive and condensed way. These fields shall be coded in the following order.

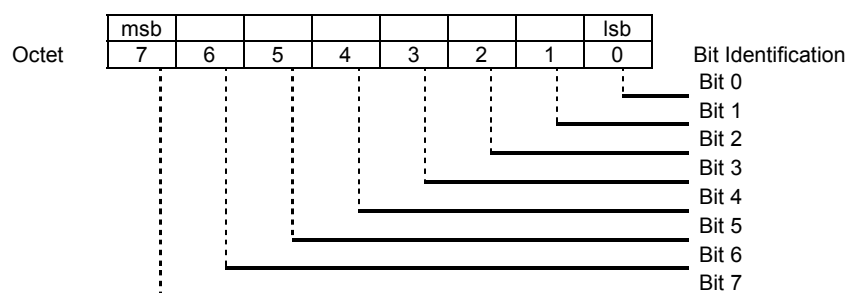


Figure 3 - Common structure of specific Fields

Bits may be grouped as group of bits. Each bit or group of bits shall be addressed by its Bit Identification (e.g. Bit 0, Bit 1 to 4). The position within the octet shall be according to the figure above. Alias names may be used for each bit or group of bits or they may be marked as reserved. The grouping of individual bits shall be in ascending order without gaps. The values for a group of bits may be represented as binary, decimal or hexadecimal values. This value shall only be valid for the grouped bits and can only represent the whole octet if all 8 bits are grouped. Decimal or hexadecimal values shall be transferred in binary values so that the bit with the highest number of the group represents the msb concerning the grouped bits.

EXAMPLE 1 Description and relation for the specific field octet

Bit 0 reserved.

Bit 1-3: Reason_Code The decimal value 2 for the Reason_Code means general error.

Bit 4-7 shall always set to one.

The octet that is constructed according to the description above looks as follows:

(msb) Bit 7 = 1, Bit 6 = 1, Bit 5 = 1, Bit 4 = 1, Bit 3 = 0, Bit 2 = 1, Bit 1 = 0, (lsb) Bit 0 = 0.

The bit combination "0-1-0" for Bit 1-3 equals the decimal value 2.

3.5.4 Conventions for Type 4

There are no Type 4 specific conventions defined.

3.5.5 Conventions for Type 5

There are no Type 5 specific conventions defined.

3.5.6 Conventions for Type 6

There are no Type 6 specific conventions defined.

3.5.7 Conventions for Type 7

There are no Type 7 specific conventions defined.

3.5.8 Conventions for Type 8

There are no Type 8 specific conventions defined.

3.6 Conventions used in State Machines

3.6.1 Conventions for Type 1

The State Machines are described as follows:

Table 7 – Conventions used for State Machines

No.	Current State	Events Actions	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"

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

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.

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

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.

Readers are strongly recommended to refer to the clauses 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.

3.6.2 Conventions for Type 2

3.6.2.1 State Machine Conventions

State changes may be triggered by events (internal or external) or service invocations. Reaction to service invocations may depend on the value(s) of the attribute(s) accessed by the service.

Behaviour of the state machine shall be defined for combinations of :

- the *event* the machine receives;
- the *state* the machine is in when it receives notification of a state-changing event.

To define behaviour in these terms, a State Transition Diagram (STD) and a State Event Matrix (SEM) are used when applicable in the state machine specification.

3.6.2.1.1 State Transition Diagram

An **event** is an external stimulus that may cause a state transition. A **State Transition Diagram (STD)** graphically illustrates the states of an object and includes events, service calls and changes of attributes that cause it to transition to another state. Figure 4 shows an example of a State Transition Diagram.

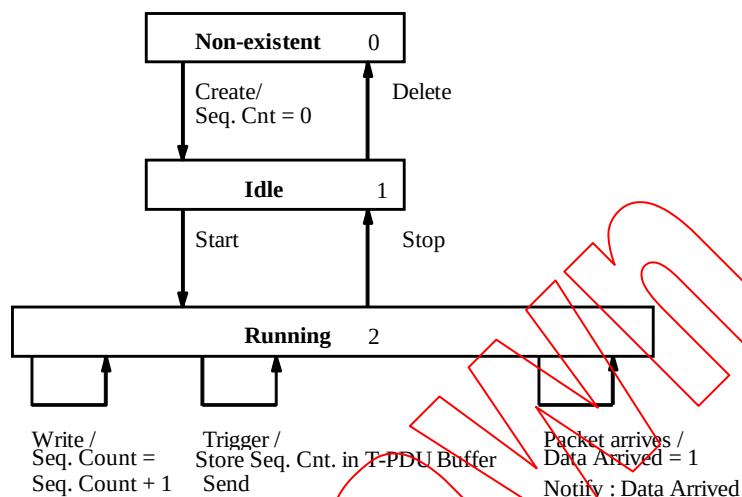


Figure 4: Example of State Transition Diagram (STD)

NOTE Event that is the primary cause of the State Transition is followed by “/” mark. Notification to upper layer is marked by “Notify :”

3.6.2.1.2 State Event Matrix

A **state** is the current active mode of operation of the state machine object (e.g. Running, Idle).

A **State Event Matrix** is a table that lists all possible events, services and changes in attribute values that initiate a state change, and indicates the response by the object to the event based on the state of the object when it receives notification of that event.

State Event Matrix format is as shown in Table 8:

Table 8: State Event Matrix Format

Event	State		
	State 1		State n
Event A description	Function triggered by event A in State 1 (if any) Notification to FAL user (if any) Transition to another state (if any)		Function triggered by event A in State n (if any) Notification to FAL user (if any) Transition to another state (if any)
.....			
Event X description	Function triggered by event X in State 1 (if any) Notification to FAL user (if any) Transition to another state (if any)		Function triggered by event X in State n (if any) Notification to FAL user (if any) Transition to another state (if any)

NOTE In absence of function, notification or transition, the corresponding entry shall not be made in the table.

3.6.2.1.3 Example State Event Matrix (Informative)

Table 9 shows an example of a state machine with three states :

- **Non-existent:** the object has not yet been created; objects transition to the existent state via the create service (if the object may be dynamically created) or at power-up (if the object is fixed by design/implementation);
- **Idle:** the object accepts services (e.g. Get_Attribute_Single), but does not produce or consume data onto or from the link;
- **Running:** the object is performing all its specified functions.

Table 9: Example State Event Matrix

Event	State		
	Non-existent	Idle	Running
Create	Transition to Idle	Error: Object already exists.	Error: Object already exists.
Delete	Error: Object does not exist. (General Error Code 0x16)	Transition to Non-existent	Error: Object State Conflict (General Error Code 0x0C)
Start	Error: Object does not exist. (General Error Code 0x16)	Transition to Running	Error: Object State Conflict (General Error Code 0x0C)
Stop	Error: Object does not exist. (General Error Code 0x16)	Error: Object State Conflict (General Error Code 0x0C)	Transition to Idle
Get_Attribute_Single	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Validate/service the request. Return response
Set_Attribute_Single	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)

3.6.3 Conventions for Type 3

The protocol sequences are described by means of State Machines.

In state diagrams states are represented as boxes state transitions are shown as arrows. Names of states and transitions of the state diagram correspond to the names in the textual listing of the state transitions.

The textual listing of the state transitions is structured as follows:

- The first row contains the name of the transition.
- The second row in capital letters define the current state.
- The third row contains an optional event in bold characters, Conditions start with a “\” as first line character followed by the conditions and Actions start with a “=>” as first line character followed by the action code.
- The last row the next state.

If the event occurs and the conditions are fulfilled the transition fires, i.e. the actions are executed and the next state is entered.

Layout of a Machine description:

TRAN- SITION	CURRENT STATE	Event \ Condition => Action	NEXT STATE

Table 10 - State Machine Description elements

Description element	Meaning
Current State	Name of the given states.
Next State	
Transition	Name of the state transition.
Event	Name or description of the event.
\Condition	Boolean expression. The preceding "\" is not part of the condition.
=> Action	List of assignments and service or function invocations. The preceding ">" is not part of the action.

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"

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

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.

```
If (condition)
```

```
  actions
```

```
else
```

```
  actions
```

```
endif
```

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.

Readers are strongly recommended to refer to the clauses for the AREP and CREP 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.

In addition the following description elements are used:

Wildcard in names

name_XXX:

“XXX” is used as wildcard string for all names beginning with “name”.

The typical use of a wildcard is an Event. In this context there are as many state transitions as possible events for this wildcard exists.

Conditional Macro

<CONDITIONAL-MACRO-NAME>

<

CONDITION1: macrobody1

CONDITION2: macrobody2

...

>

CONDITION1, CONDITION2, etc. define all possible cases for the conditional macro. The “CONDITIONAL-MACRO-NAME” acts as place holder for the “macrocode” depending on the result of the condition.

Replacement Macro

<REPLACEMENT-MACRO-NAME>

<

XXX=	Name1	:	macrobody1
------	-------	---	------------

XXX=	Name2	:	macrobody2
------	-------	---	------------

...

>

Name1, Name2, etc. define all possible cases for the replacement macro. The “REPLACEMENT-MACRO-NAME” acts as place holder for the “macrocode” depending on the value of the current use of the wildcard XXX.

Example:

<SERVICE_REQ_PARA>

<

XXX=Read: Para1, Para2

XXX=Write: Para1, Para2, Para3

>

when called in

MSAB_XXX.req(<SERVICE_REQ_PARA>)

will result in

MSAB_Read.req(Para1, Para2) or MSAB_Write.req(Para1, Para2, Para3)

3.6.4 Conventions for Type 4

There are no Type 4 specific conventions defined.

3.6.5 Conventions for Type 5

There are no Type 5 specific conventions defined.

3.6.6 Conventions for Type 6

There are no Type 6 specific conventions defined.

3.6.7 Conventions for Type 7

This section recalls the main definitions associated with the three specification techniques used : 'Evaluation Networks', 'Abstract objects', 'Service Conventions'.

3.6.7.1 Evaluation networks

To be helpful for the reader some mechanisms are explained by a graphical representation. These representations are based on evaluation network built on an oriented graph with three types of nodes:

- . The states, represented by a disk,
- . The requests represented by a rectangle,
- . The transitions, represented by a horizontal line.

The evaluation network is built according to the following principle:

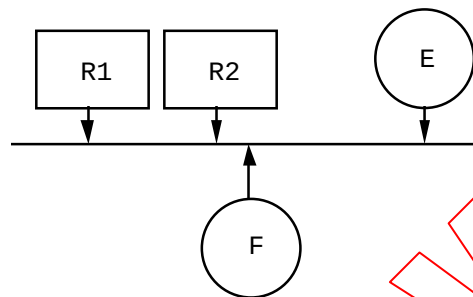


Figure 5: Example of an evaluation net

When the described system is in state E, the arrival of request R1 or R2 results in a transition which switches it state F.

On the other hand, crossing a transition takes place, by convention, in zero time.

3.6.7.2 Simple actions

Simple actions can be associated with each transition, corresponding either to the transmission of requests or to the execution of local actions.

They are materially represented by a triangle, labelled with the name of the request or of the description of the executed action.

3.6.7.3 Conditions

The crossing of a transition can concern a condition. In this case, the oriented graph associated with the representation, shows a condition associated with the arc which precedes the transition.

A corollary of this representation allows defining choices for crossing a transition from the same initial state.

In the same way as for a simple transition, the transmission of an action can be associated with the crossing of a conditional transition.

3.6.8 Conventions for Type 8

There are no Type 8 specific conventions defined.

4 Type 1

4.1 FAL Syntax Description

FalArHeader

```
FalArHeader ::= Unsigned8 {
    -- bit 8      FAL Protocol Specifier      (Always 0 for the protocol defined by this part of IEC 61158-2)
    -- bit 7 - 4  Protocol Identifier          (Identifies abstract syntax, revision, and encoding rules)
    -- bit 3      Protocol Specific bit       (Reserved for each protocol to use)
    -- bit 2-1    PDU Identifier              (Identifies a PDU type within a Protocol Identifier)
}
```

The FalArHeader shall be the first octet of all the FAL PDUs. It enables multiple protocols as well as encoding rules to coexist on the same network. subclause 4.2.1.1 defines the code points for the currently identified combinations of protocol and encoding rules.

4.1.1 FAL-AR PDU Abstract Syntax 1

The productions defined here shall be used with the Compact Encoding Rules, and it is strongly recommended that they be used with the messaging and buffered encoding rules to produce the shortest PDUs that are suitable for a limited bandwidth underlying layer.

```
FalArPDU ::= ConfirmedSend-CommandPDU
            || ConfirmedSend-AffirmativePDU
            || ConfirmedSend-NegativePDU
            || UnconfirmedSend-CommandPDU
            || UnconfirmedAcknowledgedSend-CommandPDU
            || UnconfirmedAcknowledgedSend-AffirmativePDU
            || Idle-CommandPDU
            || AR-XON-OFF-CommandPDU
            || Establish-CommandPDU
            || Establish-AffirmativePDU
            || Establish-NegativePDU
            || Abort-PDU
```

4.1.1.1 Confirmed Send Service

```
ConfirmedSend-CommandPDU ::= SEQUENCE {
    FalArHeader,
    InvokeID,
    ConfirmedServiceRequest
}

ConfirmedSend-AffirmativePDU ::= SEQUENCE {
    FalArHeader,
    InvokeID,
    ConfirmedServiceResponse
}

ConfirmedSend-NegativePDU ::= SEQUENCE {
    FalArHeader,
    InvokeID,
    ConfirmedServiceError
}
```

4.1.1.2 Unconfirmed Send Service

```
UnconfirmedSend-CommandPDU ::= SEQUENCE {
    FalArHeader,
    InvokeID,
    UnconfirmedServiceRequest
}
```

4.1.1.3 Unconfirmed Acknowledged Send Service

```
UnconfirmedAcknowledgedSend-CommandPDU ::= SEQUENCE {
    FalArHeader,
    unconfirmed-PDU [4] IMPLICIT SEQUENCE {
        InvokeID,
        UnconfirmedServiceRequest
    }
}
```

```

UnconfirmedAcknowledgedSend-AffirmativePDU ::= {
    FalArHeader
}

```

4.1.1.4 Idle Send Service

```

IdleSend-CommandPDU ::= {
    FalArHeader
}

```

4.1.1.5 AR-XON-OFF Send Service

```

AR-XON-OFF-CommandPDU ::= SEQUENCE {
    FalArHeader,
    AR-XON-OFFPDU
    Invoke ID,
    XON-OFF
}

```

[7] IMPLICIT SEQUENCE {

4.1.1.6 Establish Service

```

Establish-CommandPDU ::= SEQUENCE {
    FalArHeader,
    RespondingAREP,
    RequestingAREP,
    MAXOSCC,
    MAXOSCS,
    MAXUCSC,
    MAXUCSS,
    CIU,
    InvokeID,
    Initiate-RequestPDU
}

```

```

Establish-AffirmativePDU ::= SEQUENCE {
    FalArHeader,
    InvokeID,
    Initiate-ResponsePDU
}

```

```

Establish-NegativePDU ::= SEQUENCE {
    FalArHeader,
    InvokeID,
    Initiate-ErrorPDU
}

```

4.1.2 FAL-AR PDU Abstract Syntax 2

The productions defined here shall be used with the Traditional Encoding Rules to maintain compatibility with prior standards.

```

FalArPDU ::= ConfirmedSend-RequestPDU
|| ConfirmedSend-ResponsePDU
|| UnconfirmedSend-PDU
|| UnconfirmedAcknowledgedSend-RequestPDU
|| UnconfirmedAcknowledgedSend-ResponsePDU
|| Idle-CommandPDU
|| AR-XON-OFF-CommandPDU
|| Establish-RequestPDU
|| Establish-Request2PDU
|| Establish-ResponsePDU
|| Establish-ErrorPDU
|| Abort-PDU

```

4.1.2.1 Confirmed Send Service

```

ConfirmedSend-RequestPDU ::= SEQUENCE {
    FalArHeader,
    confirmed-RequestPDU
    InvokeID,
    ConfirmedServiceRequest
}

```

[1] IMPLICIT SEQUENCE {

```
ConfirmedSend-ResponsePDU ::= SEQUENCE {
    FalArHeader,
    pduBody CHOICE {
        confirmed-ResponsePDU [2] IMPLICIT SEQUENCE {
            InvokeID,
            ConfirmedServiceResponse
        },
        confirmed-ErrorPDU [3] IMPLICIT SEQUENCE {
            InvokeID,
            ConfirmedServiceError
        }
    }
}
```

4.1.2.2 Unconfirmed Send Service

```
UnconfirmedSend-PDU ::= SEQUENCE {
    FalArHeader,
    unconfirmed-PDU [4] IMPLICIT SEQUENCE {
        InvokeID,
        UnconfirmedServiceRequest
    }
}
```

4.1.2.3 Unconfirmed Acknowledged Send Service

```
UnconfirmedAcknowledgedSend-RequestPDU ::= SEQUENCE {
    FalArHeader,
    unconfirmed-PDU [4] IMPLICIT SEQUENCE {
        InvokeID,
        UnconfirmedServiceRequest
    }
}

UnconfirmedAcknowledgedSend-ResponsePDU ::= {
    FalArHeader
}
```

4.1.2.4 Idle Send Service

```
IdleSend-CommandPDU ::= {
    FalArHeader
}
```

4.1.2.5 AR-XON-OFF Send Service

```
AR-XON-OFF-CommandPDU ::= SEQUENCE {
    FalArHeader,
    AR-XON-OFFPDU [7] IMPLICIT SEQUENCE {
        Invoke ID,
        XON-OFF
    }
}
```

4.1.2.6 Establish Service

```
Establish-RequestPDU ::= SEQUENCE {
    FalArHeader,
    RespondingAREP,
    RequestingAREP,
    userData [6] IMPLICIT SEQUENCE {
        InvokeID,
        InitiateRequest [0] IMPLICIT Initiate-RequestPDU
    }
}
```