

Industrial communication networks — Fieldbus specifications —

Part 6-13: Application layer protocol specification — Type 13 elements

ICS 25.040.40; 35.100.70

National foreword

This British Standard is the UK implementation of EN 61158-6-13:2008. It is identical with IEC 61158-6-13:2007. Together with all of the other sections of BS EN 61158-6, it supersedes BS EN 61158-6:2004 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications — Process measurement and control, including Fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 June 2008

© BSI 2008

ISBN 978 0 580 61633 4

Amendments/corrigenda issued since publication

Date	Comments

English version

**Industrial communication networks -
Fieldbus specifications -
Part 6-13: Application layer protocol specification -
Type 13 elements
(IEC 61158-6-13:2007)**

Réseaux de communication industriels -
Spécifications des bus de terrain -
Partie 6-13: Spécification des services
des couches d'application -
Éléments de type 13
(CEI 61158-6-13:2007)

Industrielle Kommunikationsnetze -
Feldbusse -
Teil 6-13: Protokollspezifikation
des Application Layer
(Anwendungsschicht) -
Typ 13-Elemente
(IEC 61158-6-13:2007)

This European Standard was approved by CENELEC on 2008-02-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

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

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

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

CENELEC

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

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Foreword

The text of document 65C/476/FDIS, future edition 1 of IEC 61158-6-13, prepared by SC 65C, Industrial networks, of IEC TC 65, Industrial-process measurement, control and automation, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61158-6-13 on 2008-02-01.

This and the other parts of the EN 61158-6 series supersede EN 61158-6:2004.

With respect to EN 61158-6:2004 the following changes were made:

- deletion of Type 6 fieldbus for lack of market relevance;
- addition of new fieldbus types;
- partition into multiple parts numbered 6-2, 6-3, ...6-20.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2008-11-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2011-02-01

NOTE Use of some of the associated protocol types is restricted by their intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in type combinations as specified explicitly in the EN 61784 series. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 61158-6-13:2007 was approved by CENELEC as a European Standard without any modification.

CONTENTS

INTRODUCTION.....	6
1 Scope.....	7
1.1 General.....	7
1.2 Specifications.....	7
1.3 Conformance.....	7
2 Normative references	8
3 Terms, definitions, symbols, abbreviations and conventions	8
3.1 ISO/IEC 7498-1 terms	8
3.2 ISO/IEC 8822 terms	9
3.3 ISO/IEC 9545 terms	9
3.4 ISO/IEC 8824 terms	9
3.5 Terms and definitions from IEC 61158-5-13.....	10
3.6 Other terms and definitions	10
4 FAL syntax description	11
4.1 General.....	11
4.2 FAL-AR PDU abstract syntax	11
4.3 Abstract syntax of Asyn2 pduBody	14
5 Transfer syntax	20
5.1 Encoding of data types.....	20
6 FAL protocol state machines	26
7 AP context state machine	27
8 FAL service protocol machine.....	27
9 AR protocol machine	27
9.1 Buffered-network-scheduled bi-directional pre-established connection (BNB-PEC) ARPM	27
9.2 Buffered-network-scheduled uni-directional pre-established connection (BNU-PEC) ARPM.....	29
9.3 Queued user-triggered uni-directional (QUU) ARPM.....	32
9.4 Queued user-triggered bi-directional connectionless (QUB-CL) ARPM	34
9.5 Queued user-triggered bi-directional connection-oriented with segmentation (QUB-COS) ARPM	39
10 DLL mapping protocol machine	57
10.1 Primitive definitions	57
10.2 DMPM state machine	58
Annex A (normative) – Constant value assignments.....	60
A.1 Values of abort-code	60
A.2 NMT-command-ID	61
A.3 Type 13 specific error-code constants	61
A.4 EPL-node-list	63
Annex ZA (normative) Normative references to international publications with their corresponding European publications.....	65
Bibliography.....	64
Figure 1 – Encoding of Time of Day value.....	25
Figure 2 – Encoding of Time Difference value.....	25
Figure 3 – Primitives exchanged between protocol machines.....	26

Figure 4 – State transition diagram of BNB-PEC ARPM	28
Figure 5 – State transition diagram of BNU-PEC ARPM	31
Figure 6 – State transition diagram of QUU ARPM	33
Figure 7 – State transition diagram of QUB-CL ARPM	37
Figure 8 – State transition diagram of QUB-COS (CmdL) ARPM	43
Figure 9 – State transition diagram of QUB-COS (SeqL) ARPM	53
Figure 10 – State transition diagram of DMPM	58
Table 1 – Use of signaling-flags	13
Table 2 – Values of error-type	16
Table 3 – Transfer syntax for bit sequences	21
Table 4 – Transfer syntax for data type UNSIGNEDn	22
Table 5 – Transfer syntax for data type Integern	23
Table 6 – Primitives issued by user to BNB-PEC ARPM	27
Table 7 – Primitives issued by BNB-PEC ARPM to user	27
Table 8 – BNB-PEC ARPM state table – sender transactions	29
Table 9 – BNB-PEC ARPM state table – receiver transactions	29
Table 10 – Function BuildFAL-PDU	29
Table 11 – Primitives issued by user to BNU-PEC ARPM	30
Table 12 – Primitives issued by BNU-PEC ARPM to user	30
Table 13 – BNU-PEC ARPM state table – sender transactions	31
Table 14 – BNU-PEC ARPM state table – receiver transactions	31
Table 15 – Function BuildFAL-PDU	31
Table 16 – Primitives issued by user to QUU ARPM	32
Table 17 – Primitives issued by QUU ARPM to user	32
Table 18 – QUU ARPM state table – sender transactions	33
Table 19 – QUU ARPM state table – receiver transactions	34
Table 20 – Function BuildFAL-PDU	34
Table 21 – Primitives issued by user to QUB-CL ARPM	35
Table 22 – Primitives issued by QUB-CL ARPM to user	36
Table 23 – QUB-CL ARPM state table – sender transactions	38
Table 24 – QUB-CL ARPM state table – receiver transactions	39
Table 25 – Function BuildFAL-PDU	39
Table 26 – Primitives issued by user to QUB-COS (CmdL) ARPM	41
Table 27 – Primitives issued by QUB-COS (CmdL) ARPM to user	42
Table 28 – QUB-COS (CmdL) ARPM state table – sender transactions	44
Table 29 – QUB-COS (CmdL) ARPM state table – receiver transactions	47
Table 30 – Function BuildSegment	49
Table 31 – Function RoundUp	49
Table 32 – Function MoreFollows	49
Table 33 – Function AddSegment	50
Table 34 – Function GetIntermediatePDU	50

Table 35 – Primitives issued by QUB-COS (CmdL) to QUB-COS (SeqL).....	50
Table 36 – Primitives issued by QUB-COS (SeqL) to QUB-COS (CmdL).....	51
Table 37 – Parameters used with primitives exchanged between QUB-COS (SeqL) and QUB-COS (CmdL).....	51
Table 38 – QUB-COS (SeqL) ARPM states	52
Table 39 – QUB-COS (SeqL) ARPM state table – sender transactions.....	54
Table 40 – QUB-COS (SeqL) ARPM state table – receiver transactions	55
Table 41 – Function BuildFAL-PDU.....	56
Table 42 – Function IncrementCounter	57
Table 43 – Function AddToHistoryBuffer.....	57
Table 44 – Primitives issued by ARPM to DMPM	57
Table 45 – Primitives issued by DMPM to ARPM	57
Table 46 – Primitives issued by DMPM to data-link layer	58
Table 47 – Primitives issued by data-link layer to DMPM	58
Table 48 – DMPM state table – sender transactions	59
Table 49 – DMPM state table – receiver transactions.....	59
Table A.1 – Values of abort-code.....	60
Table A.2 – Values of NMTCommandID	61
Table A.3 – Type 13 specific error-code constants.....	62
Table A.4 – EPL-node-list format	63

INTRODUCTION

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

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

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-13: Application layer protocol specification – Type 13 elements

1 Scope

1.1 General

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

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 13 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard specifies interactions between remote applications and defines the externally visible behavior provided by the Type 13 fieldbus application layer in terms of

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

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

- 1) define the wire-representation of the service primitives defined in IEC 61158- 5-13, and
- 2) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 13 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI application layer structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this standard is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-13.

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

1.3 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems.

Conformance is achieved through implementation of this application layer protocol specification.

2 Normative references

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

IEC 61158-3-13, *Industrial communication networks – Fieldbus specifications - Part 3-13: Data-link layer service definition – Type 13 elements*

IEC 61158-4-13, *Industrial communication networks – Fieldbus specifications - Part 4-13: Data-link layer protocol specification – Type 13 elements*

IEC 61158-5-13, *Industrial communication networks – Fieldbus specifications - Part 5-13: Application layer service definition – Type 13 elements*

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

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

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

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

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 ISO/IEC 7498-1 terms

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

3.1.1

application entity

3.1.2

application process

3.1.3

application protocol data unit

3.1.4

application service element

3.1.5

application entity invocation

3.1.6

application transaction

3.1.7

transfer syntax

3.2 ISO/IEC 8822 terms

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

3.2.1 abstract syntax

3.3 ISO/IEC 9545 terms

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

3.3.1 application-context

3.3.2 application-process-type

3.3.3 application-service-element

3.3.4 application control service element

3.4 ISO/IEC 8824 terms

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

3.4.1 any type

3.4.2 bitstring type

3.4.3 boolean type

3.4.4 choice type

3.4.5 false

3.4.6 integer type

3.4.7 module

3.4.8 null type

3.4.9 object identifier

3.4.10 octetstring type

3.4.11 production

3.4.12 simple type

3.4.13 sequence of type

3.4.14 sequence type

3.4.15
structured type

3.4.16
tag

3.4.17
tagged type

3.4.18
true

3.4.19
type

3.5 Terms and definitions from IEC 61158-5-13

3.5.1
application relationship

3.5.2
client

3.5.3
error class

3.5.4
publisher

3.5.5
server

3.5.6
subscriber

3.6 Other terms and definitions

The following terms and definitions are used in this standard:

3.6.1
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

3.6.2
resource
processing or information capability of a subsystem

3.6.3
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

3.6.4
managing node
node that can manage the SCNM mechanism

3.6.5
controlled node
node without the ability to manage the SCNM mechanism

4 FAL syntax description

4.1 General

This description of the Type 13 abstract syntax uses formalisms similar to ASN.1, although the encoding rules differ from that standard.

4.2 FAL-AR PDU abstract syntax

4.2.1 Top level definition

```
APDU ::= CHOICE {
    [3] Isoc1
    [4] Isoc2
    [5] Asyn1
    [6] Asyn2
}
```

4.2.2 Isoc1

```
Isoc1 ::= SEQUENCE {
    message-type
    destination
    source
    reserved
    signaling-flags
    PDO-version
    reserved8
    size
    PDO-payload
}
```

4.2.3 Isoc2

```
Isoc2 ::= SEQUENCE {
    message-type
    destination
    source
    NMT-status
    signaling-flags
    PDO-version
    reserved8
    size
    PDO-payload
}
```

4.2.4 Asyn1

```
Asyn1 ::= SEQUENCE {
    message-type
    destination
    source
    NMT-status
    signaling-flags
    requested-service-ID
    requested-service-target
    EPL-version
}
```

4.2.5 Asyn2

```

Asyn2 ::= SEQUENCE {
    message-type
    destination
    source
    service-ID
    pduBody      CHOICE {
        [1h] ident-response
        [2h] status-response
        [3h] NMT-request
        [4h] NMT-command
        [5h] SDO
        [A0h...FEh] manufacturer-specific
        [FFh] reserved
    }
}

```

4.2.6 Message-type

message-type ::= Unsigned8 — Contains the context specific APDU tags

4.2.7 Addresses

destination ::= Unsigned8 — Node address (1...255)
 source ::= Unsigned8 — Node address (1...255)

4.2.8 Service-ID

service-ID ::= Unsigned8 — Contains the context specific tags for the pduBody

4.2.9 Reserved8

reserved8 ::= Unsigned8

4.2.10 Reserved16

reserved16 ::= Unsigned16

4.2.11 Reserved24

reserved24 ::= Unsigned24

4.2.12 Signaling-flags

```

signaling-flags ::= BitString {
    RD          (0)
    ER          (1)
    EA          (2)
    EC          (3)
    EN          (4)
    MS          (5)
    PS          (6)
    MC          (7)
    RS_bit1     (8)
    RS_bit2     (9)
    RS_bit3     (10)
    PR_bit1     (11)
    PR_bit2     (12)
    PR_bit3     (13)
    reserved    (14)
    reserved    (15)
}

```

The different APDU types use the flags as listed in Table 1. In all cases without "x" the flags are present but not written resp. interpreted.

Table 1 – Use of signaling-flags

	Isoc1	Isoc2	Asyn1	Id.Resp.	St.Resp.
RD	x	x			
ER			x		
EA	x		x		
EC					x
EN		x			x
MS	x	x			
PS					
MC					
RS	x	x		x	x
PR	x	x		x	x

The usage of these flags is specified in IEC 61158-3-13 and IEC 61158-4-13.

4.2.13 PDO-version

PDO-version ::= Unsigned8

— High nibble: main version; low nibble: sub version

4.2.14 Size

size ::= Unsigned8

— Size of PDO payload; max. 1490 octets due to Ethernet restrictions and protocol overhead

4.2.15 PDO-payload

PDO-payload ::= Any

4.2.16 NMT-status

NMT-status ::= CHOICE {
 NMT_GS_OFF Unsigned8 ::= 0000 0000b
 NMT_xS_NOT_ACTIVE Unsigned8 ::= 0001 1100b
 NMT_xS_PRE_OPERATIONAL_1 Unsigned8 ::= 0001 1101b
 NMT_xS_PRE_OPERATIONAL_2 Unsigned8 ::= 0101 1101b
 NMT_xS_READY_TO_OPERATE Unsigned8 ::= 0110 1101b
 NMT_xS_OPERATIONAL Unsigned8 ::= 1111 1101b
 NMT_xS_STOPPED Unsigned8 ::= 0100 1101b
 NMT_xS_BASIC_ETHERNET Unsigned8 ::= 0001 1110b
}

NOTE If sender = MN: "x" := "M"; if sender = CN: "x" := "C"

4.2.17 Requested-service-ID

requested-service-ID ::= CHOICE {
 no-service [0h] IMPLICIT Unsigned8
 ident-request [1h] IMPLICIT Unsigned8
 status-request [2h] IMPLICIT Unsigned8
 NMT-req-inv [3h] IMPLICIT Unsigned8
 manufacturer-specific [A0h]... [FEh] IMPLICIT Unsigned8
 unspecified-invite [FFh] IMPLICIT Unsigned8
}

4.2.18 Requested-service-target

requested-service-target ::= Unsigned8

— Node address (1...255); not assigned (0)

4.2.19 EPL-version

EPL-version ::= Unsigned8

— High nibble: main version; low nibble: sub version

4.3 Abstract syntax of Asyn2 pduBody

4.3.1 Ident-response

4.3.1.1 Overview

```
Ident-response ::= SEQUENCE {
    signaling-flags
    NMT-status
    reserved8
    EPL-version
    reserved8
    feature-flags          BitString          — (see 4.3.1.2)
    MTU                    Unsigned16          — size of the largest possible IP frame incl. header
    poll-in-size           Unsigned16          — actual CN setting for Isoc1 data block size
    poll-out-size          Unsigned16          — actual CN setting for Isoc2 data block size
    response-time          Unsigned32          — time required by the CN to respond to Isoc1
    reserved16
    device-type            Unsigned32          — CN's device type
    vendor-ID              Unsigned32          — CN's vendor ID
    product-code           Unsigned32          — CN's product code
    revision-number        Unsigned32          — CN's revision number
    serial-number          Unsigned32          — CN's serial number
    vendor-specific-extension-1 Unsigned64      — for vendor specific purpose, to be filled with zeros if not used
    verify-configuration-date Unsigned32        — CN's configuration date
    verify-configuration-time Unsigned32        — CN's configuration time
    application-sw-date    Unsigned32          — CN's application software date
    application-sw-time    Unsigned32          — CN's application software time
    IP-address             Unsigned32          — current IP address value of the CN
    subnet-mask            Unsigned32          — current IP subnet mask of the CN
    default-gateway        Unsigned32          — current IP default gateway of the CN
    host-name              VisibleString32     — current DNS host name of the CN
    vendor-specific-extension-2 SEQUENCE SIZE(64) OF Unsigned64
                                     —for vendor specific purpose, to be filled with zeros if not in use
}
```

NOTE Some of the above listed elements are sometimes summarized as follows:

"poll-in-size" through "response-time" are summarized under the term "cycle-timing",
 "device-type" through "serial-number" under "identity",
 "verify-configuration-date" and "verify-configuration-time" under "verify-configuration",
 "application-sw-date" and "application-sw-time" under "application-software-version",
 "vendor-specific-extension-1" and "vendor-specific-extension-2" under "vendor-specific-extensions",
 "IP-address" through "default-gateway" under "IP-address",

4.3.1.2 Feature-flags

```
feature-flags ::= BitString {
    Isochronous              (0) — device may be isochronously accessed via Isoc1
    SDO by UDP/IP            (1) — device supports SDO communication via UDP/IP
    SDO by ASnd              (2) — device supports SDO communication via ASnd
    reserved for future use  (3)
    NMT-info services        (4) — device supports NMT Info Services
    Extended NMT-state-commands (5) — device supports Extended NMT State Commands
    Dynamic PDO mapping      (6) — device supports dynamic PDO Mapping
    NMT services by UDP/IP   (7) — device supports NMT Services by UDP/IP
    Configuration manager    (8) — device supports Configuration Manager functions
    Multiplexed access       (9) — CN device supports multiplexed isochronous access.
    Node-ID setup by SW      (10) — device supports NodeID setup by software
    MN basic ethernet mode   (11) — MN device supports Basic Ethernet Mode
    Routing Type 1 support   (12) — device supports Routing Type 1 functions
    Routing Type 2 support   (13) — device supports Routing Type 2 functions
    reserved_bit1 through bit18 (13)...(31)
}
```

4.3.2 Status-response

4.3.2.1 Overview

```
Status-response ::= SEQUENCE {
    signaling-flags
    NMT-status
    reserved24
    static-error-bit-field
    List-of-errors
}
```

4.3.2.2 Static-error-bit-field

```
static-error-bit-field ::= SEQUENCE {
    error-register          BitString          — (see 4.3.2.3)
    reserved8
    specific-errors        SEQUENCE SIZE (n) OF Unsigned8 — Device profile or vendor specific errors
                                                                n depends on device configuration
}
```

4.3.2.3 Error-register

```
error-register ::= BitString {
    Generic error          (0)
    Current                (1)    OPTIONAL — 0, if no static error present, else 1
    Voltage                (2)    OPTIONAL
    Temperature            (3)    OPTIONAL
    Communication error    (4)    OPTIONAL
    Device profile specific (5)    OPTIONAL
    reserved               (6)    OPTIONAL — always 0
    Manufacturer specific  (7)    OPTIONAL
}
```

4.3.2.4 List-of-errors

```
List-of-errors ::= SEQUENCE SIZE (k) OF ErrorEntry — k >= 2, depends on device configuration
```

4.3.2.5 ErrorEntry

```
ErrorEntry ::= SEQUENCE {
    error-type          Unsigned16 — (see 4.3.2.6)
    error-code          Unsigned16 — (see 4.3.2.6 and A.3)
    time-stamp          Unsigned64
    additional-information Unsigned64
}
```

4.3.2.6 Error-type

The possible values in error-type and their meaning are listed in Table 2.

Table 2 – Values of error-type

Octet	Bit	Value	Description
0 .. 1	15 (status)	0b	ERR_History_ADOM Entry
		1b	Status entry in Status-response frame (Bit 14 shall be set to 0b)
	14 (send)	0b	ERR_History_ADOM only
		1b	Additional to the ERR_History_ADOM the entry shall also be entered in to the emergency queue of the error signaling
	13 .. 12 (mode)	0h	Not allowed in ERR_History_ADOM. Entries with this mode may only be used by the error signaling itself to indicate the termination of the history entries in the Status-response frame
		1h	An error has occurred and is active (e.g. short circuit of output detected)
		2h	An active error was cleared (e.g. no short circuit anymore) (not allowed for status entries)
		3h	An error / event occurred (not allowed for status entries)
	11 .. 0 (profile)	000h	Reserved
		001h	error-code contains a vendor specific error code
		002h	error-code contains Type 13 network specific communication profile errors (see A.3)
		003h .. FFFh	error-code contains device profile specific errors

4.3.3 NMT-request

```

NMT-request ::= SEQUENCE {
    NMT-requested-command-ID      Unsigned8      — value range see A.2
    NMT-requested-command-target  Unsigned8      — target node address
    NMT-requested-command-data    Any
}

```

4.3.4 NMT-command

```

NMT-command ::= SEQUENCE {
    NMT-command-ID CHOICE {
        NMT-state-command      Unsigned8      — value range see A.2
        NMT-info                Unsigned8
    }
    reserved8
    NMT-command-data CHOICE {
        date-time              TimeOfDay
        node-states             SEQUENCE SIZE (255) OF Unsigned8
        EPL-node-list
        NULL
    }
}

```

— only if NMT-command-ID = B0h; see A.4
— if NMT-command-ID = 96h; see A.4; reports each node state individually
— if NMT-command-ID = 40h...95h, A0h; see A.4
— else

4.3.5 SDO

4.3.5.1 Overview

```

SDO ::= SEQUENCE {
    SequenceLayer
    CommandLayer
}

```

4.3.5.2 SequenceLayer

```
SequenceLayer ::= SEQUENCE {
    SequenceFlags    Bitstring {
        rcon_bit1      (0) — 0: no connection; 1: initialisation; 2: connection valid
        rcon_bit2      (1) — 3: error response (retransmission requested)
        rsnr_bit1      (2) — 0..63
        rsnr_bit2      (3)
        rsnr_bit3      (4)
        rsnr_bit4      (5)
        rsnr_bit5      (6)
        rsnr_bit6      (7)
        scon_bit1      (8) — 0: no connection; 1: initialisation; 2: connection valid
        scon_bit2      (9) — 3: connection valid with acknowledge request
        ssnr_bit1     (10) — 0..63
        ssnr_bit2     (11)
        ssnr_bit3     (12)
        ssnr_bit4     (13)
        ssnr_bit5     (14)
        ssnr_bit6     (15)
    }
    reserved16
}
```

4.3.5.3 CommandLayer

```
CommandLayer ::= SEQUENCE {
    SDOCommandHeader    SEQUENCE {
        reserved8
        invoke-ID
        segmentation_abort_response
        command-ID
        segment-size
        reserved16
    }
    SDO-command    CHOICE {
        write-by-index-request      [1h] IMPLICIT WriteByIndex-RequestPDU
        read-by-index-request       [2h] IMPLICIT ReadByIndex-RequestPDU
        write-all-by-index-request [3h] IMPLICIT WriteAllByIndex-RequestPDU
        read-all-by-index-request  [4h] IMPLICIT ReadAllByIndex-RequestPDU
        write-multiple-by-index-request [31h] IMPLICIT WriteMultipleByIndex-RequestPDU
        read-multiple-by-index-request [32h] IMPLICIT ReadMultipleByIndex-RequestPDU
        abort                       IMPLICIT AbortPDU
        write-by-index-response     [1h] IMPLICIT WriteByIndex-ResponsePDU
        read-by-index-response      [2h] IMPLICIT ReadByIndex-ResponsePDU
        write-all-by-index-response [3h] IMPLICIT WriteAllByIndex-ResponsePDU
        read-all-by-index-response [4h] IMPLICIT ReadAllByIndex-ResponsePDU
        write-multiple-by-index-response [31h] IMPLICIT WriteMultipleByIndex-ResponsePDU
        read-multiple-by-index-response [32h] IMPLICIT ReadMultipleByIndex-ResponsePDU
    }
}
```

4.3.5.4 Invoke ID

invoke-ID ::= Unsigned8

4.3.5.5 Segmentation_abort_response

```
segmentation_abort_response ::= BitString {
    reserved      (0)
    reserved      (1)
    reserved      (2)
    reserved      (3)
    segmentation_bit1 (4) — 0: Expedited transfer; 1: initiate segment transfer
    segmentation_bit2 (5) — 2: segment; 3: segment transfer complete
    abort          (6) — 0: transfer ok; 1: abort
    response       (7) — 0: request; 1: response
}
```

4.3.5.6 Command-ID

command-ID ::= Unsigned8

— Contains context specific tag for SDO-command

4.3.5.7 Segment-size

segment-size ::= Unsigned16

— Length of segment data. Counting from the beginning of the "SDO-command". Valid value range: 0...1458

4.3.5.8 WriteByIndex services

WriteByIndex-RequestPDU ::= SEQUENCE {
 data-size — only present if in SDOCommandHeader segmentation = "initiate"
 index
 sub-index
 reserved8
 payload-data
}

WriteByIndex-ResponsePDU ::= NULL

4.3.5.9 ReadByIndex services

ReadByIndex-RequestPDU ::= SEQUENCE {
 index
 sub-index
 reserved16
}

ReadByIndex-ResponsePDU ::= SEQUENCE {
 data-size — only present if in SDOCommandHeader segmentation = "initiate"
 payload-data
}

4.3.5.10 WriteAllByIndex services

WriteAllByIndex-RequestPDU ::= SEQUENCE {
 data-size — only present if in SDOCommandHeader segmentation = "initiate"
 index
 reserved16
 payload-data
}

WriteAllByIndex-ResponsePDU ::= NULL

4.3.5.11 ReadAllByIndex services

ReadAllByIndex-RequestPDU ::= SEQUENCE {
 index
 reserved16
}

ReadAllByIndex-ResponsePDU ::= SEQUENCE {
 data-size — only present if in SDOCommandHeader segmentation = "initiate"
 payload-data
}

WriteByName-ResponsePDU ::= NULL

4.3.5.12 WriteMultipleByIndex services

```
WriteMultipleByIndex-RequestPDU ::= SEQUENCE {
    data-size                — only present if in SDOCommandHeader segmentation = "initiate"
    offset (k)               — Byte offset of next payload data set
    index
    sub-index
    padding-length
    payload-data
    offset (m)               — Byte offset of next payload data set
    index
    sub-index
    padding-length
    payload-data
    ...                      — (further write requests)
}
```

```
WriteMultipleByIndex-ResponsePDU ::= SEQUENCE {
    data-size                — only present if in SDOCommandHeader segmentation = "initiate"
    index
    sub-index
    sub-abort
    sub-abort-code
    index
    sub-index
    sub-abort
    sub-abort-code
    ...                      — (further write responses)
}
```

4.3.5.13 ReadMultipleByIndex services

```
ReadMultipleByIndex-RequestPDU ::= SEQUENCE {
    data-size                — only present if in SDOCommandHeader segmentation = "initiate"
    index
    sub-index
    reserved8
    index
    sub-index
    reserved8
    ...                      — (further read requests)
}
```

```
ReadMultipleByIndex-ResponsePDU ::= SEQUENCE {
    data-size                — only present if in SDOCommandHeader segmentation = "initiate"
    offset (k)               — offset of the next payload data set
    index
    sub-index
    sub-abort -padding-length
    payload-data / sub-abort-code
    offset (m)               — offset of the next payload data set
    index
    sub-index
    sub-abort-padding-length
    payload-data / sub-abort-code
    ...                      — (further read requests)
}
```

4.3.5.14 AbortPDU

```
AbortPDU ::= {
    abort-code    Unsigned32    — see A.1 for valid values
}
```

4.3.5.15 Data size

data-size ::= Unsigned32 — Length of transferred data block. Counting from the beginning of the "SDO-command". Valid value range: $0 \dots 2^{32}-1$.