EtherCAT Specification – Part 5

Application Layer service definition

Document: ETG.1000.5 S (R) V1.0.4

Nomenclature:

ETG-Number	ETG.1000.5
Type	S (Standard)
State	R (Release)
Version	V1.0.4

Created by: ETG
Contact: info@ethercat.org
Date 15.09.2017

EtherCAT Technology Group

Trademarks and Patents

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Disclaimer

The documentation has been prepared with care. The technology described is, however, constantly under development. For that reason the documentation is not in every case checked for consistency with performance data, standards or other characteristics. In the event that it contains technical or editorial errors, we retain the right to make alterations at any time and without warning. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Copyright

DOCUMENT HISTORY

Version	Comment
1.0	Adopted from IEC-Standard 61158-5 Type 12 for ETG use only!
1.0.1	Corrections and clarifications according to ETG1000_ES_D_V0i6_EcatSpecErrata.pdf
1.0.2	Corrections and clarifications according to ETG1000_V1i0i2_ES_R_V0i7_EcatSpecErrata.doc and IEC SC65C MT 9 editorial comments
1.0.3	Corrections and clarifications according to ETG1000_V1i0i2_ES_R_V0i8_EcatSpecErrata.doc
1.0.4	Corrections and clarifications according to ETG1000_V1i0i3_ES_R_V0i9_EcatSpecErrata.doc

CONTENTS

Figures

Tables

1 Scope

1.1 Scope of this standard and accordance to IEC Standards

The ETG.1000 series specifies the EtherCAT Technology within the EtherCAT Technology Group. It is divided into the following parts:

- ETG.1000.2: Physical Layer service definition and protocol specification
- ETG.1000.3: Data Link Layer service definition
- ETG.1000.4: Data Link Layer protocol specification
- ETG.1000.5: Application Layer service definition
- ETG.1000.6: Application Layer protocol specification

These parts are based on the corresponding parts of the IEC 61158 series Type 12. EtherCAT is named Type 12 in IEC 61158 to avoid the usage of brand names.

1.2 Overview

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 EtherCAT fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard defines in an abstract way the externally visible service provided by the different Types of the fieldbus Application Layer in terms of

a) an abstract model for defining application resources (objects) capable of being manipulated by users via the use of the FAL service,
b) the primitive actions and events of the service;
c) the parameters associated with each primitive action and event, and the form which they take; and
d) the interrelationship between these actions and events, and their valid sequences.

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

1) the FAL user at the boundary between the user and the Application Layer of the Fieldbus Reference Model, and
2) Systems Management at the boundary between the Application Layer and Systems Management of the Fieldbus Reference Model.

This standard specifies the structure and services of the IEC fieldbus Application Layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI Application Layer Structure (ISO/IEC 9545).

FAL services and protocols are provided by FAL application-entities (AE) contained within the application processes. The FAL AE is composed of a set of object-oriented Application Service Elements (ASEs) and a Layer Management Entity (LME) that manages the AE. The ASEs provide communication services that operate on a set of related application process object (APO) classes. One of the FAL ASEs is a management ASE that provides a common set of services for the management of the instances of FAL classes.

Although these services specify, from the perspective of applications, how request and responses are issued and delivered, they do not include a specification of what the requesting and responding applications are to do with them. That is, the behavioral aspects of the

applications are not specified; only a definition of what requests and responses they can send/receive is specified. This permits greater flexibility to the FAL users in standardizing such object behavior. In addition to these services, some supporting services are also defined in this standard to provide access to the FAL to control certain aspects of its operation.

1.3 Specifications

The principal objective of this standard is to specify the characteristics of conceptual application layer services suitable for time-critical communications, and thus supplement the OSI Basic Reference Model in guiding the development of application layer protocols for time-critical communications.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols.

This specification may be used as the basis for formal Application Programming-Interfaces. Nevertheless, it is not a formal programming interface, and any such interface will need to address implementation issues not covered by this specification, including

a) the sizes and octet ordering of various multi-octet service parameters, and

b) the correlation of paired request and confirm, or indication and response, primitives.

1.4 Conformance

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

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through implementation of conforming application layer protocols that fulfill any given Type of application layer services as defined in this standard.

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 61131-3, *Programmable controllers – Programming languages*

IEC/TR 61158-1:2010[1] , *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-3-12, *Industrial communication networks – Fieldbus specifications - Part 3-12: data-link layer service definition – Type 12 elements*

ISO/IEC 646:1991, *Information technology – ISO 7-bit coded character set for information interchange*

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

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

ISO/IEC 8802-3, *Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 3: Standard for Ethernet*

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

[1] To be published

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

IEEE 802.1D, *IEEE standard for local and metropolitan area networks – Common specifications – Media access control (MAC) Bridges;* available at <http://www.ieee.org>

IETF RFC 791, *Internet Protocol darpa internet program protocol specification*; available at <http://www.ietf.org>

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Reference model terms and definitions

This standard is based in part on the concepts developed in ISO/IEC 7498-1 and ISO/IEC 7498-3, and makes use of the following terms defined therein:

3.1.1 correspondent (N)-entities [7498-1]
correspondent AL-entities (N=7)

3.1.2 (N)-entity [7498-1]
AL-entity (N=7)

3.1.3 (N)-layer [7498-1]
AL-layer (N=7)

3.1.4 layer-management [7498-1]

3.1.5 peer-entities [7498-1]

3.1.6 primitive name [7498-3]

3.1.7 AL-protocol [7498-1]

3.1.8 AL-protocol-data-unit [7498-1]

3.1.9 reset [7498-1]

3.1.10 routing [7498-1]

3.1.11 segmenting [7498-1]

3.1.12 (N)-service [7498-1]
AL-service (N=7)

3.1.13 AL-service-data-unit [7498-1]

3.1.14 AL-simplex-transmission [7498-1]

3.1.15 AL-subsystem [7498-1]

3.1.16 systems-management [7498-1]

3.1.17 AL-user-data [7498-1]

3.2 Service convention terms and definitions

This standard also makes use of the following terms defined in ISO/IEC 10731 as they apply to the data-link layer:

3.2.1 acceptor

3.2.2 asymmetrical service

**3.2.3 confirm (primitive);
requestor.deliver (primitive)**

3.2.4 deliver (primitive)

**3.2.5 AL-service-primitive;
primitive**

3.2.6 AL-service-provider

3.2.7 AL-service-user

**3.2.8 indication (primitive);
acceptor.deliver (primitive)**

**3.2.9 request (primitive);
requestor.submit (primitive)**

3.2.10 requestor

**3.2.11 response (primitive);
acceptor.submit (primitive)**

3.2.12 submit (primitive)

3.2.13 symmetrical service

3.3 Application layer and data-link service terms and definitions

3.3.1
application
function or data structure for which data is consumed or produced

3.3.2
application objects
multiple object classes that manage and provide a run time exchange of messages across the network and within the network device]

3.3.3
basic slave
slave device that supports only physical addressing of data

3.3.4
bit
unit of information consisting of a 1 or a 0. This is the smallest data unit that can be transmitted

3.3.5
client
1) object which uses the services of another (server) object to perform a task

2) initiator of a message to which a server reacts

3.3.6
communication object
component that manage and provide a run time exchange of messages across the network

3.3.7
connection
logical binding between two application objects within the same or different devices

3.3.8
cyclic
events which repeat in a regular and repetitive manner

3.3.9
data
generic term used to refer to any information carried over a fieldbus

3.3.10
data consistency
means for coherent transmission and access of the input- or output-data object between and within client and server

3.3.11
data type
relation between values and encoding for data of that type. For this specification the data type definitions of IEC 61131-3 apply.

3.3.12
data type object
Entry in the object dictionary indicating a data type

3.3.13
default gateway
device with at least two interfaces in two different IP subnets acting as router for a subnet.

3.3.14
device
physical entity connected to the fieldbus composed of at least one communication element (the network element) and which may have a control element and/or a final element (transducer, actuator, etc.)

3.3.15
device profile
collection of device dependent information and functionality providing consistency between similar devices of the same device

3.3.16
diagnosis information
all data available at the server for maintenance purposes

3.3.17
distributed clocks
method to synchronize slaves and maintain a global time base

3.3.18
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.3.19
error class
general grouping for related error definitions and corresponding error codes

3.3.20
error code
identification of a specific type of error within an error class

3.3.21
event
instance of a change of conditions

3.3.22
fieldbus memory management unit
function that establishes one or several correspondences between logical addresses and physical memory

3.3.23
fieldbus memory management unit entity
single element of the fieldbus memory management unit: one correspondence between a coherent logical address space and a coherent physical memory location

3.3.24
frame
denigrated synonym for DLPDU

3.3.25
full slave
slave device that supports both physical and logical addressing of data

3.3.26
index
address of an object within an application process

3.3.27
interface
shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

3.3.28
little endian
Method for data representation of numbers greater 8 bit where the least significant octet is transmitted first.

3.3.29
master
device that controls the data transfer on the network and initiates the media access of the slaves by sending messages and that constitutes the interface to the control system

3.3.30
mapping
correspondence between two objects in that way that one object is part of the other object

3.3.31
mapping parameters
set of values defining the correspondence between application objects and process data objects

3.3.32
medium
cable, optical fibre, or other means by which communication signals are transmitted between two or more points

NOTE "media" is used as the plural of medium.

3.3.33
message
ordered series of octets intended to convey information

NOTE Normally used to convey information between peers at the application layer.

3.3.34
network
set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.3.35
node
a) single DL-entity as it appears on one local link

b) end-point of a link in a network or a point at which two or more links meet [derived from IEC 61158-2]

3.3.36
object
abstract representation of a particular component within a device

NOTE An object can be

1) an abstract representation of the capabilities of a device. Objects can be composed of any or all of the following components:

a) data (information which changes with time);

b) configuration (parameters for behavior);

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 behavior.

3.3.37
object dictionary
data structure addressed by Index and Sub-index that contains descriptions of data type objects, communication objects and application objects

3.3.38
process data
collection of application objects designated to be transferred cyclically or acyclically for the purpose of measurement and control

3.3.39
process data object
structure described by mapping parameters containing one or several process data entities

3.3.40
segment
collection of one real master with one or more slaves

3.3.41
server
object which provides services to another (client) object

3.3.42
service
operation or function that an object and/or object class performs upon request from another object and/or object class

3.3.43
slave
DL-entity accessing the medium only after being initiated by the preceding slave or the master

3.3.44
sub-index
subaddress of an object within the object dictionary

3.3.45
Sync Manager
collection of control elements to coordinate access to concurrently used objects.

3.3.46
Sync Manager channel
single control elements to coordinate access to concurrently used objects.

3.3.47
switch
MAC bridge as defined in IEEE 802.1D

3.4 Common symbols and abbreviations

AoE	Automation Device Specification (ADS) over EtherCAT

AL-	Application layer (as a prefix)
ALE	AL-entity (the local active instance of the application layer)
AL	AL-layer
APDU	AL-protocol-data-unit
ALM	AL-management
ALME	AL-management Entity (the local active instance of AL-management)
ALMS	AL-management service
ALS	AL-service
AR	Application relationship
ASE	Application service element
CAN	Controller Area Network
CiA	CAN in Automation
CoE	CAN application protocol over EtherCAT services
CSMA/CD	Carrier sense multiple access with collision detection
DC	Distributed clocks
DL	Data-link-layer
DNS	Domain name system (server for name resolution in IP networks)
E²PROM	Electrically erasable programmable read only memory
EoE	Ethernet tunneled over EtherCAT services
ESC	EtherCAT slave controller
FCS	Frame check sequence
FIFO	First-in first-out (queuing method)
FMMU	Fieldbus memory management unit
FoE	File access with EtherCAT services
HDR	Header
ID	Identifier
IETF	Internet engineering task force
IP	Internet protocol
LAN	Local area network
MAC	Medium access control
OD	Object dictionary
OSI	Open systems interconnection
PDI	Physical device internal interface (a set of elements that allows access to DL-services from the AL)
PDO	Process data object
PhL	Ph-layer
QoS	Quality of service
RAM	Random access memory
Rx	Receive
SDO	Service data object
SII	slave information interface
SM	Synchronization manager
SyncM	Synchronization manager
TCP	Transmission control protocol
Tx	Transmit
UDP	User datagram protocol
WKC	Working counter

3.5 Conventions

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

The service model, service primitives, and time-sequence diagrams used are entirely abstract descriptions; they do not represent a specification for implementation.

Service primitives, used to represent service user/service provider interactions (see ISO/IEC 10731), convey parameters that indicate information available in the user/provider interaction.

This standard uses a tabular format to describe the component parameters of the service primitives. The parameters that apply to each group of service primitives are set out in tables throughout the remainder of this standard. Each table consists of up to five columns, containing the name of the service parameter, and a column each for those primitives and parameter-transfer directions used by the service:

— the request primitive's input parameters;

— the indication primitive's output parameters;

— the response primitive's input parameters; and

— the confirm primitive's output parameters.

NOTE The request, indication, response and confirm primitives are also known as requestor.submit, acceptor.deliver, acceptor.submit, and requestor.deliver primitives, respectively (see ISO/IEC 10731).

One parameter (or part of it) is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive and parameter direction specified in the column:

M parameter is mandatory for the primitive.

U parameter is a User option, and may or may not be provided depending on the dynamic usage of the service-user. When not provided, a default value for the parameter is assumed.

C parameter is conditional upon other parameters or upon the environment of the service-user.

(blank) parameter is never present.

Some entries are further qualified by items in brackets. These may be a parameter-specific constraint:

(=) indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.

In any particular interface, not all parameters need be explicitly stated. Some may be implicitly associated with the primitive.

In the diagrams which illustrate these interfaces, dashed lines indicate cause-and-effect or time-sequence relationships, and wavy lines indicate that events are roughly contemporaneous.

4 Concepts

4.1 Common concepts

All of IEC/TR 61158-1, Clause 9 is incorporated by reference, except as specifically overridden in 4.2.

4.2 Type specific concepts

4.2.1 Operating principle

This standard and its companion EtherCAT standards describe a real-time Ethernet technology that aims to maximize the utilization of the full duplex Ethernet bandwidth. Medium access control employs the master/slave principle, where the master node (typically the control system) sends the Ethernet frames to the slave nodes, which extract data from and insert data into these frames.

From an Ethernet point of view, a EtherCAT segment is a single Ethernet device, which receives and sends standard ISO/IEC 8802-3 Ethernet frames. However, this Ethernet device is not limited to a single Ethernet controller with downstream microprocessor, but may consist of a large number of EtherCAT slave devices. These process the incoming frames directly and extract the relevant user data, or insert data and transfer the frame to the next slave device. The last slave device within the segment sends the fully processed frame back, so that it is returned by the first slave device to the master as response frame.

This procedure utilizes the full duplex capability of Ethernet: both communication directions are operated independently. Communication without switch between a master device and a EtherCAT segment consisting of one or several slave devices may be established.

Industrial communication systems have to meet different requirements in terms of the data transmission characteristics. Parameter data is transferred acyclically and in large quantities, whereby the timing requirements are relatively non-critical, and the transmission is usually triggered by the control system. Diagnostic data is also transferred acyclically and event-driven, but the timing requirements are more demanding, and the transmission is usually triggered by a peripheral device.

Process data, on the other hand, is typically transferred cyclically with different cycle times. The timing requirements are most stringent for process data communication. EtherCAT AL supports a variety of services and protocols to meet these differing requirements.

4.2.2 Communication model overview

The EtherCAT application layer distinguishes between master and slave. The communication relationship is always initiated by the master.

A EtherCAT segment consists of at least one master device and one or many slave devices. All slave devices support the EtherCAT State Machine (ESM) and support the transmission of EtherCAT process data.

The application relationship can be modeled independent of communication relationship. The master-slave relationship is the standard application relationship.

4.2.3 Application layer element description

4.2.3.1 Management

The mandatory management consists of a set of object to control the state of a slave. An interface to DL provides read access to all DL registers.

4.2.3.2 Information interface

The mandatory slave information interface (SII) consists of all objects that can be stored persistently.

4.2.3.3 Synchronization support

The optional support of isochronous operation consists of several attributes for synchronization and timestamping of binary signals.

4.2.3.4 Access to slave

The real time entity consists of an interface for network triggered exchange of data and an interface for user triggered access to slave objects. Objects mainly used for network triggered access are called PDO. SDO are the objects that are used for user triggered access.

The access methods for PDO are read and write to a buffer. The communication structure is a producer-consumer relationship as shown in Figure **1** with no direct acknowledgement of the delivery of data. The delivery of data will be monitored by additional elements in the slave and in the master (i.e. watchdog and working counter). The producer can be a master as well as a slave.

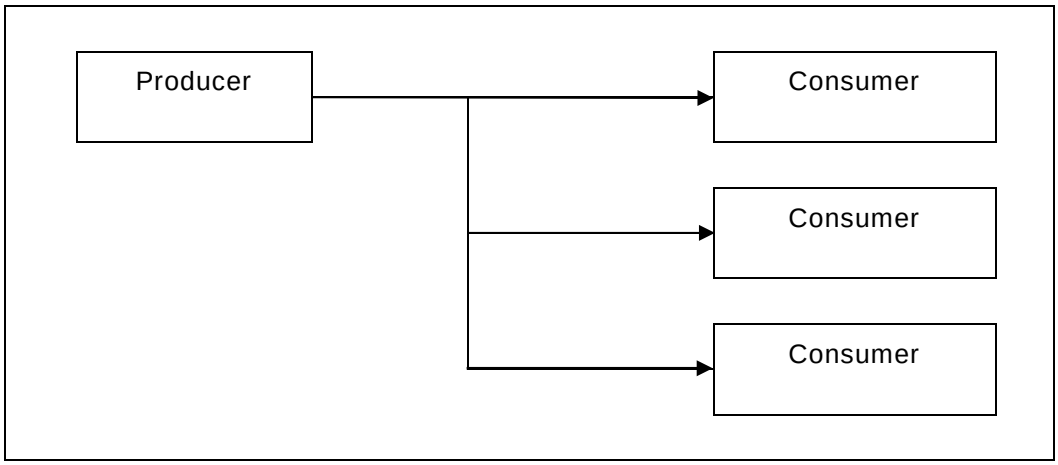

Figure 1 **– Producer consumer model**

The access to SDO follows the client server principle. The client issues a service invocation to the server. The slave starts the service execution and replies the result afterwards. There is always a response needed to conclude this type of service.

Figure **2** shows the workflow of this communication interaction.

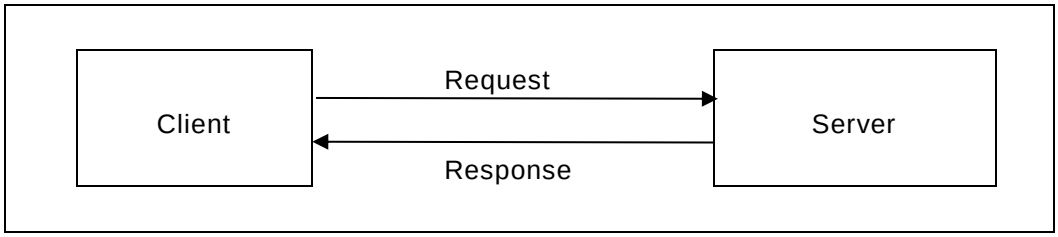

Figure 2 **– Client server model**

There may be an unsolicited type of server to client interaction as well. This model is used to convey data server triggered. This type of service called notification is shown in Figure **3**.

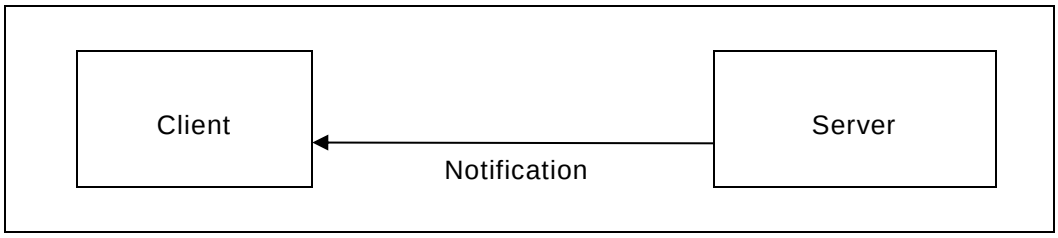

Figure 3 **– Server triggered invocation**

4.2.3.5 TCP/UDP/IP suite

The optional protocol is dedicated for slaves who will use the standard internet protocol suite. The protocols itself are defined in IETF RFC 791 and 768. EoE describes the mapping of the IP-Protocol (and similar kind of communication) to EtherCAT data-link layer. IP is a connectionless communication type with bidirectional data flow.

4.2.3.6 File access

The primary use of the file transfer is the download and upload of program files and configuration data. The file access is done with client server protocol architecture.

4.2.4 Slave reference model

4.2.4.1 Mapping onto OSI basic reference model

EtherCAT is described using the principles, methodology and model of ISO/IEC 7498 Information processing systems — Open Systems Interconnection — Basic Reference Model (OSI). The OSI model provides a layered approach to communications standards, whereby the layers can be developed and modified independently. The EtherCAT specification defines functionality from top to bottom of a full OSI stack, and some functions for the users of the stack. Functions of the intermediate OSI layers, layers 3 – 6, are consolidated into either the EtherCAT data-link layer or the EtherCAT application layer. Likewise, features common to users of the Fieldbus application layer may be provided by the EtherCAT application layer to simplify user operation, as noted in Figure 4.

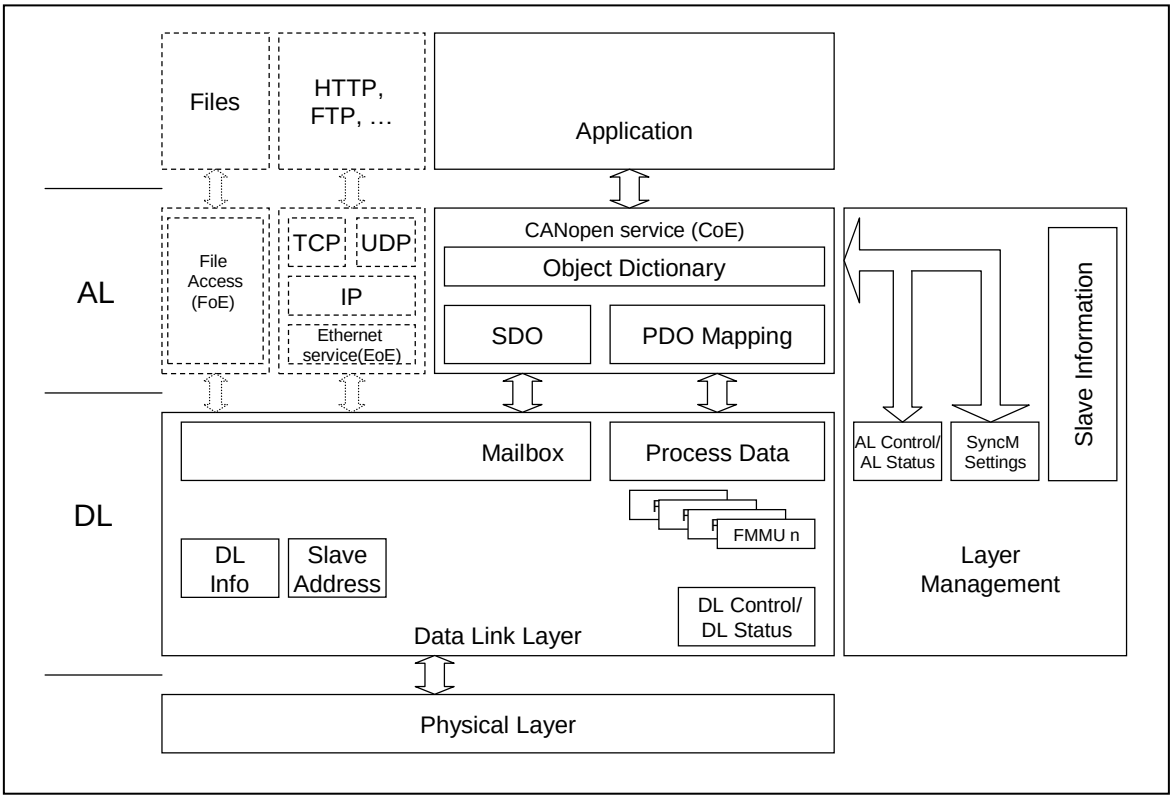

Figure 4 – Slave reference model

4.2.4.2 Data-link layer features

The data-link layer provides basic time critical support for data communications among devices connected via a EtherCAT segment. The term “time-critical” is used to describe applications having 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.

The data-link layer has the task to compute, compare and generate the frame check sequence and provide communications by extracting data from and/or including data into the Ethernet frame. This is done depending on the data-link layer parameters which are stored at pre-defined memory locations. The application data is made available to the application layer in physical memory, either in a mailbox configuration or within the process data section.

Additionally some data structures in the data-link layer will be used to allow a coordination of the interaction between Master and slave such as AL Control, Status and Event and Sync Manager settings.

4.2.4.3 Slave AL classification

4.2.4.3.1 Simple slave device

From the application layer point of view, slave devices are classified in simple devices without an application controller and complex devices with an application controller.

NOTE: The DL slave classification in basic slaves and full slaves is independent of the AL view, since DL addressing mechanisms are invisible at the AL interface.

Simple devices have a fixed process data layout, which is described in the device description file. Simple devices may confirm the AL Management services without a reaction within the local application, as noted in Figure 5. There is no special reaction needed for safe state operation (e.g. the value 0 will be processed in the same way as no valid value will be send).

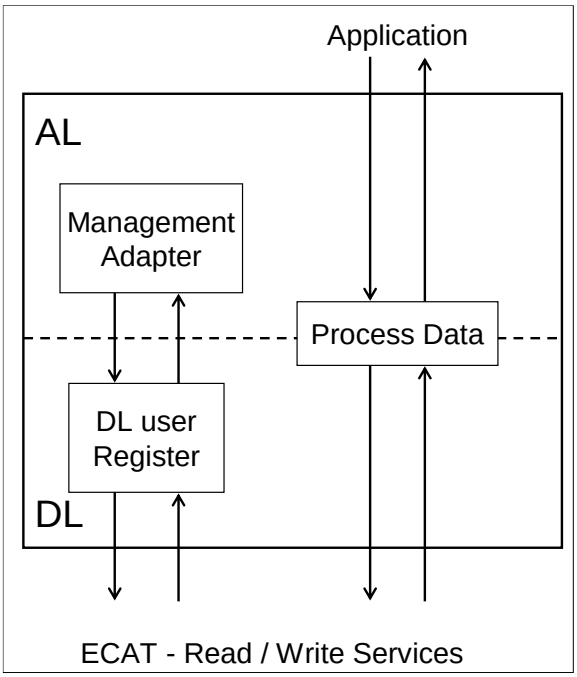

Figure 5 – Simple slave device

4.2.4.3.2 Complex slave device

As noted in Figure 6, complex devices support

- the ESM
- the Mailbox (optional),
- the CoE object dictionary (recommended if mailbox is supported),
- the SDO services to read and/or write the object dictionary data entries (recommended if mailbox is supported), and
- the SDO information service to read the defined objects in the object dictionary and each entry description in compact format (recommended if mailbox is supported).

For the process data transmission the PDO mapping objects and the Sync Manager PDO assign objects, which describe the process data layout, must be supported for reading. If an complex device supports configurable process data, the configuration is done by writing the PDO mapping and/or the Sync Manager PDO assign objects.

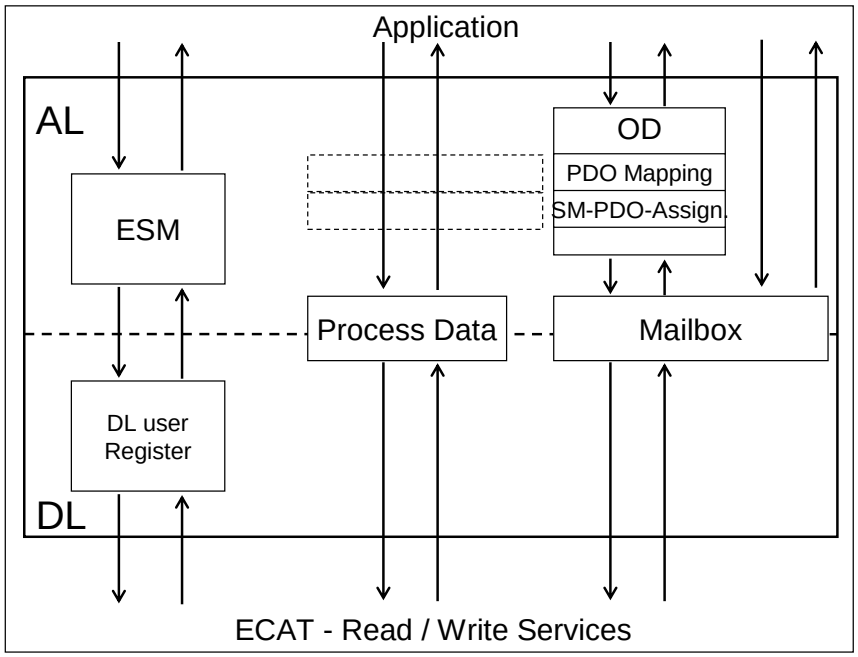

Figure 6 – Complex slave device

There are different interaction types defined in this standard:

- CAN application protocol over EtherCAT services (CoE)
- Ethernet over EtherCAT services (EoE)
- File Access over EtherCAT services (FoE)

The different types are used to address different classes of objects. These types can be mixed at a single application relationship.

4.2.5 Master reference model

4.2.5.1 Overview

The master communicates with the slaves by using the services described in the slave chapter. Additionally there is a slave Handler for each slave defined in the master to control the ESM of the slave and a Router which enables slave-to-slave communication via the mailbox, as noted in Figure 7.

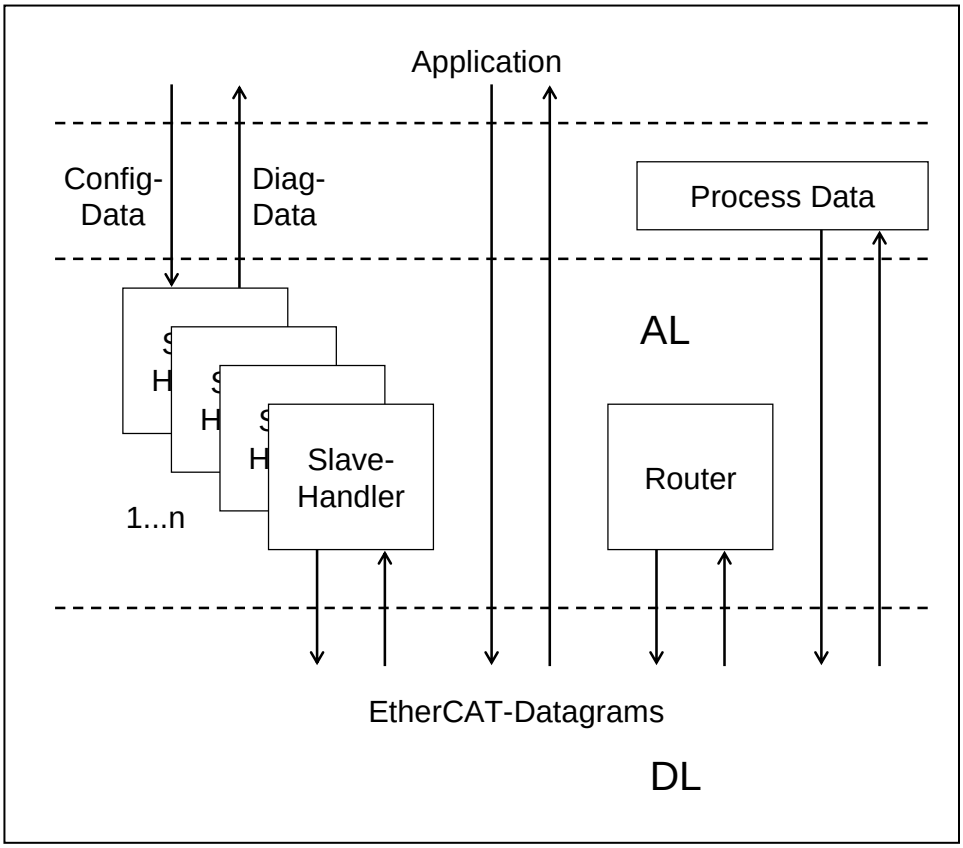

Figure 7 – Master functional overview

4.2.5.2 Slave handler

The master should support a slave handler for each slave using the state services to control the ESM of the slave. The slave Handler is the image of the slave's ESM in the master. Additionally the slave Handler may send SDO services before changing the state of the slave's ESM

Parameter

Position

This parameter specifies the position in the logical ring which is used to address the slave when reading the Identification and writing the Station Address. It is mandatory for all slaves.

Expected Identification

This parameter specifies the expected identification of the slave, which should be read and compared by the master in the Init state. This parameter is mandatory for all slaves.

Station Address

This parameter specifies the station address which is assigned in the Init state to the slave. All further services will use this station address to address the slave. This parameter is mandatory for all slaves.

Mailbox Configuration

This parameter specifies the configuration of the Sync Manager channels 0 and 1 for the mailbox which is written in the Init state to the slave. This parameter is mandatory for complex slaves.

FMMU Configuration

This parameter specifies the configuration of the FMMU channels which is written in the Pre-Operational state to the slave.

Process Data Configuration

This parameter specifies the configuration of the Sync Manager channels which is used for process data and is written in the Pre-Operational state to the slave.

PDO mapping

This parameter specifies the PDO mapping objects which may be written in the Pre-Operational state to the slave.

Sync Manager PDO assign

This parameter specifies the Sync Manager PDO assign objects which may be written in the Pre-Operational state to the slave.

Start Up Objects

This parameter specifies the objects from the object dictionary of the slave which may be written to the slave from the slave Handler during start up.

4.2.5.3 Router

The Router can be used for several applications:

- routing mailbox services from the client slave to the server slave
- routing mailbox service responses from the server slave to the client slave
- forwarding mailbox services from third party devices
- forwarding mailbox service responses to third party devices

The task of the router is to overwrite the address field of the mailbox service with the station address of the client or with a virtual address before routing the mailbox service to the server addressed by the original address field. The address field of the mailbox service response is overwritten by the router with the station address of the server before routing the mailbox service response to the client slave addressed by the original address field or to the corresponding IP Address/ MAC Address in case of a virtual address.

5 Data type ASE

5.1 General

All of IEC/TR 61158-1, 5.1 is incorporated by reference.

5.2 Formal definition of data type objects

All of IEC/TR 61158-1, 5.2 is incorporated by reference.

5.3 FAL defined data types

5.3.1 Fixed length types

5.3.1.1 Boolean types

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	1
2	Data type Name	=	Boolean
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This data type expresses a Boolean data type with the values TRUE and FALSE.

5.3.1.2 Bitstring types

5.3.1.2.1 BIT2

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	31
2	Data type Name	=	BIT2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

A BIT2 is an ordered sequence of Boolean data types, numbered from 1 to 2.

5.3.1.2.2 BIT3

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	32
2	Data type Name	=	BIT3
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

A BIT3 is an ordered sequence of Boolean data types, numbered from 1 to 3.

5.3.1.2.3 BIT4

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	33
2	Data type Name	=	BIT3
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

A BIT4 is an ordered sequence of Boolean data types, numbered from 1 to 4.

5.3.1.2.4 BIT5

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	34
2	Data type Name	=	BIT5
3	Format	=	FIXED LENGTH

4.1	Octet Length	=	1

A BIT5 is an ordered sequence of Boolean data types, numbered from 1 to 5.

5.3.1.2.5 BIT6

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	35
2	Data type Name	=	BIT6
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

A BIT6 is an ordered sequence of Boolean data types, numbered from 1 to 6.

5.3.1.2.6 BIT7

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	36
2	Data type Name	=	BIT7
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

A BIT7 is an ordered sequence of Boolean data types, numbered from 1 to 7.

5.3.1.2.7 BIT8

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	37
2	Data type Name	=	BIT8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

A BIT8 is an ordered sequence of Boolean data types, numbered from 1 to 8.

5.3.1.2.8 BITARR8

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	45
2	Data type Name	=	BITARR8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

A BITARR8 is an ordered sequence of bits, numbered from 1 to 8.

5.3.1.2.9 BITARR16

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	46
2	Data type Name	=	BITARR16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

A BITARR16 is an ordered sequence of bits, numbered from 1 to 15.

5.3.1.2.10 BITARR32

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	47
2	Data type Name	=	BITARR32

3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

A BITARR32 is an ordered sequence of bits, numbered from 1 to 31.

5.3.1.3 Currency types

There are no Currency types defined for EtherCAT.

5.3.1.4 Date/Time types

5.3.1.4.1 TimeOfDay

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	12
2	Data type Name	=	TimeOfDay
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

This data type is composed of two elements of unsigned values and expresses the time of day and the date. The first element is an Unsigned32 data type and gives the time after the midnight in milliseconds. The second element is an Unsigned16 data type and gives the date counting the days from January 1, 1984.

5.3.1.4.2 TimeDifference

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	13
2	Data type Name	=	TimeDifference
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4 or 6

This data type is composed of two elements of unsigned values that express the difference in time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The optional second element is an Unsigned16 data type that provides the difference in days.

5.3.1.5 Enumerated types

There are no Enumerated types defined for EtherCAT.

5.3.1.6 Handle types

There are no Handle types defined for EtherCAT.

5.3.1.7 Numeric types

5.3.1.7.1.1 float

This data type is the same as Float32.

5.3.1.7.1.2 Float32

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	8
2	Data type Name	=	Float32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type has a length of four octets. The format for float32 is that defined by IEC 60559 as single precision.

5.3.1.7.1.3 double

This data type is the same as Float64.

5.3.1.7.1.4 Float64

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	17
2	Data type Name	=	Float64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This type has a length of eight octets. The format for float64 is that defined by IEC 60559 as double precision.

5.3.1.7.2 Integer types

5.3.1.7.2.1 Integer8

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	2
2	Data type Name	=	Integer8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This integer type is a two's complement binary number with a length of one octet.

5.3.1.7.2.2 SINT

This IEC 61131-3 type is the same as Integer8.

5.3.1.7.2.3 char

This data type is the same as Integer8.

5.3.1.7.2.4 Integer16

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	3
2	Data type Name	=	Integer16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is a two's complement binary number with a length of two octets.

5.3.1.7.2.5 INT

This IEC 61131-3 type is the same as Integer16.

5.3.1.7.2.6 short

This data type is the same as Integer16.

5.3.1.7.2.7 Integer24

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	16
2	Data type Name	=	Integer24
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	3

This integer type is a two's complement binary number with a length of three octets.

5.3.1.7.2.8 Integer32

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	4

2	Data type Name	=	Integer32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This integer type is a two's complement binary number with a length of four octets.

5.3.1.7.2.9 DINT

This IEC 61131-3 type is the same as Integer32.

5.3.1.7.2.10 long

This data type is the same as Integer32.

5.3.1.7.2.11 Integer40

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	18
2	Data type Name	=	Integer40
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	5

This integer type is a two's complement binary number with a length of five octets.

5.3.1.7.2.12 Integer48

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	19
2	Data type Name	=	Integer48
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

This integer type is a two's complement binary number with a length of six octets.

5.3.1.7.2.13 Integer56

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	20
2	Data type Name	=	Integer56
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	7

This integer type is a two's complement binary number with a length of seven octets.

5.3.1.7.2.14 Integer64

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	21
2	Data type Name	=	Integer64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This integer type is a two's complement binary number with a length of eight octets.

5.3.1.7.2.15 LINT

This IEC 61131-3 type is the same as Integer64.

5.3.1.7.3 Unsigned types

5.3.1.7.3.1 Unsigned8

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	5
2	Data type Name	=	Unsigned8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This type has a length of one octet.

5.3.1.7.3.2 USINT

This IEC 61131-3 type is the same as Unsigned8.

5.3.1.7.3.3 unsigned char

This data type is the same as Unsigned8.

5.3.1.7.3.4 BYTE

This data type is the same as USINT.

5.3.1.7.3.5 Unsigned16

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	6
2	Data type Name	=	Unsigned16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of two octets.

5.3.1.7.3.6 UINT

This IEC 61131-3 type is the same as Unsigned16.

5.3.1.7.3.7 WORD

This type is used in the same way as UINT.

5.3.1.7.3.8 Unsigned24

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	22
2	Data type Name	=	Unsigned24
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	3

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of three octets.

5.3.1.7.3.9 Unsigned32

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	7
2	Data type Name	=	Unsigned32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of four octets.

5.3.1.7.3.10 UDINT

This IEC 61131-3 type is the same as Unsigned32.

5.3.1.7.3.11 Unsigned40

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	24
2	Data type Name	=	Unsigned40
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	5

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of five octets.

5.3.1.7.3.12 Unsigned48

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	25
2	Data type Name	=	Unsigned48
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of six octets.

5.3.1.7.3.13 Unsigned56

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	26
2	Data type Name	=	Unsigned56
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	7

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of seven octets.

5.3.1.7.3.14 Unsigned64

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	27
2	Data type Name	=	Unsigned64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of eight octets.

5.3.1.7.3.15 ULINT

This IEC 61131-3 type is the same as Unsigned64.

5.3.1.8 Pointer types

There are no Pointer types defined for EtherCAT.

5.3.1.9 OctetString types

There are no OctetString types of fixed length defined for EtherCAT.

5.3.1.10 VisibleString character types

There are no VisibleString types of fixed length defined for EtherCAT.

5.3.2 String types

5.3.2.1 OctetString

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	10
2	Data type Name	=	OctetString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

An OctetString is an ordered sequence of octets, numbered from 1 to n. For the purposes of discussion, octet 1 of the sequence is referred to as the first octet.

NOTE ETG.1000.6 defines the order of transmission.

5.3.2.2 VisibleString

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	9
2	Data type Name	=	VisibleString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

This type is defined as the ISO/IEC 646 string type.

5.3.2.3 UnicodeString

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	11
2	Data type Name	=	UnicodeString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

This type is defined as the ISO/IEC 10646 string type.

5.3.3 GUID Types

5.3.3.1 GUID

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	29
2	Data type Name	=	GUID
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	16

A GUID is a globally unique identifier with a length of 128 Bit.

5.4 Data type ASE service specification

All of IEC/TR 61158-1, 5.4 is incorporated by reference.

6 Communication model specification

6.1 ASEs

6.1.1 Process data ASE

6.1.1.1 Overview

In the EtherCAT application layer environment, each application process of slave can contain several objects for each application process instance to convey process data. It is structured

by means of PDOs. The contents of the process data can be described by the PDO Mapping and the Sync Manager PDO assign objects of the CoE ASE. For simple slave devices the process data is fixed and is defined in the device description file.

For the process data communication usually the application memory of the buffered type is used that master and slave always have access to the process data.

Additional services are provided to read acyclically the values of the process data objects and to indicate new values for the input data object and the output data object.

The process data objects are implicitly addressed through the related services. The granularity of input or output data in a server/provider is according to the correspondent configuration attributes.

The process data ASE uses the producer/consumer access model. That means that the update of the process data with the values of the inputs and the update of the outputs with the process data are decoupled from the conveyance of the data. The receipt of a new value is indicated by the Process Output Data indication service primitive.

The primitives of the Process Output Data services are mapped to the buffered type application memory primitives described in the DL. It is recommended but not required to use FMMU entities. Configured with FMMU a single Process Output Data request can result in multiple Process Output Data indications. The process data confirmation will show the master the success or failure of the update procedure.

Figure 8 shows the primitives between master and the slaves for a Process Output Data sequence.

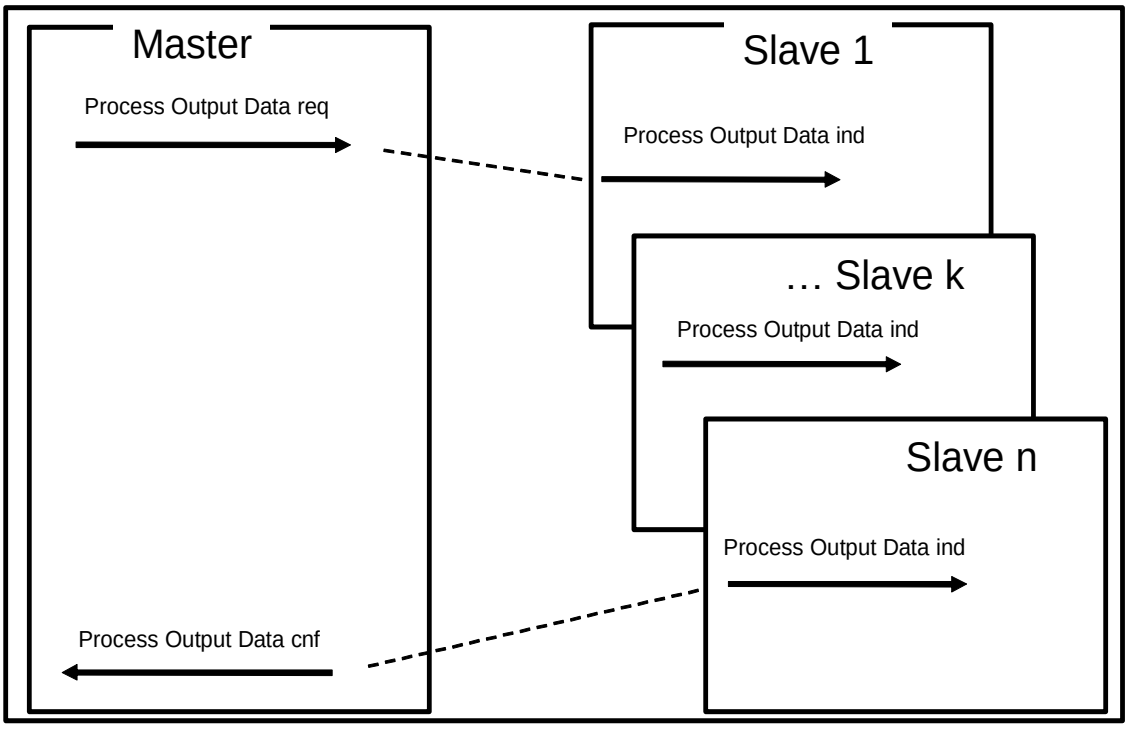

Figure 8 – Process output data sequence

The master usually sends process output data to several slaves issuing a DL write or RW service. Each slave gets the AL Event of the corresponding Sync Manager. The slave's AL controller may read the process output data at any time from the related application memory.

The primitives of the Process Input Data services are mapped to the buffered type application memory primitives described in the DL.

Figure 9 shows the primitives between master and the slaves for a Process Input Data sequence.

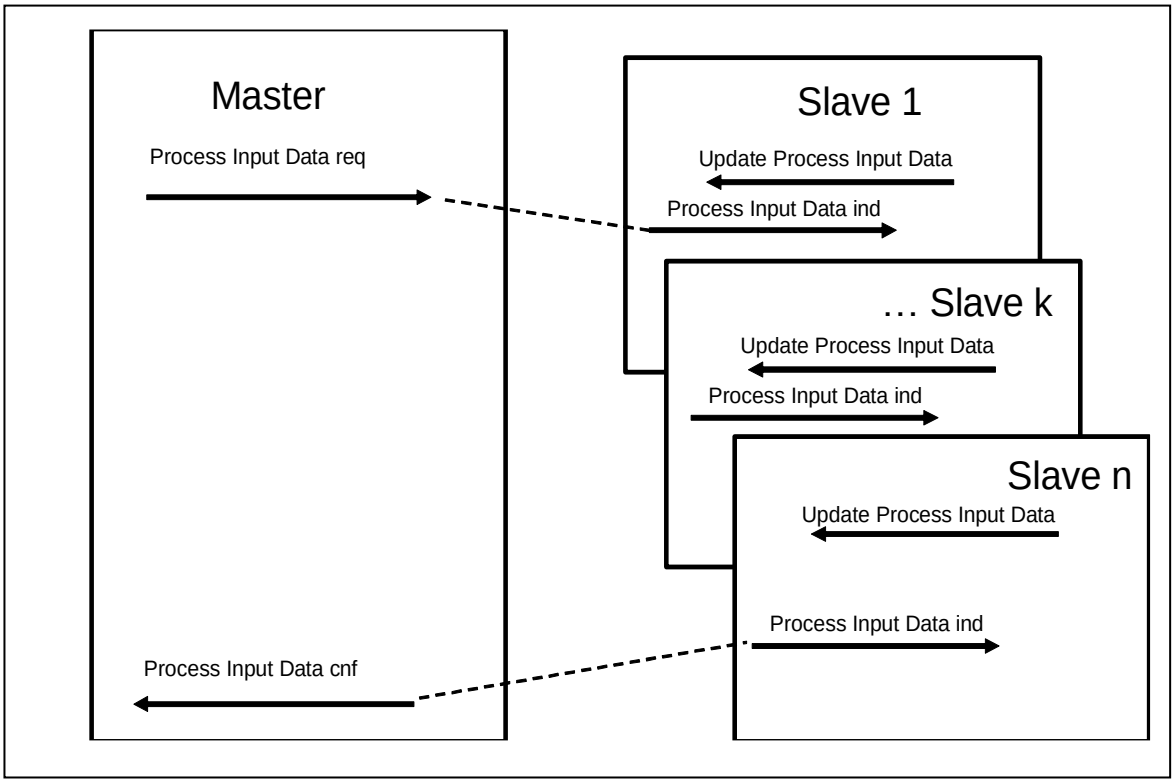

Figure 9 – Process input data sequence

The master usually reads process input data from several slaves with a logical read or RW service. The master gets the data from the previously written buffer. Each slave gets the AL Event of the corresponding Sync Manager if the input data are read out. The slave's AL controller may write the process input data at any time in the related application memory.

The formal model of the process data ASE is described next, followed by a description of its services. Furthermore, the process data ASE represents the real input and output structure of a device.

For all service primitives, parameter memory areas that are written by concurrently active service primitives should not overlap.

6.1.1.2 Process data class specification

6.1.1.2.1 Formal model

The process data object is described by the following template:

ASE:			**Process Data ASE**
CLASS:			**Process Data**
CLASS ID:			not used
PARENT CLASS:			TOP
ATTRIBUTES:			
1.	(m)	Key Attribute:	Implicit
2.	(m)	Attribute:	List of Buffer
2.1	(m)	Attribute:	BufferType
2.2	(m)	Attribute:	List of PDO
2.2.1	(m)	Attribute:	PDO Index
2.2.2	(m)	Attribute:	List of Entry
2.2.2.1	(m)	Attribute:	Index
2.2.2.2	(m)	Attribute:	Subindex
2.2.2.3	(m)	Attribute:	Bitlen
SERVICES:			
1.	(o)	OpsService:	Process Output Data
2.	(o)	OpsService:	Process Input Data
3.	(o)	OpsService:	Update Process Input Data

Objects for structuring and additional access can be found in CoE ASE.

6.1.1.2.2 Attributes

Implicit

The attribute Implicit indicates that the process data object is implicitly addressed by the services.

List of buffer
One Buffer is composed of the following list elements:

Buffertype
This attribute specifies the type of the buffer.

Allowed values: Input or Output.

List of PDO
One Buffer Element is composed of the following list elements:

PDO index
This attribute specifies to the index of the the process data object. Its permissible range is 0x1600 to 0x17FF and 0x1A00 to 0x1BFF.

List of entry
This attribute consists of the following attributes.

Index
This attribute specifies to the reference to the object. Its permissible range is 1 to 0x7FFF.

Subindex
This attribute specifies to the reference to the object entry. Its permissible range is 0 to 0xFF.

Bitlen
This attribute specifies to the length in bits of the object entry value. Its permissible range is 1 to 255.

6.1.1.2.3 Services

Process output data
This optional service is used by masters to convey output data to the slave.

Process input data
This optional service is used by slaves to publish input data.

Update process Input data
This optional service is used by slaves to provide input data to publish.

6.1.1.3 Process data ASE service specification

6.1.1.3.1 Supported services

The process data ASE defines the services

Process Output Data

Process Input Data

Update Process Input Data

6.1.1.3.2 Process output data service

6.1.1.3.2.1 Service overview

The Process Output Data service is used to convey data from the master to the slave.

6.1.1.3.2.2 Service primitives

Table 1 shows the parameters of the service.

Table 1 – Process output data

Parameter name	Req	Ind	Cnf
Argument			
List of Slave Address	M	M (=)	
List of RxPDO	M	M (=)	
Output Data	M	M (=)	
Result(+)			S
Result(–)			S

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.

Argument
The argument conveys the service specific parameters of the service request.

List of slave address
This parameter specifies the identifier for the slaves.

List of RxPDO
This parameter specifies a predefined fixed set of RxPDOs.

Output data
This parameter specifies the value of the Output Data object elements.

The allocation of the entries is done in that way, that the entries will be placed in sequential order. If the bit length is less than 8 and the number of bits fits in the unused bit of the actual octet, the object will be mapped onto the actual octet with the first bit positioned in the first unused bit.

NOTE The structuring of the output data is not fixed and can be changed with the use CoE services and FMMUs.

Result(+)
This selection type parameter indicates that the service request succeeded.
Result(–)
This selection type parameter indicates that the service request failed.

6.1.1.3.2.3 Service procedure

According to the cyclic buffer to buffer transportation characteristics the following behavior is possible:

- The Process Output Data requests are issued faster than the contents of the buffers are transported over the network. In this case the values provided with the latest Process Output Data service are conveyed over the network.
- The Process Output Data requests are issued slower than the contents of the buffers are transported over the network. In this case the values provided with each Process Output Data service are conveyed more than once over the network.
- If the Process Output Data requests are issued synchronous with the transport of the contents of the buffers the values provided with each Process Output Data service are conveyed once over the network.

6.1.1.3.3 Process input data service

6.1.1.3.3.1 Service overview

The Process Input Data service is used to convey data from the slave to the master.

6.1.1.3.3.2 Service primitives

Table 2 shows the parameters of the service.

Table 2 – Process input data

Parameter name	Req	Ind	Cnf
Argument			
List of Slave Address	M	M (=)	
List of TxPDO	M	M (=)	
Result(+)			S
Input Data			M
Result(–)			S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.			

Argument
The argument conveys the service specific parameters of the service request.

List of slave address
This parameter specifies the identifier for the slaves.

List of TxPDO
This parameter specifies a predefined fixed set of TxPDOs.

Result(+)
This selection type parameter indicates that the service request succeeded.

Input data
This parameter specifies the value of the Input Data object elements.

The allocation of the entries is done in that way, that the entries will be placed in sequential order. If the bit length is less than 8 and the number of bits fits in the unused bit of the actual octet, the object will be mapped onto the actual octet with the first bit positioned in the first unused bit.

NOTE The structuring of the input data is not fixed and can be changed with the use CoE services and FMMUs.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.1.3.3.3 Service procedure

According to the cyclic buffer to buffer transportation characteristics the following behavior is possible:

- The Process Input Data requests issued faster than the contents of the buffers are transported over the network. In this case only the values provided with the latest Update Process Input Data service are conveyed over the network.
- The Process Input Data requests issued slower than the contents of the buffers are transported over the network. In this case the values provided with each Update Process Input Data service are conveyed more than once over the network.
- If the Process Input Data requests are issued synchronous with the transport of the contents of the buffers the values provided with each Update Process Input Data service are conveyed once over the network.

6.1.1.3.4 Update process input data service

6.1.1.3.4.1 Service overview

The Update Process Input Data service is used to update input data buffer in the slave.

6.1.1.3.4.2 Service primitives

Table 3 shows the parameters of the service.

Table 3 – Update process input data

Parameter name	Req	Cnf
Argument		
List of TxPDO	M	
Input Data	M	
Result(+)		S
Result(–)		S

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.

Argument
The argument conveys the service specific parameters of the service request.

List of TxPDO
This parameter specifies a predefined fixed set of TxPDOs.

Input data
This parameter specifies the value of the Input Data object elements.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.1.3.4.3 Service procedure

- The service Procedure is described with the Process Input Data service.

6.1.2 SII ASE

6.1.2.1 Overview

In the EtherCAT application layer environment, each application process of a slave has a SII which contains all information needed for identification of a device. The slave information interface is stored persistently and specifies the boot configuration data and the application

information data. The boot configuration data contain the settings to initialize the interface of the slave controller during power on. The application information data contain the product code that can be used by the master to find the corresponding configuration file for the device. The slave information interface registers define the access to the slave information interface which is supported by the slave controller (ESC).

The formal model of the SII ASE is described next, followed by a description of its services. Furthermore, the SII ASE represents the real input and output structure of a device.

6.1.2.2 SII class specification

6.1.2.2.1 Formal model

The SII object is described by the following template:

ASE:			**SII ASE**
CLASS:			**SII**
CLASS ID:			not used
PARENT CLASS:			TOP
ATTRIBUTES:			
1.	(m)	Key Attribute:	Implicit
2.	(m)	Attribute:	PDI Control
2.1	(m)	Attribute:	PDI Type
2.2	(m)	Attribute:	AL State Control
2.3	(o)	Attribute:	Specific Settings
2.4	(o)	Attribute:	Sync Impulse Length
2.4	(o)	Attribute:	Configured station address
3.	(m)	Attribute:	Device Identification
3.1	(m)	Attribute:	Vendor ID
3.2	(m)	Attribute:	Product Code
3.3	(m)	Attribute:	Revision Number
3.4	(o)	Attribute:	Serial Number
4.	(o)	Attribute:	Mailbox Configuration
4.1	(o)	Attribute:	Bootstrap
4.1.1	(o)	Attribute:	Receive Memory Offset
4.1.2	(o)	Attribute:	Receive Size
4.1.3	(o)	Attribute:	Send Memory Offset
4.1.4	(o)	Attribute:	Send Size
4.2	(o)	Attribute:	Standard
4.2.1	(o)	Attribute:	Receive Memory Offset
4.2.2	(o)	Attribute:	Receive Size
4.2.3	(o)	Attribute:	Send Memory Offset
4.2.4	(o)	Attribute:	Send Size
4.2.5	(o)	Attribute:	Mailbox Protocols

5.	(m)	Attribute:	SII information
5.1	(m)	Attribute:	Size
5.2	(o)	Attribute:	Version
6.	(o)	Attribute:	Additional Parameter
6.1	(o)	Attribute:	Delay Correcion 1
6.1.1	(o)	Attribute:	Port0 Delay
6.1.1	(o)	Attribute:	Port1 Delay
6.2	(o)	Attribute:	Delay Correcion 2
6.2.1	(o)	Attribute:	Port0 Tx Delay
6.2.2	(o)	Attribute:	Port1 Tx Delay
6.2.3	(o)	Attribute:	Port2 Tx Delay
6.2.4	(o)	Attribute:	Port3 Tx Delay
6.2.5	(o)	Attribute:	Port0 Rx Delay
6.2.6	(o)	Attribute:	Port1 Rx Delay
6.2.7	(o)	Attribute:	Port2 Rx Delay
6.2.8	(o)	Attribute:	Port3 Rx Delay
6.3	(o)	Attribute:	Delay Correcion 3
6.3.1	(o)	Attribute:	Forward Port 0 to next Port
6.3.2	(o)	Attribute:	Forward non-Port 0 to next Port
6.3.3	(o)	Attribute:	Additive forward time closed Port
7.	(o)	Attribute:	Categories
7.1	(o)	Attribute:	List of Strings
7.1.1	(o)	Attribute:	Strings
7.2	(o)	Attribute:	General information
7.2.1	(o)	Attribute:	Group Name
7.2.2	(o)	Attribute:	Image
7.2.3	(o)	Attribute:	Order information
7.2.4	(o)	Attribute:	Name
7.2.5	(o)	Attribute:	Physical Layer
7.2.6	(o)	Attribute:	CoE Details
7.2.7	(o)	Attribute:	FoE Details
7.2.8	(o)	Attribute:	EoE Details
7.2.9	(o)	Attribute:	NoLWR
7.2.10	(o)	Attribute:	Power Budget
7.3	(o)	Attribute:	List of FMMU
7.3.1	(o)	Attribute:	Entry
7.4	(o)	Attribute:	List of Sync Manager
7.4.1	(o)	Attribute:	Sync Manager
7.5	(o)	Attribute:	List of TxPDO
7.5.1	(o)	Attribute:	Index
7.5.2	(o)	Attribute:	Related Sync Manager
7.5.3	(o)	Attribute:	Reference to DC
7.5.4	(o)	Attribute:	Name
7.5.5	(o)	Attribute:	List of Entries
7.5.5.1	(o)	Attribute:	Index
7.5.5.2	(o)	Attribute:	Subindex
7.5.5.3	(o)	Attribute:	Name
7.5.5.4	(o)	Attribute:	Data type
7.5.5.5	(o)	Attribute:	Data type
7.6	(o)	Attribute:	List of RxPDO
7.6.1	(o)	Attribute:	Index
7.6.2	(o)	Attribute:	Related Sync Manager
7.6.3	(o)	Attribute:	Reference to DC

7.6.4	(o)	Attribute:	Name
7.6.5	(o)	Attribute:	List of Entries
7.6.5.1	(o)	Attribute:	Index
7.6.5.2	(o)	Attribute:	Subindex
7.6.5.3	(o)	Attribute:	Name
7.6.5.4	(o)	Attribute:	Data type
7.6.5.5	(o)	Attribute:	Data Length
7.7	(o)	Attribute:	DC Settings
7.7.1	(o)	Attribute:	DC area

SERVICES:

1.	(o)	OpsService:	Read SII
2.	(o)	OpsService:	Write SII
3.	(o)	OpsService:	Reload SII

Objects for structuring and additional access can be found in CoE ASE.

6.1.2.2.2 Attributes

Implicit

The attribute Implicit indicates that the SII object is implicitly addressed by the services.

PDI control
This object is composed of the following elements:

PDI type
This attribute, defined in 6.2.2, specifies the initialization value for the PDI Type of AR ASE.

AL state control
This attribute, defined in 6.2.2, specifies the initialization value for the AL State Control of AR ASE.

Specific settings
This optional attribute, defined in 6.2.2, specifies the initialization value for the Specific Settings of AR ASE.

Sync impulse length
This optional attribute specifies the value for the Sync Impulse.

Configured station address
This optional attribute specifies the initialization value for the configured station address, as specified in ETG.1000.3.

Device identification
This object is composed of the following elements:

Vendor ID
This attribute specifies the initialization Vendor ID as detailed in 6.1.4.1.2.3.6. Its length is four octets.

Product code
This attribute specifies the initialization Product Code as detailed in 6.1.4.1.2.3.6. Its length is four octets.

Revision number
This attribute specifies the initialization Revision Number as detailed in 6.1.4.1.2.3.6. Its length is four octets.

Serial number
This attribute specifies the initialization Serial Number as detailed in 6.1.4.1.2.3.6. Its length is four octets.

Mailbox configuration
This object is composed of the following elements:

Bootstrap
This optional object is composed of the following elements:

Receive memory offset
This attribute specifies the offset to the memory area. Its permissible range is 0x1000 to 0xFFDA.

Receive size
This attribute specifies the receive mailbox size. Its permissible range is 38 to 1486 octets.

Send memory offset
This attribute specifies the offset to the memory area. Its permissible range is 0x1000 to 0xFFDA.

Send size
This attribute specifies the receive mailbox size. Its permissible range is 38 to 1486 octets.

Standard
This optional object is composed of the following elements:

Receive memory offset
This attribute specifies the offset to the memory area. Its permissible range is 0x1000 to 0xFFDA.

Receive size
This attribute specifies the receive mailbox size. Its permissible range is 38 to 1486 octets.

Send memory offset
This attribute specifies the offset to the memory area. Its permissible range is 0x1000 to 0xFFDA.

Send size
This attribute specifies the receive mailbox size. Its permissible range is 38 to 1486 octets.

Mailbox protocols
This attribute specifies the mailbox protocols supported. Its allowed values are EoE, CoE, FoE, and profile and application specific encodings.

SII information
This object is composed of the following elements:

Size
This attribute specifies the size of the SII structure in units of 1 Kibit (1 024 bits). Its range is 1 to 65 536 units of 1 Kibit each, reperesented as 0 to 0xFFFF.

Version
This attribute specifies the SII structure version; its value is 1

Additional parameter
This object is composed of the following elements:

Delay correction 1
This attribute consists of the following elements, each with a granularity of 100 ps.

Port0 delay
This attribute specifies a correction factor for line delay to be added if master is behind Port 0.

Port1 delay
This attribute specifies a correction factor for line delay to be added if master is behind Port 1.

Delay correction 2
This attribute consists of the following elements, each with a granularity of 100 ps.

Port0 Tx delay
This attribute specifies a correction factor for delay to be added if Port0 as sender is involved in a path. It represents the time difference between signal at the PhL media and the entry at PhL.

Port1 Tx delay
This attribute specifies a correction factor for delay to be added if Port1 as sender is involved in a path. It represents the time difference between signal at the PhL media and the entry at PhL.

Port2 Tx delay
This attribute specifies a correction factor for delay to be added if Port2 as sender is involved in a path. It represents the time difference between signal at the PhL media and the entry at PhL.

Port3 Tx delay
This attribute specifies a correction factor for delay to be added if Port3 as sender is involved in a path. It represents the time difference between signal at the PhL media and the entry at PhL.

Port0 Rx delay
This attribute specifies a correction factor for delay to be added if Port0 as receiver is involved in a path. It represents the time difference between the entry at PhL and signal at the PhL media.

Port1 Rx delay
This attribute specifies a correction factor for delay to be added if Port1 as receiver is involved in a path. It represents the time difference between the entry at PhL and signal at the PhL media.

Port2 Rx delay
This attribute specifies a correction factor for delay to be added if Port2 as receiver is involved in a path. It represents the time difference between the entry at PhL and signal at the PhL media.

Port3 Rx delay
This attribute specifies a correction factor for delay to be added if Port3 as receiver is involved in a path. It represents the time difference between the entry at PhL and signal at the PhL media.

Delay correction 3
This attribute consists of the following elements, each with a granularity of 100 ps.

Forward port 0 to next port
This attribute specifies a correction factor for delay to be added if Port0 as receiver is involved in a path. It represents the time difference between PhL to DL at Port 0 and that of DL to PhL at the next port.

Forward non-port 0 to next port
This attribute specifies a correction factor for delay to be added if a port other than Port0 as receiver is involved in a path. It represents the time difference between PhL to DL at the port that is not Port 0 and that of DL to PhL at the next port.

Additive forward time closed port
This attribute specifies a correction factor for delay to be added if a port is in the closed state. The basic time is “Additive forward time closed Port” if Port 0 is involved as virtual receiver.

Categories
This object is composed of the following elements:

List of strings
This attribute consists of the following list elements:

Strings
This attribute specifies textual elements that are reference by their list position in the following category elements.

General information
This attribute consists of the following elements:

Group name
This attribute specifies a reference to a string in the Strings collection. It is used to specify a group name for the device.

Image
This attribute specifies a reference to a string in the Strings collection. It is used to specify a name of a pictorial representation for the device.

Order information
This attribute specifies a reference to a string in the Strings collection. It is used to specify the order information for the device.

Name
This attribute specifies a reference to a string in the Strings collection. It is used to specify the name information for the device.

Physical layer
This attribute specifies the main physical layer technology of the device.

CoE details
This attribute specifies a structure with CoE information.

FoE details
This attribute specifies a structure with FoE information.

NoLRW
This Boolean attribute is true if use of the DL-service LogicalReadWrite is not supported by this device.

Power budget
This attribute specifies the operating current required by this device. Negative values indicate that the device provides power. This attribute's range is -128 to +127 mA.

List of FMMU
This attribute consists of the following list elements:

Entry
This attribute specifies the preferred use of an FMMU. A maximum of three uses can be supported.

List of Sync Manager
This attribute consists of the following list elements:

Sync Manager
This attribute specifies the preferred use of a Sync Manager, as defined in ETG.1000.3.

List of TxPDO
This optional attribute is present if there are a limited number of PDO selections that are always available for this type of device. It consists of the following list of elements:

Index
This attribute specifies the Index of a PDO. Its permissible range is 0x1A00 to 0x1BFF.

Related Sync Manager
This attribute specifies the allocation of the PDU to a Sync Manager. Its permissible range is 0 to 15.

Reference to DC sync
This attribute specifies one of up to sixteen Sync Methods as specified in the CoE object dictionary.

Name
This attribute specifies the Name as Reference to a string

List of entries
This attribute consists of the following list elements:

Index
This attribute specifies the Index of an object.

Subindex
This attribute specifies the Subindex of an object.

Name
This attribute specifies the Name as Reference to a string

Data type
This attribute specifies the data type as index to a CoE object directory.

Data length
This attribute specifies the length of the object in bits.

List of RxPDO
This optional attribute is present if there are a limited number of PDO selections that are always available for this type of device. It consists of the following list of elements:

Index
This attribute specifies the Index of a PDO in the range 0x1600 to 0x17FF

Related Sync Manager
This attribute specifies the allocation to a Sync Manager; its range is 0 to 15

Reference to DC sync
This attribute specifies the Sync Method as specified in CoE object dictionary; its range is 0 to 15

Name
This attribute specifies the Name as Reference to a string

List of entries
This attribute consists of the following list elements:

Index
This attribute specifies the Index of an object.

Subindex
This attribute specifies the Subindex of an object.

Name
This attribute specifies the Name as Reference to a string

Data type
This attribute specifies the data type as index to a CoE object directory.

Data length
This attribute specifies the length of the object in bits.

DC settings
This attribute consists of the following element:

DC area
This attribute specifies an octet string the contents will be defined in the device profiles.

6.1.2.2.3 Services

Write SII
This optional service is used by masters to convey 2 octets of data to the slave SII.

Read SII
This optional service is used by masters to convey 4 octets of data from the slave SII to the master.

Reload SII
This optional service is used by masters to convey 2 octets of data to the slave SII and update the mirror register in the Slave DL.

6.1.2.3 SII ASE service specification

6.1.2.3.1 Supported services

The SII ASE defines the services

Read SII

Write SII

Reload SII

6.1.2.3.2 SII read service

6.1.2.3.2.1 Service overview

The SII Read service is used to transfer of up to 4 octets of data from the server to client. The server answers with the result of the read operation.

6.1.2.3.2.2 Service primitives

Table 4 shows the service primitives and parameter of the SII Read service.

Table 4 – SII read

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Word Index	M	M (=)		
Result(+)			S	S
Data			M	M (=)
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the slave.

Word Index
This parameter specifies the word index to the SII area which is to be read.

Result(+)
This selection type parameter indicates that the service request succeeded.

Data
This parameter specifies the object value read.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.2.3.3 SII write service

6.1.2.3.3.1 Service overview

The SII Write service is used to transfer of up to 2 octets of data from the client to server. The server answers with the result of the write operation.

6.1.2.3.3.2 Service primitives

Table 5 shows the service primitives and parameter of the SII Write service.

Table 5 – SII write

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Word Index	M	M (=)		
Data	M	M (=)		
Result(+)			S	S
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the slave.

Word Index
This parameter specifies the word index to the SII area which is to be written.

Data
This parameter specifies the object value which is to be written.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.2.3.4 SII reload service

6.1.2.3.4.1 Service overview

The SII Reload service is used to transfer of up to 2 octets of data from the client to server and refresh the DL registers associated with the new SII values. The server answers with the result of the reload operation.

6.1.2.3.4.2 Service primitives

Table 6 shows the service primitives and parameter of the SII Reload service.

Table 6 – SII reload

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Word Index	M	M (=)		
Data	M	M (=)		
Result(+)			S	S
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the slave.

Word index
This parameter specifies the word index to the SII area which is to be written.

Data
This parameter specifies the object value which is to be written.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.3 Isochronous ASE

6.1.3.1 Overview

In the EtherCAT application layer environment, an application process of a slave may have a isochronous PDI which specifies information needed for synchronous operation of several devices. The isochronous specification assumes an interface that consists of two synchronization signals that are related to the system clock, as defined in ETG.1000.4. Additionally support for latching input signals may be supported.

The isochronous PDI registers define the access to the slave information interface if isochronous operation is supported by the slave controller (ESC).

The formal model of the isochronous ASE is described next, the services are local read and write services of the DL and local events.

6.1.3.2 Isochronous class specification

6.1.3.2.1 Formal model

The isochronous object is described by the following template:

ASE:			**Isochronous ASE**
CLASS:			**Isochronous**
CLASS ID:			not used
PARENT CLASS:			TOP
ATTRIBUTES:			
1.	(m)	Key Attribute:	Implicit
2.	(o)	Attribute:	Sync Parameter
2.1	(o)	Attribute:	Cyclic operation enable
2.2	(o)	Attribute:	SYNC0 activate
2.3	(o)	Attribute:	SYNC1 activate
2.4	(o)	Attribute:	Specific Settings
2.5	(o)	Attribute:	Sync Impulse Length
2.6	(o)	Attribute:	Interrupt 0 Status
2.7	(o)	Attribute:	Interrupt 1 Status
2.8	(o)	Attribute:	Cyclic Operation Start Time
2.9	(o)	Attribute:	SYNC0 cycle time
2.10	(o)	Attribute:	SYNC1 cycle time
3.	(m)	Attribute:	Latch Parameter
3.1	(m)	Attribute:	Latch 0 positive edge
3.2	(m)	Attribute:	Latch 0 negative edge
3.3	(m)	Attribute:	Latch 1 positive edge
3.4	(o)	Attribute:	Latch 1 negative edge
3.1	(m)	Attribute:	Latch 0 positive event
3.2	(m)	Attribute:	Latch 0 negative event
3.3	(m)	Attribute:	Latch 1 positive event
3.4	(o)	Attribute:	Latch 1 negative event
3.1	(m)	Attribute:	Latch 0 positive edge value
3.2	(m)	Attribute:	Latch 0 negative edge value
3.3	(m)	Attribute:	Latch 1 positive edge value
3.4	(o)	Attribute:	Latch 1 negative edge value
SERVICES:			
1.	(o)	OpsService:	DL-read local
2.	(o)	OpsService:	DL-write local
1.	(o)	OpsServices:	DL-read
2.	(o)	OpsServices:	DL-write

6.1.3.2.2 Attributes

Implicit

The attribute Implicit indicates that the isochronous object is implicitly addressed by the services.

Sync parameter
This object is composed of the following elements:

Cyclic operation enable
This Boolean attribute enables cyclic Sync 0 or 1 operation.

Sync0 activate
This Boolean attribute enables the Sync 0 operation.

Sync1 activate
This Boolean attribute enables the Sync 1 operation.

Specific Settings
This optional attribute, defined in 6.2.2, specifies the initialization value for the Specific Settings of AR ASE.

Sync Impulse Length
This optional attribute specifies the value for the Sync Impulse in multiples of 10 ns.

Interrupt 0 Status
This optional Boolean attribute indicates an active Sync0 interrupt.

Interrupt 1 Status
This optional Boolean attribute indicates an active Sync1 interrupt.

Cyclic Operation Start Time
This optional attribute sets a start time related to system time for cyclic operation.

Sync0 cycle time
This attribute set the cycle time of Sync0 in multiples of 1ns for Sync0.

Sync1 cycle time
This attribute set the cycle time of Sync1 in multiples of 1ns for Sync0.

Latch Parameter
This object is composed of the following elements:

Latch 0 positive edge
This Boolean attribute enables Latch 0 operation for a single event (true) or continuous latching.

Latch 0 negative edge
This Boolean attribute enables Latch 0 operation for a single event (true) or continuous latching.

Latch 1 positive edge
This Boolean attribute enables Latch 1 operation for a single event (true) or continuous latching.

Latch 1 negative edge
This Boolean attribute enables Latch 1 operation for a single event (true) or continuous latching.

Latch 0 positive event
This Boolean attribute indicates Latch 0 positive edge event.

Latch 0 negative event
This Boolean attribute indicates Latch 0 negative edge event.

Latch 1 positive event
This Boolean attribute indicates Latch 1 positive edge event.

Latch 1 negative event
This Boolean attribute indicates Latch 1 negative edge event.

Latch 0 positive edge value
This attribute stores the system time value in case of a Latch 0 positive edge event.

Latch 0 negative edge value
This attribute stores the system time value in case of a Latch 0 negative edge event.

Latch 0 positive edge value
This attribute stores the system time value in case of a Latch 0 positive edge event.

Latch 0 negative edge value
This attribute stores the system time value in case of a Latch 0 negative edge event.

6.1.3.2.3 Services

DL-read local
This optional service is used by the slave application to read out the isochronous parameter. The value of the parameter memory address should be in the range of 0x980 to 0x9FF.

DL-write local
This optional service is used by the slave application to write isochronous parameter. The value of the parameter memory address should be in the range of 0x980 to 0x9FF. Only a subset of the registers can be written.

DL-read
This optional service is used by the master to read out the isochronous parameter. The value of the parameter memory address should be in the range of 0x980 to 0x9FF.

DL-write
This optional service is used by the master to write isochronous parameter. The value of the parameter memory address should be in the range of 0x980 to 0x9FF. Only a subset of the registers can be written.

6.1.4 CoE ASE

6.1.4.1 Overview

6.1.4.1.1 General

CANopen is a communication protocol originally developed for CAN-based systems. It is a standardized embedded network with highly flexible configuration capabilities.

NOTE CANopen (CiA DS 301) is standardized as EN 50325-4.

EtherCAT uses the CAN application protocol service definition.

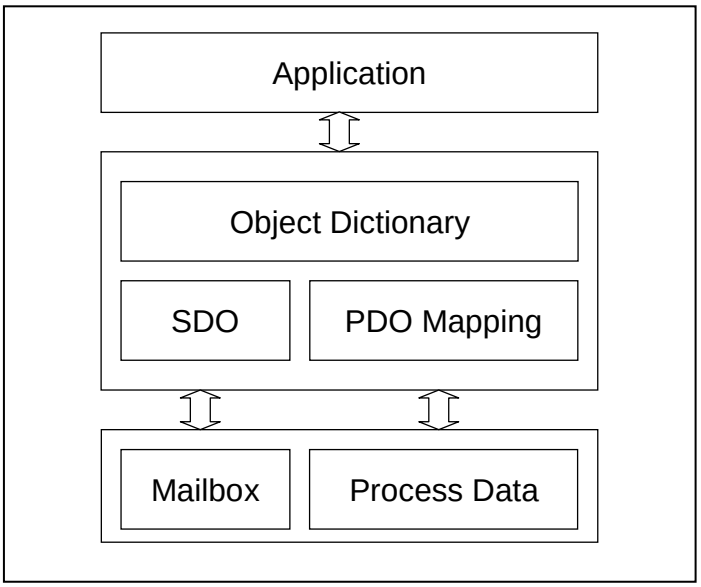

Figure 10 – CoE server model

The Object Dictionary contains parameters, application data and the mapping information between process data interface and application date (PDO mapping). Its entries can be accessed via Service Data Objects (SDO), as shown in Figure 10.

6.1.4.1.2 Object dictionary

6.1.4.1.2.1 Object dictionary structure

The Object Dictionary contains all the CoE related data objects of the device in a standardized way. It is a collection of the device parameters data structures that can be accessed with the SDO Upload and SDO Download services.

With the SDO information services the available entries of the object dictionary and the entry description of an entry in the object dictionary can be read.

The object dictionary consists of the areas as described in Table 7

Table 7 – Allocation of SDO areas

Area	Contents
Data type Area:	Definition of the data types
CoE Communication Area:	Definition of the variables which may be used for all servers and for dedicated communication purposes
Manufacturer Specific Area:	Definition of manufacturer specific variables
Device Profile Area:	Definition of the variables defined in a device profile
Reserved Area:	reserved for future use

6.1.4.1.2.2 Data type area

The Data type Area consists of the following parts:

Static Data types

Definition of general simple data types

Complex Data types

Definition of general structured data types

Manufacturer Specific Complex Data types

Definition of manufacturer specific structured data types

Device Profile Specific Static Data types

Definition of device profile specific simple data types

Device Profile Specific Complex Data types

Definition of device profile specific structured data types

Enumeration Data types

Definition of device specific enumeration data types

6.1.4.1.2.3 CoE Communication area

6.1.4.1.2.3.1 Device type

The Device Type object has the following parameter:

Parameter

Device profile

This parameter specifies the device profile that is used of the device.

Additional information

This parameter specifies additional information defined by the device profile.

6.1.4.1.2.3.2 6.1.4.1.2.3.2 Error register

The optional Error Register object has the following parameter:

Parameter

Generic error

This parameter is set to true if a generic error is present.

Current error

This parameter is set to true if a current error is present.

Voltage error

This parameter is set to true if a voltage error is present.

Temperature error

This parameter is set to true if a temperature error is present.

Communication error

This parameter is set to true if a communication error is present.

Device profile specific error

This parameter is set to true if a device profile specific error is present.

Manufacturer specific error

This parameter is set to true if a manufacturer specific error is present.

6.1.4.1.2.3.3 Manufacturer device name

The Manufacturer Device Name object has the following parameter:

Parameter

Device name

This parameter specifies the device name of the device.

6.1.4.1.2.3.4 Hardware version

The Hardware Version object has the following parameter:

Parameter

Hardware version

This parameter specifies the hardware version of the device.

6.1.4.1.2.3.5 Software version

The Software Version object has the following parameter:

Parameter

Software version

This parameter specifies the software version of the device.

6.1.4.1.2.3.6 Identity

The Identity object has the following parameter:

Parameter

Vendor ID

This parameter specifies the vendor ID of the device.

Product code

This parameter specifies the product code of the device.

Major Revision Number

This parameter specifies the major revision number of the device which identifies the functionality of the device.

Minor Revision Number

This parameter specifies the minor revision number of the device which identifies different version with the same functionality.

Serial Number

This parameter specifies the serial number of the device.

6.1.4.1.2.3.7 Receive PDO mapping

The Receive PDO Mapping object has the following parameter:

Parameter

PDO Number

This parameter specifies the number of the PDO.

Number of mapping entries

This parameter specifies the number of mapping entries.

List of mapping entries

This parameter specifies a list of mapping entries.

6.1.4.1.2.3.8 Transmit PDO mapping

The Transmit PDO Mapping object has the following parameter:

Parameter

PDO Number

This parameter specifies the number of the PDO.

Number of mapping entries

This parameter specifies the number of mapping entries.

List of mapping entries

This parameter specifies a list of mapping entries.

6.1.4.1.2.3.9 Sync manager communication Type

The Sync Manager Communication Type object has the following parameter:

Parameter

Number of Used Sync Manager entities

This parameter specifies the number of used Sync Manager entities.

List of Sync Manager Communication Types

This parameter specifies the communication type for each used Sync Manager entity.

6.1.4.1.2.3.10 Sync manager PDO assignment

The Sync Manager PDO assignment object has the following parameter:

Parameter

Channel number

This parameter specifies the number of the Sync Manager channel.

Number of assigned PDOs

This parameter specifies the number of assigned PDOs.

List of assigned PDOs

This parameter specifies the list of assigned PDOs.

6.1.4.1.2.3.11 Synchronization timing

The Sync Manager Synchronization object has the following parameter:

Parameter

Channel number

This parameter specifies the number of the Sync Manager channel.

Synchronization type

This parameter specifies the method of synchronization.

Cycle time

This parameter specifies the target length of a cycle.

Shift time

This parameter specifies an offset parameter for synchronized action.

6.1.4.1.3 SDO interactions

6.1.4.1.3.1 SDO services

With the SDO services entries of the Object Dictionary can be read or written. The SDO transport protocol allows transmitting objects of any size. The SDO protocol is equivalent to the CANopen SDO protocol in order to support reuse of existing protocol stacks.

The first octet of the first segment specifies the necessary flow control information. The next three octets of the first segment specify the index and sub-index of the Object Dictionary entry to be read or written. The next octets of the first segment are available for user data. The second and the following segments contain the control octet and user data. The receiver confirms each segment or a block of segments, so that a peer-to-peer communication (client/server) takes place.

In legacy mode the SDO protocol consists of 8 octets only to match the CAN data size. In enhanced mode the payload data is simply enlarged without changing the protocol headers. In this way the larger data size of the mailbox is applied to the SDO protocol, the transmission of large data is accelerated accordingly.

Furthermore, a mode is added that allows one to transfer the entire data located at one index of the object dictionary at one go. The data of all sub-indices is then transferred subsequently.

The confirmed services (SDO Upload, SDO Download, Download SDO Segment, and Upload SDO Segment) and the unconfirmed service (Abort SDO Transfer) are used for Service Data Objects performing the segmented/expedited transfer. The SDO Download service class is split up in SDO Download Expedited and Initiate SDO Download Normal services. The SDO Upload service class is split up in SDO Upload Expedited and Initiate SDO Upload Normal services.

The primitives of the SDO services are mapped to the primitives of the mailbox transmission services.

6.1.4.1.3.2 SDO download sequence

Figure 11 shows the primitives between client and server in case of a successful single SDO Download sequence.

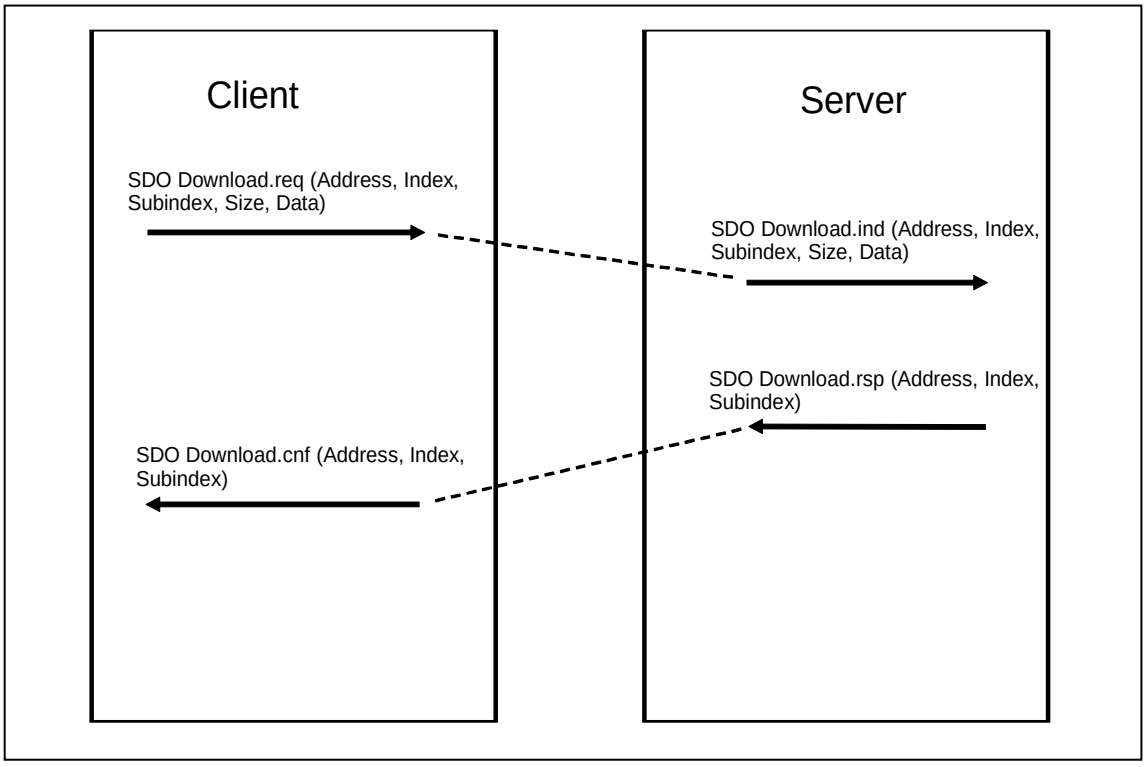

Figure 11 – Successful single SDO-Download sequence

Figure 12 shows the primitives between client and server in case of an unsuccessful SDO Download sequence. The service parameter Address, Index and Subindex should be the same in the Download request and the Abort request.

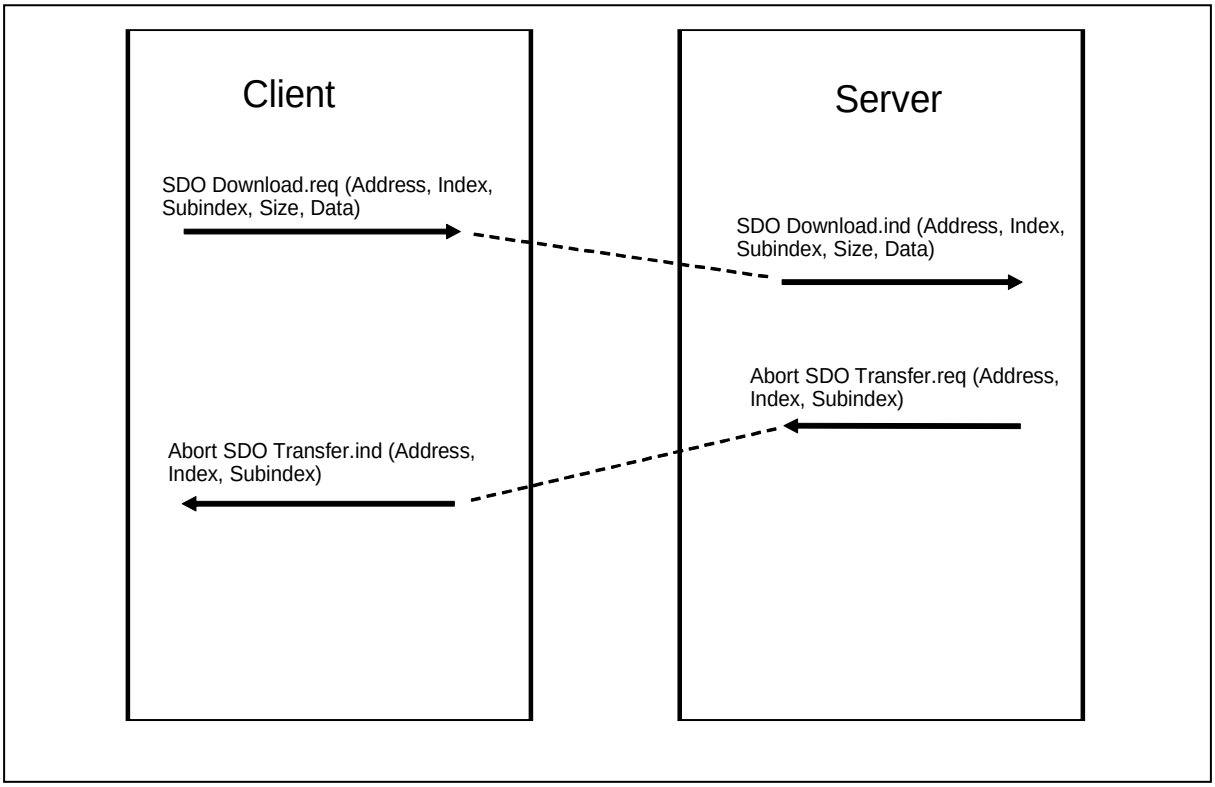

Figure 12 – Unsuccessful single SDO-Download sequence

Figure 13 shows the primitives between client and server in case of a successful segmented SDO Download sequence.

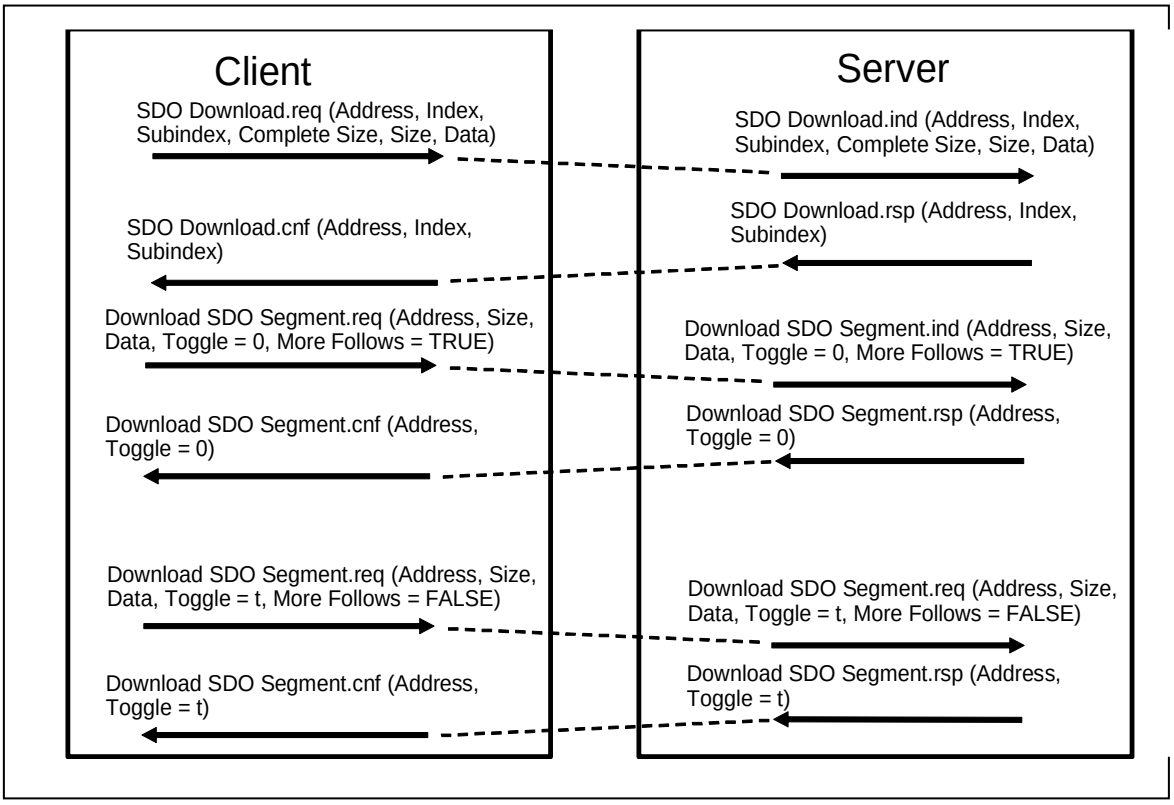

Figure 13 – Successful segmented SDO-Download sequence

6.1.4.1.3.3 SDO Upload Sequence

Figure 14 shows the primitives between client and server in case of a successful single SDO Upload sequence.

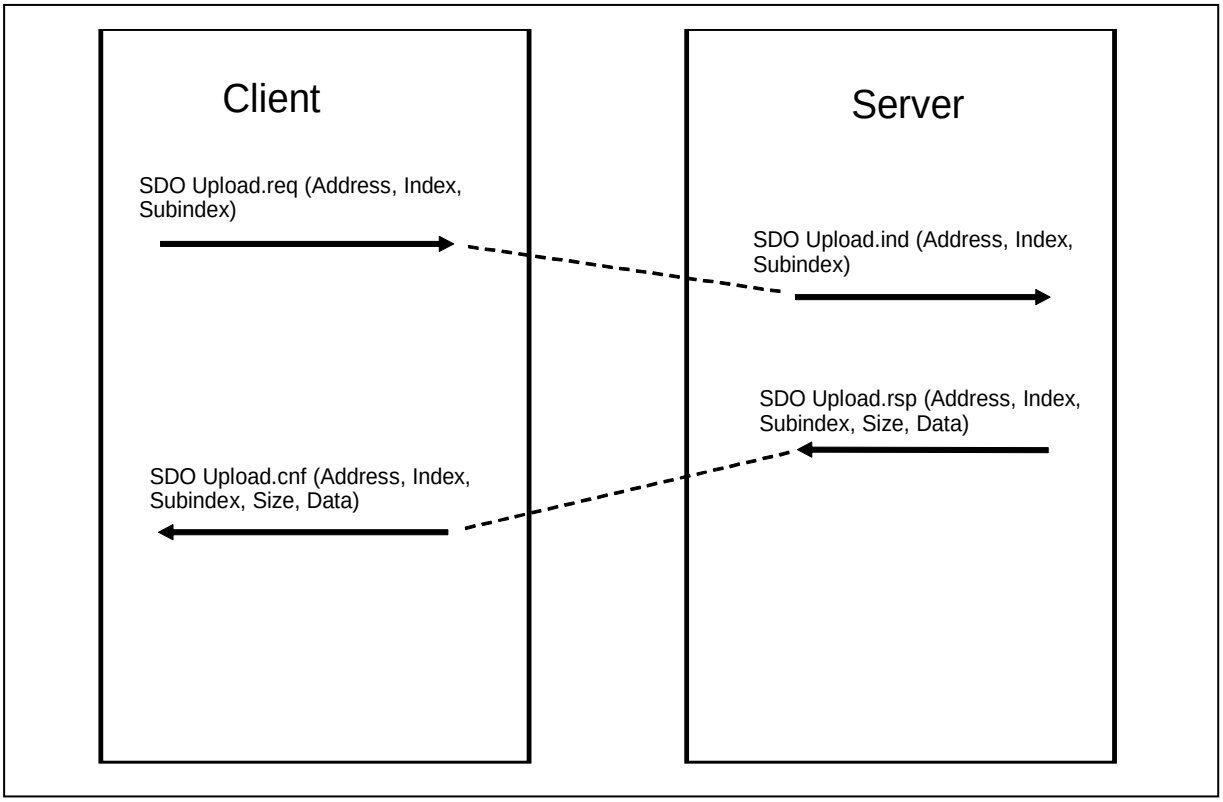

Figure 14 – Successful single SDO-Upload sequence

Figure 15 shows the primitives between client and server in case of a unsuccessful SDO Upload sequence. The service parameter Address, Index and Subindex should be the same in the Upload request and the Abort request.

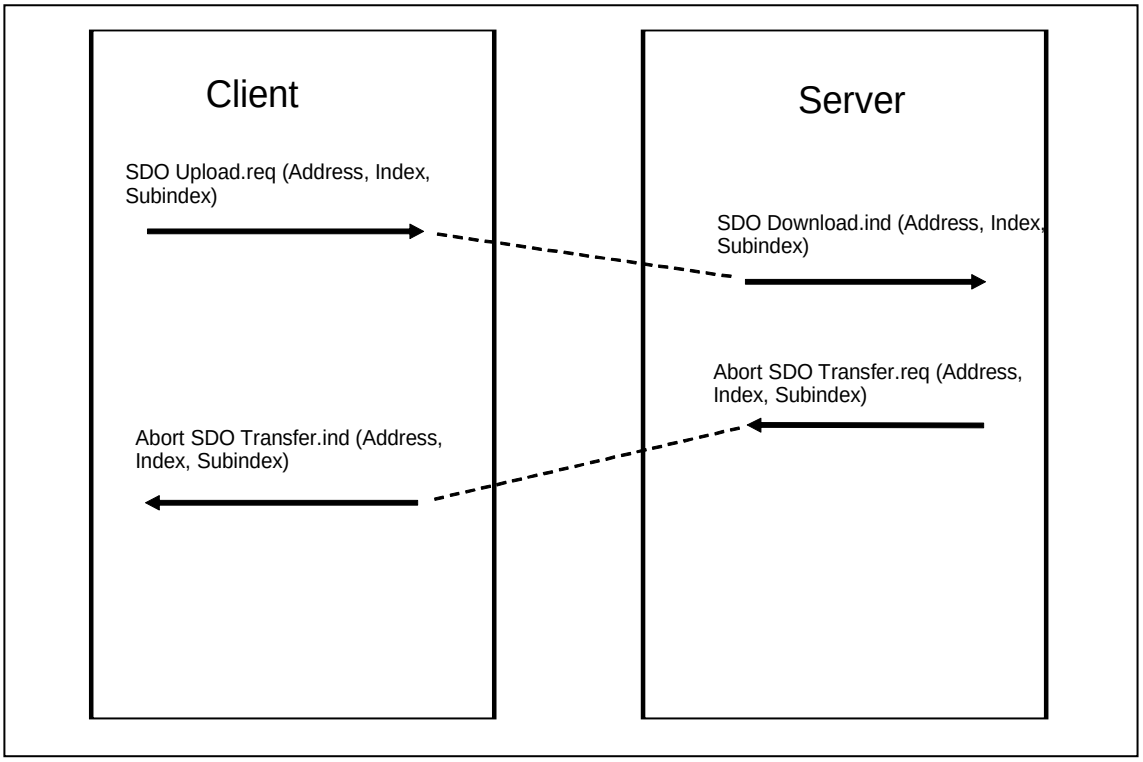

Figure 15 – Unsuccessful single SDO-Upload sequence

Figure 16 shows the primitives between client and server in case of a successful segmented SDO Upload sequence.

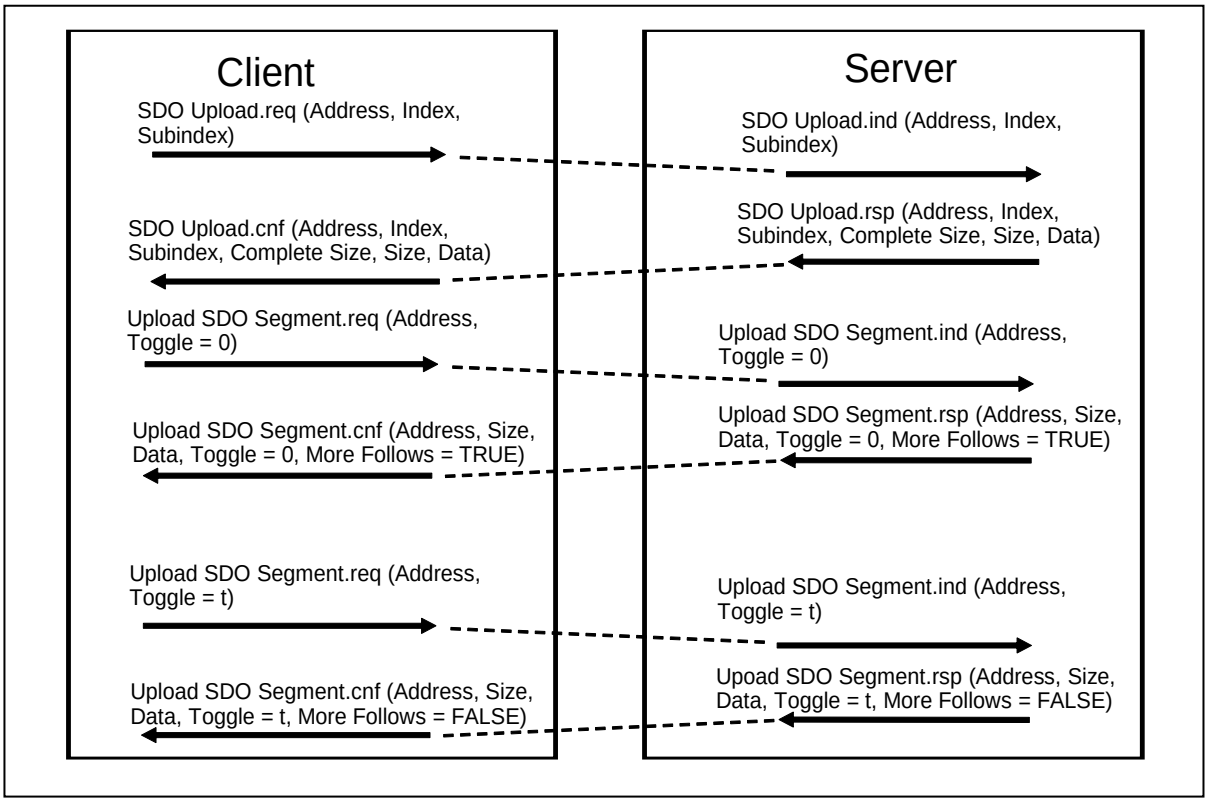

Figure 16 – Successful segmented SDO-Upload sequence

6.1.4.1.3.4 Information services

With the SDO information services the object dictionary of a server can be read by a client. The primitives of the SDO information services are mapped to the primitives of the mailbox transmission services.

Figure 17 shows the primitives between client and server which is the same for all SDO information services named as SDO_Info. The fragmentation is done with an SDO_Info_Seg service.

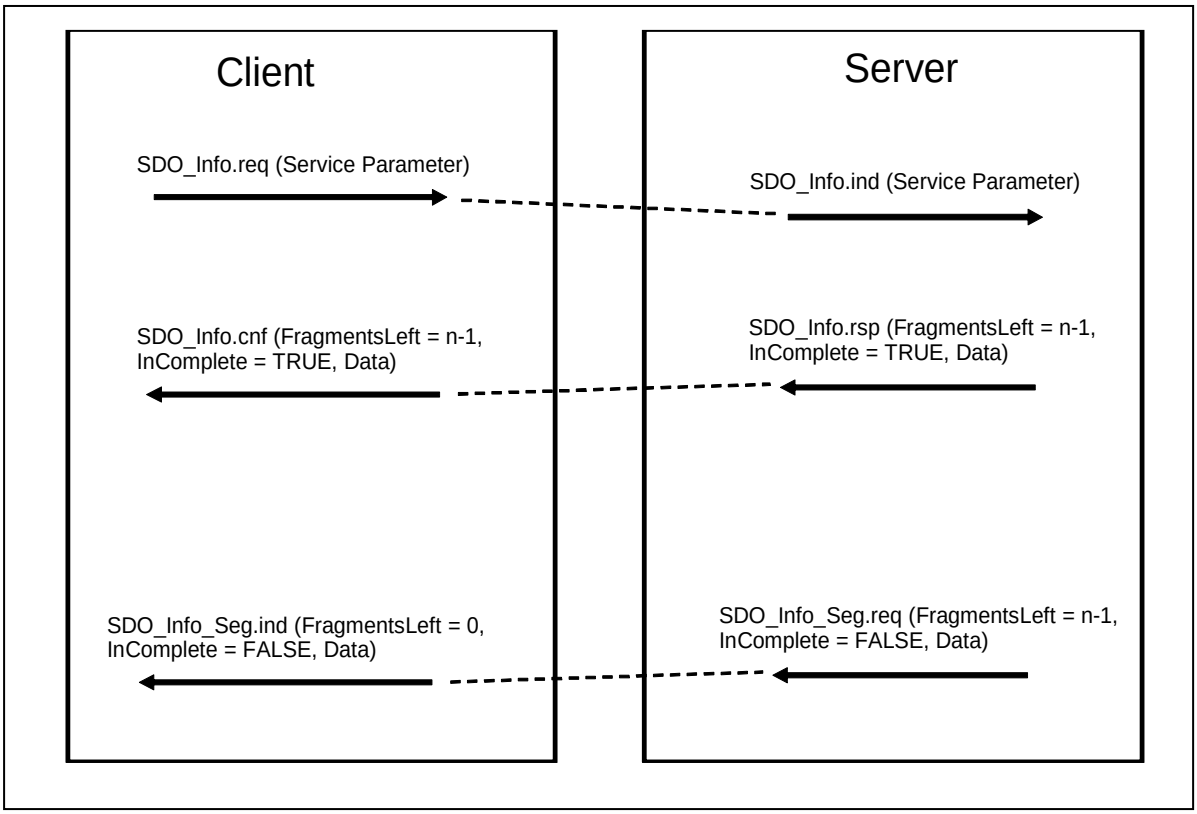

Figure 17 – SDO information sequence

6.1.4.1.4 Emergency interaction

Emergency messages are triggered by the occurrence of a device internal error situation. The transmission is executed via the mailbox interface.

The primitives of the Emergency service are mapped to the primitives of the mailbox transmission services.

Figure 18 shows the primitives between server and client in case of an Emergency service.

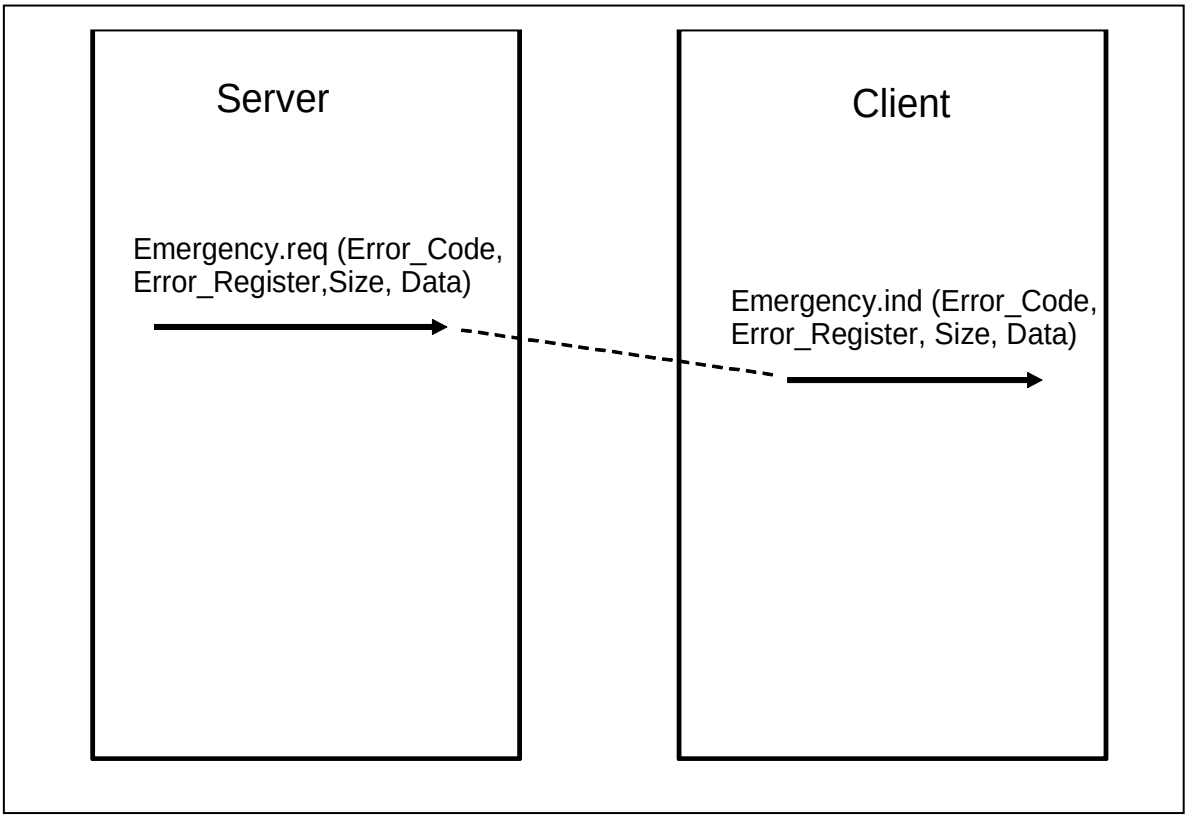

Figure 18 – Emergency service

6.1.4.1.5 Command

Command objects can be used for operations where data has to be sent in the request and the response.

An operation will be started in the server by a writing subindex 1 of the command object with the command request data by a SDO Download service. The status and the response data of the operation will be read with a SDO Upload to subindex 3 of the command object.

Figure 19 shows the primitives between server and client in case of a Command service.

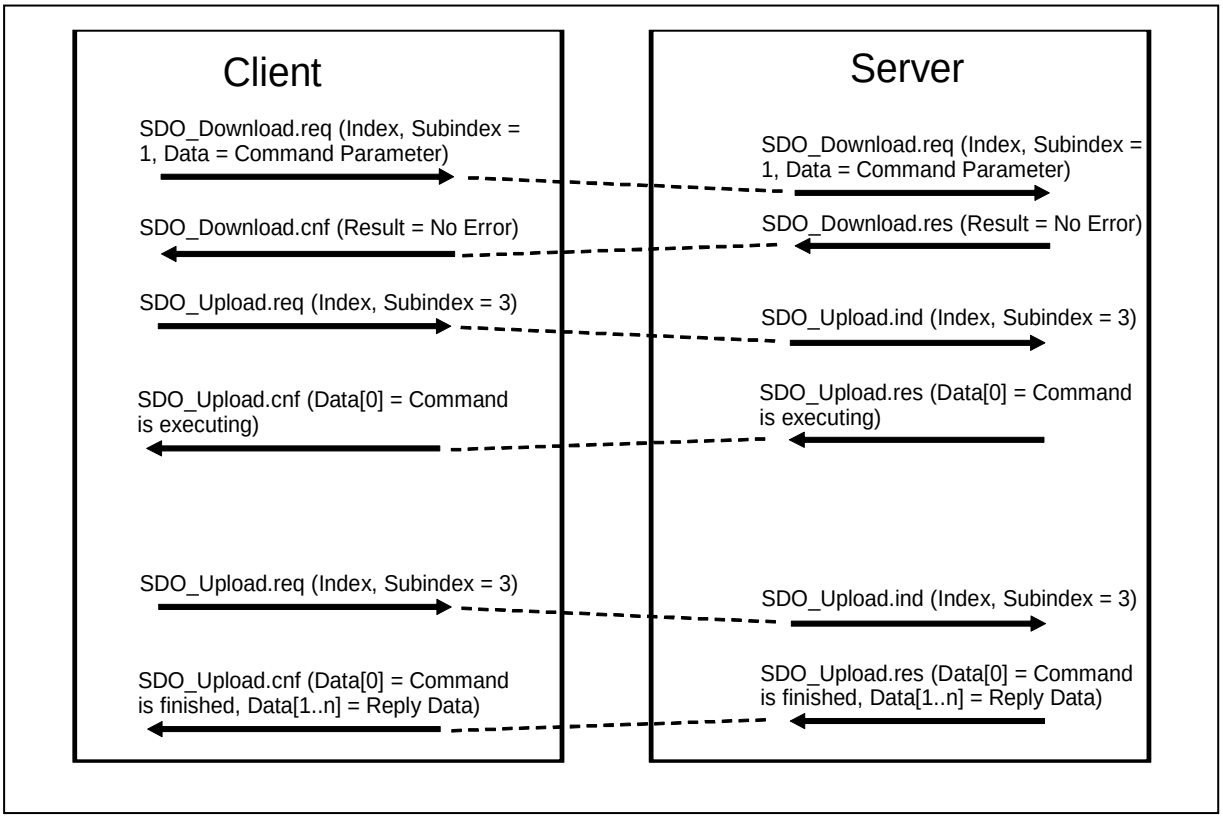

Figure 19 – Command sequence

6.1.4.1.6 Process data interaction

6.1.4.1.6.1 Mapping of objects to process data

The process data of a slave are specified in process data ASE. Each Sync Manager channel PDO assignment object describes a consistent area inside the process data ASE and consists of several (1 to 255) process data objects.

Process data objects (PDO) consists of objects in the object dictionary which can be mapped to PDOs. The PDO mapping objects contains a list of object references together with the length in bit.

PDO mapping refers to mapping of the application objects (real time process data) from the object dictionary to the PDOs. The device profiles provide a default mapping for every device type which is appropriate for most applications. E.g. the default mapping for digital I/O simply represents the inputs and outputs in their topological order in the transmit and receive PDOs.

The current mapping can be read by means of corresponding entries in the object directory. These are known as the mapping tables. The first location in the mapping table (sub-index 0) specifies the number of mapped objects that are listed after it. The tables are located in the object directory at index 0x1600 to 0x17FF for the RxPDOs and at 0x1A00 to 0x1BFF for the TxPDOs. Figure 20 shows an example of a PDO mapping.

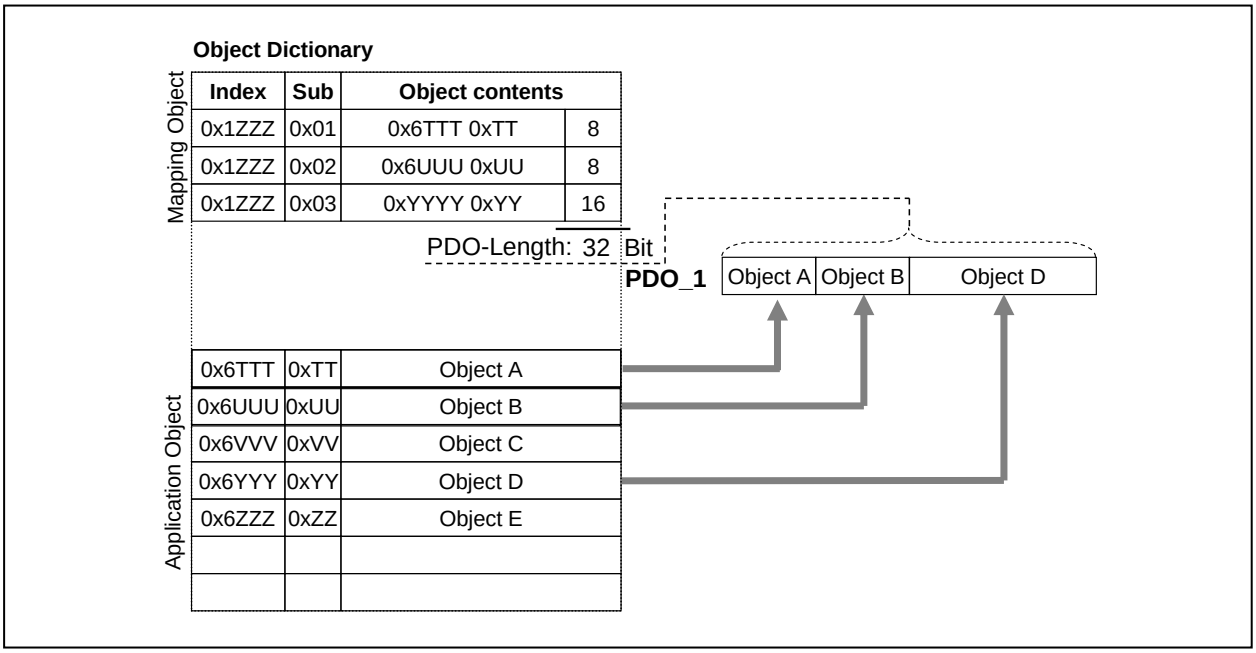

Figure 20 – PDO mapping

Sync Manager channel consists of several PDOs. The Sync Manager PDO assign objects describes how these PDOs are related to a Sync Manager.

For devices with a configurable mapping or devices allowing an access to the process data by different master tasks, the Sync Manager PDO assign objects should be writable.

The current assignment can be read by means of corresponding entries in the object directory. The first location in the Sync Manager PDO assignment table (sub-index 0) specifies the number of assigned PDOs that are listed after it. The tables are located in the object directory at index 0x1C10 to 0x1C2F. Figure 21 shows an example of a PDO assignment.

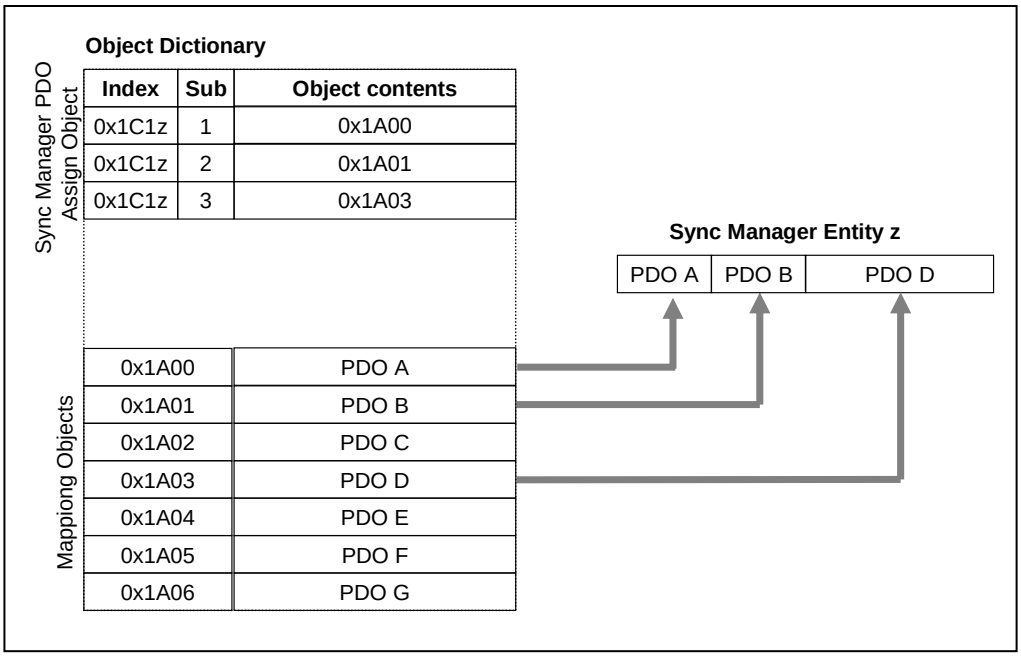

Figure 21 – Sync manager PDO assigment

Slaves with mailbox communication support the reading of the PDO mapping and the Sync Manager PDO assign objects. For slaves with a configurable mapping there are two possibilities how this configurable mapping should be supported. In the first possibility the slave supports

different mappings described in the PDO mapping objects which are only readable. The master configures the actual mapping by selecting the desired PDO mapping objects when writing the Sync Manager PDO assign objects. The second possibility for more enhanced devices allows the master to write the PDO mapping objects to get a more flexible mapping configuration possibility.

6.1.4.1.6.2 Process data objects (PDO)

Each process data object has the following parameters.

Parameter

Number

This parameter specifies the number to identify the PDO.

Direction

This parameter specifies the direction Rx (master to slave, client to server) or Tx (slave to master, server to client) of the PDO.

Number of mapped Objects

This parameter specifies the number of mapped objects which are related to the PDO.

List of mapped Objects

This parameter specifies the related objects which define the mapping of the PDO.

6.1.4.1.6.3 Sync manager PDO assignment

Each Sync Manager channel object has the following parameters.

Parameter

Number

This parameter specifies the number to identify the Sync Manager channel.

Direction

This parameter specifies the direction Output (master to slave) or Input (slave to master) of the Sync Manager channel.

Number of assigned PDOs

This parameter specifies the number of assigned PDOs which are related to the Sync Manager channel.

List of assigned PDOs

This parameter specifies the assigned PDOs to the Sync Manager channel.

6.1.4.1.6.4 PDO transmission with CoE

With the PDO services process data objects can be transmit via the mailbox Interface acyclically from the client to the server (RxPDO) or from the server to the client (TxPDO).

The primitives of the PDO services are mapped to the primitives of the mailbox transmission services.

Figure 22 shows the primitives between client and server in case of a RxPDO service.

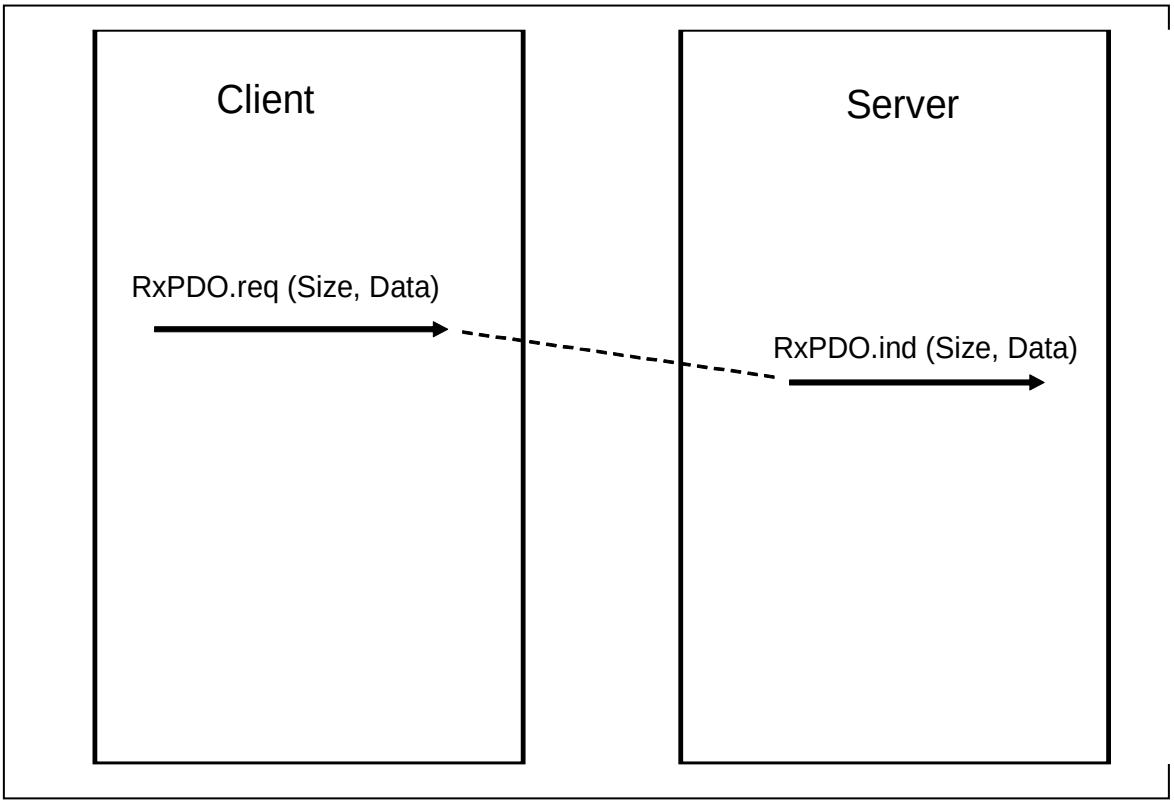

Figure 22 – RxPDO service

Figure 23 shows the primitives between server and client in case of a TxPDO service.

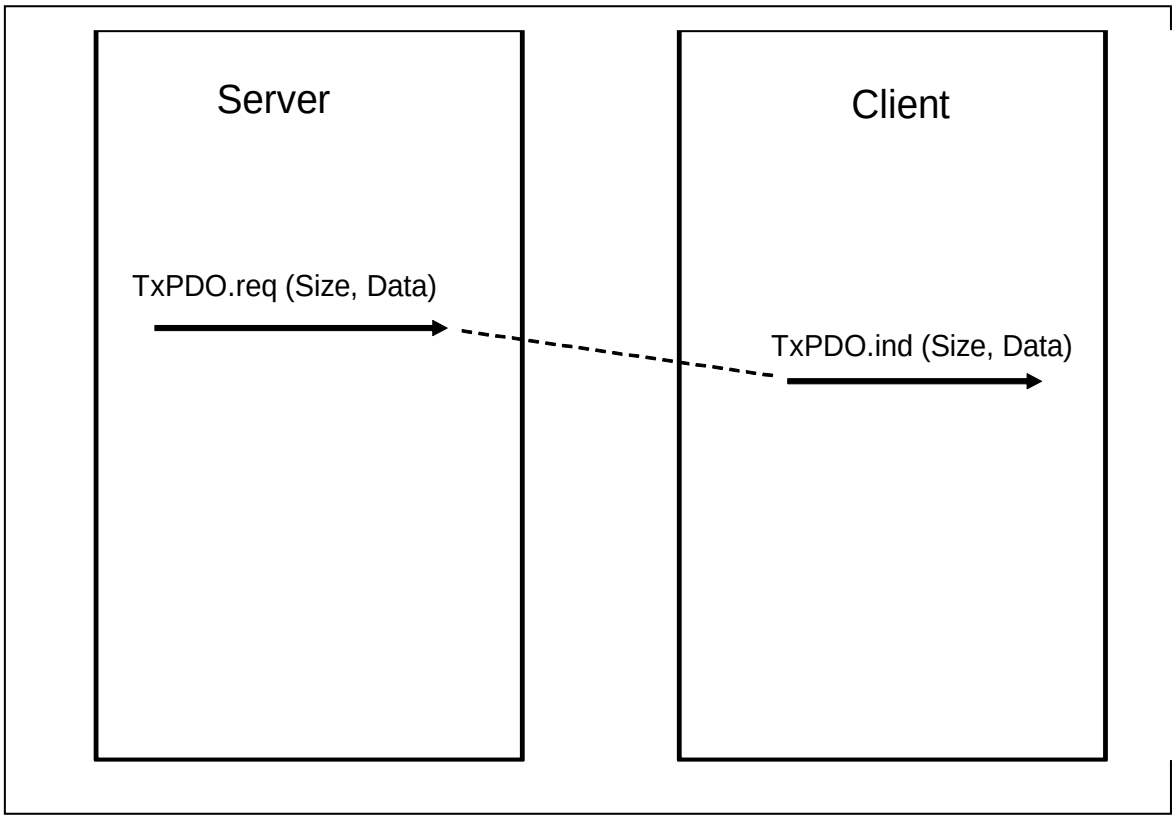

Figure 23 – TxPDO service

Figure 24 shows the primitives between client and server in case of a RxPDO Remote Transmission service.

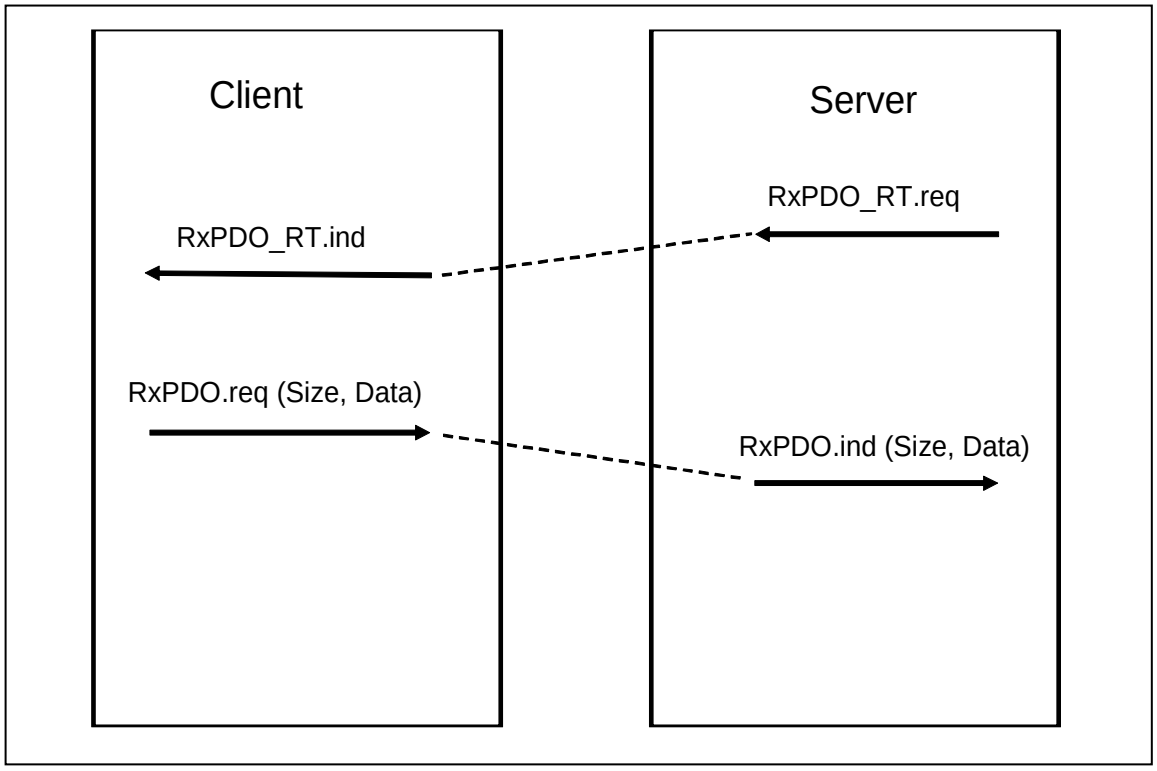

Figure 24 – RxPDO remote transmission sequence

Figure 25 shows the primitives between client and server in case of a TxPDO Remote Transmission service.

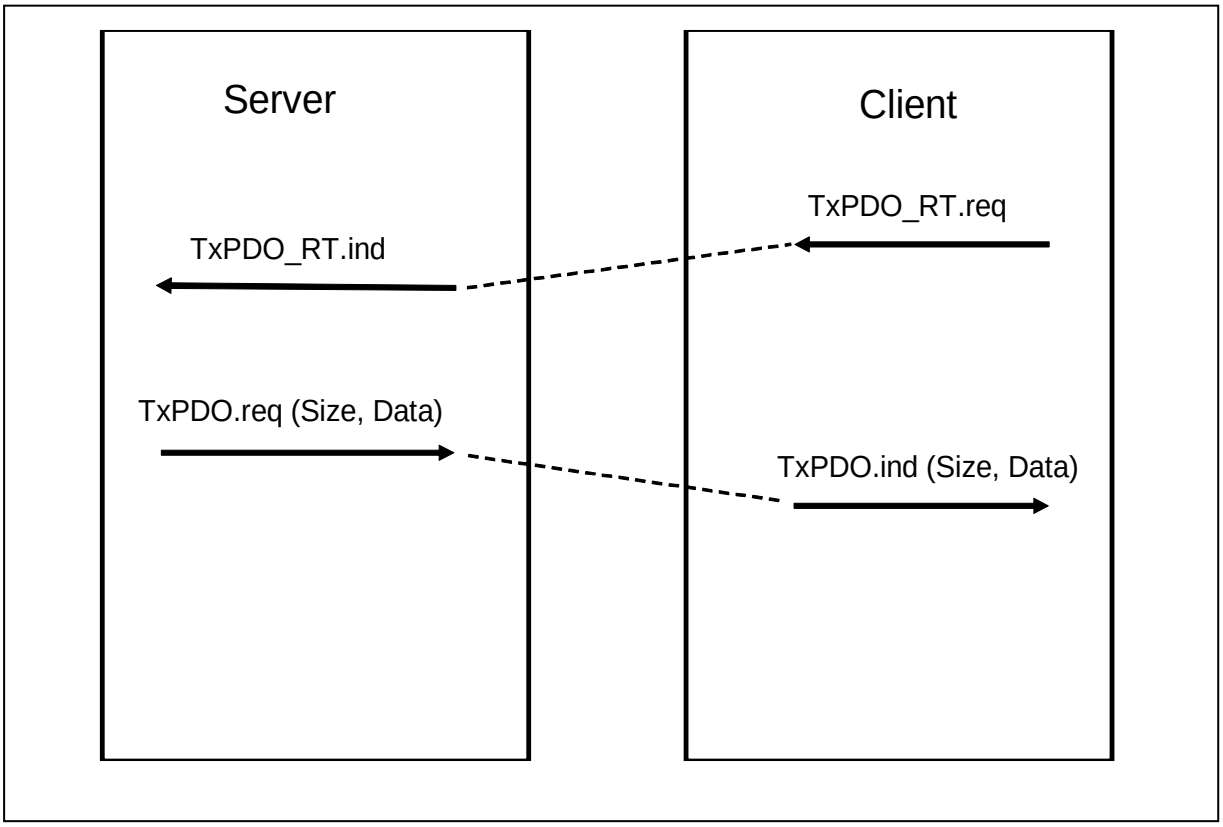

Figure 25 – TxPDO remote transmission sequence

6.1.4.2 CoE class specification

6.1.4.2.1 Formal model

The CoE Object Dictionary object is described by the following template:

ASE:			**CoE**
CLASS:			**CoE Object Dictionary**
CLASS ID:			not used
PARENT CLASS:			TOP
ATTRIBUTES:			
1.	(m)	Key Attribute:	Implicit
2.	(m)	Attribute:	List of Objects
2.1	(m)	Attribute:	Index
2.2	(m)	Attribute:	List of Entry
2.2.1	(m)	Attribute:	Subindex
2.2.2	(m)	Attribute:	Bitlen
2.2.3	(o)	Attribute:	Access
2.2.4	(o)	Attribute:	DataType
2.2.5	(o)	Attribute:	DefaultValue
2.2.6	(o)	Attribute:	MinValue
2.2.7	(o)	Attribute:	MaxValue
2.2.8	(o)	Attribute:	Description
SERVICES:			
1.	(o)	OpsService:	SDO Download Expedited
2.	(o)	OpsService:	SDO Download Normal
3.	(o)	OpsService:	Download SDO Segment
4.	(m)	OpsService:	SDO Upload Expedited
5.	(m)	OpsService:	SDO Upload Normal
6.	(o)	OpsService:	Upload SDO Segment
7.	(m)	OpsService:	Abort SDO Transfer
8.	(o)	OpsService:	Get OD List
9.	(o)	OpsService:	Get Object Description
10.	(o)	OpsService:	Get Entry Description
11.	(o)	OpsService:	OD List Segment
12.	(o)	OpsService:	Object Entry Segment
13.	(o)	OpsService:	Emergency
14.	(o)	OpsService:	RxPDO
15.	(o)	OpsService:	TxPDO
16.	(o)	OpsService:	RxPDO Remote Transmission
17.	(o)	OpsService:	TxPDO Remote Transmission

6.1.4.2.2 Attributes

Implicit

The attribute Implicit indicates that the CoE Object Dictionary object is implicitly addressed by the services.

List of objects
One Object is composed of the following list elements:

Index
This attribute specifies a numerical index of the object.

List of Entry
This attribute consists of the following attributes.

Subindex
This attribute specifies to the reference to the object entry.

Bitlen
This attribute specifies to the length in bits of the object entry value.

NOTE For PDOs, the length must be ≤ 11888.

Access
This attribute specifies the access rights to that entry. Its allowed values are

- RP (Read in Pre-Operational)
- RS (Read in Safe-Operational)
- RO (Read in Operational)
- WP (Write in Pre-Operational)
- WS (Write in Safe-Operational)
- WO (Write in Operational)
- RX (usable for RxPDO)
- TX (usable for TXPDO)

DataType
This optional attribute specifies to the data type of the object entry value.

DefaultValue
This optional attribute specifies to the default value of the object entry value.

MinValue
This optional attribute specifies to the smallest possible value of the object entry value.

MaxValue
This optional attribute specifies to the largest possible value of the object entry value.

Description
This optional attribute gives a description of the object entry value.

6.1.4.2.3 Services

SDO downloadexpedited
Writes up to four octets to the slave.

SDO download normal
Writes up to to a negotiated number of octets to the slave.

Download SDO segment
Writes additional data if the object size is greater than the negotiated number of octets.

SDO upload expedited
Reads up to four octets from the slave.

SDO upload normal
Reads p to to a negotiated number of octets from the slave.

Upload SDO segment
Reads additional data if the object size is greater than the negotiated number of octets.

Abort SDO transfer
Server abort of service in case of an errorneous condition.

Get OD list
Reads a list of available indices.

Get object description
Reads details of an index.

Get entry description
Reads details of a subindex.

OD list segment
Continuation of the list of available indices.

Object entry segment
Continuation of an entry description.

Emergency
Report of unexpected conditions.

RxPDO
Unsolicited transfer of RxPDO Data from master to slave.

TxPDO
Unsolicited transfer of TxPDO Data from slave to master.

RxPDO remote transmission
Unsolicited request from slave to master to transfer of RxPDO Data.

TxPDO remote transmission
Unsolicited request from master to slave to transfer of TxPDO Data.

6.1.4.3 CoE service specification

6.1.4.3.1 Supported services

The CoE ASE defines the services

SDO Download Expedited

SDO Download Normal

Download SDO Segment

SDO Upload Expedited

SDO Upload Normal

Upload SDO Segment

Abort SDO Transfer

Get OD List

Get Object Description

Get Entry Description

OD List Segment

Object Entry Segment

Emergency

RxPDO

TxPDO

RxPDO Remote Transmission

TxPDO Remote Transmission

6.1.4.3.2 SDO download expedited service

The SDO Download Expedited service is used for an expedited transfer of up to 4 octets of data from the client to server. The server answers with the result of the download operation. Table 8 shows the service primitives and parameter of the SDO Download Expedited service.

Table 8 – SDO download expedited

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Index	M	M (=)		
Subindex	M	M (=)		
Data	M	M (=)		
Result(+)			S	S
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Index
This parameter specifies the index in the object dictionary of the object which it to be downloaded. Its range is 0 to 0x9FFF

Subindex
This parameter specifies the subindex in the object dictionary of the object which is to be downloaded.

Data
This parameter specifies the object value, of length 1 to 4 octets, to be downloaded.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.4.3.3 SDO download normal service

The SDO Download service will be used for a single normal transfer of data from the client to the server, if the number of octets to be downloaded fit in the mailbox. Otherwise a segmented transfer will be started. Table 9 shows the service primitives and parameter of the SDO Download Normal service.

Table 9 – SDO download normal

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Index	M	M (=)		
Subindex	M	M (=)		
Complete Access	M	M (=)		
Complete Size	M	M (=)		
Data	M	M (=)		
Result(+)			S	S
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Index
This parameter specifies the index in the object dictionary of the object which is to be downloaded. Its range is 0 to 0x9FFF

Subindex
This parameter specifies the subindex in the object dictionary of the object which is to be downloaded.

Complete access
This Boolean parameter is set to true to indicate that all subindices should be written on that request.

Complete size
This parameter indicates the number of octets to be downloaded to the server. If the Complete Size is greater than the Size parameter the difference number of octets will be downloaded in the subsequent Download SDO Segment services.

Data
This parameter specifies the part of the object value, of length 1 to 1477 octets, to be downloaded.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.4.3.4 Download SDO segment service

The Download SDO Segment service is used to download the additional data from the client to the server, which did not fit in the mailbox with SDO Download Normal service. The master will start as many Download SDO Segment services as data is available. Table 10 shows the service primitives and parameter of the Download SDO Segment service.

Table 10 – Download SDO segment

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Toggle	M	M (=)		
More Follows	M	M (=)		
Data	M	M (=)		
Result(+)			S	S
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Toggle
This Boolean parameter should be toggled with every Download SDO Segment service, starting with zero.

More follows
This Boolean parameter is set to false if this service is the last Download SDO Segment service of a SDO download sequence. It is set to true if at least one Download SDO Segment service of a SDO download sequence is following.

Data
This parameter specifies the part of the object value, of length 1 to 1477 octets, to be downloaded.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.4.3.5 SDO upload expedited service

The SDO Upload Expedited service can be used for an expedited transfer of up to 4 octets of data from the server to client. The server answers with the result of the upload operation and the requested data in case of a successful operation. Table 11 shows the service primitives and parameter of the SDO Upload Expedited service.

Table 11 – SDO upload expedited

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Index	M	M (=)		
Subindex	M	M (=)		
Result(+)			S	S
Data			M	M (=)
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Index
This parameter specifies the index in the object dictionary of the object which is to be uploaded. Its range is 0 to 0x9FFF

Subindex
This parameter specifies the subindex in the object dictionary of the object which is to be uploaded.

Result(+)
This selection type parameter indicates that the service request succeeded.

Data
This parameter specifies the object value, of length 1 to 4 octets, that is being uploaded to the server.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.4.3.6 SDO upload normal service

The SDO Upload Normal service is used for a single normal transfer of data from the server to the client, if the number of octets to be uploaded fit in the mailbox, or to start a segmented transfer with more octets. The server answers with the result of the upload operation and the requested data in case of a successful operation. Table 12 shows the service primitives and parameter of the SDO Upload Normal service.

Table 12 – SDO upload normal

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Index	M	M (=)		
Subindex	M	M (=)		
Complete Access	M	M (=)		
Result(+)			S	S
Complete Size			M	M (=)
Data			M	M (=)
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Index
This parameter specifies the index in the object dictionary of the object which is to be uploaded. Its range is 0 to 0x9FFF

Subindex
This parameter specifies the subindex in the object dictionary of the object which is to be uploaded.

Complete access
This Boolean parameter is set to true to indicate that all subindices should be read on that request.

Result(+)
This selection type parameter indicates that the service request succeeded.

Complete size
This parameter indicates the number of octets to be uploaded to the server. If the Complete Size is greater than the Size parameter the difference number of octets will be uploaded in the subsequent Upload SDO Segment services.

Data
This parameter specifies the part of the object value, of length 1 to 1477 octets, that is being uploaded to the server.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.4.3.7 Upload SDO segment service

The Upload SDO Segment service is used to upload the additional data from the server to the client, which did not fit in the mailbox with the SDO Upload service response. The client will start so many Upload SDO Segment services until all data is uploaded from the server. Table 13 shows the service primitives and parameter of the Upload SDO Segment service.

Table 13 – Upload SDO segment

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Toggle	M	M (=)		
Result(+)			S	S
More Follows			M	M (=)
Data			M	M (=)
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Toggle
This Boolean parameter should be toggled with every Upload SDO Segment service, starting by zero.

Result(+)
This selection type parameter indicates that the service request succeeded.

More follows
This Boolean parameter is set to false if this service is the last Upload SDO Segment service of a SDO upload sequence. It is set to true if at least one Upload SDO Segment service of a SDO upload sequence is following.

Data
This parameter specifies the part of the object value, of length 1 to 1477 octets, that is being uploaded to the server.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.4.3.8 Abort SDO transfer service

The Abort SDO Transfer service is used from the server instead of a service response to a download or upload service in case of an error. Table 14 shows the service primitives and parameter of the Abort SDO Transfer service.

Table 14 – Abort SDO transfer

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Index	M	M (=)
Subindex	M	M (=)
Reason	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Index
This parameter specifies the index in the object dictionary of the object whose transfer is to be aborted. Its range is 0 to 0x9FFF

Subindex
This parameter specifies the subindex in the object dictionary of the object whose transfer is to be aborted.

Reason
This parameter specifies the abort reason.

6.1.4.3.9 Get OD list service

With the Get OD List service lists of objects existing in the object dictionary of the server can be read. If the response does not fit in the mailbox the response is sent with multiple fragments as described above. Table 15 shows the service primitives and parameter of the Get OD service.

Table 15 – Get OD list

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
List Type	M	M (=)		
Result(+)			S	S
Incomplete			M	M (=)
Fragments Left			M	M (=)
Index List Entry			M	M (=)
Index List			M	M (=)
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

List type
This parameter should restrict the Index List Entries to objects that have at least one entry with the appropriate object access attribute set. Its allowed values are

— ALL

— RX (usable for RxPDO)

— TX (usable for TXPDO)

— BU (Back up values)

— ST (Setting values)

Result(+)
This selection type parameter indicates that the service request succeeded.

Incomplete
This Boolean parameter is set to false if the index list is completely contained in the data parameter and set to true if OD List Segment services are following.

Fragments left
This parameter is set to zero if the index list is completely contained in the data parameter and set to the appropriate number of following OD List Segment services.

Index list entry
This parameter indicates the number of entries in the Index list, between 1 and 736.

Index list
This array parameter specifies the first part of the object value which is to be uploaded.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.4.3.10 OD list segment service

With the OD List Segment service additional lists of objects existing in the object dictionary of the server can be transferred to the client. Table 15 shows the service primitives and parameter of the OD List Segment service.

Table 16 – OD list segment

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Incomplete	M	M (=)
Fragments Left	M	M (=)
List Type	M	M (=)
Index List Entry	M	M (=)
Index List	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Incomplete
This Boolean parameter is set to false if the index list is completely contained in the data parameter and set to true if OD List Segment services are following.

Fragments left
This parameter is set to zero if the index list is completely contained in the data parameter and set to the appropriate number of following OD List Segment services.

List type
This parameter should restrict the Index List Entries to objects that have at least one entry with the appropriate object acces attribute set. Its allowed values are

— ALL

— RX (usable for RxPDO)

— TX (usable for TXPDO)

— BU (Back up values)

— ST (Setting values)

Index list entry
This parameter indicates the number of entries in the Index list, between 1 and 736.

Index list
This array parameter specifies the first part of the object value which is to be uploaded.

6.1.4.3.11 Get object description service

With the Get Object Description service an object description existing in the object dictionary of the server can be read. If the response does not fit in the mailbox the response is sent with multiple fragments as described above. Table 17 shows the service primitives and parameter of the Get Object Description service.

Table 17 – Get object description

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Index	M	M (=)		
Result(+)			S	S
Incomplete			M	M (=)
Fragments left			M	M (=)
Max Subindex			M	M (=)
Object Code			M	M (=)
Name			M	M (=)
Result(–)			S	S
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Index
This parameter specifies the index in the object dictionary of the object description which is to be read. Its range is 0 to 0x9FFF

Result(+)
This selection type parameter indicates that the service request succeeded.

Incomplete
This Boolean parameter has the value false.

Fragments left
This parameter is set to zero.

Max subindex
This parameter is set to the greatest subindex number.

Object code
This parameter specifies the general type information. Its allowed values are

— VAR

— ARRAY

— RECORD

Name
This parameter specifies the Name of the object.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.4.3.12 Get entry description service

With the Get Entry Description service an entry description existing in the object dictionary and addressed by index and subindex of the server can be read. If the response does not fit in the mailbox the response is sent with multiple fragments as described above. Table 18 shows the service primitives and parameter of the Get Entry Description service. If the response parameters do not fit in a single transfer unit of the underlying communication system, the optional parameters can be transmitted in following Entry Description Segment services.

Table 18 – Get entry description

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Index	M	M (=)		
Subindex	M	M (=)		
Request Access Rights	M	M (=)		
Request Object Category	M	M (=)		
Request Mapping Info	M	M (=)		
Request Unit Type	M	M (=)		
Request Default Value	M	M (=)		
Request Min Value	M	M (=)		
Request Max Value	M	M (=)		
Request Description	M	M (=)		
Result(+)			S	S
Incomplete			M	M (=)
Fragments left			M	M (=)
Data type			M	M (=)
Bit Length			M	M (=)
Access Rights			O	O(=)
Object Category			O	O(=)
Mapping Info			O	O(=)
Unit Type			O	O(=)
Default Value			O	O(=)
Min Value			O	O(=)
Max Value			O	O(=)
Description			O	O(=)
Result(–)			S	S

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Index
This parameter specifies the index in the object dictionary of the object entry description which is to be read. Its range is 0 to 0x9FFF

Subindex
This parameter specifies the subindex of the object entry description which is to be read.

Request access rights
This Boolean parameter is set to true if the parameter Access Rights is requested to be included in the response.

Request object category
This Boolean parameter is set to true if the parameter Object Category is requested to be included in the response.

Request mapping info
This Boolean parameter is set to true if the parameter Mapping Info is requested to be included in the response.

Request unit type
This Boolean parameter is set to true if the parameter Unit Type is requested to be included in the response.

Request default value
This Boolean parameter is set to true if the parameter Default Value is requested to be included in the response.

Request min value
This Boolean parameter is set to true if the parameter Min Value is requested to be included in the response.

Request max value
This Boolean parameter is set to true if the parameter Max Value is requested to be included in the response.

Request description
This Boolean parameter is set to true if the parameter Description is requested to be included in the response.

Result(+)
This selection type parameter indicates that the service request succeeded.

Incomplete
This Boolean parameter is set to false if the object entry is completely contained in the data parameter and set to true if Object Entry Segment services are following.

Fragments left
This parameter is set to zero if the object entry is completely contained in the data parameter and set to the appropriate number of following Object Entry Segment services.

Data type
This parameter points to the index of the data type in the object dictionary.

Bit length
This parameter defines the length in bits of the object entry value.

Access rights
This optional parameter defines the access right to the entry.

Object category
This optional parameter defines the object category of the object entry. Its allowed values are

— M (mandatory)

— O (optional)

Unit type
This optional parameter points to the index of the data type in the object dictionary.

Default value
This optional parameter shows the default value for this object entry.

Min value
This optional parameter shows the smallest possible value for this object entry.

Max value
This optional parameter shows the largest possible value for this object entry.

Description
This optional parameter specifies a textual description of the object entry.

Result(–)
This selection type parameter indicates that the service request failed.

6.1.4.3.13 Object entry segment service

With the Object Entry Segment service additional lists of objects existing in the object dictionary of the server can be transferred to the client. Table 19 shows the service primitives and parameter of the Object Entry Segment service. It is recommended that a continuation should follow the parameter boundary.

Table 19 – Object entry segment

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Incomplete	M	M (=)
Fragments left	M	M (=)
Index	M	M (=)
Subindex	M	M (=)
Unit Type	O	O(=)
Default Value	O	O(=)
Min Value	O	O(=)
Max Value	O	O(=)
Description	O	O(=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Incomplete
This Boolean parameter is set to false if the object entry is completely contained in the data parameter and set to true if Object Entry Segment services are following.

Index
This parameter specifies the index in the object dictionary of the object entry description which has been read. Its range is 0 to 0x9FFF

Subindex
This parameter specifies the subindex of the object entry description which has been read.

Fragments left
This parameter is set to zero if the index list is completely contained in the data parameter and set to the appropriate number of following Object Entry Segment services.

Unit type
This optional parameter points to the index of the data type in the object dictionary.

Default value
This optional parameter shows the default value for this object entry.

Min value
This optional parameter shows the smallest possible value for this object entry.

Max value
This optional parameter shows the largest possible value for this object entry.

Description
This optional parameter specifies a textual description of the object entry.

6.1.4.3.14 Emergency service

The Emergency service is used from the server to transmit diagnostic messages to the client. Each diagnostic event which was transmitted by the server to the client should be transmitted again with the reset error code when the diagnostic event disappears. Table 20 shows the service primitives and parameter of the Emergency service.

Table 20 – Emergency

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Error Code	M	M (=)
Error Register	C	C (=)
Data	O	O(=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server.

Error code
This parameter specifies the emergency error code.

Error register
This conditional parameter specifies the emergency error register.

Data
This optional parameter specifies the emergency error details.

6.1.4.3.15 RxPDO service

The RxPDO service is used from the client to transmit output data to the server. The mapping and the size of the data is defined in the PDO mapping objects. Table 21 shows the service primitives and parameter of the RxPDO service.

Table 21 – RxPDO

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Index	M	M (=)
Data	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server.

Index
This parameter specifies the index in the object dictionary of the RxPDO object to be written. It's permitted range is 0x1600 to 0x17FF.

Data
This parameter specifies the object value, of length 1 to 1472 octets, to be written to the server.

6.1.4.3.16 TxPDO service

The TxPDO service is used from the server to transmit input data to the client. The mapping and the size of the data is defined in the PDO mapping objects. Table 22 shows the service primitives and parameter of the TxPDO service.

Table 22 – TxPDO

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Index	M	M (=)
Size	M	M (=)
Data	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server.

Index
This parameter specifies the index in the object dictionary of the TxPDO object which is to be transfered. Its range is 0x1A00 to 0x1BFF.

Size
This parameter indicates the number of octets to be transferred to the client. Its range is 1 to 1472.

NOTE The size should be identical to that of the PDO mapping object specified by the index parameter.

Data
This parameter specifies the object value which is to be transferred to the client.

6.1.4.3.17 RxPDO remote transmission service

The RxPDO_RT service is used from the server to request output data from the client. Table 23 shows the service primitives and parameter of the TxPDO Remote Transmission service.

Table 23 – RxPDO remote transmission

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Index	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server.

Index
This parameter specifies the index in the object dictionary of the RxPDO object which is to be transmitted. Its range is 0x1600 to 0x17FF.

6.1.4.3.18 TxPDO remote transmission service

The TxPDO_RT service is used from the client to request input data from the server. Table 24 shows the service primitives and parameter of the TxPDO Remote Transmission service.

Table 24 – TxPDO remote transmission

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Index	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server.

Index
This parameter specifies the index in the object dictionary of the TxPDO object which is to be transmitted. Its range is 0x1A00 to 0x1BFF.

6.1.5 EoE ASE

6.1.5.1 Overview

In addition to the already described addressing mode for the communication with the devices, an Ethernet fieldbus is also expected to feature standard IP-based protocols such as TCP/IP, UDP/IP and all higher protocols based on these (HTTP, FTP, SNMP etc.). Ideally, individual Ethernet frames should be transferred transparently, since this avoids restrictions with regard to the protocols to be transferred.

There are two different basic approaches for transferring acyclic Ethernet frames in cyclic fieldbus mode. In the first variant, an appropriate time slice is allocated, in which the acyclic Ethernet frames can be embedded. This time slice has to be chosen sufficiently large to be able to accommodate complete Ethernet frames. The maximum frame size of ISO/IEC 8802-3 is 1526 octets (1530 in case of tagged frames), corresponding to approximately 125 µs transmission time (including inter-frame gap) at 100 Mbit/s. The minimum resulting cycle time for systems using this variant is approximately 200-250 µs. A reduction of the maximum frame size often leads to problems. While the IP protocol allows fragmentation in principle, this is often not recommended, and there may be protocols that do not support fragmentation. Data consistency problems may also occur, particularly in UDP/IP transfers.

EtherCAT utilizes the second variant, in which the Ethernet frames are tunnelled and re-assembled in the associated device, before being relayed as complete Ethernet frames. This procedure does not restrict the achievable cycle time, since the fragments can be optimized according to the available bandwidth (Ethernet fragmentation instead of IP fragmentation). In this case, EtherCAT defines a standard channel, which in principle enables any device to participate in the normal Ethernet traffic. In an intelligent drive controller that exchanges process data with a cycle time of 100 µs, for example, an HTTP server can be integrated that features its own diagnostics interface in the form of web pages.

Another application for transferred Ethernet frames will be devices with non EtherCAT Ethernet ports. They offer standard Ethernet ports at any location within the system, through which any Ethernet device may be connected. For example, this may be a service computer that communicates directly with the control and queries the web page of an intelligent device, or simply routes it to the intranet or internet via the control. The master also features software-integrated switch functionality, which is responsible for the routing of the individual Ethernet frames from and to the devices and the IP stack of the host operating system. The switch functionality is identical with that of a standard layer 2 switch and responds to the Ethernet addresses used irrespective of the protocol.

There are special EoE Parameters needed for EoE end devices within a EtherCAT segment. This parameter can be conveyed with the EoE services Set IP Parameter and Set MAC Filter.

The primitives of the EoE services are mapped to the primitives of the mailbox transmission services.

Figure 26 shows the primitives between client and server in case of a successful EoE sequence. The frame number has to be the same for all Ethernet fragments in an EoE sequence.

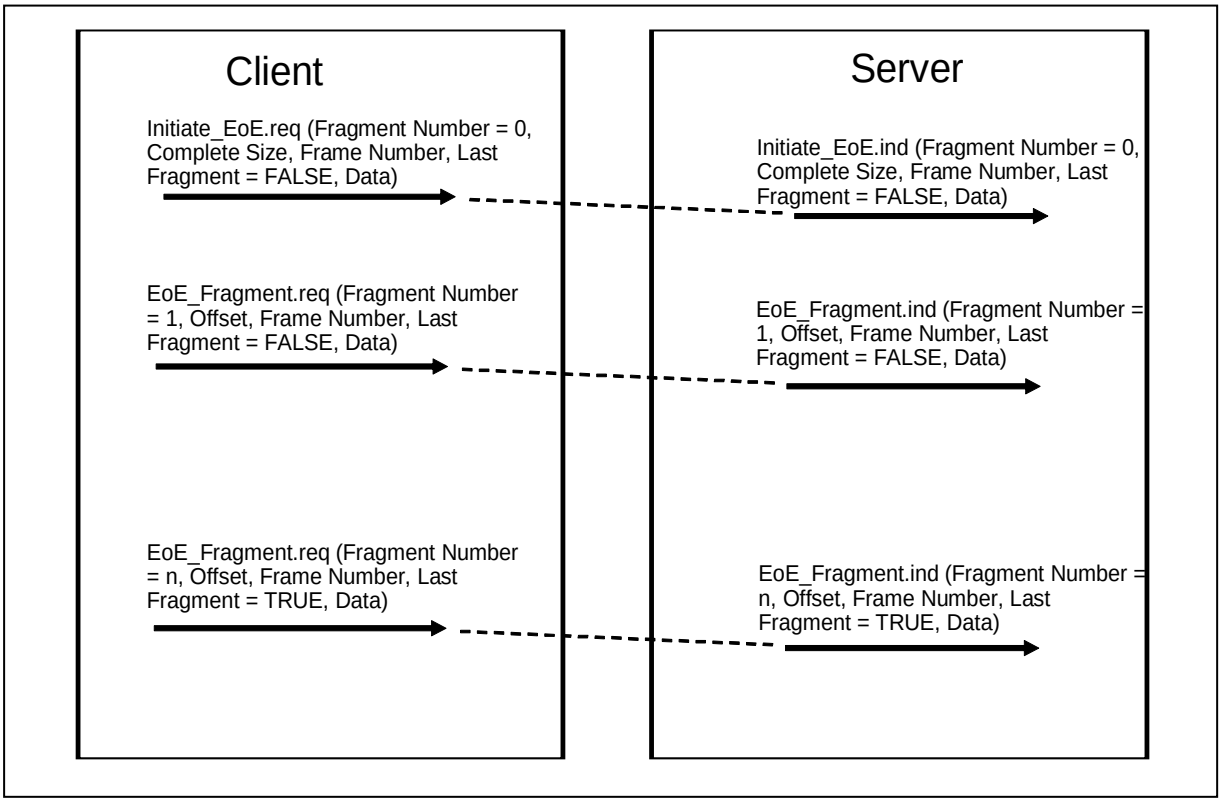

Figure 26 – EoE sequence

6.1.5.2 EoE class specification

6.1.5.2.1 Formal model

The EoE object is described by the following template:

ASE:			**EoE**
CLASS:			**EoE Parameter**
CLASS ID:			not used
PARENT CLASS:			TOP
ATTRIBUTES:			
1.	(m)	Key Attribute:	Implicit
2.	(m)	Attribute:	Virtual MAC Address
3.	(o)	Attribute:	IP Address Info
3.1	(o)	Attribute:	IP Address
3.2	(o)	Attribute:	IP Subnet mask
3.3	(o)	Attribute:	Default Gateway
3.4	(o)	Attribute:	DNS IP Address
3.5	(o)	Attribute:	Domain Name
4.	(o)	Attribute:	Filter Info
4.1	(o)	Attribute:	Broadcast Forwarding
4.2	(o)	Attribute:	List of MAC Address Filter
4.2.1	(o)	Attribute:	MAC Address
4.3	(o)	Attribute:	List of MAC Filter Mask
4.3.1	(o)	Attribute:	Filter Mask
5.	(m)	Attribute:	List of Transmit Frames
5.1	(m)	Attribute:	Frame
5.2	(o)	Attribute:	Port
5.3	(o)	Attribute:	Timestamp
6.	(m)	Attribute:	List of Receive Frames
6.1	(m)	Attribute:	Frame
6.2	(o)	Attribute:	Port
6.3	(o)	Attribute:	Timestamp
SERVICES:			
1.	(m)	OpsService:	Initiate EoE
2.	(m)	OpsService:	EoE Fragment
3.	(o)	OpsService:	Set IP Parameter
4.	(o)	OpsService:	Set MAC Filter

6.1.5.2.2 Attributes

Implicit

The attribute Implicit indicates that the EoE Parameter object is implicitly addressed by the services.

Virtual MAC address
This attribute specifies a MAC Address according to ISO/IEC 8802-3 used for identifying a node in an Ethernet network.

IP Address info
One Object is composed of the following list elements:

IP address
This optional attribute consists of an IP address according to RFC 791 to identify an IP node in the internet.

Subnet mask
This optional attribute an Subnet mask specify a group of addresses belonging to an IP subnet.

Default gateway
This optional attribute consist of an IP address according to RFC 791 identifying the next router in the internet.

DNS IP address
This optional attribute consists of an IP address of a station which supports the translation of Domain names to IP addresses.

Domain name
This optional attribute is used to store the domain name of the node.

Filter Info
One Object is composed of the following list elements:

Broadcast forwarding
This optional Boolean attribute is set to true if broadcast PDUs should be forwarded

List of MAC address filter
This attribute consists of the following attributes.

MAC address
This attribute specifies a MAC Address according to ISO/IEC 8802-3 used for forwarding frames with a match in the MAC Address filter match.

List of MAC filter mask
This attribute consists of the following attributes.

MAC address
This attribute specifies a MAC Address mask according to ISO/IEC 8802-3 used to form a MAC address group for address filtering.

List of transmit frames
This attribute consists of the following attributes.

Frame
This attribute specifies a complete Ethernet frame.

Port
This optional attribute specifies the send port of the frame. Its range is 1 to 15.

Timestamp
This optional attribute specifies the timestamp in units of ns of the local time at the send of a frame.

List of receive frames
This attribute consists of the following attributes.

Frame
This attribute specifies a complete Ethernet frame that is error free.

Port
This optional attribute specifies the receive port of the frame. Its range is 1 to 15.

Timestamp
This optional attribute specifies the timestamp in units of ns of the local time at the receipt of a frame.

6.1.5.2.3 Services

Initiate EoE
Start to transfer Ethernet frame – a timestamp response can be requested.

EoE fragment
Continuation of an Ethernet frame that can not be sent within one EtherCAT Datagram

Set IP parameter
Set up IP parameters such as MAC address and IP address.

Set MAC filter
Set up Filters for MAC addresses or Groups of MAC addresses.

6.1.5.3 EoE service specification

6.1.5.3.1 Supported services

The EoE ASE defines the services

Initiate EoE

EoE Fragment

Set IP Parameter

Set MAC Filter

6.1.5.3.2 Initiate EoE

The Initiate EoE service is used to transmit the first fragment of an Ethernet frame. Table 25 shows the service primitives and parameter of the EoE Initiate service.

Table 25 – Initiate EoE

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Port	M	M (=)		
Time Appended	M	M (=)		
Time Requested	M	M (=)		
Frame Number	M	M (=)		
Complete Size	M	M (=)		
Last Fragment	M	M (=)		
Data	M	M (=)		
Timestamp	C	C (=)		
Result			C	C (=)
Timestamp			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server.

Port
This parameter specifies a receiving/transmitting port number of the frame. Its range is 0 to 15; 0 indicates that the port is unspecified.

Time appended
This Boolean parameter is set to true if time stamp of the time of receipt (beginning of first bit of DA) is appended. It is set to false otherwise.

Time requested
This Boolean parameter is set to true if the time stamp of the time of sending (beginning of first bit of DA) should be sent in the response. It is set to false otherwise.

Frame number
This parameter specifies a sequence number of the frame modulo 8.

Complete size
This parameter indicates the number of 32 octet blocks to be transferred with this frame (excluding Preamble, SFD, FCS). Its range is 2 to 48.

Last fragment
This Boolean parameter is set to true if this service is the last fragment of the Ethernet frame. It is set to false if at least one EoE Fragment service is following.

Data
This parameter specifies the object value, of length 64 to 1472 octets, to be transferred to the client.

TimeStamp
This conditional parameter specifies a timestamp of the receipt of the frame in ns (begin of DA is timestamp-point)

Result
This parameter indicates an optional service response.

TimeStamp
This parameter specifies a timestamp of the transmission of the frame with the specified frame number

6.1.5.3.3 EoE Fragment

The EoE Fragment service is used to transmit the fragments of an Ethernet frame following the Initiate EoE service if the Ethernet frame to be transmitted is larger than the data parameter of the Initiate EoE service. Table 26 shows the service primitives and parameter of the EoE Fragment service.

Table 26 – EoE fragment

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Port	M	M (=)
Time Appended	M	M (=)
Frame Number	M	M (=)
Offset	M	M (=)
Last Fragment	M	M (=)
Data	M	M (=)
Time Stamp	C	C (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server.

Port
This parameter specifies a receiving/transmitting port number of the frame. Its range is 0 to 15; 0 indicates that the port is unspecified.

Time appended
This Boolean parameter is set to true if time stamp of the time of receipt (beginning of first bit of DA) is appended. It is set to false otherwise.

Frame number
This parameter is the same as the parameter Frame Number in the related Initiate EoE service.

Offset
This parameter multiplied with 32 indicates the offset in the Ethernet frame of the first data octet to be transferred with this frame (excluding Preamble, SFD). Its range is 2 to 47.

Last fragment
This Boolean parameter is set to true if this service is the last fragment of the Ethernet frame. It is set to false if at least one EoE Fragment service is following.

Data
This parameter specifies the object value, of length 64 to 1472 octets, to be transferred to the client.

TimeStamp
This conditional parameter specifies a timestamp of the receipt of the frame in ns (begin of DA is timestamp-point)

6.1.5.3.4 Set IP parameter

The Set IP Parameter service is used transfer IP Parameters from the client to server. The server answers with the result of the set operation. Table 27 shows the service primitives and parameter of the Set IP Parameter service.

Table 27 – Set IP parameter

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
MAC Address	O	O(=)		
IP Address	O	O(=)		
Subnet Mask	O	O(=)		
Default Gateway	O	O(=)		
DNS Server	O	O(=)		
DNS Name	O	O(=)		
Result(+)			S	S (=)
Result(–)			S	S (=)
Reason			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

MAC address
This optional parameter specifies a MAC Address according to ISO/IEC 8802-3 used for identifying a node in an Ethernet network.

IP address
This optional parameter consists of an IP address according to RFC 791 to identify an IP node in the internet.

Subnet mask
This optional parameter consists of an IP subnet mask according to RFC 791 to identify an IP subnet in the internet.

Default gateway
This optional parameter consists of an IP address according to RFC 791 to identify the standard router of this subnet to the internet.

DNS IP address
This optional parameter consists of an IP address of a station which supports the translation of Domain names to IP addresses.

Domain name
This optional parameter is used to store the domain name of the node.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

Reason
This parameter indicates the reason for the service failure.

6.1.5.3.5 Set address filter

The Set Address Filter service is used transfer Address Filters from the client to server. The server answers with the result of the set operation. Table 28 shows the service primitives and parameter of the Set Address Filter service.

Table 28 – Set address filter

Parameter name	Req	Ind	Rsp	Cnf
Argument				
Address	M	M (=)		
Broadcast Forwarding	O	O(=)		
MAC Address Filter1	O	O(=)		
..	O	O(=)		
MAC Address Filter16	O	O(=)		
MAC Filter Mask1	O	O(=)		
..	O	O(=)		
MAC Filter Mask4	O	O(=)		
Result(+)			S	S (=)
Result(–)			S	S (=)
Reason			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the source station if a master is the client or the station address of the destination if a slave is the client to allow slave to slave communication.

Broadcast forwarding
This optional Boolean parameter is set to true to enable the broadcast filter.

MAC address Filter 1 to 16
These optional parameters contain a MAC Addresses according to ISO/IEC 8802-3 used for filtering frames not dedicated to this set of addresses in an Ethernet network.

MAC filter mask 1 to 4
These optional parameters contain a MAC Addresses according to ISO/IEC 8802-3 used to mask unused bits. The Mask coresponds to the Filter with the same index.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

Reason
This parameter indicates the reason for the service failure.

6.1.6 FoE ASE

6.1.6.1 Overview

The file access mailbox command specifies a standard way to download a firmware or any other files from a client to a server or to upload a firmware or any other files from a server to a client. The protocol is similar to the TFTP protocol (trivial file transfer protocol). Both sides can initiate a read or write request via a read or write command.

The primitives of the FoE services are mapped to the primitives of the mailbox transmission services.

Figure 27 shows the primitives between client and server in case of a successful FoE Read sequence.

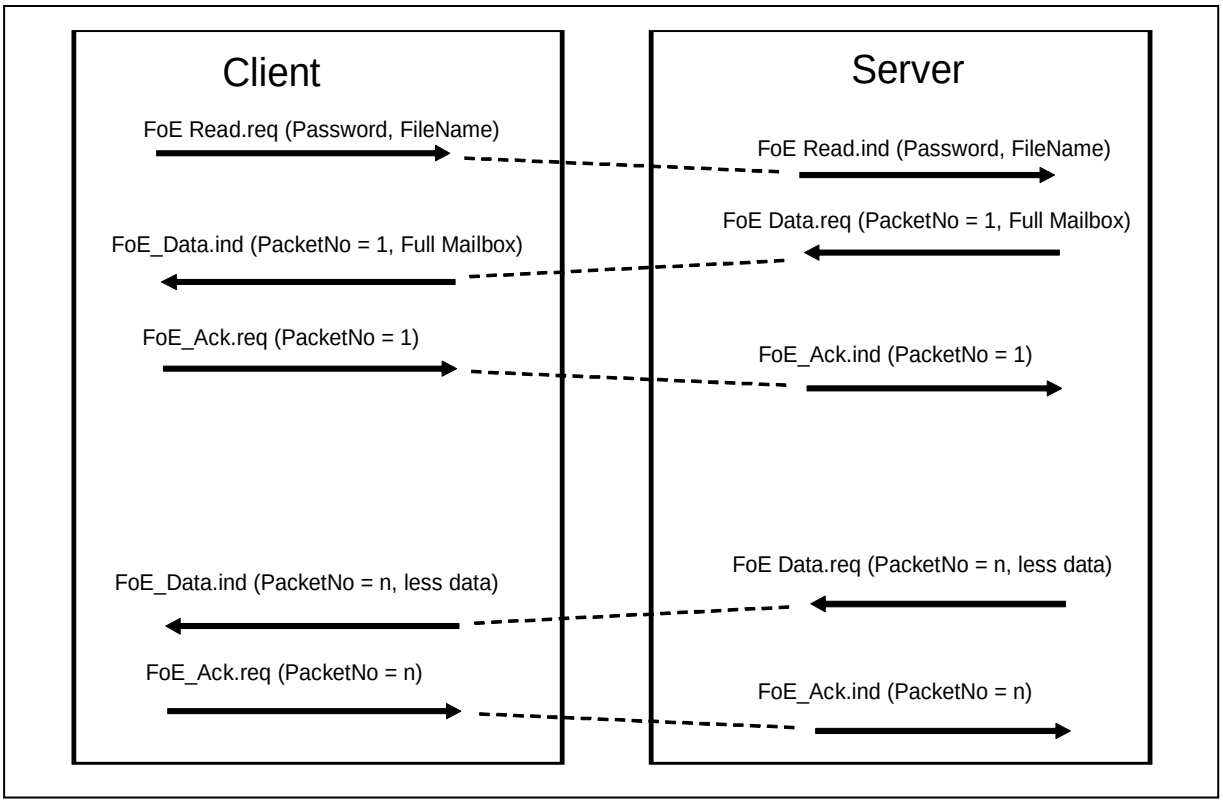

Figure 27 – FoE read sequence with success

Figure 28 shows the primitives between client and server in case of an unsuccessful FoE Read sequence.

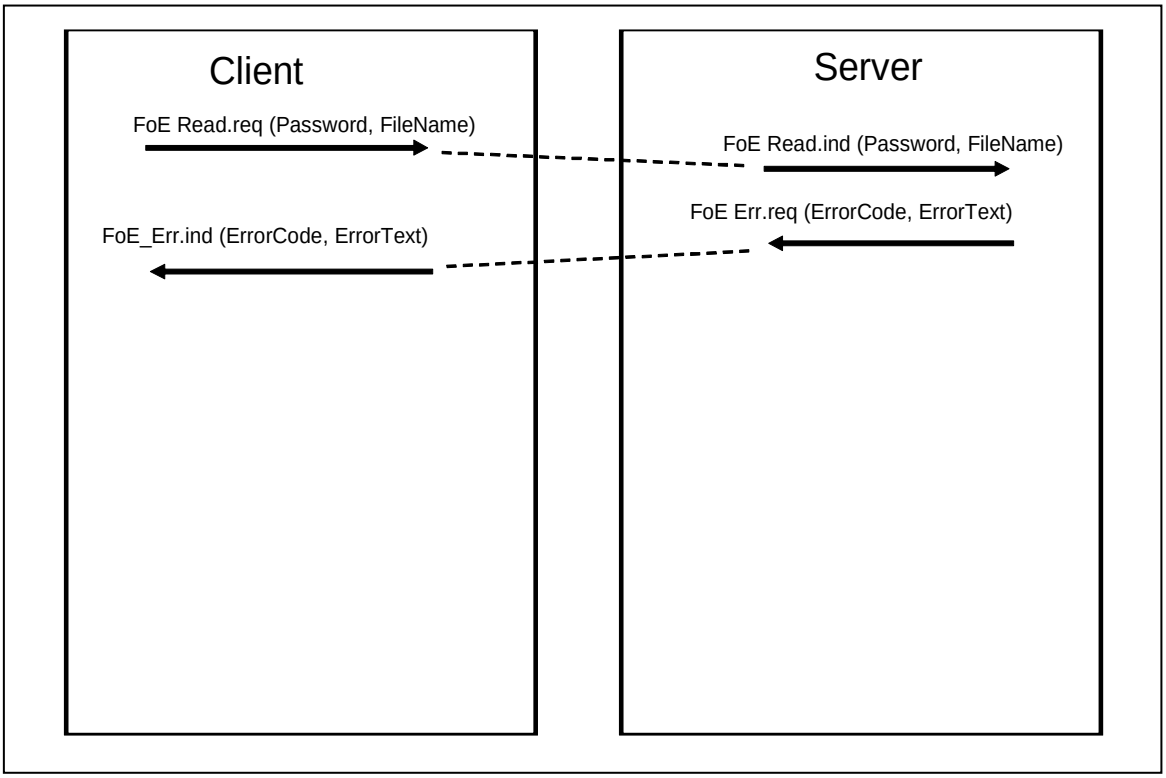

Figure 28 – FoE read sequence with error

Figure 29 shows the primitives between client and server in case of a successful FoE Write sequence.

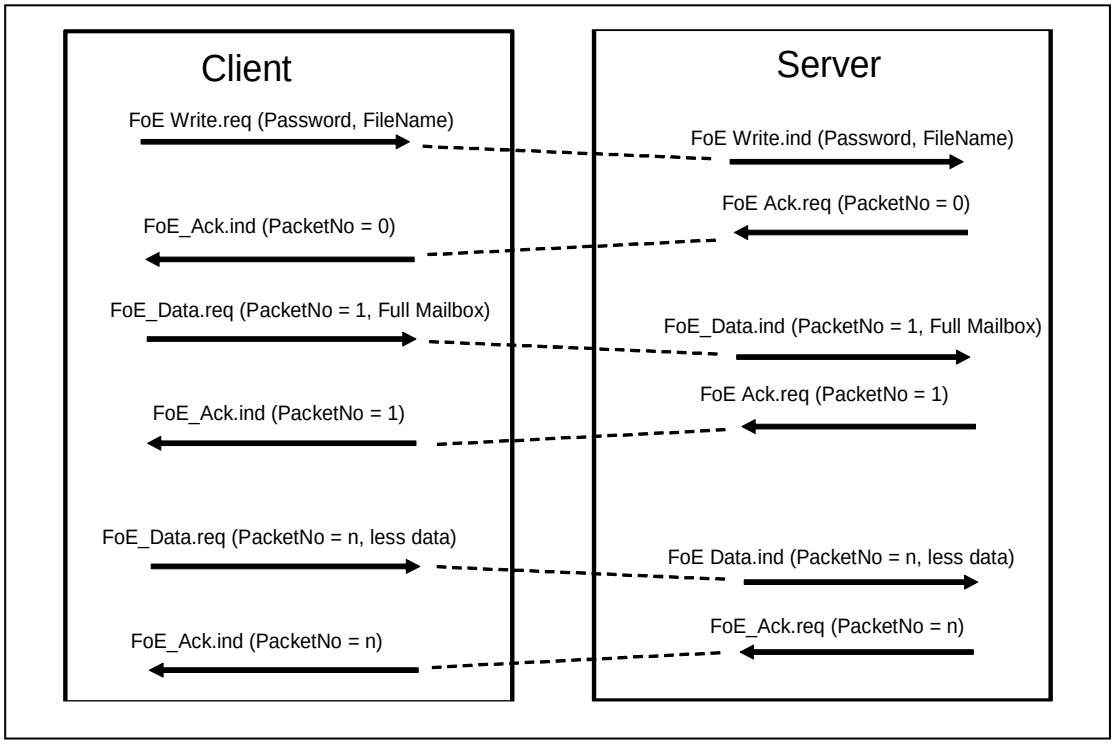

Figure 29 – FoE write sequence with success

Figure 30 shows the primitives between client and server in case of an unsuccessful FoE Write sequence.

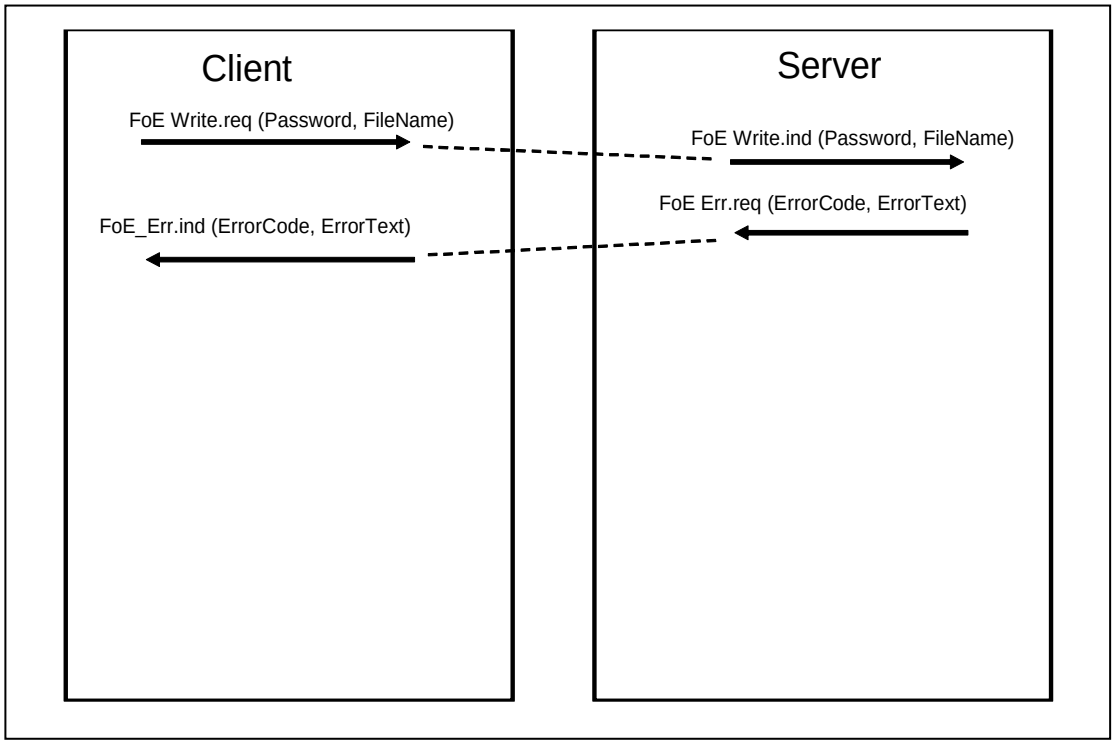

Figure 30 – FoE write sequence with error

Figure 31 shows the primitives between client and server in case of a FoE Write sequence with a busy Server interaction.

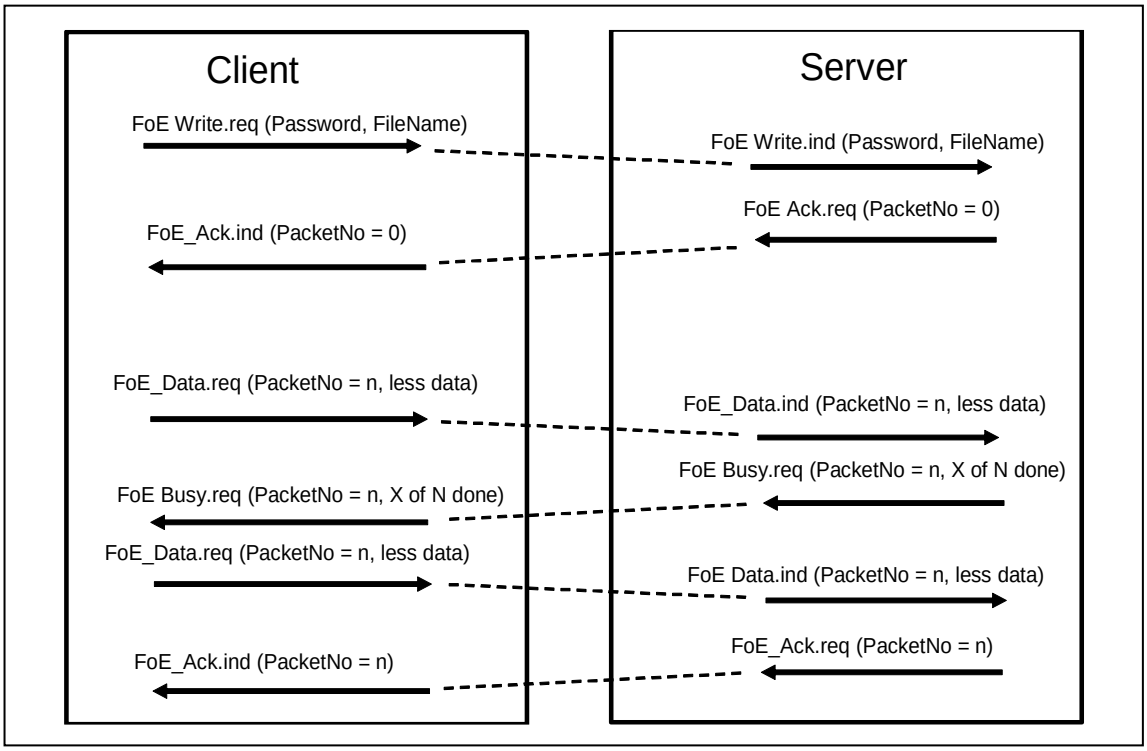

Figure 31 – FoE write sequence with busy

6.1.6.2 FoE class specification

6.1.6.2.1 Formal model

The FoE file object is described by the following template:

ASE:			**FoE**
CLASS:			**FoE File**
CLASS ID:			not used
PARENT CLASS:			TOP
ATTRIBUTES:			
1.	(m)	Key Attribute:	File Name
2.	(o)	Attribute:	Password
3.	(m)	Attribute:	Download Active
5.	(m)	Attribute:	File Info
SERVICES:			
1.	(o)	OpsService:	FoE Read
2.	(o)	OpsService:	FoE Write

6.1.6.2.2 Attributes

File name
This attribute specifies a file name to identify a file object.

Password
This optional attribute specifies a Password to prevent a file to be overwritten accidentally.

Download active
This Boolean attribute is set to true if a download is in progress and set to false otherwise.

File data
This attribute specifies a file data.

6.1.6.2.3 Services

FoE read
Initiates data transfer from the server to the client

FoE write
Initiates data transfer from the client to the server

FoE data
Conveys a piece of data

FoE ack
Acknowledge the conveyance of data

FoE busy
Reports a delay in delivery of data or acknowledgement

FoE error
Reports an error in an FoE service

6.1.6.3 FoE service specification

6.1.6.3.1 Supported services

The FoE ASE defines the services

FoE Read

FoE Write

6.1.6.3.2 FoE read

The FoE Read service is used to read a file from a server. Table 29 shows the service primitives and parameter of the FoE Read service.

Table 29 – FoE read

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Password	O	O(=)
File Name	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server if the data are sent to the slave and the address of the master otherwise.

Password
This optional parameter specifies a password for the read access.

File name
This parameter indicates the name of the file to be read.

6.1.6.3.3 FoE write

The FoE Write service is used to write a file to a server. Table 30 shows the service primitives and parameter of the FoE Write service.

Table 30 – FoE write

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Password	O	O(=)
File Name	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server if the data are sent to the slave and the address of the master otherwise.

Password
This optional parameter specifies a password for the write access.

File name
This parameter indicates the name of the file to be written.

6.1.6.3.4 FoE data

The FoE Data service is used to transmit the file data from the server in case of a read or from the client in case of a write. The first FoE Data always starts with a Packet Number of one incrementing with every following FoE Data. Table 31 shows the service primitives and parameter of the FoE Data service.

Table 31 – FoE data

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Packet Number	M	M (=)
Size	M	M (=)
Data	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server if the data are sent to the slave and the address of the master otherwise.

Packet number
This parameter specifies a sequence number of the frame starting with 1 with the first FoE Data service of a sequence. The PacketNumber is used as a sequence number.

Size
This parameter indicates the number of octets to be transferred to the client. Its range is 0 to 1472.

Data
This parameter specifies the object value which is to be transferred to the client.

6.1.6.3.5 FoE ack

The FoE Ack service is used to acknowledge a FoE Data. The Packet Number will be always the same as sent with the corresponding FoE Data before. A FoE Write will be acknowledged with a FoE Ack with a Packet Number of zero before the client starts with the first FoE Data. Table 32 shows the service primitives and parameter of the FoE Ack service.

Table 32 – FoE ack

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Packet Number	M	M (=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server if the data are sent to the slave and the address of the master otherwise.

Packet number
This parameter specifies a sequence number of the frame starting with 1 with the first FoE Data service of a sequence. The PacketNumber is used as a sequence number.

6.1.6.3.6 FoE busy

The FoE Busy service is used to indicate the client, that the server is busy to store the written file data. The client will repeat the corresponding FoE Data until the server answers with a FoE Ack. Table 33 shows the service primitives and parameter of the FoE Busy.

Table 33 – FoE busy

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Done	M	M (=)
Entire	M	M (=)
Busy Detail	O	O(=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server if the data are sent to the slave and the address of the master otherwise.

Done
This parameter specifies a percentage of the time elapsed before the FoE Ack can be issued. Its range is zero to 100 (percent).

Entire
This parameter specifies zero or the 100 percent value of done otherwise.

Busy detail
This optional parameter specifies detail information of busy..

6.1.6.3.7 FoE Error

The FoE Error service is used to indicate the client that an error has happened during file operation. Table 34 shows the service primitives and parameter of the FoE Error.

Table 34 – FoE error

Parameter name	Req	Ind
Argument		
Address	M	M (=)
Error Code	M	M (=)
Error Text	O	O(=)

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server if the data are sent to the slave and the address of the master otherwise.

Error code
This parameter indicates the type of error.

Error text
This optional parameter provides an explanation of the error.

6.1.7 MBX ASE

6.1.7.1 Overview

The mailbox commands specify a standard way to access mailbox by profile standards.

6.1.7.2 MBX class specification

6.1.7.2.1 Formal model

The MBX object is described by the following template:

ASE:			**MBX**
CLASS:			**MBX**
CLASS ID:			not used
PARENT CLASS:			TOP
ATTRIBUTES:			
1.	(m)	Key Attribute:	implicit
2.	(m)	Attribute:	Write area
3.	(m)	Attribute:	Read area
SERVICES:			
1.	(m)	OpsService:	Mailbox Read
2.	(m)	OpsService:	Mailbox Write
3.	(m)	OpsService:	Mailbox Read Upd

6.1.7.2.2 Attributes

Write area
This attribute specifies a memory area to store data send from the client to the server.

Read area
This attribute specifies a memory area to store data send from the server to the client.

6.1.7.2.3 Services

Mailbox read
Initiates data transfer from the server to the client

Mailbox write
Initiates data transfer from the client to the server

Mailbox read upd
Updates Read area in the server.

6.1.7.3 MBX service specification

6.1.7.3.1 Supported services

The MBX ASE defines the services

MBX Read

MBX Write

MBX Read Upd

6.1.7.3.2 MBX read

The MBX Read service is used to read data from a server. It can be used for profiles to convey data from the slave to the master. Table 35 shows the service primitives and parameter of the MBX Read service.

Table 35 – MBX read

Parameter name	Req	Ind	Cnf
Argument			
Address	M	M (=)	
Channel	O	O(=)	
Priority	O	O(=)	
Type	M	M (=)	
Cnt	M	M (=)	
Result(+)			S
Data			M
Result(–)			S
Reason			M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.			

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the master.

Channel
This optional parameter specifies a a channel number in the range 0 to 63

Priority
This optional parameter specifies one of four priorities

Type
This parameter specifies the type of the mailbox service.

Cnt
This parameter specifies a service counter

Result(+)
This selection type parameter indicates that the service request succeeded.

Data
This parameter specifies the data that was read

Result(–)
This selection type parameter indicates that the service request failed.

Reason
This parameter reports the reason of the failure. Possible values are

— No Slave Reaction

— No Reply Message

6.1.7.3.3 MBX write

The MBX Write service is used to write data to a server. It can be used for profiles to convey data from the master to the slave. Table 36 shows the service primitives and parameter of the MBX Write service.

Table 36 – MBX write

Parameter name	Req	Ind	Cnf
Argument			
Address	M	M (=)	
Channel	O	O(=)	
Priority	O	O(=)	
Type	M	M (=)	
Cnt	M	M (=)	
Length	M	M (=)	
Service Data	M	M (=)	
Result(+)			S
Result(–)			S
Reason			M

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server if the data are sent to the slave and the address of the master otherwise.

Channel
This optional parameter specifies a channel number.

Allowed values: 0 to 63

Priority
This optional parameter specifies one of four priorities.

Type
This parameter specifies the type of the mailbox service.

Cnt
This parameter specifies a service counter

Length
This parameter specifies the length of the service data. Its range is 0 to 1480.

Service data
This parameter specifies the user data.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

Reason
This parameter reports the reason for the failure. Possible values are

— No Slave Reaction

— No Reply Message

6.1.7.3.4 MBX read upd

The MBX Read Upd service is used to write data from the slave application to the mailbox read area. It can be used for profiles to convey data from the slave to the master in conjunction with the MBX Read service. Table 37 shows the service primitives and parameter of the MBX Read Upd service.

Table 37 – MBX read upd

Parameter name	Req	Ind	Cnf
Argument			
Address	M	M (=)	
Channel	O	O(=)	
Priority	O	O(=)	
Type	M	M (=)	
Cnt	M	M (=)	
Length	M	M (=)	
Service Data	M	M (=)	
Result(+)			S
Result(–)			S

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.

Argument
The argument conveys the service specific parameters of the service request.

Address
This parameter specifies the station address of the server if the data are sent to the slave and the address of the master otherwise.

Channel
This optional parameter specifies a channel number. Its range is 0 to 63.

Priority
This optional parameter specifies one of four priorities.

Type
This parameter specifies the type of the mailbox service.

Cnt
This parameter specifies a service counter

Length
This parameter specifies the length of the user data. Its range is 0 to 1480.

Service data
This parameter specifies the service data.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(–)
This selection type parameter indicates that the service request failed.

6.2 AR

6.2.1 Overview

6.2.1.1.1 General

There is only one AR endpoint in a Slave. This endpoint is controlled by the EtherCAT State Machine (ESM) which describes the states and state changes of the slave application. The actual state of a slave application is reflected in the AL Status Register by the application and requested state changes are indicated in the AL Control Register by the master.

The ESM is logically located between the EtherCAT slave controller (ESC) and the application.

The ESM defines four states:

- Init,
- Pre-Operational,
- Safe-Operational, and
- Operational.

All state changes are possible except for the 'Init' state, where only the transition to the 'Pre-Operational' state is possible and for the 'Pre-Operational' state, where no direct state change to 'Operational' exists.

State changes are normally requested by the master. The master requests a write to the AL Control register which results in a Register Event 'AL Control' indication in the slave. The slave responds to the change in AL Control through a local AL Status write service after a successful or a failed state change. If the requested state change failed, the slave responds with the error flag set.

The Bootstrap state is optional and there is only a transition from or to the Init state. The only purpose of this state is to download the device's firmware. In Bootstrap state the mailbox is active but restricted to the file access over EtherCAT services (FoE) protocol.

6.2.1.2 State services

6.2.1.2.1 Service overview

The primitives of the EtherCAT DL-services are mapped to the AL management primitives described in the DL as specified in Table 38.

If the slave does not support mailbox services, the slave controller should be configured from the master to confirm the state change immediately in the AL Status register.

Table 38 – AL management and ESM service primitives

AL management primitives	master service	slave service
AL Control.req/ind	Write (AL Control) – all types of write or RW allowed	Event(AL Control), ReadLocal(AL Control)
AL Control.rsp/cnf	Read (AL Status) – all types of read or RW allowed	WriteRegisterLocal(AL Status)
AL State Changed.req/ind	Read (AL Status) – all types of read or RW allowed	WriteRegisterLocal(AL Status)

6.2.1.2.2 AL control sequence

The primitives between master and slave in case of a successful AL Control sequence are specified in Figure 32.

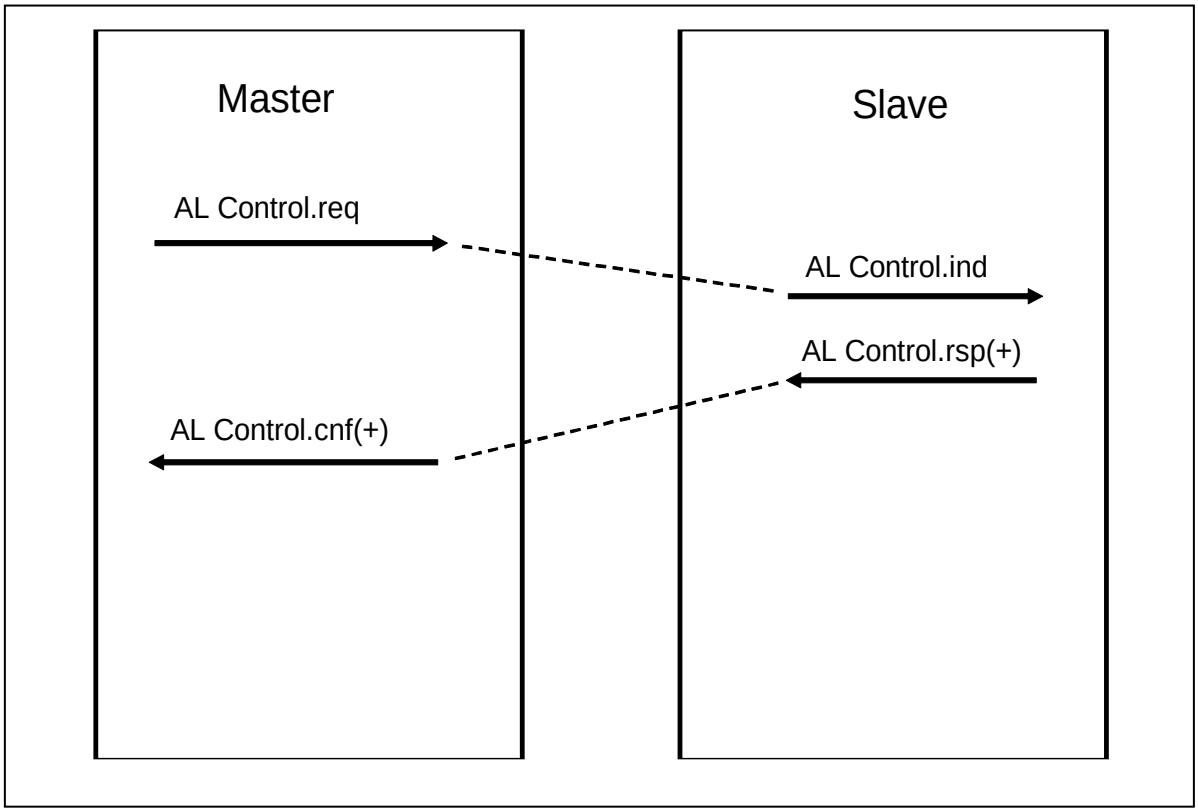

Figure 32 – Successful AL control sequence

The primitives between master and slave in case of a unsuccessful AL Control sequence are specified in Figure 33.

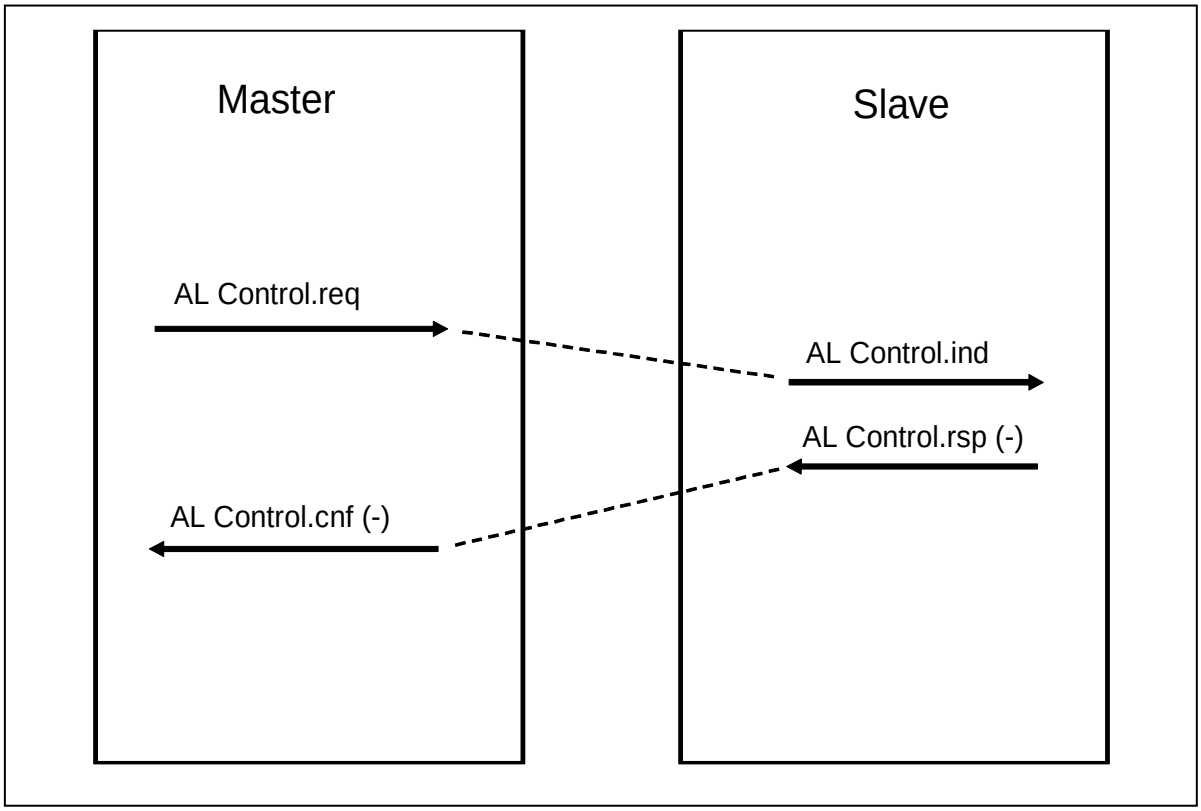

Figure 33 – Unsuccessful AL control sequence

6.2.1.2.3 AL state changed sequence

The primitives between master and slave in case of an AL State Changed sequence are specified in Figure 34.

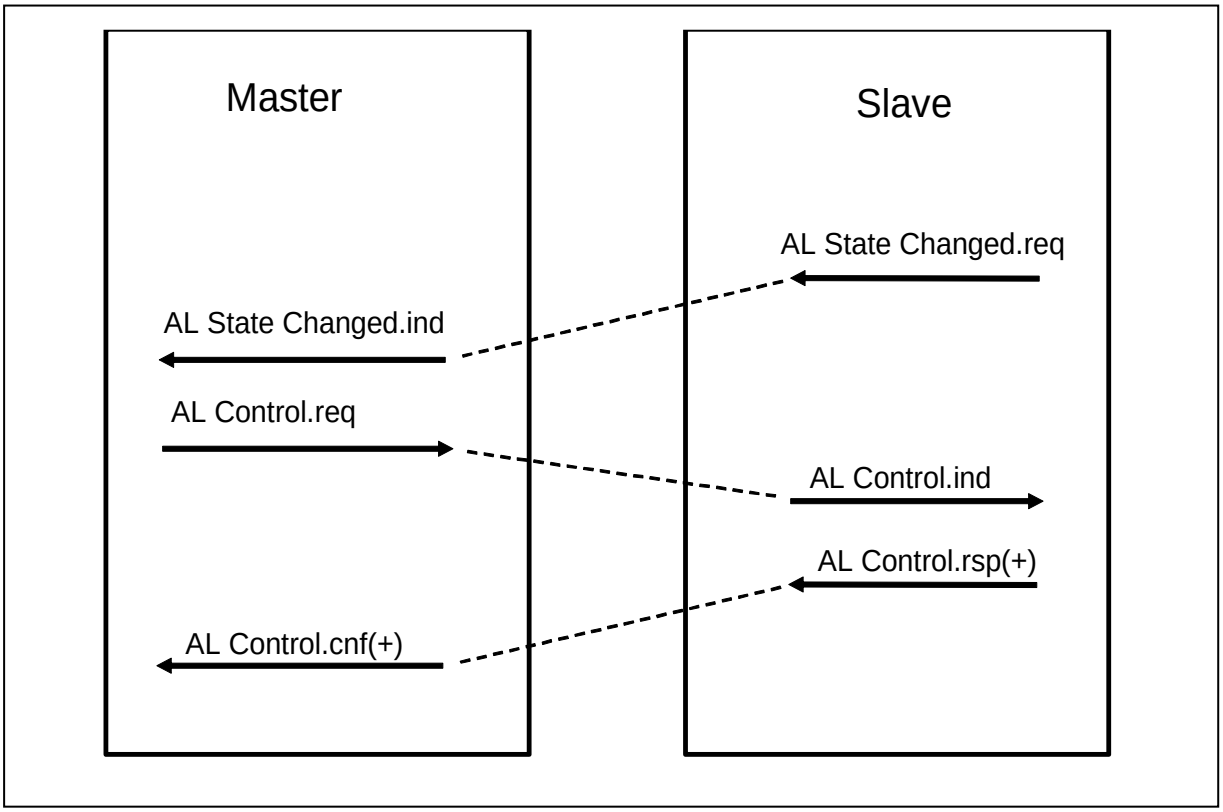

Figure 34 – AL state changed sequence

6.2.1.3 Local management services

6.2.1.3.1 Start mailbox communication

The Start Mailbox Communication Event will be issue if in the state ‘Init’ a valid AL Control service and requested AL Control State ‘Pre-Operational’ is indicated.

The master should configure the DL registers and the Sync Manager channels for the mailbox before requesting this service. Then the master should wait for the confirmation of the slave before sending other commands to the slave.

If the slave supports mailbox services, the slave’s application should read the settings of the Sync Manager and initialize its mailbox handler appropriately before confirming the state change by writing the AL Status register of the slave controller. If the Sync Manager’s configuration is correct, the slave provides a positive confirmation to this service.

6.2.1.3.2 Stop mailbox communication

The Stop Mailbox Communication Event will be issue if the slave application will enter the state ‘Init’ from ‘Pre-Operational’, ‘Safe-Operational’ or ‘Operational’.

If the slave supports mailbox services, the slave’s application has to stop its mailbox handler before issuing the state change by writing the AL Status register of the slave controller. The slave always confirms this service as successful.

6.2.1.3.3 Start input update

The Start Input Update service will be issued on a valid AL Control service and requested AL Control State the ‘Safe-Operational’ state, if the slave’s application is in the state ‘Pre-Operational’.

The master should configure the Sync Manager for the process data and the FMMU before requesting this service. After requesting the state transition the master should begin the transmission of the process data services, but should ignore the input data until the slave has confirmed the state transition.

If the slave supports mailbox services, the slave's application reads the configuration of the Sync Manager channels configured for process data transfer and checks if these settings match to its local process data configuration. If the checking was successful, the slave's application delivers valid input data before confirming the state transition. The output data of the slave should stay in a safe state. If the checking of the Sync Manager configuration was unsuccessful, the slave should confirm this service with the 'Pre-Operational' state and set the Error Flag parameter TRUE.

Start Input Update behavior is supported by complex slaves.

6.2.1.3.4 Stop input update

The Stop input Update Event will be issue if the slave application will enter the state 'Pre-Operational' or 'Init' from 'Safe-Operational' or 'Operational'.

The master should stop transmission of process data requests before requesting the state transition.

If the slave supports mailbox services, the slave's application should stop updating the input data before confirming the state transition. The slave always confirms this service as successful.

Stop Input Update behavior is supported by complex slaves.

The slave's application can use this service to indicate a local state transition, for example because of an unexpected error.

6.2.1.3.5 Start output update

The Start Output Update service will be issued on a valid AL Control service and requested AL Control State the 'Operational' state, if the slave's application is in the state 'Safe-Operational'.

The master should deliver valid output data in the process data services before requesting the state transition.

Start Output Update behavior is supported by complex slaves.

The slave's application should activate the valid output data received with the process data service before confirming the state transition. If the activation of the output data is not possible, the slave should confirm this service with the 'Safe-Operational' state and set the Error Flag parameter TRUE.

6.2.1.3.6 Stop output update

The Stop output update Event will be issue if the slave application will leave the state 'Operational'.

If the slave supports mailbox services, the slave's application should set the output in the safe state before confirming the state transition. The slave always confirms this service as successful.

Stop Output Update behavior is supported by complex slaves.

The slave's application can use this service to indicate a local state transition, for example because of an unexpected error.

6.2.1.3.7 Start bootstrap mode

The Start Mailbox Communication Event will be issue if in the state 'Init' a valid AL Control service and requested AL Control State 'Bootstrap' is indicated.

The master should configure the Sync Manager channels for the mailbox before requesting this service. The Sync Manager configuration for the mailbox in Bootstrap state can differ from the configuration in the other states. Then the master should wait for the confirmation of the slave before sending other commands to the slave.

The slave's application should read the settings of the Sync Manager and initialize its mailbox handler appropriately before confirming the state change by writing the AL Status register of the slave controller. If the slave's application supports the Bootstrap state and the configuration of the Sync Manager are correct for the Bootstrap state, the slave will confirm this service as successful.

6.2.1.3.8 Stop bootstrap mode

The Stop Mailbox Communication Event will be issue if the slave application will re-enter the state 'Init'.

If the slave supports mailbox services, the slave's application has to stop its bootstrap handler before confirming the state change by writing the AL Status register of the slave controller. The slave always confirms this service as successful.

If the slave does not support mailbox services, the slave controller should be configured from the master to confirm the state change immediately in the AL Status register.

6.2.2 AR control class specification

6.2.2.1 Formal model

The AR control object is described by the following template:

ASE:			**AR**
CLASS:			**AR control**
CLASS ID:			not used
PARENT CLASS:			TOP
ATTRIBUTES:			
1.	(m)	Key Attribute:	Implicit
2.	(m)	Attribute:	AL Control
2.1	(m)	Attribute:	AL State
2.2	(m)	Attribute:	Acknowledge
2.3	(o)	Attribute:	State info
3.	(m)	Attribute:	AL Status
3.1	(m)	Attribute:	AL State
3.2	(m)	Attribute:	Error Indication
3.3	(o)	Attribute:	ID Indication
3.4	(o)	Attribute:	State info
3.5	(m)	Attribute:	AL Status Code
4.	(m)	Attribute:	PDI Control
4.1	(m)	Attribute:	PDI Type
4.2	(m)	Attribute:	AL State Control
4.3	(o)	Attribute:	Specific settings
5.	(o)	Attribute:	Sync Control
5.1	(o)	Attribute:	Signal conditioning Sync 0
5.2	(o)	Attribute:	Enable Sync 0
5.3	(o)	Attribute:	Signal conditioning Sync 1
5.4	(o)	Attribute:	Enable Sync 1
6.	(m)	Attribute:	Events
6.1	(m)	Attribute:	AL control write event
6.2	(o)	Attribute:	Latch event
6.3	(o)	Attribute:	Sync 0 event
6.4	(o)	Attribute:	Sync 1 event
6.5	(m)	Attribute:	List of SM event
6.5.1	(m)	Attribute:	SM event
6.6	(o)	Attribute:	AL control write event mask
6.7	(o)	Attribute:	Latch event mask
6.8	(o)	Attribute:	Sync 0 event mask
6.9	(o)	Attribute:	Sync 1 event mask
6.10	(o)	Attribute:	List of SM event mask
6.10.1	(o)	Attribute:	SM event mask
7.	(o)	Attribute:	SM Settings
7.1	(o)	Attribute:	SM0
7.2	(o)	Attribute:	SM1
7.1	(o)	Attribute:	SM2
7.2	(o)	Attribute:	SM3
SERVICES:			
1.	(m)	OpsService:	AL Control
2.	(m)	OpsService:	AL State Change

NOTE The attribute are read/written by DL-services as described in ETG.1000.3 as this ASE will be the agent for management services.

6.2.2.2 Attributes

Implicit

The attribute Implicit indicates that the AR Control object is implicitly addressed by the services.

AL Control
One Object is composed of the following elements:

AL state
This parameter specifies the requested AL state. Possible values are

- Init
- Pre-operational
- Safe-operational
- Operational
- Bootstrap

Acknowledge
This Boolean attribute is set to confirm an error indication of the AL.

State info
This optional attribute consists of application-specific information set by the master.

ID Request
This Boolean attribute is set to request the Identification value of the AL

AL Status
One Object is composed of the following elements:

AL state
This attribute contain the AL State set by the local instance. Possible values are

- Init
- Pre-operational
- Safe-operational
- Operational
- Bootstrap

Error indication
This Boolean attribute consists of the information that the state control of the AL has detected an error

ID Indication
This Boolean attribute consists of the information that the AL has loaded an ID value.

State info
This optional attribute consists of application-specific information set by the slave application.

AL status code
This optional attribute is controlled by the application and reports the last error detected by the state control instance of the AL or an ID value.

PDI Control
One Object is composed of the following elements:

PDI type
This attribute specifies the process data interface type describing the hardware interface behaviour of the slave controller.

AL state control
This Boolean attribute is set to false if the AL Status operation is done by the application and is set to true if the AL Status will be emulated by the slave controller by copying the AL control into the AL status.

Specific settings
This optional attribute consists of application specific settings done to configure the PDI.

PDI Control
One Object is composed of the following elements:

Signal conditioning sync 0
This optional attribute specifies the process data interface type describing the hardware interface behaviour of the slave controller's sync signal.

Enable sync 0
This optional attribute enables the synchronisation signal. Two events sources can be selected. Allowed values are

— Interrupt

— Pulse

— Both

— None

Signal conditioning sync 1
This optional attribute specifies the process data interface type describing the hardware interface behaviour of the slave controller's sync signal.

Enable sync 1
This optional attribute enables the synchronisation signal. Two events sources can be selected. Allowed values are

— Interrupt

— Pulse

— Both

— None

Events
One Object is composed of the following elements:

AL control write event
This Boolean attribute is set if there is a write from the master to AL control. It is cleared if the AL control is read locally.

Latch event
This optional Boolean attribute is set if there is a new entry in the latch registers. It is cleared if the latch registers are read locally.

Sync 0 event
This optional Boolean attribute is set if there is a Sync 0 event. It is cleared if the event is read locally.

Sync 1 event
This optional Boolean attribute is set if there is a Sync 1 event. It is cleared if the event is read locally.

List of SM event
One Object is composed of the following elements:

SM event
This Boolean attribute is set if there is valid read or write to the SM area. It is cleared if the event is read locally.

AL control write event mask
This Boolean attribute is set if a write from the master to AL control should produce an event. It is cleared otherwise.

Latch event mask
This optional Boolean attribute is set if a new entry in the latch registers should result in an event. It is cleared otherwise.

Sync 0 event mask
This optional Boolean attribute is set if a Sync 0 event should result in an event. It is cleared otherwise.

Sync 1 event mask
This optional Boolean attribute is set if a Sync 1 event should result in an event. It is cleared otherwise.

List of SM event mask
One Object is composed of the following elements:

SM event mask
This Boolean attribute is set if valid read or write to the SM area should result in an event. It is cleared otherwise.

SM settings
One Object is composed of the following elements:

SM 0
This optional attribute consists of the SM settings as described in ETG.1000.4. SM0 is used as write mailbox.

SM 1
This optional attribute consists of the SM settings as described in ETG.1000.4. SM1 is used as read mailbox.

SM 2
This optional attribute consists of the SM settings as described in ETG.1000.4. SM2 is used as write buffer.

SM 3
This optional attribute consists of the SM settings as described in ETG.1000.4. SM3 is used as read buffer.

6.2.3 AR service specification

6.2.3.1 AL control

The AL Control service, specified in Table 39, is used by the master to request a state transition in the slave's application by writing the AL Control register. The slave confirms the state transition by writing the AL Status register, which will be read from the master to get the confirmation.

In case of an unsuccessful state transition, the master acknowledges this service with another AL Control service and Ack Flag set to one.

With an ID Request the master requests the ID value from the slave's application. The slave confirms the successful loading of the ID value to the AL Status Code register by setting the ID Flag to one.

Table 39 – AL control

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AL State	M	M (=)		
Ack Flag	M	M (=)		
ID Request	U	U (=)		
Result(+)			S	S (=)
AL Status Code			M	M (=)
Error Indication			M	M (=)
ID Indication			U	U (=)
Result(–)			S	S (=)
AL Status Code			M	M (=)
Error Indication			M	M (=)
ID Indication			U	U (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.3.				

Argument
The argument conveys the service specific parameters of the service request.

AL state
This parameter indicates the state which is requested by the master in the slave's application. Possible values are

— Init

— Pre-operational

— Safe-operational

— Operational

— Bootstrap

Ack Flag
This Boolean parameter acknowledges a negative result of an AL Control service.

ID Request
This parameter indicates an ID request

Result(+)
This selection type parameter indicates that the service request succeeded.

AL status code
This parameter indicates the actual state of the slave's application or the ID value.

Error Indication
This parameter indicates that the state control of the AL has detected an error.

ID Indication
This parameter indicates the AL has loaded an ID value.

Result(–)
This selection type parameter indicates that the service request failed.

AL status code
This parameter indicates the actual state of the slave's application.

Error Indication
This parameter indicates that the state control of the AL has detected an error.

ID Indication
This parameter indicates the AL has loaded an ID value.

6.2.3.2 AL state change

Additionally the slave's application can indicate a local state transition by writing the AL Status register as specified in Table 40. The master acknowledges the state transition with AL Control service.

Table 40 – AL state change

Parameter name	Req	Ind
Argument		
AL State	M	M (=)
AL Status Code	M	M (=)
Error Indication	M	M (=)
ID Indication	U	U (=)

Argument
The argument conveys the service specific parameters of the service request.

AL state
This parameter indicates the state which is requested by the slave's application. Possible values are

— Init

— Pre-operational

— Safe-operational

— Operational

— Bootstrap

AL status code
This parameter indicates the actual state of the slave's application.

Error Indication
This parameter indicates that the state control of the AL has detected an error. If AL state is Operational and the slave detects an error it shall change its state to a lower state.

ID Indication
This parameter indicates the AL has loaded an ID value.

Bibliography

IEC 61158-2, *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61158-4-12, *Industrial communication networks – Fieldbus specifications - Part 4-12: data-link layer protocol specification – Type 12 elements*

IEC 61158-6-12, *Industrial communication networks – Fieldbus specifications - Part 6-12: Application layer protocol specification – Type 12 elements*

IEC 61784-2, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3*

IEEE 802.1Q, *IEEE standard for Local and metropolitan area networks – Virtual bridged local area networks Bridges;* available at <http://www.ieee.org>

EN 50325-4, *Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces - Part 4: CANopen*

ETG.1000.2 *EtherCAT Specification Part 2: Physical layer specification and service definition*

ETG.1000.3 *EtherCAT Specification Part 3: Data Link Layer service definition*

ETG.1000.4 *EtherCAT Specification Part 4: Data-link layer protocol specification*

ETG.1000.5 *EtherCAT Specification Part 5: Application layer service definition*

ETG.1000.6 *EtherCAT Specification Part 6: Application layer protocol specification*

ETG.1100 *EtherCAT Specification Communication profiles*