CiA Draft Standard 309

Interfacing CANopen with TCP/IP

Part 2: Modbus/TCP mapping

Version 1.1

12 December 2006

HISTORY

Date	Changes
2004-09-30	*Publication of version 1.0 as draft standard proposal*
2006-12-12	*Publication of version 1.1 as draft standard* Minor editorial corrections and clarifications.

CAN in Automation e. V.
Kontumazgarten 3
DE - 90429 Nuremberg, Germany
Tel.: +49-911-928819-0
Fax: +49-911-928819-79
Url: www.can-cia.org
Email: headquarters@can-cia.org

Contents

1 Scope

This specification specifies the services and protocols to interface CANopen networks to a TCP/IP-based network.

This set of specifications is organized as follows:

- Part 1: General principles and services
- Part 2: Modbus/TCP mapping
- Part 3: ASCII mapping

This part of the specification defines the mapping of services defined in /CiA 309-1/ on Modbus/TCP. It is intended to access CANopen devices via a gateway device from a remotely Modbus/TCP connected device (e.g. PLC or tool).

2 References

The references given in part 1 shall apply to this part, too.

/CiA309-1/	CiA 309:2006, Interfacing CANopen with TCP/IP – Part 1: General principles and services (V1.1)
/MAP/	Modbus Application Protocol: December 2002 (www.modbus.org) (V1.1)
/RFC/	Request for comments (RFC) 791 internet protocol: September 1981

3 Abbreviations and definitions

3.1 Abbreviations

The abbreviations given in part 1shall apply to this part, too.

ADU	Application Data Unit
PDU	Protocol Data Unit
MEI	Modbus Encapsulated Interface
MEC	Modbus Exception Codes

3.2 Definitions

3.2.1 General

The definitions given in /CiA 301/ and part 1 shall apply to this part, too.

3.2.2 CANopen general reference command

The services defined in part 1 are mapped to a Modbus function code known as the *CANopen general reference command.*

The *CANopen general reference command* is an encapsulation of the services defined in /CiA 309-1/ that is used to access (read from or write to) the entries of a CANopen object dictionary as well as controlling/monitoring the gateway device, and other CANopen devices.

The networked system is intended to work within the limitations of existing Modbus networks. Therefore, the information needed to query or modify the object dictionaries in the CANopen devices is mapped into the format of a Modbus message. The command has the 253-byte limitation in both the request and the response message. *Figure 1* illustrates how the CANopen general reference MEI type is incorporated into the Modbus encapsulated interface function code.

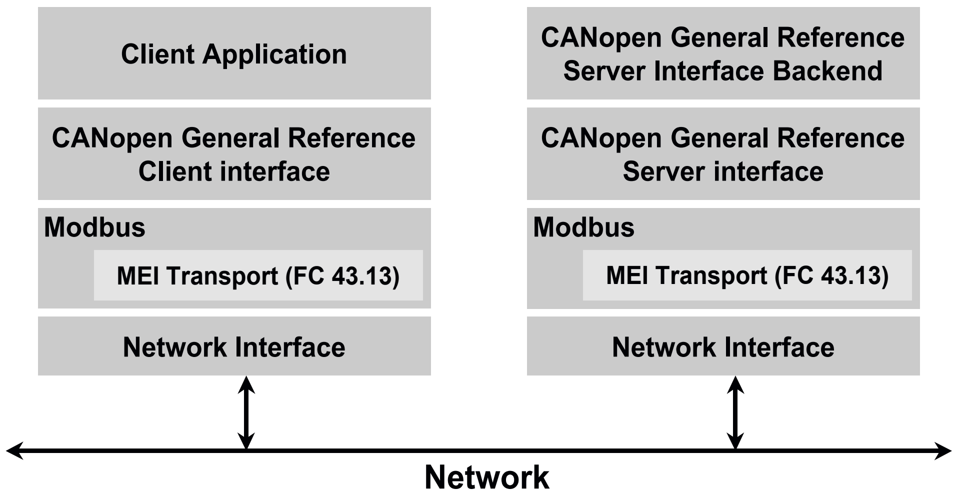

Client Application
CANopen General Reference Client interface
Modbus
MEI Transport (FC 43.13)
Network Interface
CANopen General Reference Server Interface Backend
CANopen General Reference Server interface
Modbus
MEI Transport (FC 43.13)
Network Interface
Network

Fig. 1: Device view of both the client and server modules

The *Network interface* can be any communication stack used to send Modbus PDUs, such as TCP/IP, serial line, or Modbus Plus. When TCP/IP is used, the Modbus service is available at TCP port 502 /MAP/.

The *MEI Transport Service* is a general service used to encapsulate and transport interface methods associated with a particular Modbus encapsulation interface type.

3.2.3 Request and response definition

3.2.3.1 General

This chapter defines the *CANopen general reference request and response PDU.* Like other Modbus request PDUs, this function has the 253-byte limit in both the request and the response message ADU /MAP/.

Field name	**Byte size and order**	**Example/range**
Function code	1 byte	43 ($2B_h$)
MEI type	1 byte	13 ($0D_h$)
Protocol option fields	1 to m byte	*See* 3.2.3.4
Address and data fields	n byte	refer to description

3.2.3.2 Function code

The *function code* is 43 ($2B_h$).

3.2.3.3 MEI type

The *MEI type* for the CANopen general reference is 13 ($0D_h$).

3.2.3.4 Protocol option fields

3.2.3.4.1 General

The following fields are defined:

Field name	**Byte size and order**	**Example/range**
Protocol control	1 to 2 byte	*See* 3.2.3.4.2
Reserved field	1 byte	*See* 3.2.3.4.3
[Optional] Counter byte	1 byte	*See* 3.2.3.4.4
[Optional] Network ID	1 byte	*See* 3.2.3.4.5
[Optional] Encoded data	1 byte	*See* 3.2.3.4.6

3.2.3.4.2 Protocol control

The *protocol control* field contains the flags needed to control the message protocol. The protocol control bytes is defined as follows:

0	7
Protocol control byte 1	
MSB	LSB

8	15
Protocol control byte 2	
MSB	LSB

The most significant bit (MSB) is bit 0 for *protocol control byte 1*, and bit 8 for protocol control byte 2. The least significant byte (LSB) is bit 7 for *protocol control byte 1*, and bit 15 for *protocol control byte 2*. The bits are numbered in transmission order following the data transmission order defined in Appendix B in /RFC/.

Note: This bit numbering is used throughout this specification.

Bit	Field name	Description
0	*Extend flag*	0 = No multiple message transaction, or end of multiple message transaction; 1 = Part of a multiple message transaction
1	*Extend protocol control*	0 = 1 byte, 1 = 2 byte
2	*Counter byte option*	*See* description below
3	*Reserved*	0
4	*Reserved*	0
5	*Network ID option*	*See* description below
6	*Encoded data option*	*See* description below
7	*Access flag*	0 = read, 1 = write
8 to 15	*Reserved*	0

Extend flag

This bit is used when the object dictionary data set is larger than what fits in one Modbus command. The data set is then *extended* over multiple Modbus messages, each message being a fragment of the data set.

Extend protocol control

This bit indicates the length of the protocol control.

Counter byte option

This bit is set to 1 to indicate that the counter byte field is used in this message. If this bit is set to 0 the counter byte does not exist in this message.

Network ID option

This bit is set to 1 to indicate that the network ID field is used in this message. If this bit is set to 0 the network ID does not exist in this message.

Encoded data option

This bit is set to 1 to indicate that the encoded data field is used in this message. If this bit is set to 0 the encoded data does not exist in this message.

Access flag

This bit indicates the access method of the request command.

3.2.3.4.3 Counter byte

This byte is intended for a rolling counter to be used during extended read and write operations. This bit counter increases for each subsequent fragment that is downloaded. The first fragment sets the *counter byte* to 0. The *counter byte* is the same for the request and it's response message. After reaching 255, the counter rolls over to 0. If the receiver detects a gap, it aborts the processing of the extended read/write with an exception response (*see* chapter 5.2 and Appendix A).

3.2.3.4.4 Reserved field

This byte is reserved.

3.2.3.4.5 Network ID

This byte is used to identify a particular CANopen network interface for a gateway managing several CANopen networks. If the receiver receives a request with a network ID not corrresponding to an existing network, it aborts the processing with an extended exception response (*see* chapter 5.2 and Appendix A). Networks are numbered from 1 to 255. The network ID option bit controls the existence of this field in the CANopen general reference command.

3.2.3.4.6 Encoded data

This byte is used to encode the data types. The following shows encoding values for data types in the encoded fields when selected for Download SDO and Upload SDO commands as well for PDO access command.

Note: The data is encoded as defined in /CiA301/.

Data types	Values
Boolean	30_h
Unsigned8	00_h
Unsigned16	01_h
Unsigned24	02_h
Unsigned32	03_h
Unsigned40	04_h
Unsigned48	05_h
Unsigned56	06_h
Unsigned64	07_h
Integer8	10_h
Integer16	11_h
Integer24	12_h
Integer32	13_h
Integer40	14_h
Integer48	15_h
Integer56	16_h
Integer64	17_h
Real32	20_h
Real64	21_h
Time of day	40_h
Visible string	50_h
Domain	60_h

3.2.3.5 Address and data fields

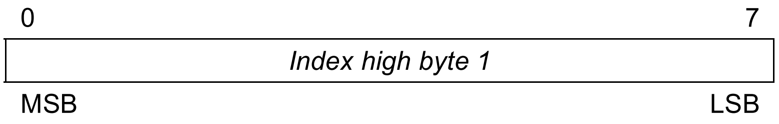
The following fields are defined:

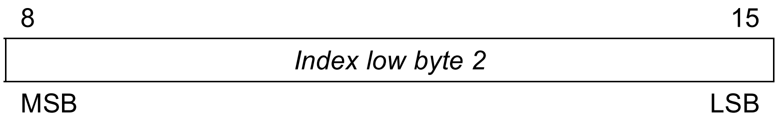
Field name	**Byte size and order**	**Example/range**
Node ID	1 byte	01_h to $7F_h$
Index	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Sub-index	1 byte	00_h to FF_h
Starting address	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Number of data values	1 byte, high	0000_h to $00FD_h$
	1 byte, low	
Read/write data	n byte	-

Node ID

This byte is used to select a particular NMT slave device on the CANopen network (01_h to $7F_h$). The value of $7F_h$ is used for special purposes (*see* sub-index definition).

Index





These two bytes are used to access a particular object in a CANopen SDO server device. The value of $FFFF_h$ is used for special purposes (*see* sub-index definition).

Sub-index

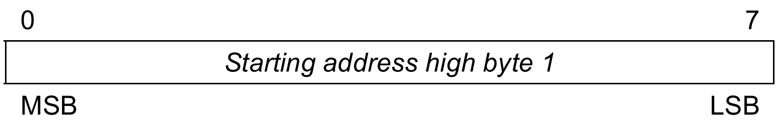
This byte is used usually to access a particular sub-object in a CANopen SDO server device. If the node-ID is $7F_h$ and the index is $FFFF_h$, the following definitions apply (*Table 1*).

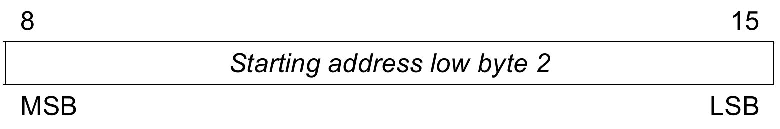
Table 1: Command codes used in the sub-index field

Code	**Command**
00_h	Reserved
01_h	GATEWAY_INITIALIZATION
02_h to 03_h	Reserved
04_h	START_ALL_NODES
05_h	PRE_OP_ALL_NODES
06_h	STORE_CONFIGURATION
07_h	Reserved
08_h	RESTORE_CONFIGURATION
09_h to 10_h	Reserved
11_h	STOP_ALL_NODES

Code	Command
12_h	START_ONE_NODE
13_h	STOP_ONE_NODE
14_h	PRE_OPERATIONAL_ONE_NODE
15_h	RESET_NODE
16_h	RESET_COMM_NODE
17_h	ENABLE_NODE_GUARDING
18_h	DISABLE_NODE_GUARDING
19_h	START_HEARTBEAT_CONSUMER
20_h	DISABLE_HEARTBEAT_CONSUMER
21_h to 22_h	ERROR
23_h	SET_SDO_TIMEOUT
24_h	RESET_ALL_NODE
25_h	RESET_COMM_ALL_NODE
26_h to $2F_h$	Reserved
30_h	SET_BIT_RATE
31_h	SET_SERVER_HEARTBEAT
32_h	SET_SERVER_NODE_ID
33_h	SET_DEFAULT_NETWORK
34_h to $5F_h$	Reserved
60_h	RESET_CONTROLLER
61_h	RUN_CONTROLLER
62_h	STOP_CONTROLLER
63_h to FF_h	Reserved

Starting address





These two bytes indicate the starting offset into the selected object. It may be non-zero when performing segmented SDO transfers. It does not map to the CANopen SDO transfer protocol directly, but is used by the client and server applications for address synchronization.

Number of data value

0		7
	Number of data value high 1	
MSB		LSB

8		15
	Number of data value low 2	
MSB		LSB

These two bytes provide a count of the desired number of data values to be read or to be written. This count uses the defined encoded data value to calculate the number of data values transferred in the message. If the encoded data field is not present or is not supported by the device, the default data type is Unsigned8, and this field represents the number of data bytes.

Read/write data

These bytes contain the data to be read or to be written. The *CANopen general reference request* contains the write data when the request is a download request. The *CANopen general reference response* contains the read data when the request is a upload request.

4 Network access command specification

4.1 SDO access commands

4.1.1 General

This chapter defines the mapping of SDO_UPLOAD and SDO_DOWNLOAD services to the CANopen general reference PDU.

SDO_UPLOAD and SDO_DOWNLOAD commands are intended to give access to all the elements of the gateway/server object dictionary as well as all the elements of a node object dictionary on the CANopen sub networks.

Simple or extended Modbus transfer may be realized according the size of the objects. For extended transfers the counter byte shall be used in order to secure the exchanges. Extended transfers are required for read/write data larger than 238 bytes.

The default data type shall be Unsigned8. To request a specific datatype the optional encoded data flag shall be enabled. A device is not required to implement the encoded data field. In this case all data is transferred as Unsigned8.

For simple and extended SDO_UPLOAD and SDO_DOWNLOAD that access large objects, several SDO transfer on the CANopen networks shall be required and in order not to block the Modbus network, the Modbus server on the gateway shall immediately send the standard exception response “Acknowledge” and proceed with the request. The client shall simply have to poll the server by resendig the request until it got the response.

The client shall poll by re-sending the identical MB 43-13 Request PDU (Note: for MB/TCP with the same MBAP TransactionID /MAP/ also). The expected responses to the poll shall be either: 1) another ACKNOWLEDGE, 2) a normal MB 43-13 response indicating a completed SDO data transfer, or 3) another MB Exception response indicating an abort situation.

4.1.2 Upload SDO command

SDO_UPLOAD.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	*See* Appendix C
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	00_h to FF_h
[Optional] Network ID	1 byte	1 to 255
[Optional] Encoded data	1 byte	*See* 3.2.3.4.6

Field name	Byte size and order	Example/range
Node ID	1 byte	01_h to $7F_h$
Index	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Sub-index	1 byte	00_h to FF_h
Starting address	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Number of data values	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Read/write data	0 byte	not applicable

SDO_UPLOAD.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	*See* Appendix C
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	00_h to FF_h
[Optional] Network ID	1 byte	1 to 255
[Optional] Encoded data	1 byte	*See* 3.2.3.4.6

Field name	Byte size and order	Example/range
Node ID	1 byte	01_h to $7F_h$
Index	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Sub-index	1 byte	00_h to FF_h
Starting address	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Number of data values	1 byte, high	0001_h to $FFFF_h$
	1 byte, low	
Read/write data	n byte	Data received

SDO_UPLOAD.resp(NotOk)

See chapter 5.

4.1.3 Download SDO command

SDO_DOWNLOAD.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	*See* Appendix D
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	00_h to FF_h
[Optional] Network ID	1 byte	1 to 255
[Optional] Encoded data	1 byte	*See* 3.2.3.4.6

Field name	Byte size and order	Example/range
Node ID	1 byte	01_h to $7F_h$
Index	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Sub-index	1 byte	00_h to FF_h
Starting address	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Number of data values	1 byte, high	0001_h to $FFFF_h$
	1 byte, low	
Read/write data	n byte	Data to be written

SDO_DOWNLOAD.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	*See* Appendix D
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	00_h to FF_h
[Optional] Network ID	1 byte	1 to 255
[Optional] Encoded data	1 byte	*See* 3.2.3.4.6

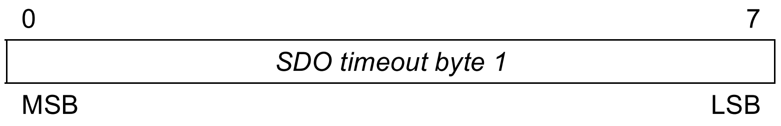
Field name	Byte size and order	Example/range
Node ID	1 byte	01_h to $7F_h$
Index	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Sub-index	1 byte	00_h to FF_h
Starting address	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	n byte	0001_h to $FFFF_h$

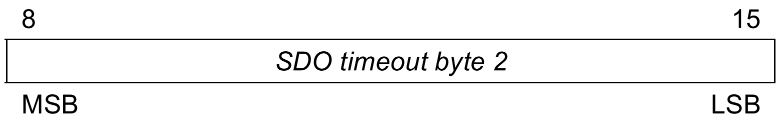
SDO_ DOWNLOAD.resp(NotOk)

See chapter 5.

4.1.4 Configure SDO timeout command

Read/write data





SET_SDO_TIMEOUT.req

Field name	**Byte size and order**	**Example/range**
Protocol control	1 byte	05$_h$: specified network 01$_h$: default network
Reserved field	1 byte	00$_h$
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01$_h$) 1 to 255 (if protocol control is 05$_h$)
[Optional] Encoded data	1 byte	not applicable

Field name	**Byte size and order**	**Example/range**
Node ID	1 byte	7F$_h$
Index	1 byte, high	FFFF$_h$
	1 byte, low	
Sub-index	1 byte	23$_h$
Starting address	1 byte, high	0000$_h$
	1 byte, low	
Number of data values	1 byte, high	0002$_h$
	1 byte, low	
Read/write data	1 byte high	SDO timeout in ms
	1 byte low	

SET_SDO_TIMEOUT.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	23_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

SET_SDO_TIMEOUT.resp(NotOk)

See chapter 5.

4.2 PDO access commands

4.2.1 General

The Modbus/TCP mapping specification does not support the PDO access services, and defines no PDO access commands.

4.2.2 Configure RPDO command

This command is not supported.

4.2.3 Configure TPDO command

This command is not supported.

4.2.4 Read PDO data command

This command is not supported.

4.2.5 Write PDO data command

This command is not supported.

4.2.6 RPDO received command

This command is not supported.

4.3 CANopen NMT commands

4.3.1 General

The following command definitions shall be used to implement the CANopen NMT services as defined in /CiA 309-1/. The supported services depend on the gateway class.

4.3.2 Start node command

START_NODE.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	04_h: start all nodes 12_h: start one node
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h: if sub-index is 04_h
	1 byte, low	0001_h: if sub-index is 12_h
Read/write data	1 byte	not applicable (if number of data values is 0000_h) 01_h to $7F_h$: target node-ID (if number of data values is 0001_h)

START_NODE.resp(Ok)

Field name	**Byte size and order**	**Example/range**
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	**Byte size and order**	**Example/range**
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	04_h: start all nodes 12_h: start one node
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h: if sub-index is 04_h
	1 byte, low	0001_h: if sub-index is 12_h
Read/write data	1 byte	not applicable if number of data values is 0000_h 01_h to $7F_h$: target node-ID if number of data values is 0001_h (class 3: after node has started successfully)

START_NODE.resp(NotOk)

See chapter 5.

4.3.3 Stop node command

STOP_NODE.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	11_h: stop all nodes 13_h: stop one node
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h: if sub-index is 11_h
	1 byte, low	0001_h: if sub-index is 13_h
Read/write data	1 byte	not applicable (if number of data values is 0000_h) 01_h to $7F_h$: target node-ID (if number of data values is 0001_h)

STOP_NODE.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	11_h: stop all nodes 13_h: stop one node
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h: if sub-index is 11_h
	1 byte, low	0001_h: if sub-index is 13_h
Read/write data	1 byte	not applicable if number of data values is 0000_h 01_h to $7F_h$: target node-ID if number of data values is 0001_h (class 3: after node has stopped successfully)

STOP_NODE.resp(NotOk)

See chapter 5.

4.3.4 Set node to pre-operational command

PRE_OP_NODE.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	1 to 255 if protocol control is 05_h
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	7F$_h$
Index	1 byte, high	FFFF$_h$
	1 byte, low	
Sub-index	1 byte	05$_h$: set all nodes to pre-operational 14$_h$: set one node to pre-operational
Starting address	1 byte, high	0000$_h$
	1 byte, low	
Number of data values	1 byte, high	0000$_h$: if sub-index is 05$_h$
	1 byte, low	0001$_h$: if sub-index is 14$_h$
Read/write data	1 byte	not applicable (if number of data values is 0000$_h$) 01$_h$ to 7F$_h$: target node-ID (if number of data values is 0001$_h$)

PRE_OP_NODE.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05$_h$: specified network 01$_h$: default network
Reserved field	1 byte	00$_h$
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01$_h$) 1 to 255 (if protocol control is 05$_h$)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	7F$_h$
Index	1 byte, high	FFFF$_h$
	1 byte, low	
Sub-index	1 byte	05$_h$: set all nodes to pre-operational 14$_h$: set one node to pre-operational
Starting address	1 byte, high	0000$_h$
	1 byte, low	
Number of data values	1 byte, high	0000$_h$: if sub-index is 05$_h$
	1 byte, low	0001$_h$: if sub-index is 14$_h$
Read/write data	1 byte	not applicable if number

		of data values is 0000$_h$ 01$_h$ to 7F$_h$: target node-ID if number of data values is 0001$_h$ (class 3: after node has been set to preoperated successfully)

PRE_OP_NODE.resp(NotOk)

See chapter 5.

4.3.5 Reset node command

RESET_NODE.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05$_h$: specified network 01$_h$: default network
Reserved field	1 byte	00$_h$
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01$_h$) 1 to 255 (if protocol control is 05$_h$)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	7F$_h$
Index	1 byte, high	FFFF$_h$
	1 byte, low	
Sub-index	1 byte	24$_h$: reset all nodes 15$_h$: reset one node
Starting address	1 byte, high	0000$_h$
	1 byte, low	
Number of data values	1 byte, high	0000$_h$: if sub-index is 24$_h$
	1 byte, low	0001$_h$: if sub-index is 15$_h$
Read/write data	1 byte	not applicable (if number of data values is 0000$_h$) 01$_h$ to 7F$_h$: target node-ID (if number of data values is 0001$_h$)

RESET_NODE.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	24_h: reset all nodes 15_h: reset one node
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h: if sub-index is 24_h
	1 byte, low	0001_h: if sub-index is 15_h
Read/write data	1 byte	not applicable if number of data values is 0000_h 01_h to $7F_h$: target node-ID if number of data values is 0001_h (class 3: after node has been reset successfully)

RESET_NODE.resp(NotOk)

See chapter 5.

4.3.6 Reset communication node command

RESET_COMM_NODE.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	25_h: reset communication all nodes 16_h: reset communication one node
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h: if sub-index is 25_h
	1 byte, low	0001_h: if sub-index is 16_h
Read/write data	1 byte	not applicable (if number of data values is 0000_h) 01_h to $7F_h$: target node-ID (if number of data values is 0001_h)

RESET_COMM_NODE.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	25_h: reset communication all nodes 16_h: reset communication one node
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h: if sub-index is 25_h
	1 byte, low	0001_h: if sub-index is 16_h
Read/write data	1 byte	not applicable if number of data values is 0000_h 01_h to $7F_h$: target node-ID if number of data values is 0001_h (class 3: after node has been reset communication successfully)

RESET_COMM_NODE.resp(NotOk)

See chapter 5.

4.3.7 Enable node guarding command

ENABLE_NODE_GUARDING.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	17_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0004_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: target node-ID
	1 word	GuardTime /CiA 301/
	1 byte	LifetimeFactor /CiA 301/

ENABLE_NODE_GUARDING.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	17_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: target node-ID (class 3: after node guarding has started successfully)

ENABLE_NODE_GUARDING.resp(NotOk)

See chapter 5.

4.3.8 Disable node guarding command

DISABLE_NODE_GUARDING.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h for a specified network 01_h for default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	18_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	Target node ID: 01_h to $7F_h$

DISABLE_NODE_GUARDING.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	18_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: target node ID (class 3: after node guarding has stopped success fully)

DISABLE_NODE_GUARDING.resp(NotOk)

See chapter 5

4.3.9 Start heartbeat consumer command

START_HEARTBEAT_CONSUMER.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h for a specified net-work 01_h for default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if proto-col control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	19_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0005_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: target node ID
	2 word	Heartbeat consumer

		time /CiA 301/

START_HEARTBEAT_CONSUMER.resp(Ok)

Field name	**Byte size and order**	**Example/range**
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	**Byte size and order**	**Example/range**
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	19_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: target node ID (after first heartbeat has been received successfully)

START_HEARTBEAT_CONSUMER.resp(NotOk)

See chapter 5.

4.3.10 Disable heartbeat consumer command

DISABLE_HEARTBEAT_CONSUMER.req

Field name	**Byte size and order**	**Example/range**
Protocol control	1 byte	05_h for a specified network 01_h for default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	20_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: target node ID

DISABLE_HEARTBEAT_CONSUMER.rsp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
[Optional] Reserved field	1 byte	not applicable
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	20_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: target node ID

DISABLE_HEARTBEAT_CONSUMER.resp(NotOk)

See chapter 5.

4.3.11 Error control event received

This service is not supported.

4.4 Device failure management commands

4.4.1 General

The following command definitions shall be used to implement the device failure management services as defined in /CiA 309-1/.

4.4.2 Read device error command

ERROR.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	04_h: specified network 00_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 00_h) 0: request the last error report 1 to 255 (if protocol control is 04_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	21_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	00_h: if last error report is requested 01_h to $7F_h$: target node-ID

ERROR.resp(Generic)

Field name	Byte size and order	Example/range
Protocol control	1 byte	04_h: specified network 00_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 00_h) 1 to 255 (if protocol control is 04_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	21_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0003_h to $FFFF_h$
	1 byte, low	
Read/write data	1 byte	Target node ID: 01_h to $7F_h$
	1 byte high	Error Msg Nbr: 0000_h to $FFFF_h$ (*see* Appendix B)
	1 byte low	
	n byte	Error message string (*see* Appendix B)

ERROR.resp(EMCY)

Field name	Byte size and order	Example/range
Protocol control	1 byte	04_h: specified network 00_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	0 to 255 (if protocol control is 04_h)
[Optional] Encoded data	1 byte	not applicable

Field name	**Byte size and order**	**Example/range**
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	22_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0003_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: target node ID
	1 word	0000_h to $FFFF_h$: EMCY error code /CiA 301/

ERROR.resp(NotOk)

See chapter 5.

4.4.3 Emergency event received command

This service is not supported.

4.5 CANopen interface configuration commands

4.5.1 General

The following command definitions shall be used to implement the CANopen interface configuration services as defined in /CiA 309-1/.

4.5.2 Initialize gateway command

GATEWAY_INITIALIZATION.req

Field name	**Byte size and order**	**Example/range**
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	01_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h: if default bit-rate is used 0001_h: if bit-rate is set
	1 byte, low	
Read/write data	0 or 1 byte	00_h to 09_h: /CiA 305/

GATEWAY_INITIALIZATION.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h; specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	01_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

GATEWAY_INITIALIZATION.resp(NotOk)

See chapter 5.

4.5.3 Store configuration command

This command shall cause the the non-volotile strange of all CANopen parameters that are savable.

STORE_CONFIGURATION.req

Field name	**Byte size and order**	**Example/range**
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	**Byte size and order**	**Example/range**
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	06_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

STORE_CONFIGURATION.resp(Ok)

Field name	**Byte size and order**	**Example/range**
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	06_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

STORE_CONFIGURATION.resp(NotOk)

See chapter 5.

4.5.4 Restore configuration command

This command shall cause the restorage of all CANopen parameters that are restorable.

RESTORE_CONFIGURATION.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	08_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

RESTORE_CONFIGURATION.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	08_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

RESTORE_CONFIGURATION.resp(NotOk)

See chapter 5.

4.5.5 Set heartbeat producer command

SET_SERVER_HEARTBEAT.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	31_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0002_h
	1 byte, low	
Read/write data	1 word	0000_h to $FFFF_h$: Heartbeat producer time of the gateway /CiA 301/

SET_SERVER_HEARTBEAT.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	31_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0002_h
	1 byte, low	
Read/write data	1 byte high	0000_h to $FFFF_h$: Current heartbeat producer time of the gateway
	1 byte low	

SET_SERVER_HEARTBEAT.resp(NotOk)

See chapter 5.

4.5.6 Set node ID

SET_GATEWAY_NODE_ID.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	32_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: gateway node ID

SET_GATEWAY_NODE_ID.resp(Ok)

Field name	**Byte size and order**	**Example/range**
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	**Byte size and order**	**Example/range**
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	32_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	01_h to $7F_h$: gateway node ID

SET_GATEWAY_NODE_ID.resp(NotOk)

See chapter 5.

4.6 Gateway management command

4.6.1 General

The following command definitions shall be used to implement the gateway management services as defined in /CiA 309-1/.

4.6.2 Set default network

SET_DEFAULT_NETWORK.req

Field name	**Byte size and order**	**Example/range**
Protocol control	1 byte	01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	33_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	1 to 127: Default net-work number

SET_DEFAULT_NETWORK.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	33_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0001_h
	1 byte, low	
Read/write data	1 byte	1 to 127: default network number

SET_DEFAULT_NETWORK.resp(NotOk)

See chapter 5.

4.6.3 Set the default node ID command

This service is not supported.

4.6.4 Get version

GET_VERSION.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	04_h: specified network 00_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 00_h) 1 to 255 (if protocol control is 04_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	34_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	

GET_VERSION.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	04_h: specified network 00_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 00_h) 1 to 255 (if protocol control is 04_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	7F$_h$
Index	1 byte, high	FFFF$_h$
	1 byte, low	
Sub-index	1 byte	33$_h$
Starting address	1 byte, high	0000$_h$
	1 byte, low	
Number of data values	1 byte, high	0019$_h$
	1 byte, low	
Read/write data	2 word	Vendor ID /CiA 301/
	2 word	Product code /CiA 301/
	2 word	Revision number /CiA 301/
	2 word	Serial number /CiA 301/
	1 byte	01$_h$: gateway class 1 02$_h$: gateway class 2 03$_h$: gateway class 3
	1 byte high	Protocol version (manu-facturer-specific)
	1 byte low	
	1 byte high	Implementation class (manufacturer-specific)
	1 byte low	

GET_VERSION.resp(NotOk)

See chapter 5.

4.7 Controller management command

4.7.1 General

The following commands are intended to manage a controller bundled with the gateway or in a remote node on a CANopen sub-network.

Note: Upload and download of the controller application program is done by using SDO_UPLOAD and SDO_DOWNLOAD commands.

4.7.2 Controller reset command

RESET_CONTROLLER.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	60_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

RESET_CONTROLLER.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	60_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

RESET_CONTROLLER.resp(NotOk)

See chapter 5.

4.7.3 Start controller command

START_CONTROLLER.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	61_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

START_CONTROLLER.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID.	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	61_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

RUN_CONTROLLER.resp(NotOk)

See chapter 5.

4.7.4 Stop controller command

STOP_CONTROLLER.req

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if protocol control is 01_h) 1 to 255 (if protocol control is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID.	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	62_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

STOP_CONTROLLER.resp(Ok)

Field name	Byte size and order	Example/range
Protocol control	1 byte	05_h: specified network 01_h: default network
Reserved field	1 byte	00_h
[Optional] Counter byte	1 byte	not applicable
[Optional] Network ID	1 byte	not applicable (if proto-col control is 01_h) 1 to 255 (if protocol con-trol is 05_h)
[Optional] Encoded data	1 byte	not applicable

Field name	Byte size and order	Example/range
Node ID	1 byte	$7F_h$
Index	1 byte, high	$FFFF_h$
	1 byte, low	
Sub-index	1 byte	62_h
Starting address	1 byte, high	0000_h
	1 byte, low	
Number of data values	1 byte, high	0000_h
	1 byte, low	
Read/write data	0 byte	not applicable

STOP_CONTROLLER.resp(NotOk)

See chapter 5.

5 CANopen general reference exception PDU

5.1 CANopen general reference standard exception PDU

Field name	Byte size and order	Example/range
Function code	1 byte	171 = 43 + 128 ($2B_h$ + 80_h) (indicates error)

Modbus exception code (MEC)	1 byte	*See* table 2

The MEI transport service includes nothing that is specific to any MEI type. All MEI type interface behaviors and policies are implemented by the interface. However it is possible and/or desirable to map some interface failure conditions to general predefined Modbus exception codes.

Table 2: Modbus exception codes

MEC	**Modbus name**	**MEI meaning**
01_h	Illegal function code	Function code 43 not supported or function code 43 and MEI type not supported.
02_h	Illegal data address	MEI type specific meaning
03_h	Illegal data values	MEI type specific meaning
04_h	Server failure	The server failed to execute a MEI type method
05_h	Acknowledge	The server accepted the method invocation but the method requires a relatively long time to execute. The server therefore returns only an acknowledgement of the method invocation receipt. The disposition of method completion is an aspect of the MEI type interface design and is transparent to the MEI transport service.
06_h	Server busy	The server was unable to accept the method invocation transported by the MEI transport service. The client application has the responsibility of deciding if and when to resend the method invocation.

5.2 CANopen general reference extended exception PDU

This chapter defines the CANopen general reference extended exception response PDU. This response shall be generated when there is an exception with an associated sub exception code.

Field name	**Byte size and order**	**Example/range**
Function code	1 byte	171 = 43 + 128 ($2B_h$ + 80_h) (indicates error)
Modbus exception code	1 byte	FF_h (extended exception)
Extended exception length	2 byte	1 to n+1
MEI type	1 byte	13 ($0D_h$)
Exception data	n byte	implementation-specific

Modbus exception code

The *Modbus exception code* is set to FF_h to indicate that this exception response is an extended response.

Extended exception length

The *extended exception length* represents the length of the extended exception response. It is the addition of MEI type length and exception data length.

Exception data

The *exception data* identifies implementation specific error information that may be reported when processing the CANopen general reference request.

5.2.1 Unsupported options exception PDU

This chapter defines the CANopen general reference extended exception response PDU that is generated when the protocol control options selected in the message are not supported**.**

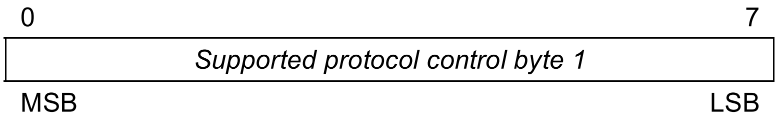
Field name	Byte size and order	Example/range
Function code	1 byte	171 = 43 + 128 (2B$_h$ + 80$_h$) (indicates error)
Modbus exception code	1 byte	FF$_h$ (extended exception)
Extended exception length	2 bytes	2 + length of Supported protocol control
MEI type	1 byte	13 (0D$_h$)
Exception code	1 byte	AE$_h$
Supported protocol control	1 to 2 byte	*See* description below

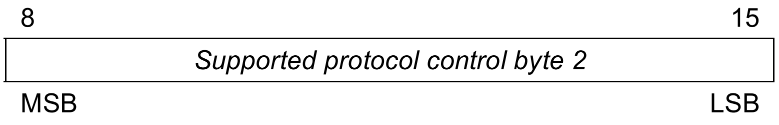
Exception code

The code of AE$_h$ shall indicate that a protocol version exception has occurred. The exception occurs when the server does not support a protocol version requested by the client.

Supported protocol control

Provides the protocol control options supported by the server.





If a protocol option is supported, the server shall set the corresponding bit as defined in 3.2.3.4.2 Protocol control to 1 in the exception PDU response.

If the bit 1 that corresponds to the *extend protocol control* is set to 1, then the supported protocol control field shall be 2 byte long.

If the bit 1 that corresponds to the extend protocol control is set to 0, then the supported protocol control field shall be 1 byte long.

5.2.2 SDO/PDO services, network management and server services complete

This chapter defines the CANopen general reference extended exception response PDU that is generated when the server is unable to complete successfully a client request

In this format a complete exception and error code is provided according the underlying used SDO services.

Field name	Byte size and order	Example/range
Function code	1 byte	171 = 43 + 128 ($2B_h$ + 80_h) (indicates error)
Modbus exception code	1 byte	FF_h (extended exception)
Extended exception length	2 byte	6
MEI Type	1 byte	13 ($0D_h$)
Exception code	1 byte	CE_h
Error code	4 byte	*See* description below

Exception code

The value of CE_h shall indicate that an exception has occurred while performing a requested services.

Error code

Provides the type of error that has occurred as defined in Appendix A.

6 Usage

6.1 General

This chapter illustrates the general processing of the CANopen general reference request.

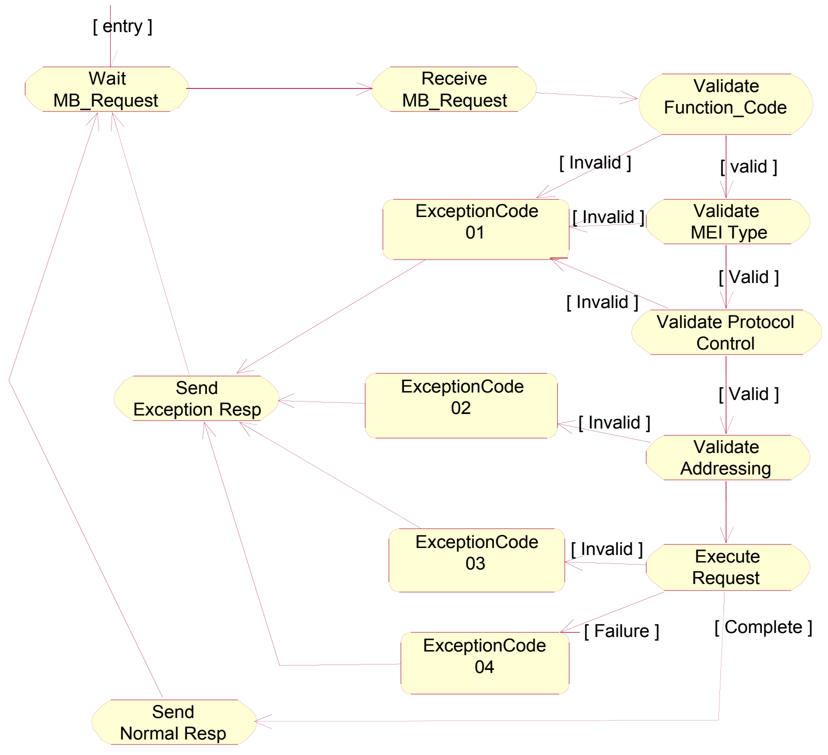


Fig. 2: General request processing

6.2 Simple Modbus request

If an object dictionary entry is accessed using one Modbus message then a simple Modbus request is used. All the necessary information needed to complete the access of the object dictionary is contained within one Modbus request.

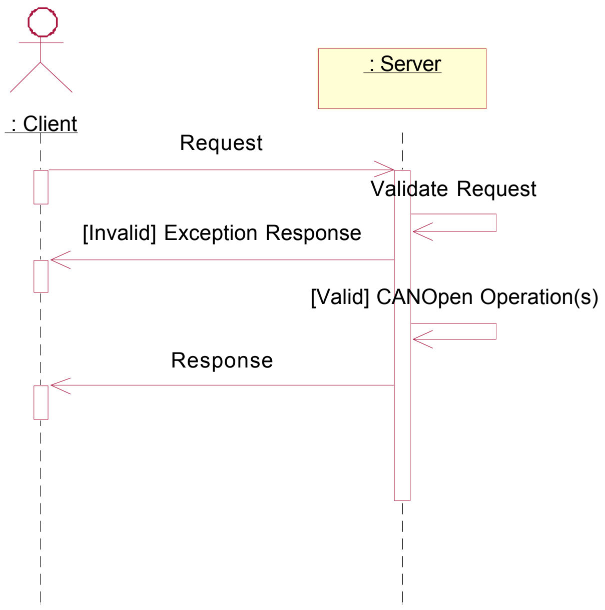


Fig. 3: Simple CANopen request

Request

Represents the CANopen general reference request generated by the client.

Validate request

The *server validates* the content and protocol options of the request.

[Invalid] Exception response

Represents the *exception response* (regular or extended) generated by the server if there is an error in the content or protocol options.

[Valid] CANopen operation(s)

If the request is valid, performing one or more *CANopen* operations processes the request. The number and type of CANopen operations to be performed is implementation specific.

Response

Represents the CANopen general reference response generated by the server. The response could be a normal response, exception response, or extended exception response depending on what occurs when processing the request.

6.3 Extended request

6.3.1 General

If an object dictionary entry is larger than a Modbus message, the extended *CANopen general reference command* shall be used to access the object dictionary entry. The object dictionary entry is then *extended* over multiple Modbus messages, each message being a fragment of the data set.

6.3.2 Extended request with the counter field

The following scenario is an example with the following conditions used to describe using the extended request using the counter field. The object dictionary entry is *extended* over three fragments.

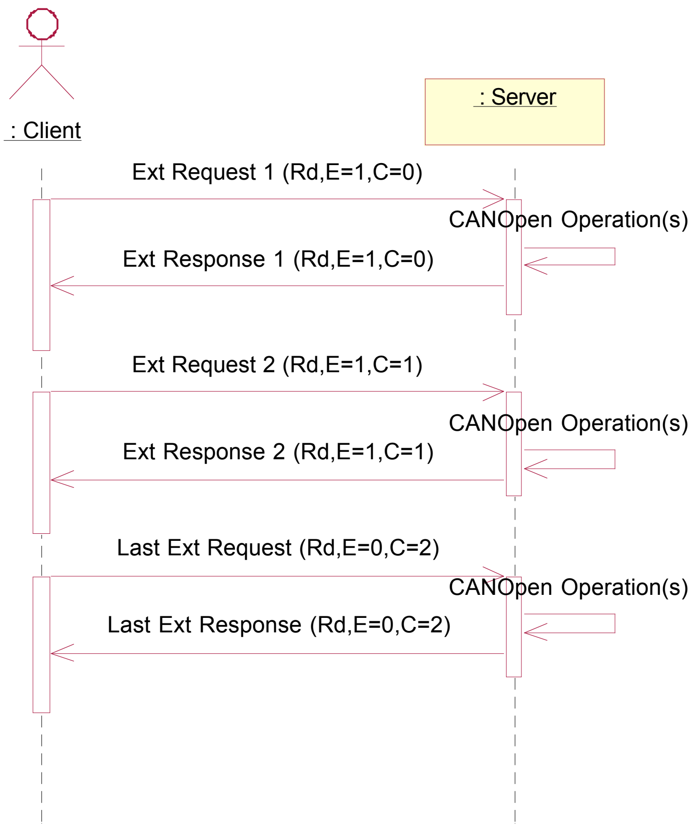

: Client
: Server
Ext Request 1 (Rd,E=1,C=0)
CANOpen Operation(s)
Ext Response 1 (Rd,E=1,C=0)
Ext Request 2 (Rd,E=1,C=1)
CANOpen Operation(s)
Ext Response 2 (Rd,E=1,C=1)
Last Ext Request (Rd,E=0,C=2)
CANOpen Operation(s)
Last Ext Response (Rd,E=0,C=2)

Fig. 4: Extended CANopen request with counter

Ext request 1 (Rd, E=1, C=0)

Represents a valid extended CANopen general reference request that contains the first fragment of the object dictionary entry. The *Rd* indicates that this is a read request. The *E*=1 indicates that extend flag = 1, the start of an extended request. The *C=0* (Counter) indicates that this is the first fragment of an extended request.

CAN operation(s)

This is representative of performing one or more CANopen operations to process a particular request. The number and type of CANopen operations to be performed is implementation specific.

Ext response 1 (Rd, E=1, C=0)

Represents the extended CANopen general reference response generated by the server. The response is a normal extended response that reflects the request.

Ext request 2 (Rd, E=1, C=1), Ext response 2 (Rd, E=1, C=1)

Represents a valid extended CANopen general reference request/response operation for the second fragment of object dictionary entry. The *Rd* is to indicate that all fragments shall contain the same access operation. The *E=1* indicates extend flag = 1, the continuation of an extended request. The *C=1* indicates that the this is the next extended request fragment in sequence.

Last ext request (Rd, E=0, C=2), Last ext response (Rd, E=0, C=2)

Represents a valid extended CANopen general reference request/response operation for the last fragment of object dictionary entry, as indicated by the *E=0.* The *C=2* indicates this is the next extended request fragment in sequence.

6.3.3 Extended request – Invalid fragment

This section defines the example of an *invalid fragment* failure that occurs during the *extended request* processing.

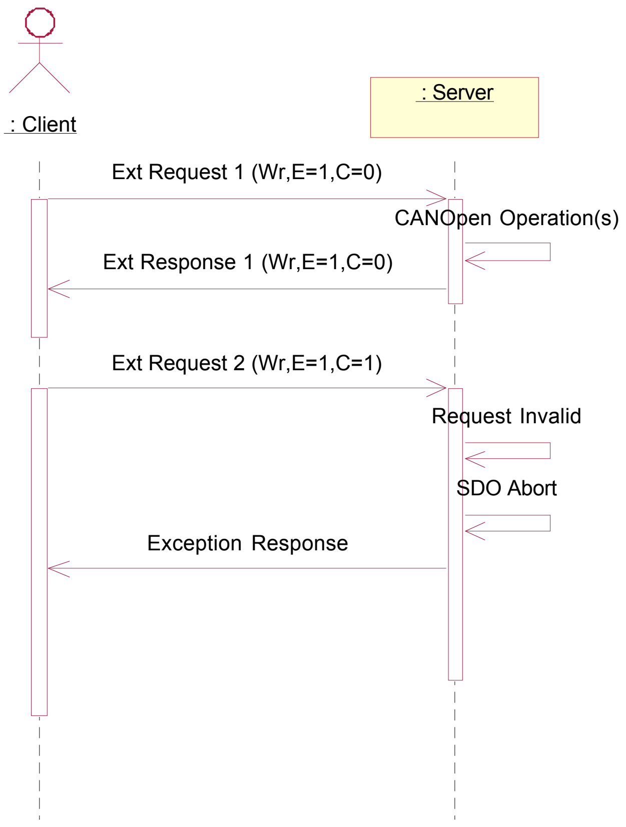

: Client
: Server
Ext Request 1 (Wr,E=1,C=0)
CANOpen Operation(s)
Ext Response 1 (Wr,E=1,C=0)
Ext Request 2 (Wr,E=1,C=1)
Request Invalid
SDO Abort
Exception Response

Fig. 5: Extended request failure – Invalid fragment

Ext request 1 (Wr, E=1, C=0), Ext response 1 (Wr, E=1, C=0)

Represents a valid extended CANopen general reference request/response operation for first fragment of the object dictionary entry. The *E=1* indicates extend flag=1, the start of an extended request. The *C=0* indicates that this is the first fragment of an extended request.

CANopen operation(s)

This is representative of performing one or more CANopen operations to process a particular request. The number and type of CANopen operations to be performed is implementation specific.

Ext request 2 (Wr, E=1, C=1), Request invalid

Represents an invalid extended CANopen general reference request for the second fragment of object dictionary entry.

SDO abort, exception response

An SDO abort should occur to clear the object dictionary entry update. An exception response is generated to indicate an error in the request

6.3.4 Incomplete extended request

6.3.4.1 Occurred during extended request processing

This chapters gives the example of an *incomplete extended request* failure that occurs during the *extended request* processing.

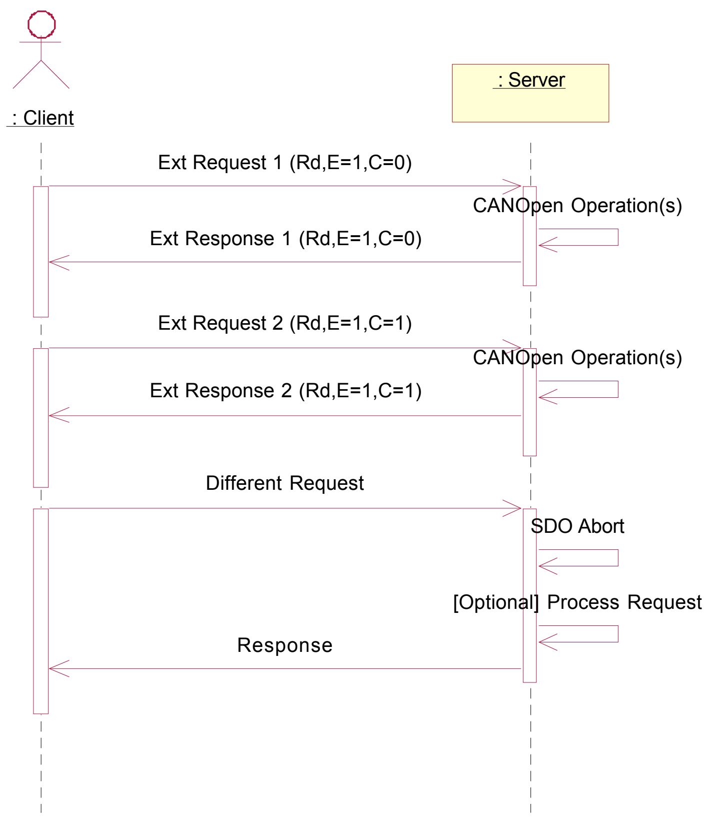

: Client
: Server
Ext Request 1 (Rd,E=1,C=0)
CANOpen Operation(s)
Ext Response 1 (Rd,E=1,C=0)
Ext Request 2 (Rd,E=1,C=1)
CANOpen Operation(s)
Ext Response 2 (Rd,E=1,C=1)
Different Request
SDO Abort
[Optional] Process Request
Response

Fig. 6: Extended request failure – Incomplete request

Ext request 1 (Rd, E=1, C=0), Ext response 1 (Rd, E=1, C=0)

Represents a valid extended CANopen general reference request/response operation for first fragment of the object dictionary entry. The *E=1* indicates that this is the start of an extended request. The *C=0* indicates that this is the first fragment of an extended request.

Ext request 2 (Rd, =1, C=1), Ext response 2 (Rd, E=1, C=1)

Represents a valid extended CANopen general reference request/response operation for the second fragment of the object dictionary entry.

Different request

Represents the server not getting the last extended request but getting a different Modbus request.

SDO abort

An SDO abort should occur to clear the object dictionary entry update.

[Optional] Process request

Depending on the implementation the new request may be processed after the SDO abort has completed.

Response

A Modbus response shall be generated. Depending on the implementation, the response may be an exception response indicating an extended request failure, or a response based on the results of processing the new request.

6.3.4.2 Client tries to resume prior extended transfer

This is an extension of the “Incomplete Extended request” use case.

After processing with the “Different request”, the server receives the “ExtRequest3” that was expected previously after “ExtRequest2”. As the server has aborted the SDO services related to this request, an extended exception response with the exception response with the code FFFF 1004_h will be send by the server.

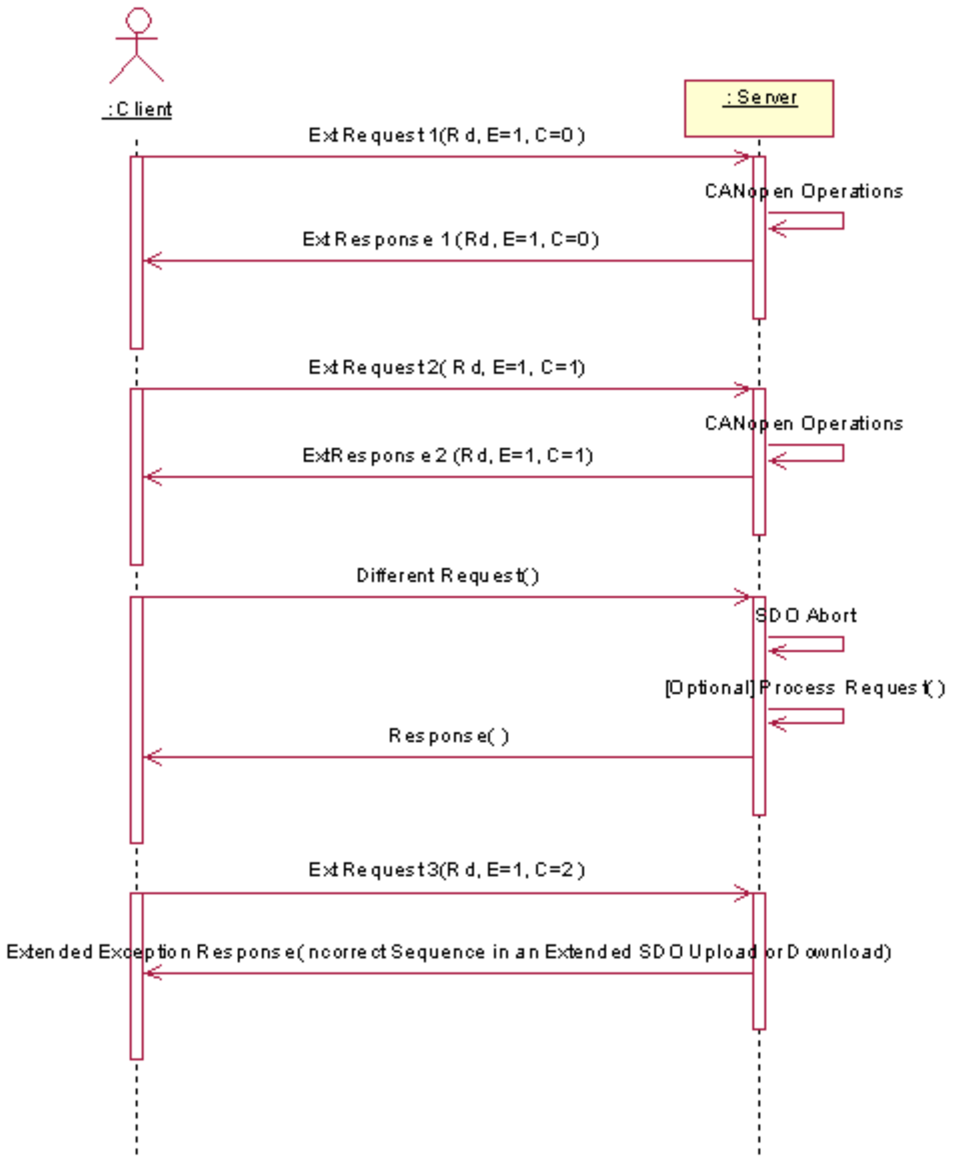

: Client
: Server
ExtRequest1(Rd, E=1, C=0)
CANopen Operations
ExtResponse 1 (Rd, E=1, C=0)
ExtRequest2(Rd, E=1, C=1)
CANopen Operations
ExtResponse 2 (Rd, E=1, C=1)
Different Request()
SDO Abort
[Optional] Process Request()
Response()
ExtRequest3(Rd, E=1, C=2)
Extended Exception Response(ncorrect Sequence in an Extended SDO Upload or Download)

Fig 7: Extended Request Failure – Icomplete Request – Client Try to Resume sequence

Ext Request 1 (Rd,E=1,C=0), Ext Response 1 (Rd,E=1,C=0)

Represents a valid Extended CANopen General Reference Request/Response operation for first fragment of the Object Dictionary Entry. The **"E=1"** indicates that this is the start of an extended request. The **"C=0"** indicates that this is the first fragment of an extended request.

Ext Request 2 (Rd,E=1,C=1), Ext Response 2 (Rd,E=1,C=1)

Represents a valid Extended CANopen General Reference Request/Response operation for the second fragment of the Object Dictionary Entry.

Different Request

Represents the server not getting the last Extended Request but getting a different MODBUS request.

SDO Abort

An SDO Abort should occur to clear the Object Dictionary entry update.

[Optional] Process Request

Depending on the implementation the new request may be processed after the SDO Abort has completed.

Response

A MODBUS Response shall be generated. Depending on the implementation, the response may be an exception response indicating an Extended Request failure, or a Response based on the results of processing the new request.

Ext Request 3 (Rd,E=1,C=2), Extended Exception Response(ncorrect Sequence in an Extended SDO Upload or Download)

Represents an invalid Extended CANopen General Reference Request/Response operation for the third fragment of the Object Dictionary Entry.

6.3.4.3 Client starts a new extended request

This is an extension of the "Incomplete Extended request" use case.

After processing with the "Different request", the server receives the "ExtRequest1" of a completely new extended SDO_DOWNLOAD or SDO_UPLOAD transfer. As the server has aborted the SDO services it is ready to proceed to a new extended transfer and then it will handle this request.

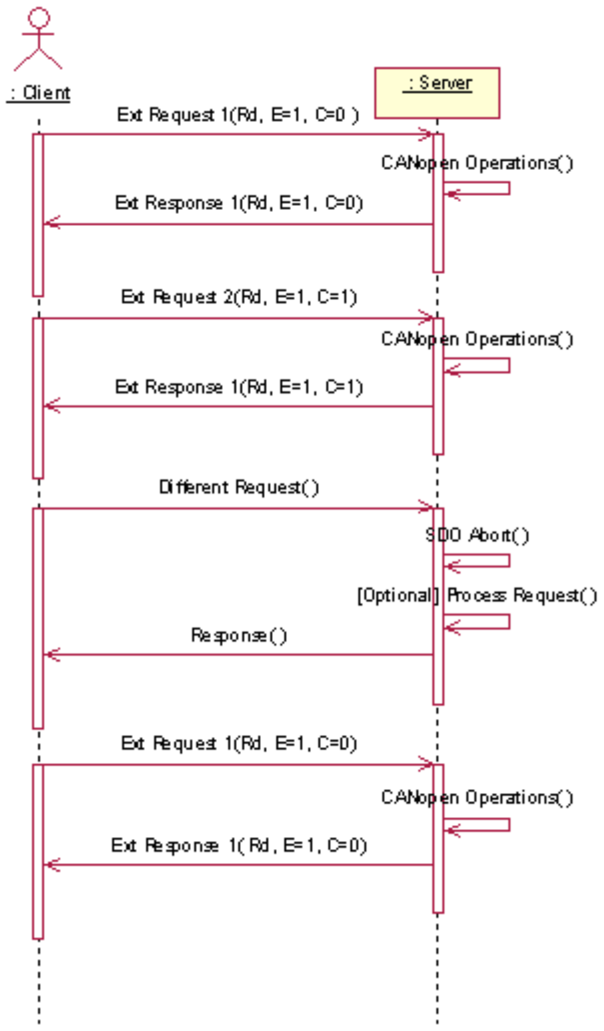

: Client
: Server
Ext Request 1(Rd, E=1, C=0)
CANopen Operations()
Ext Response 1(Rd, E=1, C=0)
Ext Request 2(Rd, E=1, C=1)
CANopen Operations()
Ext Response 1(Rd, E=1, C=1)
Different Request()
SDO Abort()
[Optional] Process Request()
Response()
Ext Request 1(Rd, E=1, C=0)
CANopen Operations()
Ext Response 1(Rd, E=1, C=0)

Fig 8: Extended request failure – complete request – Client starts a new extended requests

Ext Request 1 (Rd,E=1,C=0), Ext Response 1 (Rd,E=1,C=0)

Represents a valid Extended CANopen General Reference Request/Response operation for first fragment of the Object Dictionary Entry. The **"E=1"** indicates that this is the start of an extended request. The **"C=0"** indicates that this is the first fragment of an extended request.

Ext Request 2 (Rd,E=1,C=1), Ext Response 2 (Rd,E=1,C=1)

Represents a valid Extended CANopen General Reference Request/Response operation for the second fragment of the Object Dictionary Entry.

Different Request

Represents the server not getting the last Extended Request but getting a different MODBUS request.

SDO Abort

An SDO Abort should occur to clear the Object Dictionary entry update.

[Optional] Process Request

Depending on the implementation the new request may be processed after the SDO Abort has completed.

Response

A MODBUS Response shall be generated. Depending on the implementation, the response may be an exception response indicating an Extended Request failure, or a Response based on the results of processing the new request.

Ext Request 1 (Rd,E=1,C=0), Ext Response 1 (Rd,E=1,C=0)

Represents a valid Extended CANopen General Reference Request/Response operation for first fragment of the Object Dictionary Entry. The **"E=1"** indicates that this is the start of an extended request. The **"C=0"** indicates that this is the first fragment of an extended request.

6.3.5 Extended request – Missing fragment

This chapters gives the example of an extended request failure that occurs because of a missing fragment.

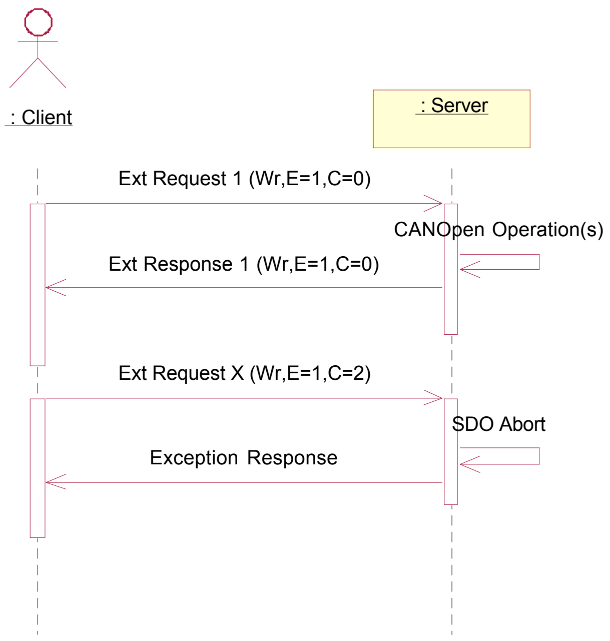

: Client
: Server
Ext Request 1 (Wr,E=1,C=0)
CANOpen Operation(s)
Ext Response 1 (Wr,E=1,C=0)
Ext Request X (Wr,E=1,C=2)
SDO Abort
Exception Response

Fig. 9: Extended request – Missing fragment

Ext request 1 (Wr, E=1, C=0), Ext response 1 (Wr, E=1, C=0)

Represents a valid extended CANopen general reference request/response operation for first fragment of the object dictionary entry. The *E=1* indicates that this is the start of an extended request. The *C=0* indicates that this is the first fragment of an extended request.

Ext request X (Wr, E=1, C=2)

Represents a fragment that is out of sequence. This is indicated by the counter not in sequence.

SDO abort, exception response

An SDO abort should occur to clear the object dictionary entry update. An exception response is generated to indicate an error in the request.

6.3.6 Receiving extended SDO request from a 2nd client while performing extended request from a 1st client

If the gateway is already processing an extended SDO_UPLOAD or SDO_DOWNLOAD request and that no more SDO server are accessible on the node (the second client tries to access the same network and node as the first client) or that no more SDO client are accessible on the gateway, the modbus server on the gateway will immediately send the standard exception response "SERVER BUSY" and will not proceed with the request. The client will simply have to poll the server by re-sending the request until it gets the response. The gateway will proceed with request when a SDO client and server became available.

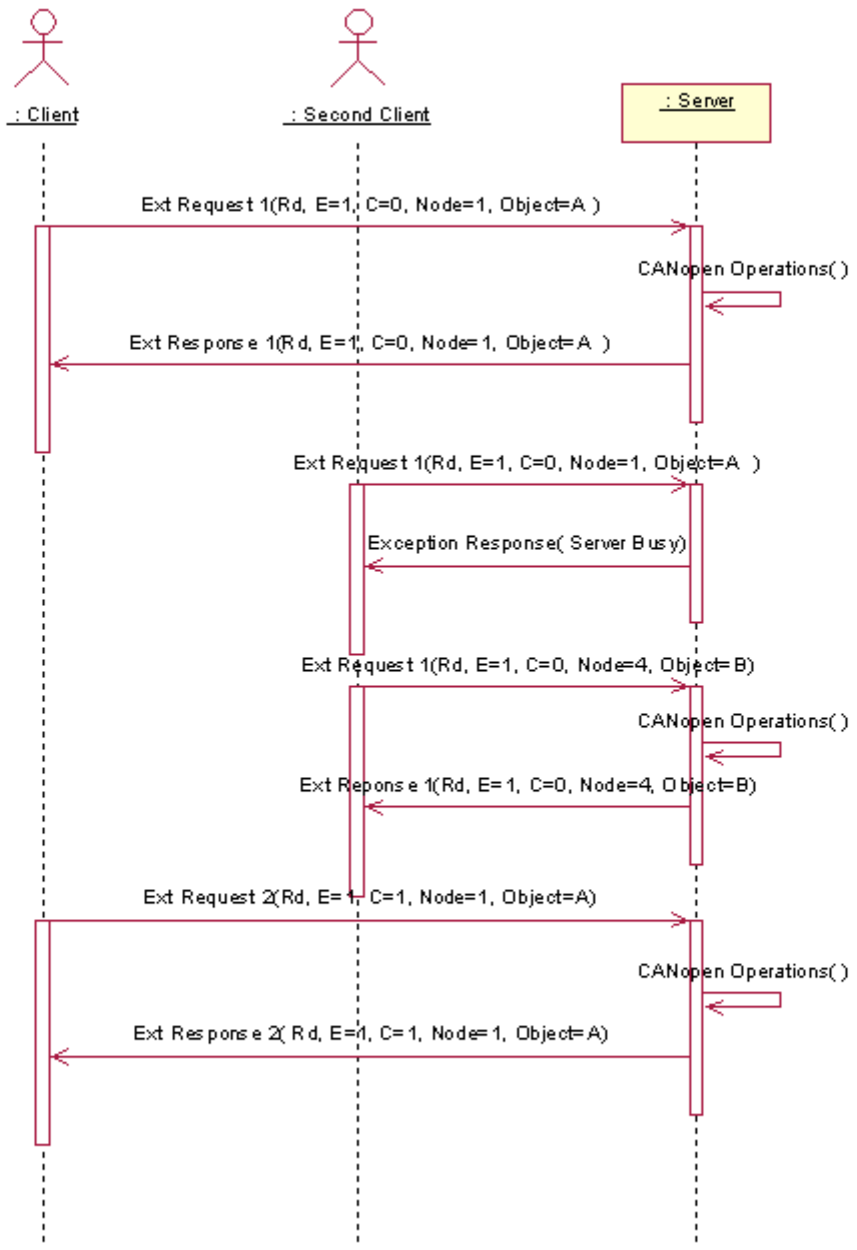

: Client
: Second Client
: Server
Ext Request 1(Rd, E=1, C=0, Node=1, Object=A)
CANopen Operations( )
Ext Response 1(Rd, E=1, C=0, Node=1, Object=A)
Ext Request 1(Rd, E=1, C=0, Node=1, Object=A)
Exception Response(Server Busy)
Ext Request 1(Rd, E=1, C=0, Node=4, Object= B)
CANopen Operations( )
Ext Reponse 1(Rd, E=1, C=0, Node=4, Object=B)
Ext Request 2(Rd, E= 1, C=1, Node=1, Object=A)
CANopen Operations( )
Ext Response 2(Rd, E=1, C=1, Node=1, Object=A)

Fig 10: Two clients tries to access the same object/node

Ext Request 1 (Rd,E=1,C=0, Node=1, Object=A), Ext Response 1 (Rd,E=1,C=0, Node=1, Object=A)

Represents a valid Extended CANopen General Reference Request/Response operation for first fragment of the Object Dictionary Entry A in Node 1 for the second client. The **"E=1"** indicates that this is the start of an extended request. The **"C=0"** indicates that this is the first fragment of an extended request.

Ext Request 1 (Rd,E=1,C=0, Node=1, Object=A), Exception Response (Server Busy)

Represents a impossible to achieve Extended CANopen General Reference Request/Response operation for first fragment of the Object Dictionary Entry A in Node 1 for the second client. The **“E=1”** indicates that this is the start of an extended request. The **“C=0”** indicates that this is the first fragment of an extended request.

Ext Request 1 (Rd,E=1,C=0, Node=4, Object=B), Ext Response 1 (Rd,E=1,C=0, Node=4, Object=B)

Represents a valid Extended CANopen General Reference Request/Response operation for first fragment of the Object Dictionary Entry B in Node 4 for the second client. The **“E=1”** indicates that this is the start of an extended request. The **“C=0”** indicates that this is the first fragment of an extended request.

Ext Request 2 (Rd,E=1,C=1, Node=1, Object=A), Ext Response 1 (Rd,E=1,C=1, Node=1, Object=A)

Represents a valid Extended CANopen General Reference Request/Response operation for first fragment of the Object Dictionary Entry A in Node 1 for the first client.

6.3.7 Simple modbus request cannot be satisfied with a simple Modbus response

This scenario occurs when a client sends an SDO_UPLOAD request to a server in a simple Modbus request, but the server cannot send the data in a simple response due to the large amount of data requested.

In this case, the server shall respond with a Modbus extended exception PDU, with 8 bytes of exception data. The first four bytes containing the error code, and the second four bytes containing the number of data values in the object dictionary entry that was requested.

Appendix A

This appendix is composed of several tables to indicate the error code to be reported in case of failure during the execution of SDO services, PDO services, network management and server services.

If the service failed as a result of the underlying SDO services used, the SDO abort code reported by the SDO client and/or server is to be transmitted as is. In that case the SDO abort code is defined by CiA 301 [1] or any relevant document.

SDO abort code starting with FFFF $xxxx_h$ are reserved and defined for abort code generated by the gateway server itself.

The table 3 shows the SDO abort code for SDO_UPLOAD and SDO_DOWNLOAD services to be used when the service execution is not related to the underlying SDO service on the CANopen sub-network.

Table 3: Error codes for SDO services

Reason code	**Reason type**
FFFF 0000_h	ok
FFFF 1001_h	Node does not exist
FFFF 1002_h	Network does not exist
FFFF 1003_h	Service not supported
FFFF 1004_h	A gap has been detected in the counter byte of the protocol control field.
FFFF 0003_h	Client/server command specifier not valid or unknown.
FFFF 0007_h	Out of memory.
FFFF 0008_h	Unsupported access to an object.
FFFF 0009_h	Attempt to read a write only object.
FFFF $000A_h$	Attempt to write a read only object.
FFFF $000B_h$	Object does not exist in the object dictionary.
FFFF $000C_h$	Object cannot be mapped to the PDO.
FFFF $000D_h$	The number and length of the objects to be mapped would exceed PDO length.
FFFF $000E_h$	General parameter incompatibility reason.
FFFF $000F_h$	General internal incompatibility in the device.
FFFF 0010_h	Access failed due to an hardware error.
FFFF 0011_h	Data type does not match, length of service parameter does not match
FFFF 0012_h	Data type does not match, length of service parameter too high
FFFF 0013_h	Data type does not match, length of service parameter too low
FFFF 0014_h	Sub-index does not exist.
FFFF 0015_h	Values range of parameter exceeded (only for write access).
FFFF 0016_h	Values of parameter written too high.
FFFF 0017_h	Values of parameter written too low.

FFFF 0018$_h$	Maximum values is less than minimum values.
FFFF 0019$_h$	General error
FFFF 001A$_h$	Data cannot be transferred or stored to the application.
FFFF 001B$_h$	Data cannot be transferred or stored to the application because of local control.
FFFF 001C$_h$	Data cannot be transferred or stored to the application because of the present device state.
FFFF 001D$_h$	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error).
FFFF 001E$_h$	Requested data object is too large to fit in a single message

Table 4: Error codes for CANopen NMT services and device failure management services

Reason code	Error type
FFFF 0000$_h$	ok
FFFF 1001$_v$	Node does not exist
FFFF 1002$_h$	Network does not exist
FFFF 1003$_h$	Service not supported

Table 5: Error codes for device failure management services

Reason code	Error type
FFFF 0000$_h$	ok
FFFF 1001$_h$	Node does not exist
FFFF 1002$_h$	Network does not exist
FFFF 1003$_h$	Service not supported

Table 6: Error codes for gateway management services

Reason code	Error type
FFFF 0000$_h$	ok
FFFF 1001$_h$	Node does not exist
FFFF 1002$_h$	Network does not exist
FFFF 1003$_h$	Service not supported

Table 7: Error codes for controller management services

Reason code	**Error type**
FFFF 0000$_h$	ok
FFFF 1001$_h$	Node does not exist
FFFF 1002$_h$	Network does not exist
FFFF 1003$_h$	Service not supported
FFFF 1004$_h$	Incorrect sequence in an extended SDO upload or download (e.g. the counter byte of the protocol field has not the exprected values according previous request or response)
FFFF 4001$_h$	Controller can’t be reset
FFFF 4002$_h$	Controller can’t be stopped
FFFF 4003$_h$	Controller can’t be started

Appendix B

Table 8: Error messages for the monitoring error service

Error Msg Nbr	Error Ms text
0	No error
100	Request not supported
101	Syntax error
200	Lost guarding message
201	Lost connection
300	Error passive
301	Bus off

Appendix C

Table 9: Protocol control values for SDO upload

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Description
0	0	0	0	0	0	0	0	Simple transfer Default network number Default data type
0	0	0	0	0	1	0	0	Simple transfer Specified network number Default data type
1	0	1	0	0	0	0	0	Extended transfer Default network number Default data type All request but last one
0	0	1	0	0	0	0	0	Extended transfer Default network number Default data type Last request of the extended transfer
1	0	1	0	0	1	0	0	Extended transfer Specified network number Default data type All request but last one
0	0	1	0	0	1	0	0	Extended transfer Specified network number Default data type Last request of the extended transfer
0	0	0	0	0	0	1	0	Simple transfer Default network number Specified data type
0	0	0	0	0	1	1	0	Simple transfer Specified network number Specified data type
1	0	1	0	0	0	1	0	Extended transfer Default network number Specified data type All request but last one
0	0	1	0	0	0	1	0	Extended transfer Default network number Specified data type Last request of the extended transfer
1	0	1	0	0	1	1	0	Extended transfer Specified network number Specified data type All request but last one
0	0	1	0	0	1	1	0	Extended transfer Specified network number Specified data type Last request of the extended transfer

Appendix D

Table 10: Protocol control values for SDO download

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Request type
0	0	0	0	0	0	0	1	Simple transfer Default network number Default data type
0	0	0	0	0	1	0	1	Simple transfer Specified network number Default data type
1	0	1	0	0	0	0	1	Extended transfer Default network number Default data type All request but last one
0	0	1	0	0	0	0	1	Extended transfer Default network number Default data type Last request of the extended transfer
1	0	1	0	0	1	0	1	Segmented transfer Specified network number Default data type All request but last one
0	0	1	0	0	1	0	1	Extended transfer Specified network number Default data type Last request of the extended transfer
0	0	0	0	0	0	1	1	Simple transfer Default network number Specified data type
0	0	0	0	0	1	1	1	Simple transfer Specified network number Specified data type
1	0	1	0	0	0	1	1	Extended transfer Default network number Specified data type All request but last one
0	0	1	0	0	0	1	1	Extended transfer Default network number Specified data type Last request of the extended transfer
1	0	1	0	0	1	1	1	Extended transfer Specified network number Specified data type All request but last one
0	0	1	0	0	1	1	1	Extended transfer Specified network number Specified data type Last request of the extended transfer