Anybus®

Safety Interface Guide

SCM-1202-024 1.4 en-US ENGLISH

Important User Information

Disclaimer

Table of Contents Page

1 Preface

1.1 Related Documents

Document	Author
Anybus CompactCom 40 Software Design Guide	HMS
Anybus CompactCom M40 Hardware Design Guide	HMS
Anybus CompactCom Host Application Implementation Guide	HMS
CIP Safety on EtherNet/IP, Generic Porting Guide	HMS / IXXAT

1.2 Document History

Version	Date	Description
1.0	2016-11-16	First Release
1.1	2016–12–06	PROFIsafe chapter added. Minor changes.
1.2	2017-09-29	Fail Safe over EtherCAT (FSoE) chapter added. Minor changes.
1.3	2019-03-07	Updated trademark information Rebranded
1.4	2020-12-21	Added the Read_Vendor_Block bootloader command. Added the PROFIsafe GetSupportedSpdus command. Updated SafetyReset command. Minor changes.

1.3 Document Conventions

Numbered lists indicate tasks that should be carried out in sequence:

1. First do this
2. Then do this

Bulleted lists are used for:

- Tasks that can be carried out in any order
- Itemized information

► An action

→ and a result

User interaction elements (buttons etc.) are indicated with bold text.

```
Program code and script examples
```

Cross-reference within this document: *Document Conventions, p. 3*

External link (URL): www.hms-networks.com

WARNING
Instruction that must be followed to avoid a risk of death or serious injury.

Caution
Instruction that must be followed to avoid a risk of personal injury.

Instruction that must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.

Additional information which may facilitate installation and/or operation.

1.4 Acronym List

Acronym	**Description**
CSAL	CIP Safety Adaptation Layer
CSS	CIP Safety Stack
HALC	Hardware Abstraction Layer Communication
PDO	Process Data Object
PDU	Process Data Unit
SDO	Service Data Object
SPDU	Safe Process Data Unit

1.5 Trademark Information

Anybus® is a registered trademark of HMS Networks.

EtherCAT® Conformance tested

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Safety over EtherCAT®

Safety over EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

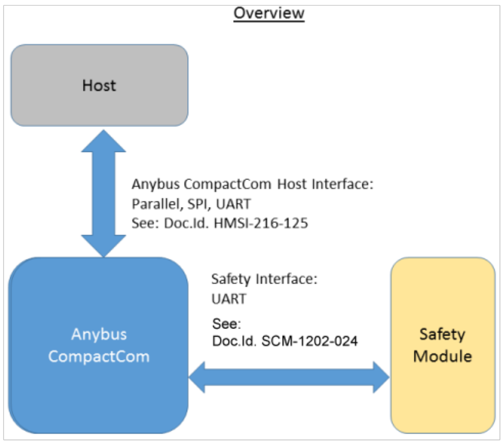
All other trademarks are the property of their respective holders.

2 About the Anybus Safety Interface Guide

2.1 General

The Anybus Safety Interface Guide describes the communication between an Anybus CompactCom device and a safety module. This guide is aimed for all users who will need to interface with the Anybus safety channel, either by developing a safety module or by developing a communication interface.

Using a safety module, it is possible for the application to transmit and receive safe I/Os and messages, embedded in a safety protocol suitable for the network.



2.2 Communication Settings

The network related communication settings should be set to the following:

- UART
- 8-bit data, no parity, 1 stop bit
- Serial communication bit order, LSB first
- No hardware flow control signals
- Baud rate accuracy should be within ±2% of configured baud rate

2.3 Endianness

All CompactCom and safety module telegrams should be transferred in a little-endian format.

2.4 Safety Processors

It is assumed that µC1 has address 0xA5 and µC2 has address 0xAA.

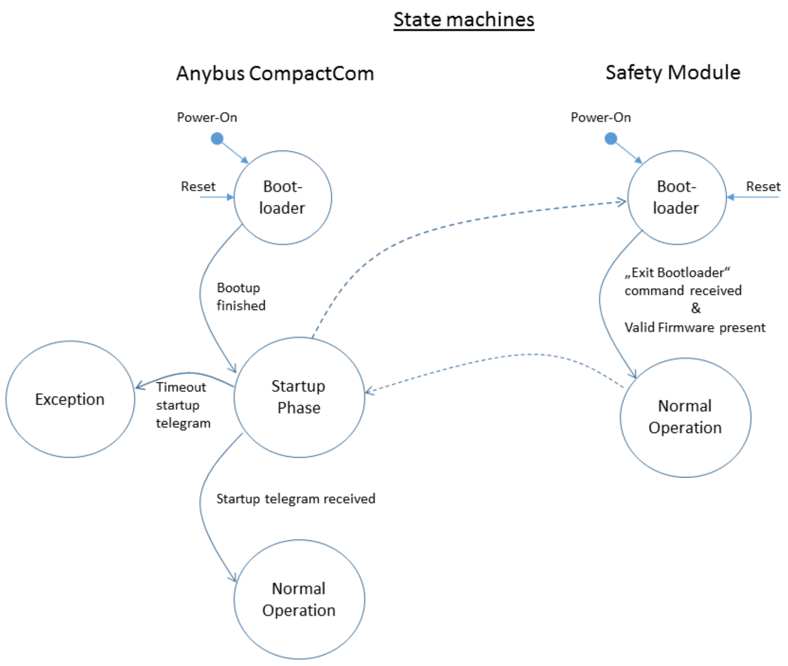
2.5 CRC Calculation

The polynomial value used for the 16-bit CRC calculation is 0xA001. The initial value is 0xFFFF.

For more information and examples concerning the CRC calculation, see the CRC Calculation (16-bit) chapter in the Anybus CompactCom 40 Software Design Guide.

3 Bootloader Mode

The Anybus CompactCom device will assume that the safety module starts up in bootloader mode. It is important that the safety module and the CompactCom device are powered on and reset at the same time, by the host application. In bootloader mode, the safety module can perform start-up tasks and system tests. The safety module must stay in bootloader mode until the CompactCom has sent an exit bootloader command.



3.1 Start-up Tasks

During bootloader mode, the CompactCom device will send commands to the safety module that it needs to be able to handle and respond to. They will be repeated every 100 ms for a total maximum of 10 times. If the safety module fails to respond during this time, the CompactCom will enter exception. The commands, and the command and response telegram structures, are listed in the tables below.

When all commands have been handled and properly responded to, the CompactCom will signal the end of bootloader mode by issuing the Exit_BL command to each of the two safety processors µC1 and µC2.

These are the commands that the safety module needs to be able to respond to during bootloader mode.

Command Code	Name	Description
0x02	Read_BL_ Version	Reads the software version of the bootloader.
0x03	Read_Appl_ Version	Reads the software version of the application software in the flash.
0x09	Exit_BL	Exit bootloader.
0x0A	Read_Module_ Type	Reads the hardware (the type of safety module).
0x0B	Read_Vendor_ Block	Reads vendor-specific information. Optional.

If the safety module receives an unknown or an unsupported command from the CompactCom, the bootloader should respond with the message "unknown command".

3.1.1 Read_BL_Version

Command

Byte	Name	Value	Description
0	Start byte	Adr. µC1/ µC2	Processor address
1 — 2	Length	0x03	Length of telegram (including CRC)
3	Command code	0x02	Command to read bootloader version
4 — 5	CRC	N/A	16 bit CRC

Response

Byte	Name	Value	Description
0	Start byte	Adr. µC1/µC2	Processor address
1 — 2	Length	0x06	Length of telegram (including CRC)
3	Command-response	0x82	This value should be calculated as the command code added by the most significant bit of the byte, eg. for command 0x02 the command-response is 0x82
4	Response code	N/A	See *Response Codes, p. 10*
5 — 6	Data	N/A	Two bytes SW version of bootloader
7 — 8	CRC	N/A	16 bit CRC

3.1.2 Read_Appl_Version

Command

Byte	Name	Value	Description
0	Start byte	Adr. µC1/ µC2	Processor address
1 — 2	Length	0x03	Length of telegram (including CRC)
3	Command code	0x03	Command to read version of application SW
4 — 5	CRC	N/A	16 bit CRC

Response

Byte	Name	Value	Description
0	Start byte	Adr. µC1/µC2	Processor address
1 — 2	Length	0x07	Length of telegram (including CRC)
3	Command-response	0x83	This value should be calculated as the command code added by the most significant bit of the byte, eg. for command 0x03 the command-response is 0x83
4	Response code	N/A	See *Response Codes, p. 10*
5 — 7	Data	N/A	Three bytes SW version of application SW. If no valid SW is found in flash, values should be 0 and the response code should be set accordingly
8— 9	CRC	N/A	16 bit CRC

3.1.3 Exit_BL

Command

Byte	Name	Value	Description
0	Start byte	Adr. µC1/ µC2	Processor address
1 — 2	Length	0x04	Length of telegram (including CRC)
3	Command code	0x09	Command to exit bootloader mode
4	-	-	Reserved
5 — 6	CRC	N/A	16 bit CRC

Response

Byte	Name	Value	Description
0	Start byte	Adr. µC1/µC2	Processor address
1 — 2	Length	0x04	Length of telegram (including CRC)
3	Command-response	0x89	This value should be calculated as the command code added by the most significant bit of the byte, eg. for command 0x09 the command-response is 0x89
4	Response code	N/A	See *Response Codes, p. 10*
5 — 6	CRC	N/A	16 bit CRC

3.1.4 Read_Module_Type

Command

Byte	Name	Value	Description
0	Start byte	Adr. µC1/ µC2	Processor address
1 — 2	Length	0x03	Length of telegram (including CRC)
3	Command code	0x0A	Command to read module type
4 — 5	CRC	N/A	16 bit CRC

Response

Byte	Name	Value	Description
0	Start byte	Adr. µC1/µC2	Processor address
1 — 2	Length	0x0E	Length of telegram (including CRC)
3	Command-response	0x8A	This value should be calculated as the command code added by the most significant bit of the byte, eg. for command 0x0A the command-response is 0x8A
4	Response code	N/A	See *Response Codes, p. 10*
5 — 6	Module type	N/A	Two byte module type Vendor specific value, provided by HMS
7 — 8	Network type	N/A	Two byte network type 0x00A6 for Safety over EtherCAT 0x00A5 for CIP Safety on EtherNet/IP 0x00A1 for PROFIsafe on PROFINET
9 — 10	HW ID	N/A	Two byte hardware ID
11 — 14	Serial number	N/A	Four byte serial number
15 — 16	CRC	N/A	16 bit CRC

3.1.5 Read_Vendor_Block

Command

Byte	Name	Value	Description
0	Start byte	Adr. µC1/ µC2	Processor address
1 — 2	Length	0x03	Length of telegram (including CRC)
3	Command code	0x0B	Command to read Vendor Data Block
4 — 5	CRC	N/A	16 bit CRC

Response

Byte	Name	Value	Description
0	Start byte	Adr. µC1/µC2	Processor address
1 — 2	Length	4+n	Length of telegram (including CRC)
3	Command-response	0x8B	This value should be calculated as the command code added by the most significant bit of the byte, eg. for command 0x0B the command-response is 0x8B
4	Response code	N/A	See *Response Codes, p. 10*
5 —4 + n	Vendor specific	N/A	Vendor specific data, where n = number of vendor specific bytes and n > 0. The information in this response is mapped to attribute 12 and 13 in the Functional Safety Module object in CompactCom.
5 + n — 6 + n	CRC	N/A	16 bit CRC

3.1.6 Response Codes

Value	Description
0x00	Command successful
0x01	Command in progress
0x02	Error, unknown command
0x03	Checksum error of the Intel HEX file
0x04	Error, invalid/no application software in the flash memory
0x06	Error, unable to access flash memory
0x07	Error, invalid parameter

4 System Start

The following figure depicts the boot and start-up process.

In the example, an IXXAT Safe T100 safety module from HMS is used. For that module, the start-up telegram is sent every 25 ms. For the T100, the configured cycle time for cyclic telegrams is 2 ms.

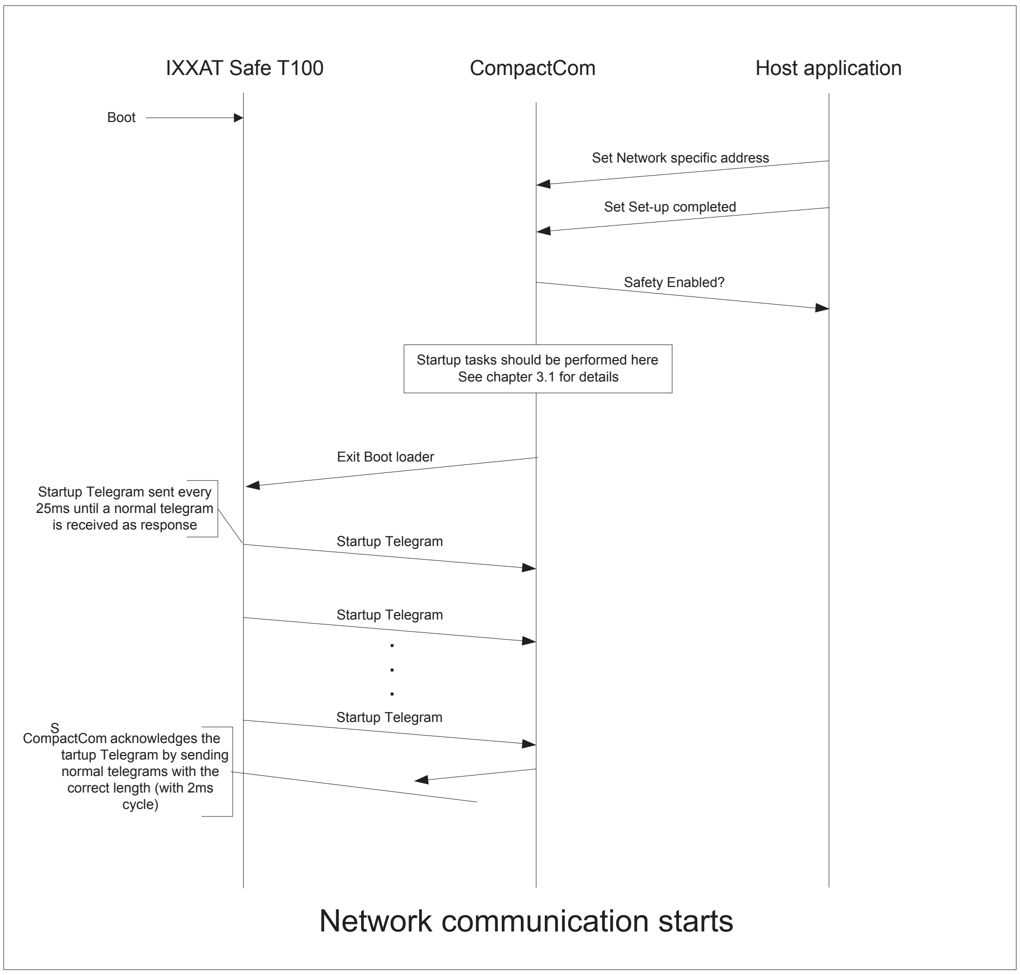

IXXAT Safe T100
CompactCom
Host application
Boot
Set Network specific address
Set Set-up completed
Safety Enabled?
Startup tasks should be performed here
See chapter 3.1 for details
Exit Boot loader
Startup Telegram sent every 25ms until a normal telegram is received as response
Startup Telegram
Startup Telegram
Startup Telegram
S
CompactCom acknowledges the tartup Telegram by sending normal telegrams with the correct length (with 2ms cycle)
Network communication starts

Fig. 1 Network communication using a CompactCom and an IXXAT Safe T100 safety module

The safety module should exit the bootloader mode after the CompactCom device has sent the exit bootloader command. After exit, the safety module should send a start-up telegram to the CompactCom, with information about itself and input and output data sizes. This telegram should be sent repeatedly, with a delay of between 10 and 100 ms, until the safety device receives a correct response telegram, with the correct output data size, from the CompactCom device.

If the CompactCom does not receive a start-up telegram within 1000 ms after receiving the exit bootloader response, the device will enter Exception mode.

This marks the end of the start-up phase, and normal safe communication can start.

4.1 The Start-up Telegram

Byte	Name	Value
0–1	Vendor ID	Identifies vendor. Apply at HMS for a unique Vendor ID.
2–3	IO Channel ID	IO configuration descriptor. Vendor specific. Refer to the safety module documentation for details.
4–6	Firmware version	The firmware version number of the safety module. The version 1.12.03 corresponds to 0x03, 0x0C, 0x01.
7–10	Serial number	Vendor specific 32-bit unique serial number.
11	Safe output PDU size	Size (in bytes) for the safe output PDU data transferred from the PLC to the CompactCom device.
12	Output data size	Data size in bytes of the safety module’s safe outputs.
13	Safe input PDU size	Size (in bytes) for the safe input PDU data transferred from the CompactCom device to the PLC.
14	Input data size	Data size in bytes of the safety module’s safe inputs.
15-16	CRC	-

5 Cyclic Telegrams

The data frames transferred on the serial line between the Anybus CompactCom device and the safety module will be called telegrams.

These telegrams are produced cyclically by both the CompactCom device and the safety module. The communication between the devices does not require synchronization.

5.1 The Anybus Telegram Structure

After the initial phase, where the I/O lengths are determined, the Anybus CompactCom device will transfer the following telegram.

Name	**Size in Bytes**	**Description**
Ctrl/Status	1	Control and status information. See table below.
Msg	16	Message part of the telegram. It is used as a fragmented protocol on top of the cyclical data exchange. Refer to *Messages, p. 16*.
SPDU	SPDU_OUT_LEN	Safety protocol data unit from the network. Size specified in startup telegram.
CRC	2	16-bit CRC field.

5.1.1 Ctrl/Status Byte Information

The Ctrl/Status byte consists of the following.

Bit	Name	Description
0-2	Anybus State	State of the Anybus device. See the Anybus State tables below.
3-5	Reserved	-
6	MsgFragSafe	Bit used for fragmentation of messages. Refer to *Messages, p. 16*.
7	MsgFragAnybus	Bit used for fragmentation of messages. Refer to *Messages, p. 16*.

Anybus State

In the Anybus State field, the CompactCom device informs the safety module which state it expects the safety module to be in.

Anybus State for PROFIsafe on PROFINET

Value	State	Description
0	R_PRM	New parameterization data shall be allowed. I/O data is not valid.
1	R_STOP	Connected to PLC. I/O data is not valid.
2	R_RUN	Connected to PLC. I/O data is valid.
3..7	Reserved	-

Anybus State for Fail Safe over EtherCAT (FSoE)

Value	State	Description
0	R_BOOT	Requests to transit to Bootstrap state
1	R_INIT	Requests to transit to Init state
2	R_PREOP	Requests to transit to Preoperational state
3	R_SAFEOP	Requests to transit to Safe-operational state
4	R_OP	Requests to transit to Operational state
5..7	Reserved	-

Anybus State for CIP Safety on EtherNet/IP

Value	State	Description
0	R_NO	No request
1	R_ABORT	Requests to transit to abort state
3..7	Reserved	-

5.2 The Safety Module Telegram Structure

After the initial phase, where the I/O lengths are determined, the safety module shall transfer the following telegram.

Name	Size in bytes	Description
Ctrl/Status	1	Control and status information
Msg	16	Message part of the telegram.
SPDU	SPDU_IN_LEN	Safety protocol data unit to the network. Size specified in startup telegram.
Data In	DATA_IN_LEN	Value of safe inputs. Size specified in startup telegram.
Data Out	DATA_OUT_LEN	Value of safe outputs. Size specified in startup telegram.
CRC	2	16-bit CRC field

5.2.1 Ctrl/Status

The Ctrl/Status byte consists of the following:

Bit	Name	Description
0-3	State	State of the safety module. See the safety module state tables below.
4-5	Reserved	-
6	MsgFragSafe	Bit used for fragmentation of messages. Refer to *Messages, p. 16*.
7	MsgFragAnybus	Bit used for fragmentation of messages. Refer to *Messages, p. 16*.

Safety Module State

In the State field, the safety module informs the CompactCom device which state it expects the CompactCom device to be in.

The safety module may send other states, but those below are the only ones the CompactCom device acknowledges.

Safety Module State for PROFIsafe on PROFINET

Value	State	Description
15	FAIL SAFE	Fail safe state. The transition to this state may happen at any time, also during an ongoing fragmented message transfer. In order to detect this, the CompactCom aborts this fragmented transfer.

Safety Module State for Fail Safe over EtherCAT (FSoE)

Value	State	Description
15	FAIL SAFE	Fail safe state. The transition to this state may happen at any time, also during an ongoing fragmented message transfer. In order to detect this, the CompactCom aborts this fragmented transfer.

Safety Module State for CIP Safety on EtherNet/IP

Value	State	Description
3	WAIT_TUNID	Waiting for TUNID state.
13	ABORT	Abort state.
15	FAIL SAFE	Fail safe state. The transition to this state may happen at any time, also during an ongoing fragmented message transfer. In order to detect this, the CompactCom aborts this fragmented transfer.

6 Messages

For acyclic data exchange, where time is noncritical, a fragmented message protocol is available. A message part is embedded in all telegrams transferred between the CompactCom device and the safety module.

In the Ctrl/Status word, there are two bits that are used to control message fragmentation: MsgFragSafe and MsgFragAnybus. These bits are used in pairs, to indicate new fragments and the valid reception of fragments. The MsgFragSafe bit pair is used for messages transferred by the safety module, and the MsgFragAnybus bit pair for messages transferred by the Anybus CompactCom.

A new fragment may only be transferred, if the bits are equal. If they are equal, the transmitting part puts the next fragment into the telegram to be sent, and toggles the corresponding bit in the Ctrl/Status byte. That same fragment must then be sent in every telegram, until the receiving part acknowledges the reception by toggling the corresponding bit so that both the receiving part’s bit and the transmitting part’s bit has the same value.

A message may consist of 1 to several fragments. The maximum amount of fragments is network specific.

To be able to know when a message has been received in full, the length of all received fragments should be calculated and matched towards the length of the message.

A response to a specific message should not be generated until the message has been received in full.

The message header should only be part of the first fragment.

An error response can be issued for any request. See *Error Response, p. 17*.

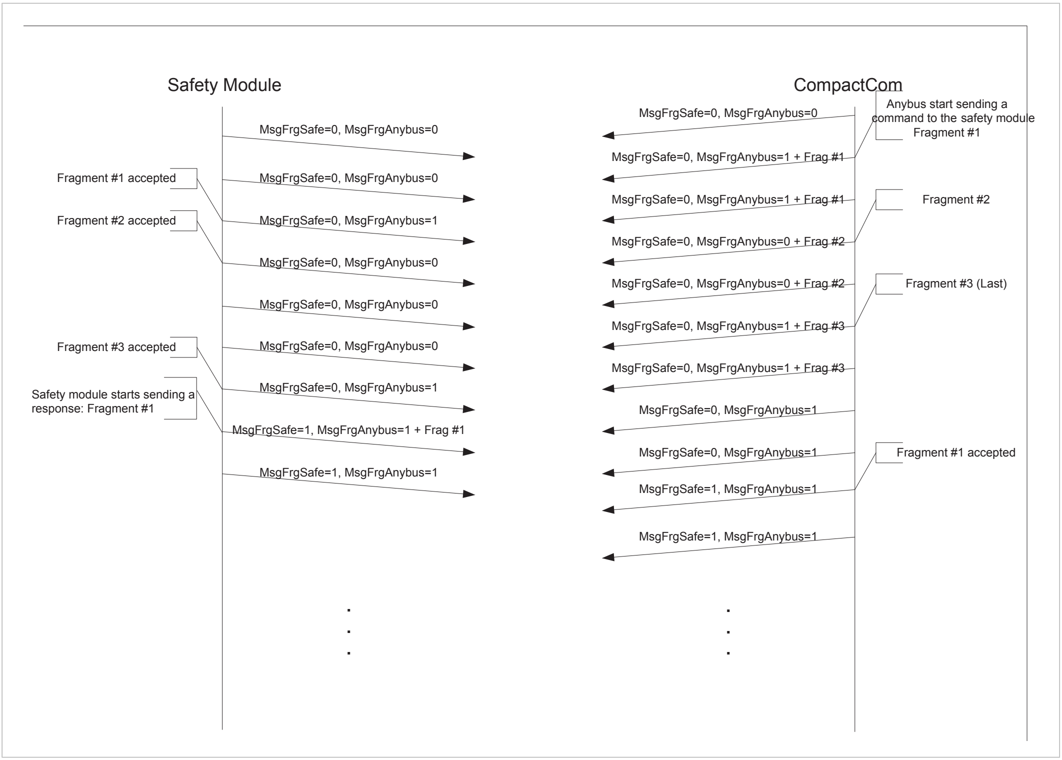

Safety Module
CompactCom
MsgFrgSafe=0, MsgFrgAnybus=0
Anybus start sending a command to the safety module Fragment #1
MsgFrgSafe=0, MsgFrgAnybus=0
MsgFrgSafe=0, MsgFrgAnybus=1 + Frag #1
Fragment #1 accepted
MsgFrgSafe=0, MsgFrgAnybus=0
MsgFrgSafe=0, MsgFrgAnybus=1 + Frag #1
Fragment #2
Fragment #2 accepted
MsgFrgSafe=0, MsgFrgAnybus=1
MsgFrgSafe=0, MsgFrgAnybus=0 + Frag #2
MsgFrgSafe=0, MsgFrgAnybus=0
MsgFrgSafe=0, MsgFrgAnybus=0 + Frag #2
Fragment #3 (Last)
MsgFrgSafe=0, MsgFrgAnybus=0
MsgFrgSafe=0, MsgFrgAnybus=1 + Frag #3
Fragment #3 accepted
MsgFrgSafe=0, MsgFrgAnybus=0
MsgFrgSafe=0, MsgFrgAnybus=1 + Frag #3
MsgFrgSafe=0, MsgFrgAnybus=1
Safety module starts sending a response: Fragment #1
MsgFrgSafe=0, MsgFrgAnybus=1
MsgFrgSafe=1, MsgFrgAnybus=1 + Frag #1
MsgFrgSafe=0, MsgFrgAnybus=1
Fragment #1 accepted
MsgFrgSafe=1, MsgFrgAnybus=1
MsgFrgSafe=1, MsgFrgAnybus=1
MsgFrgSafe=1, MsgFrgAnybus=1

Fig. 2 Fragmentation Example

6.1 Message Header

Name	Size in bytes	Description
ID	1	The ID is used to tie a request to a corresponding response, in case there are several simultaneous requests at the same time. The CompactCom device and the safety module may use the ID byte independently from each other. This means that the CompactCom can use ID = 1 for a request, even though the safety module has already issued a request with ID = 1. For more information about the limitations concerning the number of simultaneous requests, see the respective user manuals.
Req/Rsp	2	Describes what is contained in the data part of the message. See the table below
Length	1–2 (Network specific)	Amount of bytes in the data field (0-255/0-65535: Network specific)
Data	Length	Message specific information.

6.1.1 Request/Response

Bit	Name	Description
15	ERR	Error bit which can be set in a response message in case of error message
14	CMD	Bit set to indicate that this is a command request. Cleared for a response message.
13-8	Reserved	Reserved for future use. Set to zero.
7-0	Function	Number of the function. Mirrored in the response.

6.2 Error Response

Byte	Name	Value	Description
1	ID	ID	Mirrored from request
2-3	Req/Rsp	0x80XX	XX = Function code mirrored from request
4	Length	0x02	2 bytes of error code following
5-6	Data	0xYYXX	Error code. See table below for response error codes.

Response Error Codes		
Value	**Error code**	**Description**
0x0001	Unsupported function	The received request with the specified function code is not supported
0x0002	Invalid request length	The length of the received request didn’t match the expected length.
0x0003	Invalid data	The data passed is invalid in some way (e.g., out-of-range).
0x0004	Invalid state	The responder is in a state where the function cannot be accepted.
0x0005	Out of resources	There are no resources left to handle this request at the moment.

6.3 Anybus CompactCom Messages

The messages below can be sent from the CompactCom device to the safety module at any time during normal operation.

6.3.1 GetStatus

This message reads the current status from the safety module.

Request

Byte	Name	Value
1	ID	-
2-3	Request	0x4000
4	Length	0x00

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0000
4	Length	0x0C
5–16	Status Data	Bytes: 5–6: Error counter, DR 7–8: Error counter, SE 9–10: Event information, event 1 (newest event) 11–12: Event information, event 2 13–14: Event information, event 3 15–16: Event information. event 4 (oldest event) For information about the error counters DR and SE, see the Functional Safety Module Object (11h) chapter in the Anybus CompactCom 40 Software Design Guide. Every safety module can have its own representation of the error counters.

6.3.2 SetConfigString

With this message, it is possible to configure the safety module’s safety parameter sets. This message is safety protocol specific.

The intention with this configuration string is to provide an alternate method of providing setup parameters for the configuration of the safe application. This string is normally not something the end-user is able to change, but is more a method of letting the manufacturer hard-code certain settings. Normally these parameters are transferred from the safety PLC (where the end-user can configure these parameters) to the safety module during establishment of the safe connection.

The host application can use this functionality to transfer application specific, safety-related configuration data from the non-safe host to the safety module. Setup and coding of the data is completely (safe) application specific.

Request

Byte	Name	Value
1	ID	-
2-3	Request	0x4001
4	Length	n
5–(5+n)	Config string	Byte string of n bytes. Length and content evaluated by the safety module.

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0001
4	Length	0x00

6.3.3 ErrorConfirmation

This message is used to initiate an error confirmation from the host application.

When received by the safety module, the safety module shall try to reset any errors unless prohibited by safety constraints or prohibited by the local configuration of the safety module.

For more information, see the Functional Safety Module Object (11h) in the Anybus CompactCom 40 Software Design Guide.

Request

Byte	Name	Value
1	ID	-
2-3	Request	0x4002
4	Length	0x00

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0002
4	Length	0x00

6.4 Safety Module Messages

The messages below can be sent from the safety module to the CompactCom device at any time during normal operation.

6.4.1 FatalErrorEntry

This message is sent to the CompactCom device when the safety module encounters a fatal error. Since the safety module might be in a non-communicating state after sending this message, the CompactCom device may omit the response.

Request

Byte	Name	Value
1	ID	-
2-3	Request	0x4000
4	Length	0x02
5–6	Event information	Error type for the event causing the fatal error entry.

Response (Optional)

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0000
4	Length	0x00

7 PROFIsafe Specific Details

7.1 CompactCom Messages

7.1.1 SetFAddress

With this request, the desired F-address for the safety module is set. This request is always sent during start-up phase. The originator of the F-address is the host application (for example, a switch, or a keypad parameter), using the CompactCom Network Configuration Object (04h), Instance #21. The F-address is an alias address used during the connection phase. The value of this parameter must match what is set by the end user in the configuration tool (e.g. Step7), otherwise the PROFIsafe communication cannot be established.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4080
4	Length	0x02
5-6	FAddress	F-address to be used. Valid range: 1 – 65534

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0080
4	Length	0x00

7.1.2 WriteAcyclicRecord

When an acyclic request to write data to the safety module is received by the CompactCom from the IO controller, this request is sent to the safety module.

This request can, for example, be used to set the F-parameters (F_Destination_Add, F_Source_ Add, F_Wd_Time, F_SIL_Level, F_CRC1).

Please note that the “Data” part of network specific telegrams may be coded in a different endianness.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4081
4	Length	Depending on passed data length (2 + passed data length)
5-6	Index	The index to write (will signify the kind of data that is passed)
7-(7+n)	Data	Data of n bytes. Length and content evaluated by the safety module

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0081
4	Length	0x00

7.1.3 ReadAcyclicRecord

When an acyclic request to read data from the safety module is received by the CompactCom from the IO controller, this request is sent to the safety module.

Please note that the “Data” part of network specific telegrams may be coded in a different endianness.

This command is optional to implement.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4082
4	Length	6
5-6	Index	The index to read (will signify the kind of data that is passed)
7-10	Max data length	The max amount of data which the IO Controller will read (that is, the number of bytes returned can be equal than or less to this value)

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0082
4	Length	n
5-(5+n)	Data	Data of n bytes

7.1.4 GetSupportedSpdus

This request is sent to the safety module to receive a list of information of the supported SPDUs.

At least one SPDU must be supported by the safety module. If the safety module supports only one SPDU, the command is optional to implement. If multiple SPDUs are supported, the command is mandatory to implement.

If the GetSupportedSpdus command is not supported, or the response is empty, the SPDU ID will be 0 and the I/O lengths will be set according to the Start-up telegram. See *The Start-up Telegram, p. 12*.

The SPDUs are identified by 32–bit unique SubModuleIdentNumbers. When multiple SPDU configurations are supported, the active configuration is selected by the controller through the SubModuleIdentNumber of the expected identification. The SPDU IDs are represented in bits 24–31 of each SubModuleIdentNumber defined for the safety module.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4083
4	Length	0x00

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0083
4	Length	Length of Data, 3*n (n – number of supported SPDUs, n>=1)
5-(4+3*n)	Data	See the *Response Data, p. 23* table

Response Data

Byte	Name	Value
1	SPDU #1 ID	Identifies SPDU #1
2	SPDU #1 input length	Input SPDU length of SPDU #1
3	SPDU #1 output length	Output SPDU length of SPDU #1
...	...	...
3*(n-1)+1	SPDU #n ID	Identifies SPDU #n
3*(n-1)+2	SPDU #n input length	Input SPDU length of SPDU #n
3*(n-1)+3	SPDU #n output length	Output SPDU length of SPDU #n

7.2 Safety Module Messages

7.2.1 SendChannelDiagAlarm

This request is sent to the CompactCom when there is a need to transmit an alarm message to the connected IO controller.

If the max number of active alarms for the CompactCom is reached, the CompactCom will reject the message with error code “Out-of-resources”.

The maximum number of active alarms the CompactCom supports, is specified in the Network Guide.

This command is optional to implement.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4081
4	Length	6
5-6	Channel Number	0...0x7FFF: Channel number 0x8000: Refers to the submodule itself, not a specific channel
7	Channel Properties	0: Other 1: 1-bit 2: 2-bit 3: 4-bit 4: Byte 5: Word (2 bytes) 6: Double Word (4 bytes) 7: Long Word (8 bytes)
8	Channel Direction	0: Manufacturer specific 1: Input (data to I/O controller) 2: Output (data from I/O controller) 3: Input/output (data to/from I/O controller)
9-10	Channel Error	Safety layer diagnosis messages (sent transparently to the I/O controller) Predefined values, to be implemented in the safety module, can be found in the PROFIsafe specification.

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0081
4	Length	1
5	Alarm ID	This ID is used when the alarm is removed

7.2.2 RemoveAlarm

This request is sent to the CompactCom when an alarm shall be removed.

If the SendChannelDiagAlarm command is implemented in the safety module, this command is mandatory to implement.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4082
4	Length	0x01
5	Alarm ID	Reference to the Alarm which shall be removed

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0082
4	Length	0x00

8 FSoE Specific Details

8.1 CompactCom Messages

8.1.1 SetFsoeSlaveAddress

With this request, the FSoE slave address for the safety module is set. This request is always sent during the start-up phase. The origin of this address is the host application, using the CompactCom Network Configuration Object (04h), Instance #21. The FSoE slave address is an alias address used during the connection phase. The value of this parameter must match what is set by the end user in the configuration tool, otherwise the FSoE communication cannot be established.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4080
4	Length	0x02
5-6	FsoeSlaveAd-dress	FSoE slave address to be used. Valid range: 1 – 65535

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0080
4	Length	0x00

8.1.2 GetCoeObjects

This request is sent from the CompactCom to receive a list of all objects implemented inside the safety module. The information returned by this command will be used to implement the EtherCAT service “Get OD list”.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4083
4	Length	0x02
5-6	ListType	Requested List type: 0x00: List of list-length shall be delivered 0x01: ALL (all objects shall be delivered) 0x02: RX (objects mappable in a RxPDO shall be delivered) 0x03: TX (objects mappable in a TxPDO shall be delivered) 0x04: BU (objects for backup / device replacement shall be delivered) 0x05: ST (objects used as startup parameter shall be delivered)

Response (ListType 0x00)

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0083
4-5	Length	0x000A
6-7	Length ALL	Byte length of list for ListType 0x01 (ALL) (number of object IDs*2)
8-9	Length RX	Byte length of list for ListType 0x02 (RX) (number of object IDs*2)
10-11	Length TX	Byte length of list for ListType 0x03 (TX) (number of object IDs*2)
12-13	Length BU	Byte length of list for ListType 0x04 (BU) (number of object IDs*2)
14-15	Length ST	Byte length of list for ListType 0x05 (ST) (number of object IDs*2)

Response (ListType 0x01 - 0x05)

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0083
4-5	Length	Depending on the number of object IDs (2 * n)
6-7	Object ID 1	Object Index 1
8-9	Object ID 2	Object Index 2
...	...	...
(2*n+4) – (2*n+5)	Object ID n	Object Index n

The size of the response will typically exceed the maximum size allowed for the message. This means that the response will be sent in more than one fragment.

The object indices added in this list must be entered in ascending order. If they are not in ascending order, they will not be visible on the EtherCAT network.

8.1.3 GetPdoMapping

This request is sent from the CompactCom to receive the PDO mapping information for RxPDO and TxPDO. The information returned by this command will be used to implement the PDO mapping objects (e.g. 0x1600 and 0x1A00).

The safety module mapping is for information only. The mapping itself is already done on the safety module. This means that the SPDU sent from the safety module to the CompactCom contains the mapped PDO to be copied 1:1 to the network.

Dynamic mapping of the safety module is not supported. The PDO mapping objects are read-only.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4084
4	Length	0x01
5	Direction	0x00: RxPDO mapping 0x01: TxPDO mapping

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0084
4	Length	Depending on the number of mapped objects (2 + (4 * n))
5-6	ObjectIndex	Object index of the PDO mapping object (e.g. 0x1600 for Rx and 0x1A00 for Tx)
7-10	MappedObject 1	Mapped object #1. Corresponds to subindex 1 E.g. 0x70010120 to map object 0x7001, subindex 0x01 with size 0x20 (32) bits
11-14	MappedObject 2	Mapped object #2. Corresponds to subindex 2
(4*n+3) - (4*n+6)	MappedObject n	Mapped object #n. Corresponds to subindex n

8.1.4 GetObjectInfo

This request is sent from the CompactCom to receive detailed information about a single object. The information returned by this command will be used to implement the EtherCAT service “Get object description”.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4085
4	Length	0x02
5-6	Index	The index of the object which info is to be read

Response (Success)

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0085
4	Length	Depending on name length (8 + name length)
5-8	Status	0 - No error
9-10	DataType	Data type of this object
11	MaxSubIn-dex	Greatest subindex number of this object
12	ObjectCode	Object code of this object 0x07: VAR 0x08: ARRAY 0x09: RECORD
13-n	Name	Name of this object (character string without null terminator)

Response (Error)

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0085
4	Length	4
5-8	Status	SDO Abort Code

8.1.5 GetEntryInfo

This request is sent from the CompactCom to receive detailed information about a single entry. The information returned by this command will be used to implement the EtherCAT service "Get entry description".

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4086
4	Length	0x03
5-6	Index	The index of the object which entry info is to be read
7	Subindex	The subindex of the object which entry info is to be read

Response (Success)

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0086
4	Length	Depending on name length (10 + name length)
5-8	Status	0 - No error
9-10	DataType	Data type of this entry
11-12	BitLength	Length in bits of the object entry value
13-14	Object Access	Access rights of this entry: Bit 0: read access in Pre-Operational state Bit 1: read access in Safe-Operational state Bit 2: read access in Operational state Bit 3: write access in Pre-Operational state Bit 4: write access in Safe-Operational state Bit 5: write access in Operational state Bit 6: object is mappable in RxPDO Bit 7: object is mappable in TxPDO Bit 8: object can be used for backup Bit 9: object can be used for settings Bit10: object is mappable in SafeInputs Bit11: object is mappable in SafeOutputs Bit12: object is mappable in Safety Parameter Set, writeable only via safe configuration Bit 13-15: reserved
15-n	Name	Name of this entry (character string without null terminator) Note: The entry name for subindex 0 of an ARRAY object or a RECORD object is fixed and assigned by the CompactCom ("Number of entries"). The transferred name is not used here

Response (Error)

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0086
4	Length	4
5-8	Status	SDO Abort Code

8.1.6 SdoDownload

This request is sent to the safety module when an acyclic SDO write access request to write data to the safety module from the EtherCAT master is received.

Since the CompactCom does not do any access right validation, this is the task of the safety module when the service is requested.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4081
4-5	Length	4+n
6-7	Index	The index to write (will signify what kind of data that is passed)
8	Subindex	The subindex to write (will signify what kind of data that is passed)
9	Flags	Bit 0: • 1 = Object shall be written completely • 0 = Index with subindex is written Bit 1-7: Reserved
10-(9+n)	Data	Data of n bytes Length and content is evaluated by the safety module If the value n (data length) is 0 (zero), data does not exist

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0081
4	Length	4
5-8	Status	0 - No error Any other value represents an SDO Abort Code

8.1.7 SdoUpload

This request is sent to the safety module when an acyclic SDO request to read data from the safety module from the EtherCAT master is received.

Since the CompactCom does not do any access right validation, this is task of the safety module when the service is requested.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4082
4	Length	0x06
5-6	Index	The index to read (will signify what kind of data that is passed)
7	Subindex	The subindex to read (will signify what kind of data that is passed)
8	Flags	Bit 0: • 1 = Object shall be read completely • 0 = Index with subindex is read Bit 1-7: Reserved
9-10	Request length	The maximum amount of data which will be read (the number of bytes returned can be equal or less to this value)

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0082
4-5	Length	4+n
6-9	Status	0 - No error Any other value represents an SDO Abort Code
10-(9+n)	Data	Data of n bytes Data is only sent if the Status field equals 0 (No error) If the value n (data length) is 0 (zero), data does not exist

8.2 Safety Module Messages

8.2.1 LEDstate

The safety module can provide state information that can be used to implement a FSoE status LED. The state is transmitted on change.

Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4080
4	Length	0x01
5	Status LED	0x00: Initializing 0x01: Ready/Start-Up 0x02: Normal operation 0x03: Failsafe Data 0x04: Unspecified FSoE connection error 0x05: Communication parameter error 0x06: Application parameter error 0x07: Wrong Safety Address 0x08: Wrong Command 0x09: Watchdog error 0x0A: CRC error 0xFE: Reserved (for CompactCom usage e.g. to host) 0xFF: Reserved (for CompactCom usage e.g. to host)

Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0080
4	Length	0x00

9 CIP Safety Specific Details

The data exchange between the CIP Safety Stack (CSS) and the non-safe CIP stack (CSAL (CIP Safety Adaptation Layer)) is handled by Hardware Abstraction Layer — Communication (HALC) messages.

For a complete HALC command reference, see the CIP Safety on EtherNet/IP, Generic Porting Guide from HMS / IXXAT.

9.1 HalcCsalMessage

All HALC messages that are sent from the non-safe part, running on the CompactCom, to the safe part, running on the safety module, should be transmitted using HalcCsalMessages.

The only exception to this rule are messages with the HALC command identifier CSOS_k_CMD_IXCO_IO_DATA. They contain cyclical safety data and will be transferred using the SPDU (Safety Process Data Unit) part of the message.

9.2 HalcCsalMessage Structure

The length field of the HalcCsalMessage is 2 bytes instead of 1 byte, because HALC messages can be larger than 255 bytes.

For more information, see *Message Header, p. 17*.

9.2.1 Request

HALCS_t_MSG is a structure used in the IXXAT CIP Safety stack. For other CIP Safety stacks, an adaptation layer implementation might be necessary.

Byte	Name	Value / Description	HALCS_t_MSG
1	ID	-	-
2-3	Request	0x4080	-
4–5	Length	x (length of HALC command and HALC addInfo + n (data length))	u16_len
6–7	HALC command	HALC command identifier	u16_cmd
8–11	HALC addInfo	Additional information field of the HALC message	u32_addInfo
12–(11+n)	Data	Data field of the HALC message If the value n (data length) is 0 (zero), data does not exist.	pb_data: Pointer to data

9.2.2 Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0080
4	Length	0x00

9.3 SetInitData

This message is sent from the CompactCom device with the purpose of initializing the safety module. It contains data CSS needs for start-up. Safe communication is only possible after the safety module has received this command.

9.3.1 Request

Byte	Name	Value
1	ID	-
2-3	Request	0x4081
4	Length	18
5–8	Serial number	This identifies the product on the network. It shall be a unique number per unit, together with the vendor ID. The CompactCom will provide the serial number.
9–12	Node ID	The IP address of the CompactCom device.
13–14	Vendor ID	User defined. The CompactCom will provide the vendor ID.
15–16	Device type	Device type (product type). The CompactCom will provide the device type.
17–18	Product code	User defined. The CompactCom will provide the product code.
19–20	Incarnation ID	Incarnation ID of EtherNet/IP.
21	Major revision	User defined. The CompactCom will provide the major revision value.
22	Minor revision	User defined. The CompactCom will provide the minor revision value.

9.3.2 Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0081
4	Length	0x00

9.4 GetClassIDs

This message is sent from the CompactCom device to the safety module, with the purpose of receiving a list of all objects implemented in the safety module.

This information is needed for the CompactCom to be able to route explicit requests to these objects to the safety module.

It is mandatory to list the Safety Supervisor Object and the Safety Validator Object. More objects can be added as needed.

! The Assembly Class (ID 4) shall not be included in this list. Requests to Assembly Instances according to the GetAssemblyInstIDs response will be routed to the safety module automatically.

9.4.1 Request

Byte	Name	Value
1	ID	-
2-3	Request	0x4082
4	Length	0x00

9.4.2 Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0082
4	Length	n (length of the following part in bytes) The CompactCom can handle up to 20 registered classes.
5–6	Class ID 1	-
7–8	Class ID 2	-
...		
(n+3)-(n+4)	Class ID m	-

9.5 GetAssemblyInstIDs

This message is sent by the CompactCom device, with the purpose of receiving a list of all assembly instances inside the safety module.

This information is needed for the CompactCom to be able to route explicit requests to these objects to the safety module.

In the case that the safety module does not support Assembly Instance IDs, it shall respond to the request with a zero length response.

9.5.1 Request

Byte	Name	Value
1	ID	-
2-3	Request	0x4084
4	Length	0x00

9.5.2 Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0084
4	Length	n (length of the following part in bytes) The CompactCom can handle up to 20 registered assembly instances.
5–6	Assembly Instance ID 1	-
7–8	Assembly Instance ID 2	-
...		
(n+3)-(n+4)	Assembly Instance ID m	-

9.6 LinkStatus

This message is sent by the CompactCom device to report the link status. Using this command, the Link Object (TCP/IP object) can report to the safety module that the underlying network status has changed.

9.6.1 Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4083
4	Length	0x01
5	Link Status	0: link is down / not available 1: link is up / available

9.6.2 Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0083
4	Length	0x00

9.7 HalcCssMessage

All HALC messages that are sent from the safe part, running on the safety module, to the non-safe part, running on the CompactCom device, should be transmitted using HalcCssMessages.

The only exception to this rule are messages with the HALC command identifier CSOS_k_CMD_IXCO_IO_DATA. They contain cyclical safety data and will be transferred using the SPDU part of the message.

9.8 HalcCssMessage Structure

! The length field of the HalcCssMessage is 2 bytes instead of 1 byte, because HALC messages can be larger than 255 bytes.

9.8.1 Request

HALCS_t_MSG is a structure used in the IXXAT CIP Safety stack. For other CIP Safety stacks, an adaptation layer implementation might be necessary.

Byte	Name	Value / Description	HALCS_t_MSG
1	ID	-	-
2-3	Request	0x4080	-
4–5	Length	x (length of HALC command and HALC addInfo + n (data length))	u16_len
6–7	HALC command	HALC command identifier	u16_cmd
8–11	HALC addInfo	Additional information field of the HALC message	u32_addInfo
12–(11+n)	Data	Data field of the HALC message If the value n (data length) is 0 (zero), data does not exist.	pb_data: Pointer to data

9.8.2 Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0080
4	Length	0x00

9.9 LEDstate

CSS generates the states for the Module and Network Status LEDs. The LED states are transmitted on change, from the safety module to the CompactCom device.

9.9.1 Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4081
4	Length	2
5	Module Status LED	0: Off 1: Solid green 2: Flashing green 3: Flashing red 4: Solid red 5: Flashing red and green
6	Network Status LED	0: Off 1: Solid green 2: Flashing green 3: Flashing red 4: Solid red 5: Flashing red and green

9.9.2 Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0081
4	Length	0x00

9.10 SafetyReset

This message should be sent from the safety module after a ‘Safety Reset Service’ has been acknowledged by the CSS.

The service shall “Emulate as closely as possible cycling power on the device”, and this is achieved by sending the ‘SafetyReset’ message to the CompactCom device which forwards the request to the non-safe host.

The CompactCom device will delay the reset command to the non-safe host for 1000 ms, to ensure that the ‘Safety Reset Service’ response is sent out on the network before the reset is executed.

9.10.1 Request

Byte	Name	Value / Description
1	ID	-
2-3	Request	0x4082
4	Length	2
5	Safety reset type	Safety supervisor safety reset types supported: 0x00: Emulate as closely as possible cycling power on the device. 0x01: Reset values to default configuration, and emulate as closely as possible cycling power on the device. 0x02: Reset values to default configuration, except the parameters specified in the attribute bit map, and emulate as closely as possible cycling power on the device.
6	Attribute bitmap	For safety reset type 0 and 1 this field is not used and is set to 0. For safety reset type 2 this field is set to the corresponding attribute bitmap parameter which is received in the safety reset service. The assignments are defined as: **Bit 0** When set, preserve soft-set MAC ID. Processed by the CompactCom. **Bit 1** When set, preserve soft-set baud rate. Processed by the CompactCom. **Bit 2** When set, preserve the TUNID. **Bit 3** When set, preserve the password. **Bit 4** When set, preserve the CFUNID. **Bit 5** When set, preserve the OCPUNID. **Bit 6** Reserved, always 0. **Bit 7** Use extended map.

9.10.2 Response

Byte	Name	Value
1	ID	Mirrored from request
2-3	Response	0x0082
4	Length	0x00

9.11 SPDU

SPDU telegrams are transmitted cyclically, independently of data change, in both directions. Whenever the SPDU data is updated, a data update indicator is incremented. Only updated data shall be provided to the safety protocol stacks.

For more information about the SPDU, see *The Anybus Telegram Structure, p. 13* and *The Safety Module Telegram Structure, p. 15*.

The length of the safe IO data is decided during start-up. See *The Start-up Telegram, p. 12* for more information.

9.11.1 The CompactCom SPDU

The CompactCom SPDU is sent from the CompactCom device to the safety module.

Normally all HALC messages contain a HALC command, but in this case it is implicitly known (CSOS_k_CMD_IXCO_IO_DATA in IXXAT CSS), and will not be transmitted.

The CIP Safety message is encapsulated inside the HALC message.

A time coordination message will also be sent, and it is placed at the end of the SPDU. The time coordination message needs a separate HALC addInfo, and this is also placed inside the SPDU. See the table below.

When a connection is closed, the HALC length field shall be set to the maximally supported CompactCom SPDU size, and HALC addInfo/HALC addInfo2 shall be set to 0xFFFF.

Anytime a connection does not provide valid data (e.g. when a connection is not open or haven’t produced any data yet), the data/data2 areas shall be filled with 0xFE values.

When a connection transports less data than the SPDU container maximum, there will be unused bytes right after the data field. The position of HALC addInfo2 shall stay at its constant position within the container.

If the safety module only produces safe data, the HALC length shall be set to zero and HALC addInfo to 0xFFFF. Consequently, the data field shall not be transmitted (i.e. consumer number and pad byte are also not transmitted).

If the safety module only consumes data, the HALC addInfo2 and data2 fields shall contain values as if the connection is closed.

Byte	Name	Value / Description	HALC_t_MSG
FOR IO SAFETY DATA CONSUMER			
		Not transmitted (fixed value): CSOS_k_CMD_IXCO_IO_DATA	u16_cmd
1–2	HALC length	n (length of data)	u16_len
3–4	HALC addInfo	Safety validator instance ID for safety IO data: 2 bytes	u16_addInfo
5–(4+n)	data	Byte 1: consumer number Byte 2: pad byte Byte 3–n: data message	Pb_data: pointer to data
FOR IO SAFETY DATA PRODUCER			
		Not transmitted (fixed value): CSOS_k_CMD_IXCO_IO_DATA	u16_cmd
		Not transmitted (fixed value): length of data2: 8 bytes	u16_len
(5+n)-(6+n)	HALC addInfo2	Safety validator instance ID for time coordination message: 2 bytes	u16_addInfo
(7+n)-(14+n)	data2	Byte 1: consumer number Byte 2: pad byte Byte 3–8: time coordination message	Pb_data: pointer to data2
DATA UPDATE INDICATOR FOR SAFETY DATA CONSUMER AND SAFETY DATA PRODUCER			
(15+n)	DUI_data	Data update indicator for data Incremented when the data field is updated After reaching 0xFF, this value will start again at 0x00	-
(16+n)	DUI_data2	Data update indicator for data2 Incremented when the data field is updated After reaching 0xFF, this value will start again at 0x00	-

9.11.2 The Safety Module SPDU

The safety module SPDU is sent from the safety module to the CompactCom device.

Normally all HALC messages contain a HALC command, but in this case it is implicitly known (CSOS_k_CMD_IXSVO_IO_DATA in IXXAT CSS), and will not be transmitted.

The CIP Safety message is encapsulated inside the HALC message.

A time coordination message will also be sent, and it is placed at the end of the SPDU. The time coordination message needs a separate HALC addInfo, and this is also placed inside the SPDU. See the table below.

When a connection is closed, the HALC length field shall be set to the maximally supported safety module SPDU size, and HALC addInfo/HALC addInfo2 shall be set to 0xFFFF.

Anytime a connection does not provide valid data (e.g. when a connection is not open or haven't produced any data yet), the data/data2 areas shall be filled with 0xFE values.

When a connection transports less data than the SPDU container maximum, there will be unused bytes right after the data field. The position of HALC addInfo2 shall stay at its constant position within the container.

If the safety module only consumes safe data, the HALC length shall be set to zero and HALC addInfo to 0xFFFF. Consequently, the data field shall not be transmitted.

If the safety module only produces data, the HALC addInfo2 and data2 fields shall contain values as if the connection is closed.

Byte	Name	Value / Description	HALCS_t_MSG
FOR IO SAFETY DATA PRODUCER			
		Not transmitted (fixed value): CSOS_k_CMD_IXSVO_IO_DATA	u16_cmd
1–2	HALC length	n (length of data)	u16_len
3–4	HALC addInfo	Safety validator instance ID for safety IO data: 2 bytes	u16_addInfo
5–(4+n)	data	Byte 1–n: data message	pb_data: pointer to data
FOR IO SAFETY DATA CONSUMER			
		Not transmitted (fixed value): CSOS_k_CMD_IXSVO_IO_DATA	u16_cmd
		Not transmitted (fixed value): length of data2: 6 bytes	u16_len
(5+n)-(6+n)	HALC addInfo2	Safety validator instance ID for time coordination message: 2 bytes	u16_addInfo
(7+n)-(12+n)	data2	Byte 1–6: time coordination message	pb_data: pointer to data2
DATA UPDATE INDICATOR FOR SAFETY DATA CONSUMER AND SAFETY DATA PRODUCER			
(13+n)	DUI_data	Data update indicator for data Incremented when the data field is updated After reaching 0xFF, this value will start again at 0x00	-
(14+n)	DUI_data2	Data update indicator for data2 Incremented when the data field is updated After reaching 0xFF, this value will start again at 0x00	-

9.12 SPDU Calculations and Examples

The lengths of the input and output SPDUs depend on the safe I/O data lengths to be transported. There are devices that offer several assemblies with different lengths. Such devices report the size of the longest assemblies in the startup telegram. During runtime the actual length in the SPDUs is adjusted to the data length actually used by the corresponding connection. This is considered as dynamic SPDUs.

The following examples are with “data” equalling 2 bytes and 4 bytes (in reality, at least 7 bytes), and “data2” equalling 4 bytes (in reality 8 bytes, but reduced due to clarity).

9.12.1 Dynamic CompactCom SPDU

CONNECTION NOT OPENED, INVALID DATA

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Name	HALC length		HALC addInfo		data				HALC addInfo2		data2				DUI_ data	DUI_ data2
Value	4	0	0xff	0xff	0xfe	0xfe	0xfe	0xfe	0xff	0xff	0xfe	0xfe	0xfe	0xfe	xx	xx

CONNECTION NOT OPENED, VALID DATA

Not possible

CONNECTION OPENED, INVALID DATA

4 bytes data (maximum SPDU size)

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Name	HALC length		HALC addInfo		data				HALC addInfo2		data2				DUI_ data	DUI_ data2
Value	4	0	1	0	0xfe	0xfe	0xfe	0xfe	2	0	0xfe	0xfe	0xfe	0xfe	xx	xx

2 bytes data

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Name	HALC length		HALC addInfo		data		Unused		HALC addInfo2		data2				DUI_ data	DUI_ data2
Value	2	0	1	0	0xfe	0xfe	0	0	2	0	0xfe	0xfe	0xfe	0xfe	xx	xx

The unused bytes may have any value, but for debugging purposes always using the same value (e.g. 0x00 or 0xFF) would be useful.

CONNECTION OPENED, VALID DATA

Same as *CONNECTION OPENED, INVALID DATA, p. 46*. Instead of 0xFE, real values are used.

9.12.2 Producer-Only CompactCom SPDU

When a device only produces data then the HALC length in the CompactCom SPDU is zero and the data field is not present.

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Name	HALC length		HALC addInfo		HALC addInfo2		data2				DUI_ data	DUI_ data2
Value	0	0	0xff	0xff	1	0	0xfe	0xfe	0xfe	0xfe	xx	xx

This analogously applies to a Consumer-Only safety module SPDU (not shown here).

9.12.3 SPDU Length Relations

The lengths of the input and output SPDUs depend on the safe I/O data lengths to be transported. It shall be considered that there are devices that have only inputs, only outputs or both inputs and outputs. Additionally, CIP Safety uses different frame formats depending on the length of the safe payload data. Thus the SPDU lengths are determined by the safety module and communicated to the CompactCom via the Startup telegram.

Example 1

The Startup telegram reports output SPDU length 26 bytes and input SPDU length 22 bytes. Conclusively, the device has 2 bytes of safe outputs and 2 bytes of safe inputs.

The CompactCom SPDU contents:

HALC length	2 bytes
HALC addInfo	2 bytes
data: consumer number Pad Byte Safe Payload CIP Safety Overhead of 1-2 bytes format	1 byte 1 byte 2 bytes 6 bytes
HALC addInfo2	2 bytes
data 2: consumer number Pad byte Time Coordination Message	1 byte 1 byte 6 byte
DUI_data DUI_data2	1 byte 1 byte
sum	**26 bytes**

The safety module SPDU contents:

HALC length	2 bytes
HALC addInfo	2 bytes
data: Safe Payload CIP Safety Overhead of 1-2 bytes format	2 bytes 6 bytes
HALC addInfo2	2 bytes
data 2: Time Coordination Message	6 byte
DUI_data DUI_data2	1 byte 1 byte
sum	**22 bytes**

Example 2

The Startup telegram reports output SPDU length 36 bytes and input SPDU length 14 bytes. Conclusively, the device has 5 bytes of safe outputs and no safe inputs.

The CompactCom SPDU contents:

HALC length	2 bytes
HALC addInfo	2 bytes
data: consumer number Pad Byte Safe Payload CIP Safety Overhead of 3-250 bytes format	1 byte 1 byte 5 bytes (=N) 13 bytes (N+8)
HALC addInfo2	2 bytes
data 2: consumer number Pad byte Time Coordination Message	1 byte 1 byte 6 byte
DUI_data DUI_data2	1 byte 1 byte
sum	**36 bytes**

The safety module SPDU contents:

HALC length	2 bytes
HALC addInfo	2 bytes
data: Safe Payload CIP Safety Overhead	0 bytes 0 bytes
HALC addInfo2	2 bytes
data 2: Time Coordination Message	6 byte
DUI_data DUI_data2	1 byte 1 byte
sum	**14 bytes**

SCM-1202-024 1.4 en-US / 2020-12-21 / 21169