Software Manual

CIP Safety on EtherNet/IP

Generic Porting Guide

HMS Technology Center Ravensburg GmbH
Helmut-Vetter-Straße 2
88213 Ravensburg
Germany

Tel.: +49 751 56146-0
Fax: +49 751 56146-29
Internet: www.hms-networks.de
E-Mail: info-ravensburg@hms-networks.de

Support

In case of unsolvable problems with this product or other HMS products please contact HMS in written form:

Fax: +49 751 56146-29
E-Mail: support@ixxat.de

Further international support contacts can be found on our webpage www.hms-networks.de

Copyright

Registered trademarks

Document number: 4.02.0501.20002
Version: 1.3

1 Introduction

1.1 Introduction

This manual describes the porting of the CIP (Common Industrial Protocol) Safety Software (CSS) to EtherNet/IP devices.

It is assumed that the reader is familiar with the CIP [1], EtherNet/IP [2] and CIP Safety [3] specifications. This manual is an add-on to the CSS manual [4] and describes the characteristics for using CSS in combination with EtherNet/IP. This is a generic description so no assumption on a specific EtherNet/IP implementation is made.

If the EtherNet/IP Adapter Developers Kit (EADK) from Pyramid Solutions is used then [6] describes further specific porting issues as an addition to this document.

1.1.1 Terms, definitions, acronyms

The general CIP Safety definitions and acronyms used, as well as information about syntax and other generally applicable conventions and agreements, are listed in Chapter 1 of the CIP Safety Manual Part 1 [4]. Similarly, there is an introduction to CIP Safety there, so you are urgently recommended to read [4] first.

1.1.2 References

[1] The CIP Networks Library Volume 1 Common Industrial Protocol (CIP), ODVA Inc.

[2] The CIP Networks Library Volume 2: EtherNet/IP Adaptation of CIP, ODVA Inc.

[3] The CIP Networks Library Volume 5: CIP Safety, ODVA Inc.

[4] Manual CIP Safety Part 1: Generic protocol software and target 4.02.0500.20000_Manual_CSS_EN.pdf

[5] Manual CIP Safety Part 2: Originator 4.02.0500.20100_Manual_CSS_Orig_EN.pdf

[6] Porting Guide for EADK based Systems 4.02.0501.20003_Porting Guide_EIP_EN_Target_EADK.pdf

1.1.3 History

Date	version	Change	Editor
15.06.15	1.0	Document released	A. Kramer
30.07.15	1.1	Added chapter 4.2.2 (Changing the IP address) Description of the commands extended (explaining the data field and the other HALC struct members in more detail) Added information to CSOS_k_CMD_IXSCE_SCLOSE_RES that CSAL needs to close the connection Extended description of command CSOS_k_CMD_IXSSC_TERMINATE	A. Kramer
07.10.15	1.2	Clarified in chapter 2 which parts of the system are safety relevant.	A. Kramer
07.06.16	1.3	Added hint that EtherNet/IP stack shall not do timeout checking for safety connections Added hint that when CSAL receives commands CSOS_k_CMD_IXSCE_SCLOSE_RES or CSOS_k_CMD_IXSCE_CNXN_CLOSE the connection may already be closed (then ignore these commands) Removed command CSOS_k_CMD_IXLOS_LINK_FAILURE Corrected length information for CSOS_k_CMD_IXSCE_SOPEN_RES command Replaced IXXAT by HMS on title page, copyright, headers and footers	A. Kramer

1.2 Installation of the CIP Safety on EtherNet/IP Software

Instructions on how to install the CIP Safety Software and a guide for the first steps with the provided demo programs can be found in [4].

For the demo the CSAL for Sercos is provided as a library (instead of source code). In deviation from the description in [4] the library and the corresponding interface headers can be found in the sub-directory src\Demo\CSOS\Target\CSAL_lib.

2 Software Concept

As can be seen from Figure 2-1 and the CIP Safety software manual [4], CIP Safety on EtherNet/P consists of several software parts. The CIP Safety Stack running on the safe controllers Controller 1 and Controller 2 and the non-safe software running on Controller 3.

The main task of the CSS on a CIP Safety device is to create and handle CIP Safety messages with cyclical process data as specified in [3], and to determine communications errors according to correction measures [3] (see Section 2-1.2).

All software on Controller 1 and Controller 2 is safety relevant. There is pre-certification for CSS. [4] lists the requirements for integration of CSS. The system integrator is responsible for certification of the other software parts (Safety Application, HALCS) as well as for the complete system.

The Software on Controller 3 consists of an EtherNet/IP stack and the corresponding lower layers (e.g. TCP/IP stack). Most likely an additional software layer is needed that adapts the interface of the EtherNet/IP stack to the HALC interface to CSS. Further on we will call this CIP Safety Adaptation Layer (CSAL). In case the EtherNet/IP stack doesn't provide all the functionality necessary for running CIP Safety over EtherNet/IP modifications inside the EtherNet/IP stack may be required.

All software on Controller 3 including this manual is not safety relevant.

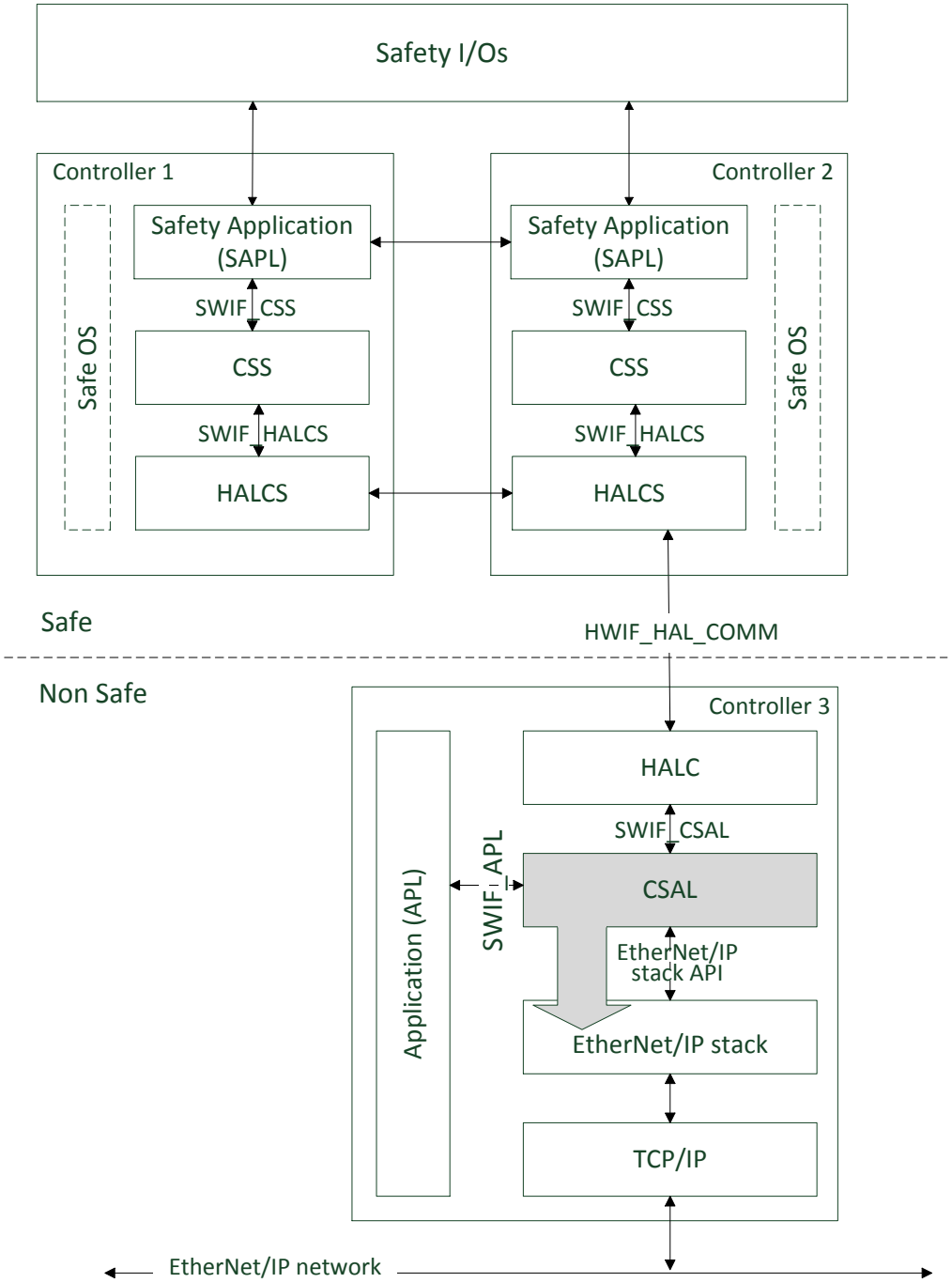

Figure 2-1: 3-Controller system architecture

Alternatively it is possible to combine the functionality of Controller 2 and Controller 3 into one Controller (see [4]). For ease of description this document continues to describe the 3 Controller architecture as shown in Figure 2-1.

If non-safe software is run on the same controller as the safe software, the user must take suitable measures to ensure that the non-safe software cannot have negative effects on the safe software.

2.1 Hardware-Interfaces

As can be seen in Figure 2-1, Controller 3 is connected to Controller 2 via the interface HWIF_HAL_COMM (in a two-controller architecture, this can simply be a purely software interface).

These interfaces are operated by the HALC unit of each microcontroller system. The internal structure of the HALC and thus also the communication between the controllers is not part of this manual. Only the software interface to the HALC defined in Section 2.2.2 is considered below.

2.2 Software interfaces

In addition to the hardware interfaces, the following interfaces between the individual software parts also exist within Controller 3.

2.2.1 SWIF_APL

This interface is used to communicate between the (unsafe) application and the CIP Safety Adaptation Layer.

This interface is not subject to this document and thus not further described.

2.2.2 SWIF_CSAL

Interface between HALC and the CIP Safety Adaptation Layer. All data exchanged between CSAL and CSS flow across this interface.

Since the data formats for Controller 3 and Controller 1 or 2 may differ, the data on this interface is always exchanged in Little Endian format.

The API of this interface is described in Section 4.

2.2.3 Other software interfaces

Other software interfaces (e.g. Socket API between TCP/IP stack and EtherNet/IP stack) are not subject to this document.

3 Specifics of CIP Safety on EtherNet/IP

This section lists some specific aspects that must be considered when implementing a CIP Safety on EtherNet/IP device in contrast to a regular (non-safety) EtherNet/IP device.

3.1 I/O connections

In [2] the use of transport class 0 and 1 are specified for the transfer of I/O data. While non-safe EtherNet/IP practically only uses class 1, [1] (chapter 3-1) specifies the use of class 0 for CIP Safety on EtherNet/IP.

3.2 Quality of Service

According to [3] (chapter 2-10.4) CIP Safety on EtherNet/IP devices must support the Quality of Service mechanism. See also [2] chapter 3-7 and 5-6.

3.3 Duplicate IP Detection

According to [3] (chapter 2-10.3) CIP Safety on EtherNet/IP devices shall implement the IPv4 address conflict detection.

4 General description of CSAL functionality

4.1 Identity Object

Prior to starting the CSAL and EtherNet/IP stack the identity information (Vendor ID, Device Type, Product Code, Revision, Serial Number, Product Name) must be present in the respective attributes of the Identity Object. These values must correspond to those in the Safety Supervisor Object in CSS. These values are not part of the HALC message exchange but the stack user shall make sure they are available on Controller 3 (either stored separately on each controller or by data exchange via the HALC interface prior to starting CSS/CSAL).

The value of the identity object state of CIP Safety devices is determined by the Safety Supervisor Object. Thus CSS sends the device state attribute value whenever it changes. For this the HALC command CSOS_k_CMD_IXSSO_DEV_STATE is used. A description of all HALC commands can be found in chapter 5 of this document.

Care must be taken during system startup or reset that the Identity Object's Status attribute is still synchronized with the device state of CSS.

Safety devices shall not support the Reset Service of the Identity Object. Standard EtherNet/IP stacks used for CIP Safety must be modified accordingly!

4.2 TCP/IP Interface Object

4.2.1 Safety Network Number

For safety devices the TCP/IP Interface Object must support the attribute Safety Network Number. The value to be shown in this attribute is received from CSS with the HALC command CSOS_k_CMD_IXSSO_SNN.

Care must be taken during system startup or reset that the command is received and the SNN gets stored.

4.2.2 Changing the IP address

In general CIP Safety uses the Node ID as an abstract but unique identification for a safety device. Depending on the non-safe network CIP Safety is running on, the node ID is either equal to the IP address (EtherNet/IP), the SDID (Sercos) or the MAC address (DeviceNet).

If the device offers the possibility to change the IP address via the TCP/IP Interface object several safety related issues must be considered:

- It is recommended that after setting the IP address the device needs a reset to the Safety Supervisor Object. Setting a new IP address shall not automatically disrupt safety connections by resetting or immediately restarting the device.
- Changing the IP address will also require a change of the TUNID as the IP address is part of the TUNID data field "Node ID".
- Changing of the IP address requires a restart of CSS. The IP address (i.e. Node ID for CIP Safety on EtherNet/IP) is part of the parameters passed to CSS during initialization (IXSSC_Init()).

The following sequence is an example of how the changing of the IP address can be realized:

1. A tool writes a new IP address configuration to the Interface Configuration Attribute of the TCP/IP Interface Object.
2. The tool requests a Safety_Reset Service of the Safety Supervisor Object.
3. The device resets.
4. During startup the new IP address is transferred from the non-safe controller to the safe controllers.
5. On the safe controllers the new IP address is passed to IXSSC_Init()
6. CSS compares the TUNID that it has stored against the new IP address and will detect that they differ.
7. The Safety Supervisor Object will go into the state “Waiting for TUNID”.
8. The user can use the Propose_TUNID and Apply_TUNID services to set the new TUNID.

4.3 Message Routing

Explicit requests to objects contained within the CSS (Safety Supervisor Object and Safety Validator Object) must be routed via HALC to CSS. Also requests to safety application objects implemented inside the safe application must be routed. These request messages must be transferred using HALC messages of the type CSOS_k_CMD_IXMRO_EXPL_REQ

In the u32_addInfo field the CSAL can tag the request with a unique identifier to be able to match the response message (CSOS_k_CMD_IXSMR_EXPL_RES) to this request. This response message must then be sent to the client who has initiated this explicit message over the same connection respectively the same unconnected messaging session.

Forward_Open and Forward_Close requests use separate specific commands (see 4.4).

For Originators CSS responds with the special response message CSOS_k_CMD_IXSMR_EXPL_RES_INS on requests to read the configuration data of a connection. This can either be a response to a Get_Attribute_All service to a Connection Configuration Object instance or the response to a Get_Attribute_Single of the “Target Config Data” attribute. CSAL then has to insert the configuration data into this message. See also section 4.6.

4.4 I/O Connections

For CIP Safety I/O connections the opening, closing and the transport of the I/O data in both directions must be considered. Also single- and multi-cast must be distinguished. There are also differences if the device is the Target or the Originator of this connection. Details will be explained in the following subsections.

4.4.1 Target

4.4.1.1 Opening an I/O connection

Received Forward_Open requests for Safety Connections must be passed to CSS with the CSOS_k_CMD_IXCMO_SOPEN_REQ HALC command. The CSS then process the request and responds with CSOS_k_CMD_IXSCE_SOPEN_RES. In case this is a success response the CSAL shall set up a Class 0 transport connection in the EtherNet/IP stack and store the Safety Validator Instance ID/Consumer Number relation for this connection. The response data received with the command must be returned to the Originator.

4.4.1.2 Closing I/O connections

Similar to the connection opening received Forward_Close requests to connection instances associated with safety connections must be passed via HALC to CSS using the CSOS_k_CMD_IXCMO_SCLOSE_REQ command. In turn a CSOS_k_CMD_IXSCE_SCLOSE_RES message is received and in case of success the connection must be removed and the response must be sent to the Originator. If CSAL detects that the connection is already non-existent (e.g. because a CSOS_k_CMD_IXSCE_CNXN_CLOSE was received earlier) CSAL shall ignore closing but still send the response to the Originator.

4.4.2 Originator

4.4.2.1 Opening an I/O connection

CSS initiates the sending of a SafetyOpen either by sending a CSOS_k_CMD_IXCCO_SOPEN1_REQ or CSOS_k_CMD_IXCCO_SOPEN2_REQ (depending if Type1 or Type2). In case of a Type1 SafetyOpen CSAL has to insert the configuration data into the message (see section 4.6).

When the response of the Target is received it must be forwarded to CSS by means of a CSOS_k_CMD_IXCMO_SOPEN_RES message. In case it is a success response and it is accepted by CSS it will then send to CSAL a CSOS_k_CMD_CNXN_OPEN to indicate to CSAL that it has to set up the transport connection.

4.4.2.2 Closing I/O connections

For a scheduled closing of a connection CSS sends a SafetyClose by means of a CSOS_k_CMD_IXCCO_CLOSE_REQ message (for an unsolicited close see section 4.4.3).

When the response of the Target is received it must be forwarded to CSS by means of a CSOS_k_CMD_IXCMO_SCLOSE_RES message.

4.4.3 Common (Target and Originator)

In case of an unsolicited connection close the CSS sends a CSOS_k_CMD_IXSCE_CNXN_CLOSE command. CSAL shall then also remove its connection.

For multicast connections the Consumer Number parameter can be zero. This means that not only the connection to one consumer must be closed but the whole connection (including all consumers).

CSAL shall be prepared to receive CSOS_k_CMD_IXSCE_CNXN_CLOSE commands for connections that are already non-existent. In such cases this command shall be ignored.

4.4.4 I/O data exchange

For CSAL the transport of safe I/O data and Time Coordination / Time Correction messages is equivalent. Thus in the following sections only the term I/O data is used but can refer also to Time Coordination / Time Correction messages.

4.4.4.1 Transmission of I/O data

Whenever it is time to send an I/O message CSS transmits a HALC command of type CSOS_k_CMD_IXSVO_IO_DATA containing the transmit data. CSAL then needs to send this out on the corresponding I/O connection identified by the Safety Validator Instance ID allocated during connection opening. This can either be a single- or multi-cast connection.

The transmission of safety I/O messages is triggered by CSS (not a time triggered transmission).

4.4.4.2 Receiving I/O data

CSAL must match a received Safety I/O data message to one of the open safety connections and send it to CSS with the HALC message

CSOS_k_CMD_IXCO_IO_DATA. With this message CSAL must also provide the Safety Validator Instance ID and Consumer Number received from CSS during connection opening.

Timeout detection for connections used for Safety I/O data is provided by CSS. Thus CSAL or the EtherNet/IP stack shall not perform timeout checking for safety connections (in particular automatic deleting of the connection shall not occur).

4.5 General Control commands

4.5.1 Stack termination

When the safe application terminates CSS this event is reported to CSAL by means of the command CSOS_k_CMD_IXSSC_TERMINATE. It is recommended that all EtherNet/IP transport connections that are used for safety connections are also terminated and that HALC sends no further HALC commands to CSS.

4.6 Originator only: Target Configuration Data

With CIP Safety the configuration data for the Target devices can be stored on the Originator. Instead of storing this data in the safe software part the concept of CSS realizes this by storing the data in the non-safe part and inserting the data on the fly when this data have to be sent by the Originator. Commands that need insertion of data (CSOS_k_CMD_IXCCO_SOPEN1_REQ and CSOS_k_CMD_IXSMR_EXPL_RES_INS) provide an offset that indicates at which position the configuration data block has to be inserted in the message and an Application Reference ID that identifies the connection instance. The value of Application Reference ID can be freely assigned by the tool that manages the configuration data for CSS and CSAL. For more details see [5].

5 HALC commands

The following sub-chapters provide detailed information about the HALC commands that are exchanged between CSS and the EtherNet/IP protocol stack.

The description of each command contains a table with these fields:

relevant	This command is used either for Targets only, Originators only or both on Targets and Originators.
Source	Stack unit that is the source of this command. This can be a unit inside CSS for commands that are transmitted from CSS to the EtherNet/IP stack or an EtherNet/IP stack unit for commands that are sent to CSS. The source unit is also part of the command's name.
Destination	Stack unit that is the destination of this command. The destination unit is encoded in the numerical value defined for this command and can be determined using the CSOS_UNIT_HANDLE_GET() macro.
u16_len	Refers to the length field in the HALCS_t_MSG / HALC_t_MSG structure and is equal to the number of bytes contained in the pb_data field. See [4] section 5.3 for the definition of this struct.
u32_addInfo	Describes the value in the Additional Information field of the HALCS_t_MSG / HALC_t_MSG structure.
pb_data	Describes the data associated with the HALCS_t_MSG / HALC_t_MSG structure.

All data is transferred in little endian format.

The command codes are defined in CSOSapi.h.

5.1 CSOS_k_CMD_IXSSO_DEV_STATE

With this command CSS reports any state change of the Safety Supervisor Object state machine to CSAL. The Identity Object state attribute shall reflect the Safety Supervisor Object state.

relevant	Target and Originator
Source	CSS:IXSSO
Destination	CSAL:IXIDO
u16_len	0
u32_addInfo	New Device State (see CSOS_t_SSO_DEV_STATUS in CSOSapi.h)
pb_data	-

5.2 CSOS_k_CMD_IXSSO_SNN

With this command CSS reports its Safety Network Number to the link object of the CIP protocol stack (e.g. TCP/IP Object). CSS sends this message after initialization and whenever the SNN is changed (e.g. after successful Apply_TUNID service).

relevant		Target and Originator
Source		CSS:IXSSO
Destination		CSAL:IXLO
u16_len		6
u32_addInfo		-
pb_data	**0**	SNN.time
	1	
	2	
	3	
	4	SNN.date
	5	

5.3 CSOS_k_CMD_IXSSC_TERMINATE

This command is sent to CSAL when SAPL terminates CSS (by calling IXSSC_Terminate()). It is recommended that all EtherNet/IP transport connections that are used for safety connections are also terminated and that HALC sends no further HALC commands to CSS.

relevant	Target and Originator
Source	CSS:IXSSC
Destination	CSAL:ALC
u16_len	0
u32_addInfo	-
pb_data	-

5.4 CSOS_k_CMD_IXMRO_EXPL_REQ

With this command an Explicit Request message that has been received by the CIP stack is forwarded to CSS (exception: separate commands exist for Forward_Open/Forward_Close).

relevant	Target and Originator
Source	CSAL:IXMRO
Destination	CSS:IXSMR
u16_len	variable
u32_addInfo	Explicit Message Response Channel Index (can be freely assigned by CSAL)
pb_data	raw receive data [0 .. u16_len-1]: all data from the Message Router Request format (starting with Service Code, see [1] section 2-4.1).

5.5 CSOS_k_CMD_IXSMR_EXPL_RES

With this command CSS transfers the Explicit response message from CSS to the CIP stack that has been generated as a reply to the corresponding CSOS_k_CMD_IXMRO_EXPL_REQ command.

relevant	Target and Originator
Source	CSS:IXSMR
Destination	CSAL:IXMRO
u16_len	variable
u32_addInfo	echo from CSOS_k_CMD_IXMRO_EXPL_REQ request
pb_data	raw transmit data [0 .. u16_len-1]: all data from the Message Router Response format (starting with Reply Service Code, see [1] section 2-4.2).

5.6 CSOS_k_CMD_IXSMR_EXPL_RES_INS

This command is a special version of the CSOS_k_CMD_IXSMR_EXPL_RES where the CIP stack has to insert Target configuration data into the Explicit response message (i.e. Connection Configuration Object Instance Get_Attributes_All response or Get_Attribute_Single Attribute 10 response).

relevant		Originator
Source		CSS:IXSMR
Destination		CSAL:IXMRO
u16_len		variable
u32_addInfo		echo from CSOS_k_CMD_IXMRO_EXPL_REQ request
pb_data	**0**	Application Reference ID (Identifier to be freely assigned by the configuration tool to match originated connections in safe application and non-safe application. See also: description of IXCCO_Create in [5])
	1	
	2	
	3	
	4	Config Data Offset (counting starts right after this field)
	5	
	6..N	raw transmit data (incomplete, Configuration Data must be implanted) [0 .. u16_len-7]: data from the Message Router Response format (starting with Reply Service Code). At the position where Config Data Offset points to the configuration data must be inserted.

5.7 CSOS_k_CMD_IXCMO_SOPEN_REQ

This command transfers a Forward_Open for a safety connection to CSS. CSAL should parse this request (except for the Safety Network Segment which can be ignored) and keep all the fields which are necessary to set up the transport connection used for safety. CSS will respond with a CSOS_k_CMD_IXSCE_SOPEN_RES.

relevant	Target
Source	CSAL:IXCMO
Destination	CSS: IXSCE
u16_len	variable
u32_addInfo	Explicit Message Response Channel Index (can be freely assigned by CSAL)
pb_data	raw received Forward_Open Request data [0 .. u16_len-1]: Request_Data of the Message Router Request (starting with Priority/Time_tick. See [3], section 3-3.1).

5.8 CSOS_k_CMD_IXSCE_SOPEN_RES

This command transfers the Forward_Open Response generated as a reply to CSOS_k_CMD_IXCMO_SOPEN_REQ from CSS to CSAL. This can be a success or error response. In case of a success response CSAL shall set up the transport connection.

<table>
<tr><td colspan="2">relevant</td><td colspan="2">Target</td></tr>
<tr><td colspan="2">Source</td><td colspan="2">CSS: IXSCE</td></tr>
<tr><td colspan="2">Destination</td><td colspan="2">CSAL:IXCMO</td></tr>
<tr><td colspan="2">u16_len</td><td colspan="2">variable
(in case of success: 46 (Base Format), 50 (Extended Format))</td></tr>
<tr><td colspan="2">u32_addInfo</td><td colspan="2">echo from CSOS_k_CMD_IXCMO_SOPEN_REQ request</td></tr>
<tr><td rowspan="11">pb_data</td><td>0</td><td rowspan="2">Safety Validator Instance ID *)</td><td rowspan="3">CSAL must store this together with the parameters of this connection because these values will be used to match safety data to a connection in subsequent commands.</td></tr>
<tr><td>1</td></tr>
<tr><td>2</td><td>Consumer Number *)</td></tr>
<tr><td>3</td><td colspan="2">Connection Type *)
See definition of CSOS_k_CNXN_xxx in CSOSapi.h.</td></tr>
<tr><td>4</td><td colspan="2" rowspan="2">Connection Point (assembly instance that is involved in this connection: for a client the assembly that produces the data of this connection, for a server the assembly that consumes the data of this connection)</td></tr>
<tr><td>5</td></tr>
<tr><td>6</td><td>Reply Service Code</td><td rowspan="6">raw transmit data [6 .. u16_len-7]: data from the Message Router Response format (starting with Reply Service Code, see [1] section 2-4.2).</td></tr>
<tr><td>7</td><td>reserved</td></tr>
<tr><td>8</td><td>General Status</td></tr>
<tr><td>9</td><td>Size Additional Status</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>N</td><td>…</td></tr>
</table>

*) 0xFF/0xFFFF in case of an error response

5.9 CSOS_k_CMD_IXCMO_SCLOSE_REQ

This command transfers a Forward_Close request for a safety connection to CSS. CSAL shall not yet close the addressed safety connection. Instead it shall wait for a CSOS_k_CMD_IXCMO_SCLOSE_RES or CSOS_k_CMD_IXSCE_CNXN_CLOSE to do so.

relevant	Target
Source	CSAL:IXCMO
Destination	CSS: IXSCE
u16_len	variable
u32_addInfo	Explicit Message Response Channel Index (can be freely assigned by CSAL)
pb_data	raw received Forward_Close Request data [0 .. u16_len-1]: Request_Data of the Message Router Request (starting with Priority/Time_tick, see [1] section 3-5.5.4).

5.10 CSOS_k_CMD_IXSCE_SCLOSE_RES

This command transfers the Forward_Close response generated as a reply to CSOS_k_CMD_IXCMO_SCLOSE_REQ from CSS to CSAL. This can be a success or error response. In case of a success response CSAL shall close the transport connection. If the connection is already non-existent (e.g. because a CSOS_k_CMD_IXSCE_CNXN_CLOSE was received earlier) CSAL shall ignore closing but still send the response to the Originator.

<table>
<tr><td colspan="2">relevant</td><td colspan="3">Target</td></tr>
<tr><td colspan="2">Source</td><td colspan="3">CSS: IXSCE</td></tr>
<tr><td colspan="2">Destination</td><td colspan="3">CSAL:IXCMO</td></tr>
<tr><td colspan="2">u16_len</td><td colspan="3">variable</td></tr>
<tr><td colspan="2">u32_addInfo</td><td colspan="3">echo from CSOS_k_CMD_IXCMO_SCLOSE_REQ request</td></tr>
<tr><td rowspan="10">pb_data</td><td>0</td><td rowspan="2">Safety Validator Instance ID *)</td><td rowspan="3" colspan="2">Values that had been assigned in CSOS_k_CMD_IXSCE_SOPEN_RES</td></tr>
<tr><td>1</td></tr>
<tr><td>2</td><td>Consumer Number *)</td></tr>
<tr><td>3</td><td colspan="3">Pad Byte **)</td></tr>
<tr><td>4</td><td>Reply Service Code</td><td rowspan="6" colspan="2">raw transmit data [4 .. u16_len-5] data from the Message Router Response format (starting with Reply Service Code, see [1] section 2-4.2).</td></tr>
<tr><td>5</td><td>reserved</td></tr>
<tr><td>6</td><td>General Status</td></tr>
<tr><td>7</td><td>Size Additional Status</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>N</td><td>…</td></tr>
</table>

*) 0xFF/0xFFFF in case of an error response

**) Pad Bytes are always 0

5.11 CSOS_k_CMD_IXCO_IO_DATA

This command transfers received I/O data (Safety I/O data messages as well as Time Coordination messages) to CSS.

relevant		Target and Originator	
Source		CSAL:IXCO	
Destination		CSS: IXSVO	
u16_len		variable (2 + I/O frame length)	
u32_addInfo		Safety Validator Instance ID	Values that had been assigned in CSOS_k_CMD_IXSCE_SOPEN_RES(or for Originators in CSOS_k_CMD_IXCCO_SOPEN1_REQ / CSOS_k_CMD_IXCCO_SOPEN2_REQ)
pb_data	**0**	Consumer Number	
	1	Pad Byte **)	
	2	raw receive data [2 .. u16_len-3]: starts with first byte of the application data field of the Class 0 message.	
	…		
	N		

**) Pad Bytes are always 0

5.12 CSOS_k_CMD_IXSVO_IO_DATA

This command transfers transmit I/O data (Safety I/O data messages as well as Time Coordination messages) to the CIP stack.

relevant		Target and Originator	
Source		CSS: IXSVO	
Destination		CSAL:IXCO	
u16_len		variable (I/O frame length)	
u32_addInfo		Safety Validator Instance ID	Value that had been assigned in CSOS_k_CMD_IXSCE_SOPEN_RES (or for Originators in CSOS_k_CMD_IXCCO_SOPEN1_REQ / CSOS_k_CMD_IXCCO_SOPEN2_REQ)
pb_data	**0**	raw transmit data [0 .. u16_len-1] : starts with first byte of the application data field of the Class 0 message.	
	…		
	N		

5.13 CSOS_k_CMD_IXCCO_SOPEN1_REQ

With this command CSS sends a Type 1 SafetyOpen request to the CIP stack.

relevant		Originator
Source		CSS:IXCCO
Destination		CSAL:IXCMO
u16_len		variable
u32_addInfo		Connection Configuration Object Instance ID (= Safety Validator Instance ID). CSAL must store this together with the parameters of this connection because these values will be used to match safety data to a connection in subsequent commands.
pb_data	**0**	Application Reference ID (Identifier to be freely assigned by the configuration tool to match originated connections in safe application and non-safe application. See also: description of IXCCO_Create in [5])
	1	
	2	
	3	
	4	Config Data Offset (counting starts right after this field)
	5	
	6	Forward_Open Request Data (Type 1, incomplete, Configuration Data must be implanted) [6 .. u16_len-7] ***) Includes Message Router Request Format (see [1] section 2-4.1) and Forward_Open for Safety (see [3] section 3-3.1)
	…	
	N	

***) Simple Data Segment Identifier (0x80) and Segment Length (= ConfigDataSize) has already been included by CSS

5.14 CSOS_k_CMD_IXCCO_SOPEN2_REQ

With this command CSS sends a Type 2 SafetyOpen request to the CIP stack.

relevant		Originator
Source		CSS:IXCCO
Destination		CSAL:IXCMO
u16_len		variable
u32_addInfo		Connection Configuration Object Instance ID (= Safety Validator Instance ID). CSAL must store this together with the parameters of this connection because these values will be used to match safety data to a connection in subsequent commands.
pb_data	**0**	Application Reference ID (Identifier to be freely assigned by the configuration tool to match originated connections in safe application and non-safe application. See also: description of IXCCO_Create in [5])
	1	
	2	
	3	
	4	Forward_Open Request Data (Type 2, complete) [4 .. u16_len-5] Includes Message Router Request Format (see [1] section 2-4.1) and Forward_Open for Safety (see [3] section 3-3.1)
	…	
	N	

5.15 CSOS_k_CMD_IXCCO_SCLOSE_REQ

This command is transferred from CSS to the CIP stack to send a SafetyClose request.

relevant		Originator
Source		CSS:IXCCO
Destination		CSAL:IXCMO
u16_len		variable
u32_addInfo		Connection Configuration Object Instance ID (= Safety Validator Instance ID) Value that had been assigned in CSOS_k_CMD_IXCCO_SOPEN1_REQ / CSOS_k_CMD_IXCCO_SOPEN2_REQ
pb_data	**0**	Application Reference ID (Identifier to be freely assigned by the configuration tool to match originated connections in safe application and non-safe application. See also: description of IXCCO_Create in [5])
	1	
	2	
	3	
	4	Forward_Close Request Data [4 .. u16_len-5] Includes Message Router Request Format (see [1] section 2-4.1) and Forward_Open for Safety (see [1] section 3-5.5.4)
	…	
	N	

5.16 CSOS_k_CMD_IXCMO_SOPEN_RES

This command passes the received Forward_Open response to CSS.

relevant	Originator
Source	CSAL:IXCMO
Destination	CSS:IXCCO
u16_len	variable
u32_addInfo	Safety Validator Instance ID Value that had been assigned in CSOS_k_CMD_IXCCO_SOPEN1_REQ / CSOS_k_CMD_IXCCO_SOPEN2_REQ
pb_data	raw received Forward_Open Response data [0 .. u16_len-1]: all data from the Message Router Response format (starting with Reply Service Code, see [1] section 2-4.2).

5.17 CSOS_k_CMD_IXCMO_SCLOSE_RES

This command passes the received Forward_Close response to CSS.

relevant	Originator
Source	CSAL:IXCMO
Destination	CSS:IXCCO
u16_len	variable
u32_addInfo	Safety Validator Instance ID Value that had been assigned in CSOS_k_CMD_IXCCO_SOPEN1_REQ / CSOS_k_CMD_IXCCO_SOPEN2_REQ
pb_data	raw received Forward_Close Response data [0 .. u16_len-1] : all data from the Message Router Response format (starting with Reply Service Code, see [1] section 2-4.2).

5.18 CSOS_k_CMD_IXSCE_CNXN_OPEN

This command is transferred from CSS to CSAL to initiate the establishment of a safety connection.

relevant		Originator
Source		CSS:IXSCE
Destination		CSAL:IXCMO
u16_len		5
u32_addInfo		Safety Validator Instance ID Value that had been assigned in CSOS_k_CMD_IXCCO_SOPEN1_REQ / CSOS_k_CMD_IXCCO_SOPEN2_REQ
pb_data	**0**	Application Reference ID (Identifier to be freely assigned by the configuration tool to match originated connections in safe application and non-safe application. See also: description of IXCCO_Create in [5])
	1	
	2	
	3	
	4	Connection Type See definition of CSOS_k_CNXN_xxx in CSOSapi.h.

5.19 CSOS_k_CMD_IXSCE_CNXN_CLOSE

This command is transferred from CSS to CSAL to initiate closing a safety connection. In case the addressed connection doesn't exist CSAL shall ignore this message.

relevant		Target and Originator	
Source		CSS:IXSCE	
Destination		CSAL:IXCMO	
u16_len		1	
u32_addInfo		Safety Validator Instance ID	Values that had been assigned in CSOS_k_CMD_IXCCO_SOPEN1_REQ / CSOS_k_CMD_IXCCO_SOPEN2_REQ
pb_data	**0**	Consumer Number (if zero this indicates that the whole connection including all consumers must be closed)	