System Conformance Testing

8

Test Suite Supplement A

A

LT-R and LTS testing

WARNING
This document contains test specifications for the LT-reflex and the LT-supervised Mode. The examples contained in the document are based on the TP0 medium, which has been phased out. Also the specifications refer to the HSOBS test tool, which is no longer available. As it has not yet been decided by KTB, whether or not to also phase out the LT-reflex and the LT-supervised, it has been opted to leave these specifications untouched

Version 01.01.01 is a KNX Approved Standard.

This document is part of the KNX Specifications v2.1.

Document Updates

Version	Date	Modifications
1.0	2002.02.08	Approved Standard
1.1	2009-06	Readying document for publication in V2.0 of KNX specifications – keeping the TSS in view of references in TSS B
01.02.01	2013.10.24	Editorial updates for the publication of KNX Specifications 2.1.

Filename: 08_TSSA System Conformace Testing - LT-R and LT-S Tests v01.01.01 AS.docx
Version: 01.01.01
Status: Approved Standard
Savedate: 2013.10.24
Number of pages: 79

Contents

1 Presentation of the document

1.1 Scope

This document contains the test specifications for the LT-reflex and the LT-supervised mode. The examples contained in this document are based on the TP0 medium. Test specifications dealing with other KNX standardised easy-modes are contained in other documents. During the execution of the relevant underneath tests, consultation of the BDUT's PIXIT is indispensable (e.g. to verify which configuration mode the BDUT supports, to check whether the BDUT is a client or server (or both), etc).

1.2 Reference documents

KNX specifications v1.0.

1.3 Glossary. Abbreviations

HSOBS: PL132 and TP0 observer and frame generator with external acquisition board.

LT: Logical Tag.

PB: Push-button.

SNA: Subnetwork address.

BDUT: Bus Device Under Test.

x/y: high/low for the logical tags.

1.4 Description of test sheets

Each test case is written in the following format:

ID	CEMO/XXX/YYY/NN test identifier: XXX test group, YYY test sub-group, NN test number
NAME	Name of the test
PROP	Description of the test to be performed
NOTE	Additional comments
CONFIG	Test set-up
PROC	Description of the complete test procedure
CONFOR	Defines the expected results

Note: the way a product has to be stimulated to send a frame, depends on its type. This shall be checked in the BDUT's PIXIT proforma.

1.5 General Configuration

The following tools are necessary when carrying out the LT-R and LT-S conformity tests when based on the TP0 medium:

- A TP0 power supply
- A PC equipped with the Observer - Generator HSOBS and its TP0 external acquisition board

Configuration for the tests:

- Open the *CEMODLTR*.SES session file with HSOBS. Then save files under name CEMOxx.

The following underneath **wiring diagram** shall be used during this phase (if not indicated otherwise in the test):

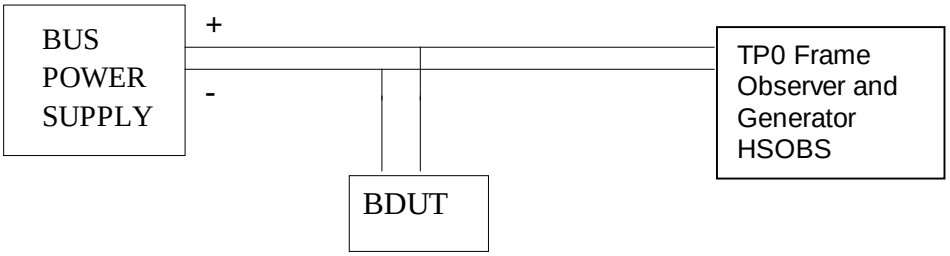

Figure 1 -Wiring diagram

Additional hints:

For carrying out all tests, the BDUT shall either have a fixed application program or shall be loaded with an application program (e.g. in the case of testing a new system component like a BIM). This program shall be properly described. An example is shown in paragraph 5.1.

For some products (e.g. a new system component like a BIM), it might be necessary to power it down and up again before it assumes new “logical tag” values.

When carrying out the underneath tests on the PL medium, one uniform domain address shall be chosen (e.g. AB82h or F0F0h).

2 E-Mode LT Reflex testing

This clause describes the testing of LT-R devices only.

The following items are tested:

- Correspondence between LT and address:
  - Specific values (tag from 3C to 3F) and default value for Individual Address
- SNA through Network Parameter Write
- Device identification (a whole structure of data, different types)
  - Usual
  - Connected mode
- Group address check
- Wildcard number as Logical tag
- Channels
- Link management

  -Read → response
- FOCI
- Parameter management

2.1 CEMO/LTR/AD

The objective of the test presented below is to validate the correspondence between the setting of the local selector (on which the logical tag depends) and the supported Group Address(es). The first test covers the 'normal' local selector settings, the second the 'special' local selector settings.

ID	**CEMO/LTR/AD/1**
NAME	Group address for LT reflex ('normal' values)
PROP	Check whether the BDUT correctly calculates its supported Group Address(es) based on the attributed/set Logical Tag (range 8000h to BFFFh). It is reminded, that the logical tag of a product is made up of two parts x/y, where • x represents values between 1 and 16 • y represents values between 1 and 4. The logical tag takes up 6 bits of the Group Address used by the BDUT. As another part of the Group Address used by the BDUT is taken up by the connection code (see PIXIT of the supplied product), there are as many Group Addresses as different connection codes inside the product. In the following examples, a connection code of 01 is assumed.
NOTE	During this test, it shall also be checked whether the BDUT supports xxFF as source address, where xx is the default SNA depending on the used medium (e.g. 01 on TP0). A number of possible values of the logical tag shall be set (for examples, see underneath table).

CONFIG

PROC In case the BDUT is a sensor,

- Start up the observer.
- stimulate the product to send a number of frames (depending on the number of supported channels).
- Check that the destination Group Address(es) as used in the generated frames corresponds to the one (to those) which can be derived from the set logical tag and supported connection codes (see PIXIT)

In case the BDUT is an actuator,

- Start up the frame generator.
- send a number of Group Write.req frames on the Group Addresses supposedly supported by the BDUT.
- Check that the BDUT reacts to the sent frames according to the behaviour specified in the PIXIT.
- When the BDUT is a two-channel product and y={1 or 3}, verify that the result is the same if LT=x/y+1. If the BDUT is a four-channel product and y=1, verify that the result is the same if LT=x/y+1, x/y+2, x/y+3.

CONFOR

Result		Y/N
The frames as generated by the sensor- BDUT contain the expected Group Addresses – the used default device address is FFh with correct SNA (according medium). The frames are accepted by the actuator-BDUT and lead to the expected results (according PIXIT-proforma).	Satisfactory	

Examples

- CEMOAD1.SEQ sequence file, 1 cycle

Example of supported Group Address(es) according to the setting of the code wheels:

Code wheel setting X/Y = Local Selector	**Coding wheel logical value high**	**Coding wheel logical value low**	**Tag**	**Group address expected if 1 channel**	**Group addresses expected if 2 channels**	**Group addresses expected if 4 channels**
1/1	0	0	00	8001h	8001h and 8101h	8001h,8101h,8201h,8301h
2/2	1	1	05	8501h	8401h and 8501h	8401h,8501h,8601h,8701h
4/3	3	2	0E	8E01h	8E01h and 8F01h	8C01h,8D01h,8E01h,8F01h
8/4	7	3	1F	9F01h	9E01h and 9F01h	9C01h,9D01h,9E01h,9F01h
15/4	E	3	3B	BB01h	BA01h and BB01h	B801h,B901h,BA01h,BB01h

Calculation of tag on basis of code wheel setting X/Y :

Coding wheel (bits) :	0	0	0	1	0	1					
Coding wheel (CW) :	CW High				CW Low						
Tag (decimal) :	0		5								
Tag (bits) :	0	0	0	1	0	1					
Tag :	TAG High		TAG Low								

In case of two channel device all settings of local selector lead to an odd number (1 or 3) of the low part of the LT-Base. In case of 4 channel device the low part of the LT-Base has the value 1.

ID **CEMO/LTR/AD/2**

NAME Group address for LT reflex (‘special’ settings)

PROP The reaction of the BDUT to the special local selector values shall be checked.

NOTE

CONFIG

PROC CASE A

If possible (depending on the number of channels supported by the BDUT), turn the tag to 16/3 “general” tag.

CASE B

If possible (depending on the number of channels supported by the BDUT), turn the tag to 16/1 and 16/2.
Note : For some product such a selection may be impossible.

CASE C

For one channel device, turn the tag to 16/4. (or for two channel device to 16/3 respectively for four channel device to 16/1).
Note : For some product such a selection may be impossible.

CONFOR

Result		Y/N
CASE A The sensor shall send the appropriate Group Addresses and the actuator react according to the supplied PIXIT. CASE B and C The sensor shall not send any frame and the actuator shall not react to the sent frames.	Satisfactory	

Examples CEMOAD2.SEQ sequence file, 1 cycle

2.2 CEMO/LTR/CH

The objective of this test is to check whether the channels implemented in the BDUT are in accordance to the relevant channel code specifications and (if applicable) whether multiple-channel products correctly calculate the supported Group Addresses on the basis of the set logical selector and the supported connection codes (the latter can be read via the descriptor type 2).

ID	**CEMO/LTR/CH/1**

NAME

Channel test

Note: When a BDUT has four inputs, each input represents a channel.

If the BDUT only supports only one channel, the multiple channel test (c) can be skipped.

PROP

a) check in the supplied documentation (PIXIT) which easy channels are supported by the BDUT

b) check the supported channels according to the channel code specifications and Volume 8/7.

c) (if applicable) check that the BDUT supporting more than one channel uses consecutive logical tags for the generation of the supported Group Addresses: it is reminded that the basis for the calculation of these consecutive logical tags is the set local selector.

NOTE

Create a sequence for a BDUT-actuator composed of

a) a frame requesting to supply the device descriptor 2

b) frames with supposedly supported Group Addresses for each channel (depending on the read out channel codes of device descriptor 2 and the set local selector).

For a BDUT-sensor, start up the Observer and stimulate the BDUT to generate frames for its various supported channels.

CONFIG

PROC

Start up the observer and the generator.

CONFOR

Result		Y/N
The frames sent by the BDUT-sensor are correct with appropriate Group Address(es). The frames sent to the BDUT-actuator are accepted when the correct Group Addresses are used and rejected with unsupported Group Addresses. The channels as implemented in the BDUT are in accordance to the relevant channel code specifications. In the Descriptor type 2, channel codes are conform to those indicated in the PIXIT of the product.	Satisfactory	

Examples

- (for BDUT two channel actuator) CEMOCH1.SEQ sequence file, 1 cycle.

2.3 CEMO/LTR/SN

The objective of this test is to check whether the BDUT accepts new SNA values from a SNA client.

ID	**CEMO/LTR/SN**
NAME	Writing of Subnetwork address.
PROP	Check that the BDUT accepts a new SNA via the A_NetworkParameter_Write service (broadcast). This can be checked in the next frame transmitted by the BDUT after the sending of the A_NetworkParameter_Write service.
NOTE	Prior to the test, the product sends frames with the default SNA value. After reception of a new SNA, it shall immediately take it into account
CONFIG	
PROC	• Start up the observer and the generator. • Send a A_NetworkParameter_Write service (IO type 1d, Prop ID 57d, value of new SNA 88h) broadcast • Verify whether the BDUT adopts the new SNA in the next frames it sends. Check whether the BDUT (optionally) reports Individual Address change via the A_NetworkParameter_Write (IO type 1d, Prop ID 60d, Test info = serial number (if not supported 00h) • Verify whether the BDUT accepts frames sent to it with the new SNA and ignores frames with the previous SNA (e.g. by read the device descriptor type 2 of the device) • Send a A_NetworkParameter_Write service with the default SNA again • Check that the BDUT resumes sending frames with this SNA • Check that the BDUT rejects A_NetworkParameter_Write services that are sent multicast (i.e. via a Group Address supported by the device) or point-to-point (i.e. via Individual Address supported by the device).

CONFOR

Result		Y/N
The frames sent by the BDUT after reception of the A_NetworkParameter_Write service (broadcast) with a new SNA are correct (new value inside) – the BDUT rejects A_NetworkParameter_Write services sent multicast or point to point	Satisfactory	

Examples

- Set up the frame generator using the CEMOSN1.SEQ sequence file, 1 cycle.
- TPDU : 03 E4 00 00 39 88 (88 is the new SNA value; 00 00 is the IO-Type of Device Object).

2.4 CEMO/LTR/DI

The objective of the tests presented below is to check the supported device identification of the product via the DeviceDescriptor-Read service (APCI = 1100 (request) and APCI = 1101 (response) – the descriptor types are coded 0 for type 0, 1 for type 1, etc…).

ID	**CEMO/LTR/DI/1**

NAME Device Identification reading, individual connectionless – negative test on multicast

PROP Check that the BDUT returns correctly its mask version (device descriptor type 0), its device identification (type 2) and an error message for all other read but undefined types.

NOTE Create a sequence with device descriptor read requests for all existing descriptor types 0 to 8 and with at least one other value between 9 and 63. The priority value may be set to any value.

CONFIG

- The generator shall be set to provide the Ack if requested by messages coming from the BDUT. In the underneath test example, the Individual Address of the generator is set to 0167h.
- Destination Address of the request shall be the Individual Address of the BDUT. The destination address of the response shall be the source address of the request.
- The BDUT shall ignore any requests that are sent via multicast (group) communication.

PROC

- Start up the observer, followed by the generator.

CONFOR

Result		Y/N
The BDUT shall supply the supported device descriptor 0 and 2 (according to the formats as specified in supplement 15 “Easy Common Parts” and the PIXIT of the BDUT) and use the correct Individual Address. For any unsupported device descriptor it shall return the “error” device descriptor 3Fh. The BDUT shall ignore any messages sent by group communication	Satisfactory	

Examples

- Set up the frame generator using the CEMODI1.SEQ. Subsequently run the CEMODI2.SEQ sequence file, 1 cycle.

TPDU of the answer for type 0: 03 40 30 12

TPDU of the answer for type 2: 03 42 (rest of the data is an example) 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 (total of 16 bytes)
TPDU for error message response (in case of unsupported device identification): 03 3F

ID	**CEMO/LTR/DI/2**

NAME Device Identification reading, individual connection oriented

PROP Check that the BDUT returns correctly its mask version (device descriptor type 0), its device identification (type 2) and an error message for all other read but undefined types.

NOTE Create a sequence with a T-Connect, followed by device descriptor read requests for all existing descriptor types 0 to 8 and with at least one other value between 9 and 63. The priority value may be set to any value.

CONFIG

- The generator shall be set to provide the Ack if requested by messages coming from the BDUT. In the underneath test example, the Individual Address of the generator is set to 0167h.
- Destination Address of the request shall be the Individual Address of the BDUT. The destination address of the response shall be the source address of the request.

PROC

- Start up the observer, followed by the generator.

CONFOR

Result		Y/N
The BDUT shall supply the supported device descriptor 0 and 2 (according to the formats as specified in supplement 15 “Easy Common Parts” and the PIXIT of the BDUT) and use the correct Individual Address. For any unsupported device descriptor it shall return the “error” device descriptor 3Fh.	Satisfactory	

Examples Set up the frame generator using the CEMODI5.SEQ sequence file, 1 cycle

2.5 CEMO/LTR/GAcheck

The objective of the tests below is to check whether an LTR-device supports the A_NetworkParameter_Read-service (type Group Address Check), allowing an external device/client to check whether the BDUT supports a particular Group Address. Devices (servers) shall be able to support Group Address check over the entire address range (not only C000h to DFFFh)!! However in case of LTR devices, these addresses can not be set outside the range.

ID	**CEMO/LTR/GA/1**

NAME Correct handling of a Group address check via A_NetworkParameter_Read (type Group Address Check)

PROP Check that the BDUT supports correctly the Group Address check mechanism, with the defined service.

NOTE The device shall be loaded with a number of addresses:

Example:

GA (h)	linked to Group Object
C234	0
C235	0
C236	1 and 2
C237	no link

A sequence shall be established containing a number of A-NetworkParameter_Read requests (point to point connectionless – Interface Object type 1d, PropID 23d) where in

Case A: the BDUT shall not support the Group Address given in the range. In the above example, make a A_NetworkParameter_Read with range 1d – Group Address C238h

Case B: the device supports at least one Group Address given in the range. In the above example, make a A_NetworkParameter_Read with range 5d – Group Address C230h

Case C: the device supports the Group Address given in the range and this Group Address is linked to more than one Group Object. In the above example, make a A_NetworkParameter_Read with range 1d – Group Address C236h

Case D: the device supports two Group Addresses given in the range, both linked to the same Group Object. In the above example, make a A_NetworkParameter_Read with range 2d – Group Address C234h

Case E: the device supports the Group Addresses given in the range, but this is not linked to a Group Object. In the above example, make a A_NetworkParameter_Read with range 1d – Group Address C237h

By means of a last frame in the sequence, one shall try to execute a A_NetworkParameter_Read on a multicast-basis, i.e. on a Group Address supported by the device

CONFIG TPDU of the request is:

Case A: 03 DA 00 01 17 01 C2 38

Case B: 03 DA 00 01 17 05 C2 30

Case C: 03 DA 00 01 17 01 C2 36

Case D: 03 DA 00 01 17 02 C2 34

Case E: 03 DA 00 01 17 01 C2 37

PROC

- Start up the observer and the generator.

CONFOR	Result		Y/N
	The BDUT sends the correct A_NetworkParameter_Responses – the BDUT shall not show any reaction when receiving A_NetworkParameter_Read on multicast	Satisfactory	

Examples

- Set up the frame generator using the CEMOGA1.SEQ sequence file, 1 cycle.
- The TPDU of the response shall be:

Case A: no response

Case B: 03 DB 00 01 17 05 C2 30

Case C: 03 DB 00 01 17 01 C2 36

Case D: 03 DB 00 01 17 02 C2 34

Case E: 03 DB 00 01 17 01 C2 37

ID **CEMO/LTR/GA/2 (mandatory if supported)**

NAME Generation of the A_NetworkParameter_Read-service by the BDUT working as a client. Check the PIXIT proforma, whether this feature is supported by the BDUT.

PROP Check that the BDUT is able to carry out a **Group Address check** of a number of connected server devices.

NOTE The BDUT shall initialise a Group Address check and use the Group Address for which it receives no response (i.e. which is not used in the connected installation).

CONFIG

- See PIXIT of the product

PROC

- Start up the observer and the generator.

CONFOR	Result		Y/N
	The BDUT correctly sends the A_NetworkParameter_Read-service and uses the Group Address for which no answer has been received.	Satisfactory	

Examples

2.6 CEMO/LTR/WN

The objective of this test is to check the use of the wildcard number as logical tag (in case of sender) and the handling of the wildcard number as a receiver.

ID	**CEMO/LTR/WN/1**

NAME LT = 3Eh, wildcard number for logical tag reflex products.

PROP Test of correct handling of wildcard number in case of receiver.

Check that the BDUT understands frames with wildcard number in the logical tag sub-field of the Group Address. The BDUT shall act in the same way as with the defined tag.

NOTE A sequence shall be created containing frames for every Group Object supported by the product: however the set logical tag shall be replaced by the wildcard number in the LT sub-field of the Group Address.

CONFIG As an example, when replacing the supported Group Address by the wildcard number, the GA 8C0F becomes BE0F (wildcard number 3Eh replaces the 6 bits 0Ch - connection code remains 0Fh) and 9F01 becomes BE01.

PROC Start up the observer and the generator. The sequence shall first send frames with the normal supported Group Addresses followed by frames with the Group Addresses containing the wildcard number: BDUT shall react in both cases in the same way.

CONFOR

Result		Y/N
Frames with the wildcard number are accepted by the BDUT.	Satisfactory	

Examples Set up the frame generator using the CEMOWN1.SEQ sequence file, 1 cycle.

ID	**CEMO/LTR/WN/2 (mandatory if supported)**

NAME LT = 3Eh, wildcard number *chosen as tag* for a logical tag reflex product.

PROP Test of the **sending** of frames by the BDUT, of which the set tag is identical to the wildcard number.

Note: For the wildcard number, the logical value of the coding wheels shall be set to F2 and their X/Y values therefore correspond to 16/3.

NOTE Stimulate the BDUT to send frames. The frames transmitted by the BDUT contain the wildcard number in the LT subfield of the Group Address for every Group Object known by the product.

CONFIG When setting the local selector to the wildcard number, the GA 9040 becomes BE40 (wildcard number 3Eh replaces 10h [on 6 bits] – connection code remains 40h).

PROC Start up the observer and the generator.

CONFOR

Result		Y/N
The frames sent by the BDUT contain the wildcard number.	Satisfactory	

Examples Set up the frame generator using the CEMOWN2.SEQ sequence file, 1 cycle.

ID **CEMO/LTR/WN/3 (mandatory if supported)**

NAME LT = 3Eh, wildcard number *chosen as tag* for a logical tag reflex product.

PROP Test of the **reception** of frames by a device, of which the logical tag is identical to the wildcard number.

Check that the BDUT accepts any frame regardless the contents of the logical tag sub-field of the Group Address (all logical tag sub-fields values are accepted). As an example, a BDUT normally supporting the GA BE40 will accept also values such as 9040.

Note: For the wildcard number, the logical value of the coding wheels shall be set to F2 and their X/Y values therefore correspond to 16/3.

NOTE The frames as sent to the BDUT shall contain a connection code that is supported by it (see PIXIT proforma).

CONFIG

PROC Start up the observer and the generator.

CONFOR

Result		Y/N
The BDUT accepts all frames, provided they contain a supported connection code	Satisfactory	

Examples Set up the frame generator using the CEMOWN3.SEQ sequence file, 1 cycle

2.7 CEMO/LTR/LM (MANDATORY IF SUPPORTED)

The objective of these tests is to check the correct handling of the link management The Link Management consists of the frame types A-Link-Read and A-Link-Response.

ID **CEMO/LTR/LM/1 (mandatory if supported)**

NAME Link Management read (A-Link-Read)

PROP Check that the BDUT responds correctly to this sent request. Ensure that the BDUT supports a number of Group Objects, each having one or more links.

Example :

GO 0 C230 (S), C231

GO 1 C240 (S), C241, C242, C243, C244, C245, C246, C247, C248, C249

GO 2 C250 (S), C251

GO 3 C260 (S), C261, C262, C263, C264, C265, C266, C267, C268, C269, C26A, C26B, C26C, C26D, C26E, C26F, C270, C271, C272, C274, C275

(S) = sending Group Address

NOTE Create a sequence containing A_Link_Read frames (individual connectionless) for every Group Object supported by the BDUT.

CASE A: send an A_Link_Read frame for an unsupported Group Object

CASE B: send an A_Link_Read frame for a supported Group Object with two Group Addresses with start index = 1 and start index = 2

CASE C: send an A_Link_Read frame for a supported Group Object with 10 Group Addresses with start index = 5

CASE D: send an A_Link_Read frame for a supported Group Object with 2 Group Addresses with start index = 3

CASE E: send an A_Link_Read frame for a supported Group Object with 21 Group Addresses with start index = 15

Try to send A_Link_Read frame also on multicast communication (on a Group Address supported by BDUT).

CONFIG

PROC Start up the observer and the generator.

CONFOR

Result		Y/N
The A-Link-Response messages contain the correct links of the read Group Object – An A_Link_Read on multicast is ignored.	Satisfactory	

Examples

- Set up the frame generator using the CEMOLM1.SEQ sequence file, 1 cycle.
- TPDU of the request:

CASE A: 03 E5 04 01

CASE B1: 03 E5 00 01

CASE B2: 03 E5 00 02

CASE C: 03 E5 01 05

CASE D: 03 E5 02 03

CASE E: 03 E5 03 0F

- TPDU of the response:

CASE A: 03 E6 04 00

CASE B1: 03 E6 00 11 C2 30 C2 31

CASE B2: 03 E6 00 12 C2 31

CASE C: 03 E6 01 15 C2 44 C2 45 C2 46 C2 47 C2 48 C2 49

CASE D: 03 E6 02 00

CASE E: 03 E6 03 1F C2 6E C2 6F C2 70 C2 71 C2 72 C2 74

ID	**CEMO/LTR/LM/2 (mandatory if supported)**
NAME	Link Management and Direct_Memory_Write service
PROP	In the case where a BDUT supports both Link Management as well as Direct Memory access (check the PICS proforma), usage of Direct Memory_Write shall disable the Link Management procedures. The reversing of the disabling of the Link Management procedures is manufacturer dependant.
NOTE	
CONFIG	Set up the frame generator
PROC	(If possible, see PICS proforma) Set BDUT to E-Mode (e.g. by setting an appropriate flag) Send a number of A_Link_Read services to Group Objects supported by the BDUT. Check whether it returns A_Link_Responses. Open connection and send Direct_Memory messages to the BDUT Check that the BDUT ignores any further Link Messages, e.g. by sending it A_Link_Read-services to supported Group Objects. Check in the PICS proforma how the BDUT can only be reset to E-Mode (e.g. through full re-programming of its memory with the ex factory settings).

CONFOR	Result		Y/N
	The BDUT ignores any further Link Messages after having received Direct_Memory messages	Satisfactory	

Examples

- set up the frame generator using the CEMOLM4.SEQ sequence file, 1 cycle.

2.8 CEMO/LTR/NS

The objective of the tests presented below is to validate the understanding of the read service on “serial number” and “object type”. These two items can be accessed via the Reduced Interface Object mechanisms or (optionally) via the A_NetworkParameter_Read.

ID **CEMO/LTR/NS/1**

NAME Reading of Serial number property via Reduced interface object mechanisms or (optionally) via A_NetworkParameter_Read service

PROP Check that the BDUT correctly interprets either message type and returns the value of the serial number in its response.

NOTE

CONFIG

PROC Start the observer and the generator.

CONFOR	Result		Y/N
	The BDUT returns the correct responses and the correct serial number.	Satisfactory	

Examples

- Set up the frame generator using the CEMONS1.SEQ sequence file, 1 cycle.
- TPDU of the A-NetworkParameter-Read request: 03 DA 00 00 0B 00.
  TPDU of the A-NetworkParameter-Response: 03 DB 00 00 0B 00 (+ 6 bytes serial number).
- TPDU of the A-PropertyValue-Read: 03 D5 00 0B 10 01,
  TPDU of the A-PropertyValue-Response: 03 D6 00 0B 10 01 (+ 6 bytes serial number).

ID	**CEMO/LTR/NS/2**
NAME	Checking of rejection of service A_PropertyDescription-Read in case of support of Reduced interface object mechanisms
PROP	Check that the BDUT rejects the service A_PropertyDescription-Read in case of support of Reduced interface object mechanisms
NOTE	
CONFIG	
PROC	Start the observer and the generator.

CONFOR	Result		Y/N
	The BDUT ignores any A_PropertyDescription-Read services sent to it.	Satisfactory	

Examples

- Set up the frame generator using the CEMONS3.SEQ sequence file, 1 cycle.
- TPDU of the A-PropertyDescriptionRead: 03 D8 00 0B 01, no response is sent by the BDUT.

2.9 CEMO/LTR/FOCI (mandatory if supported)

The objective of the tests presented below is to validate whether

a) the BDUT is able to handle Functions of Common Interest if supported

b) the BDUT ignores Functions of Common Interest that are not supported.

ID	**CEMO/LTR/FO/1**
NAME	Test of FOCIs = Functions of common interest [if present in the application]
PROP	Check that the BDUT returns values requested via a supported FOCI Group Address or ignores FOCI Group Addresses not supported by it.
NOTE	Create a sequence of Group-Read frames, containing supported and unsupported FOCI Group Addresses.
CONFIG	
PROC	Start up the observer and the generator.

CONFOR	Result		Y/N
	When receiving a Group Read on a supported FOCI Group Address, the appropriate value is returned. When receiving a Group Read on an unsupported FOCI Group Address, the BDUT ignores the message.	Satisfactory	

Examples

- Set up the frame generator using the CEMOFO1.SEQ sequence file, 1 cycle.
- In the example given, a read request is sent on Group Address F001 (FOCI date). In the received response the date is coded according to the KNX Datapoint type 11.001

2.10 CEMO/LTR/PARA

ID **CEMO/LTR/PA/1**

NAME Parameter reading – (optional) Parameter writing

PROP Checking of the correct implementation of easy channel parameters in devices. Checking of handling of A_PropertyValue_Read and (if supported) A_PropertyValue_Write requests to the channel parameters of the BDUT

NOTE It shall be checked in the supplied documentation (PIXIT), which easy channels are supported by the BDUT.

CONFIG

PROC Depending on the supported channels, a number of parameter settings shall be read and if supported written. Test procedure shall be in accordance with Volume 8/3/7 § 2.13 and 2.14 (appropriately adapted to connectionless operation)

CONFOR	Result		Y/N
	The supported channels are in accordance with the relevant channel code specifications and the behavior of the BDUT is in accordance with the channel code specifications after parameter settings (if supported).	Satisfactory	

Examples

- Set up the frame generator using the CEMOPA1.SEQ sequence file
- TPDU of A_PropertyValue_Write is : 03 D7 00 65 20 01 22 22 ,
- “00” is the n° of “Device Object”,
- “65” is the Property_ID “channel 1 parameter block”,
- “2” from “20” indicates number of elements of the block,
- “1” from “01” indicates start-index,
- 22 22 are the written bytes

3 E-Mode LT Supervised testing

Following items are tested:

- Correspondence between LT and Individual address
- SNA through Network Parameter Write
- Device identification
  - Type 2
  - Type 0
- Group address check
- Link management

  -Read → response

  -Write → response
- FOCI
- Reduced Interface Objects

  -Serial number read

  -Object type read
- Parameter management

3.1 CEMO/LTS/SNPM

In this introduction, a brief overview is given how an Individual Address is assigned to an LTS product.

a) via the logical tag from the selector (x/y).

The high part of the logical tag shall be set to a value between 1 and 16, the low part between 5 and 16. In the underneath examples, 8/9 is used as the local selector value (where applicable).

It is reminded that if the low part of the selector is set to less than 4, the BDUT is switched to Reflex-Mode (where the device address is automatically set to FF and the subnetwork address to a value according to the medium). This shall be checked according to the above test specifications.

b) Assignment (or reading) of Individual Address via serial number (see appropriate tests in Volume 8/3/7)

This is the case when the selector has its default value, i.e. 16/16.

c) Assignment (or reading) of Individual Address by setting the device in programming mode (see appropriate tests in Volume 8/3/7)

This is the case when the selector has its default value, i.e. 16/16.

Note:

In case of multiple-channel devices, the link manager may assign random Group (there is no obligation to use successive Group Addresses as for LTR)

3.2 CEMO/LTS/SN

The objective of this test is to check whether the BDUT accepts new SNA values from a SNA client.

ID	**CEMO/LTS/SN**
NAME	Writing of Subnetwork address.
PROP	Check that the BDUT accepts a new SNA via the A_NetworkParameter_Write service (broadcast or point to point connectionless). This can be checked in the next frame transmitted by the BDUT after the sending of the A_NetworkParameter_Write service.
NOTE	Prior to the test, the product sends frames with the default SNA value. After reception of a new SNA, it shall immediately take it into account
CONFIG	
PROC	• Start up the observer and the generator. • Send a A_NetworkParameter_Write service (IO type 1d, Prop ID 57d, value of new SNA 88h) either broadcast or point to point connectionless • Verify whether the BDUT adopts the new SNA in the next frames it sends. Check whether the BDUT (optionally) reports Individual Address change via the A_NetworkParameter_Write (IO type 1d, Prop ID 60d, Test info = serial number (if not supported 00h) • Verify whether the BDUT accepts frames sent to it with the new SNA and ignores frames with the previous SNA (e.g. by read the device descriptor type 2 of the device) • Send a A_NetworkParameter_Write service with the default SNA again • Check that the BDUT resumes sending frames with this SNA • Check that the BDUT rejects A_NetworkParameter_Write services that are sent multicast (i.e. via a Group Address supported by the device) or point to point

CONFOR	Result		Y/N
	the frames sent by the BDUT after reception of the A_NetworkParameter_Write service (either broadcast or point to point connectionless) with a new SNA are correct (new value inside) – the BDUT rejects A_NetworkParameter_Write services sent multicast or point to point	Satisfactory	

Examples

- Set up the frame generator using the CEMOSN2.SEQ sequence file, 1 cycle.
- TPDU : 03 E4 00 00 39 88 (88 is the new SNA value).

3.3 CEMO/LTS/DI

The objective of the tests presented below is to check the supported device identification of the product via the DeviceDescriptor-Read service (APCI = 1100 (request) and APCI = 1101 (response) – the descriptor types are coded 0 for type 0, 1 for type 1, etc…).

ID	**CEMO/LTS/DI/1**

NAME Device Identification reading, individual connectionless – negative test on multicast

PROP Check that the BDUT returns correctly its mask version (device descriptor type 0), its device identification (type 2) and an error message for all other read but undefined types.

NOTE Create a sequence with device descriptor read requests for all existing descriptor types 0 to 8 and with at least one other value between 9 and 63. The priority value may be set to any value.

CONFIG

- The generator shall be set to provide the Ack if requested by messages coming from the BDUT. In the underneath test example, the Individual Address of the generator is set to 0167h.
- Destination Address of the request shall be the Individual Address of the BDUT. The destination address of the response shall be the source address of the request.
- The BDUT shall ignore any requests that are sent via multicast (group) communication.

PROC

- Start up the observer, followed by the generator.

CONFOR

Result		Y/N
The BDUT shall supply the supported device descriptor 0 and 2 (according to the formats as specified in supplement 15 clause 1.3.3.2 and the PIXIT of the BDUT) and use the correct Individual Address. For any unsupported device descriptor it shall return the device descriptor 3Fh. The BDUT shall ignore any messages sent by group communication	Satisfactory	

Examples

- Set up the frame generator using the CEMODI3.SEQ and the CEMODI4.SEQ, 1 cycle. x/y=8/9.

TPDU of the answer for type 0: 03 40 30 12

TPDU of the answer for type 2: 03 42 (rest of the data is an example) 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 (total of 16 bytes)
TPDU for error message response (in case of unsupported device identification): 03 3F

ID	**CEMO/LTS/DI/2**

NAME Device Identification reading, individual connection oriented

PROP Check that the BDUT returns correctly its mask version (device descriptor type 0), its device identification (type 2) and an error message for all other read but undefined types.

NOTE Create a sequence with a T-Connect, followed by device descriptor read requests for all existing descriptor types 0 to 8 and with at least one other value between 9 and 63. The priority value may be set to any value.

CONFIG

- The generator shall be set to provide the Ack if requested by messages coming from the BDUT. In the underneath test example, the Individual Address of the generator is set to 0167h.
- Destination Address of the request shall be the Individual Address of the BDUT. The destination address of the response shall be the source address of the request.

PROC

- Start up the observer, followed by the generator.

CONFOR

Result		Y/N
The BDUT shall supply the supported device descriptor 0 and 2 (according to the formats as specified in supplement 15 clause 1.3.3.2 and the PIXIT of the BDUT) and use the correct Individual Address. For any unsupported device descriptor it shall return the device descriptor 3Fh.	Satisfactory	

Examples Set up the frame generator using the CEMODI6.SEQ sequence file, 1 cycle. x/y=8/9.

3.4 CEMO/LTS/GAcheck

The objective of the tests below is to check whether an LTS-device supports the A_NetworkParameter_Read-service (type Group Address Check), allowing an external device/client to check whether the BDUT supports a particular Group Address. Devices (servers) shall be able to support Group Address check over the entire address range (not only C000h to DFFFh)!

ID	**CEMO/LTS/GA/1**

NAME Correct handling of a Group address check via A_NetworkParameter_Read (type Group Address Check)

PROP Check that the BDUT supports correctly the Group Address check mechanism, with the defined service.

NOTE The device shall be loaded with a number of addresses:

Example:

GA (h) linked to Group Object

C234	0
C235	0
C236	1 and 2
C237	no link

A sequence shall be established containing a number of A-NetworkParameter_Read requests (point to point connectionless – Interface Object type 1d, PropID 23d) where in

Case A: the BDUT shall not support the Group Address given in the range. In the above example, make a A_NetworkParameter_Read with range 1d – Group Address C238h

Case B: the device supports at least one Group Address given in the range. In the above example, make a A_NetworkParameter_Read with range 5d – Group Address C230h

Case C: the device supports the Group Address given in the range and this Group Address is linked to more than one Group Object. In the above example, make a A_NetworkParameter_Read with range 1d – Group Address C236h

Case D: the device supports two Group Addresses given in the range, both linked to the same Group Object. In the above example, make a A_NetworkParameter_Read with range 2d – Group Address C234h

Case E: the device supports the Group Addresses given in the range, but this is not linked to a Group Object. In the above example, make a A_NetworkParameter_Read with range 1d – Group Address C237h

By means of a last frame in the sequence, one shall try to execute a A_NetworkParameter_Read on a multicast-basis, i.e. on a Group Address supported by the device

CONFIG TPDU of the request is:

Case A: 03 DA 00 01 17 01 C2 38

Case B: 03 DA 00 01 17 05 C2 30

Case C: 03 DA 00 01 17 01 C2 36

Case D: 03 DA 00 01 17 02 C2 34

Case E: 03 DA 00 01 17 01 C2 37

PROC

- Start up the observer and the generator.

CONFOR

Result		Y/N
The BDUT sends the correct A_NetworkParameter_Responses – the BDUT shall not show any reaction when receiving A_NetworkParameter_Read on multicast	Satisfactory	

Examples

- Set up the frame generator using the CEMOGA1.SEQ sequence file, 1 cycle.
- The TPDU of the response shall be:

Case A: no response

Case B: 03 DB 00 01 17 05 C2 30

Case C: 03 DB 00 01 17 01 C2 36

Case D: 03 DB 00 01 17 02 C2 34

Case E: 03 DB 00 01 17 01 C2 37

ID	**CEMO/LTS/GA/2 (mandatory if supported)**
NAME	Generation of the A_NetworkParameter_Read-service by the BDUT working as a client. Check the PIXIT proforma, whether this feature is supported by the BDUT.
PROP	Check that the BDUT is able to carry out a **Group Address check** of a number of connected server devices.
NOTE	The BDUT shall initialise a Group Address check and use the Group Address for which it receives no response (i.e. which is not used in the connected installation).
CONFIG	• See PIXIT of the product
PROC	• Start up the observer and the generator.

CONFOR

Result		Y/N
The BDUT correctly sends the A_NetworkParameter_Read-service and uses the Group Address for which no answer has been received.	Satisfactory	

Examples

3.5 CEMO/LTS/LM

The objective of these tests is to validate the link management.

It is composed of:

- A-Link-Write (adding and deleting)
- A-Link-Read
- A-Link-Response (6 max readable Group Address for one Group Object, in one frame).

ID	**CEMO/LTS/LM/1**

NAME Link Management read (A-Link-Read)

PROP Check that the BDUT responds correctly to this sent request. Ensure that the BDUT supports a number of Group Objects, each having one or more links.

Example :

GO 0 C230 (S), C231

GO 1 C240 (S), C241, C242, C243, C244, C245, C246, C247, C248, C249

GO 2 C250 (S), C251

GO 3 C260 (S), C261, C262, C263, C264, C265, C266, C267, C268, C269, C26A, C26B, C26C, C26D, C26E, C26F, C270, C271, C272, C274, C275

(S) = sending Group Address

NOTE Create a sequence containing A_Link_Read frames (individual connectionless) for every Group Object supported by the BDUT.

CASE A: send an A_Link_Read frame for an unsupported Group Object

CASE B: send an A_Link_Read frame for a supported Group Object with two Group Addresses with start index = 1 and start index = 2

CASE C: send an A_Link_Read frame for a supported Group Object with 10 Group Addresses with start index = 5

CASE D: send an A_Link_Read frame for a supported Group Object with 2 Group Addresses with start index = 3

CASE E: send an A_Link_Read frame for a supported Group Object with 21 Group Addresses with start index = 15

Try to send A_Link_Read frame also on multicast communication (on a Group Address supported by BDUT).

CONFIG

PROC Start up the observer and the generator.

CONFOR

Result		Y/N
The A-Link-Response messages contain the correct links of the read Group Object – An A_Link_Read on multicast is ignored.	Satisfactory	

Examples

- Set up the frame generator using the CEMOLMD1.SEQ sequence file, 1 cycle.
- TPDU of the request:

CASE A: 03 E5 04 01

CASE B1: 03 E5 00 01

CASE B2: 03 E5 00 02

CASE C: 03 E5 01 05

CASE D: 03 E5 02 03

CASE E: 03 E5 03 0F

- TPDU of the response:

CASE A: 03 E6 04 00

CASE B1: 03 E6 00 11 C2 30 C2 31

CASE B2: 03 E6 00 12 C2 31

CASE C: 03 E6 01 15 C2 44 C2 45 C2 46 C2 47 C2 48 C2 49

CASE D: 03 E6 02 00

CASE E: 03 E6 03 1F C2 6E C2 6F C2 70 C2 71 C2 72 C2 74

ID	**CEMO/LTS/LM/2**
NAME	Link Management write (add respectively delete a link on a given Group Object).
PROP	
NOTE	As an example, the BDUT supports GO0 with Group Address C234 (Sending), C235, C236 GO1 with no attributed Group Address Create a sequence of frames for a Group Object supported by the BDUT, by means of which : - case A : a range of new Group Addresses is added to a supported Group Object (e.g. C240 to C246 to GO1) - case B : a Group Address is added to a non-supported Group Object (e.g. C241 to GO2) - case C : another Group Address is set to sending than the one defined (e.g. C236 assigned to GO0) - case D: an assigned Group Address is deleted (e.g. C235 assigned to GO0)
CONFIG	
PROC	• Start up the observer and the generator. • Transmission of the frames. Case A : Check responses generated by the BDUT and check via a Group Write whether the GO1 accepts new values via the newly assigned Group Addresses Case B: Check responses generated by the BDUT and check via a Group Read whether the BDUT does not react to a Group Read on the Group Address Case C: Check responses generated by the BDUT and check via a Group Read

whether the GO0 sends its value on the newly assigned (sending) Group Address

Case D: Check responses generated by the BDUT and check via a Group Write that the BDUT no longer reacts to messages sent to it via the deleted Group Address.

CONFOR	Result		Y/N
	The BDUT makes the necessary adjustments to the assigned Group Addresses.	Satisfactory	

Examples With the generator send the sequence CEMOLMD2.seq then the CEMOLMD3.seq.

TPDU of the request

Case A:

03 E7 01 01 C2 40,

03 E7 01 00 C2 41,

03 E7 01 00 C2 42,

03 E7 01 00 C2 43,

03 E7 01 00 C2 44,

03 E7 01 00 C2 45,

03 E7 01 00 C2 46

Case B: 03 E7 02 00 C2 41.

Case C: 03 E7 00 01 C2 36.

Case D: 03 E7 00 02 C2 35.

TPDU of the response

Case A:

03 E6 01 11 C2 40,

03 E6 01 11 C2 40 C2 41,

03 E6 01 11 C2 40 C2 41 C2 42,

03 E6 01 11 C2 40 C2 41 C2 42 C2 43,

03 E6 01 11 C2 40 C2 41 C2 42 C2 43 C2 44,

03 E6 01 11 C2 40 C2 41 C2 42 C2 43 C2 44 C2 45,

03 E6 01 12 C2 41 C2 42 C2 43 C2 44 C2 45 C2 46 (or alternatively e.g. 03 E6 01 17 C2 46 depending on start index value, in this case 7)

Case B: 03 E6 02 00

Case C: 03 E6 00 31 C2 34 C2 35 C2 36

Case D: 03 E6 00 21 C2 34 C2 36

Note: Check in the PICS-proforma under which conditions it is possible to generate negative responses (start index 00) in case of adding or deleting a Group Address.

ID	**CEMO/LTS/LM/3**
NAME	Link Management and Direct_Memory_Write service
PROP	In the case where a BDUT supports both Link Management as well as Direct Memory access (check the PICS proforma), usage of Direct Memory_Write shall disable the Link Management procedures. The reversing of the disabling of the Link Management procedures is manufacturer dependant.
NOTE	
CONFIG	Set up the frame generator
PROC	(If possible, see PICS proforma) Set BDUT to E-Mode (e.g. by setting an appropriate flag) Send a number of A_Link_Read services to Group Objects supported by the BDUT. Check whether it returns A_Link_Responses. Open connection and send Direct_Memory messages to the BDUT Check that the BDUT ignores any further Link Messages, e.g. by sending it A_Link_Read-services to supported Group Objects. Check in the PICS proforma how the BDUT can only be reset to E-Mode (e.g. through full re-programming of its memory with the ex factory settings).

CONFOR

Result		Y/N
The BDUT ignores any further Link Messages after having received Direct_Memory messages	Satisfactory	

Examples

- set up the frame generator using the CEMOLMD4.SEQ sequence file, 1 cycle.

3.6 CEMO/LTS/NS

The objective of the tests presented below is to validate the understanding of the read service on “serial number” and “object type”. These two items can be accessed via the Reduced Interface Object mechanisms or (optionally) via the A_Network_Read parameter.

ID	**CEMO/LTS/NS/1**
NAME	Reading of Serial number property via Reduced interface object mechanisms or (optionally) via A_NetworkParameter_Read service
PROP	Check that the BDUT correctly interprets either message type and returns the value of the serial number in its response.
NOTE	
CONFIG	

PROC Start the observer and the generator.

CONFOR

Result		Y/N
The BDUT returns the correct responses and the correct serial number.	Satisfactory	

Examples

- Set up the frame generator using the CEMONS4.SEQ sequence file, 1 cycle.
- TPDU of the A-NetworkParameter-Read request: 03 DA 00 00 0B 00.
TPDU of the A-NetworkParameter-Response: 03 DB 00 00 0B 00 (+ 6 bytes serial number).
- TPDU of the A-PropertyValue-Read: 03 D5 00 0B 10 01,
TPDU of the A-PropertyValue-Response: 03 D6 00 0B 10 01 (+ 6 bytes serial number).

ID **CEMO/LTS/NS/2**

NAME Checking of rejection of service A_PropertyDescription Read in case of support of Reduced interface object mechanisms

PROP Check that the BDUT rejects the service A_PropertyDescription Read in case of support of Reduced interface object mechanisms

NOTE

CONFIG

PROC Start the observer and the generator.

CONFOR

Result		Y/N
The BDUT ignores any A_PropertyDescription Read services sent to it.	Satisfactory	

Examples

- Set up the frame generator using the CEMONS6.SEQ sequence file, 1 cycle.
- TPDU of the A-PropertyDescriptionRead: 03 D8 00 0B 01, no response is sent by the BDUT.

3.7 CEMO/LTS/FOCI (mandatory if supported)

The objective of the tests presented below is to validate whether

a) the BDUT is able to handle Functions of Common Interest if supported.

b) the BDUT ignores Functions of Common Interest that are not supported.

ID	**CEMO/LTS/FO/1**
NAME	Test of FOCIs = Functions of common interest [if present in the application]
PROP	Check that the BDUT returns values requested via a supported FOCI Group Address or ignores FOCI Group Addresses not supported by it.
NOTE	Create a sequence of Group-Read frames, containing supported and unsupported FOCI Group Addresses.
CONFIG	
PROC	Start up the observer and the generator.

CONFOR

Result		Y/N
When receiving a Group Read on a supported FOCI Group Address, the appropriate value is returned. When receiving a Group Read on an unsupported FOCI Group Address, the BDUT ignores the message.	Satisfactory	

Examples

- Set up the frame generator using the CEMOFO2.SEQ sequence file, 1 cycle.
- In the example given, a read request is sent on Group Address F001 (FOCI date). In the received response the date is coded according to the KNX Datapoint type 11.001

3.8 CEMO/LTS/ECH-PARA

The objective of the tests presented below is to validate the handling of parameters.

ID	**CEMO/LTS/PA/1**
NAME	Channel reading/writing
PROP	Checking of the correct implementation of easy channels and their parameters in devices (as functional blocks). Checking of handling of A_Property_Read and A_Property_Write (if supported) requests to the channel parameters of the BDUT
NOTE	It shall be checked in the supplied documentation (PIXIT), which easy channels are supported by the BDUT.
CONFIG	
PROC	Depending on the supported channels and the channel codes, a) the in-and output Group Objects of the implemented channels shall be checked according to the channel code specifications. Test procedure shall be in accordance with Volume 8/7. b) a number of parameter settings shall be read and (if supported) written. Test procedure shall be in accordance with Volume 8/3/7 § 2.13 and

2.14 (appropriately adapted to connectionless operation)

CONFOR	Result		Y/N
	The channels as implemented in the BDUT are in accordance to the relevant channel code specifications and the behavior of the BDUT is in accordance with the channel code specifications after parameter settings (if supported).	Satisfactory	

Examples

- Set up the frame generator using the CEMOPA2.SEQ sequence file
- TPDU of A_PropertyValue_Write is : 03 D7 00 65 20 01 22 22 ,
- “00” is the n° of “Device Object”,
- “65” is the Property_ID “channel 1 parameter block”,
- “2” from “20” indicates number of elements of the block,
- “1” from “01” indicates start-index,
- 22 22 are the written bytes

4 E-Mode LT supervisor validation

These tests are intended for the testing of LT-Supervised products that act as a supervisor (in some cases it may also act as a normal LT-S product – check PICS and PIXIT).

When acting as a normal LTS-device the device shall be submitted to the tests of a normal LTS-device described in the previous paragraph

When acting as a LTS Supervisor (client), the following additional tests shall be carried out:

- (on open media only) Domain Address assignment
- SNA update through Network Parameter Write
- Distribution of an Individual Address to a device in programming state (mandatory if supported)
- Distribution of an Individual Address to a device through serial number (mandatory if supported)
- Device identification
- Link management

  -Read → response

  -Write → response
- Group address check
- Reduced Interface Objects (mandatory if supported)

  -Serial number read

  -Object type read
- Parameter management (mandatory if supported)

The supplied PICS and PIXIT proforma of a LTS-device shall be consulted to check whether:

- the logical tag supervisor functions are sequentially operated
- the logical tag supervisor functions are automatically operated
- the logical tag supervisor functions are only operated after human action

It is possible that the BDUT only has supervisor functions. In this case only the underneath tests have to be carried out.

Note : The way a supervisor is updated with knowledge of new easy channels, is manufacturer specific and therefore not a part of these test specifications.

As the supervisor needs as any other connected product its own Individual Address, this has to be assigned prior to the tests (the assignment is manufacturer specific, e.g. by coding wheels).

Wiring diagram used during this phase (if not otherwise indicated in the test):

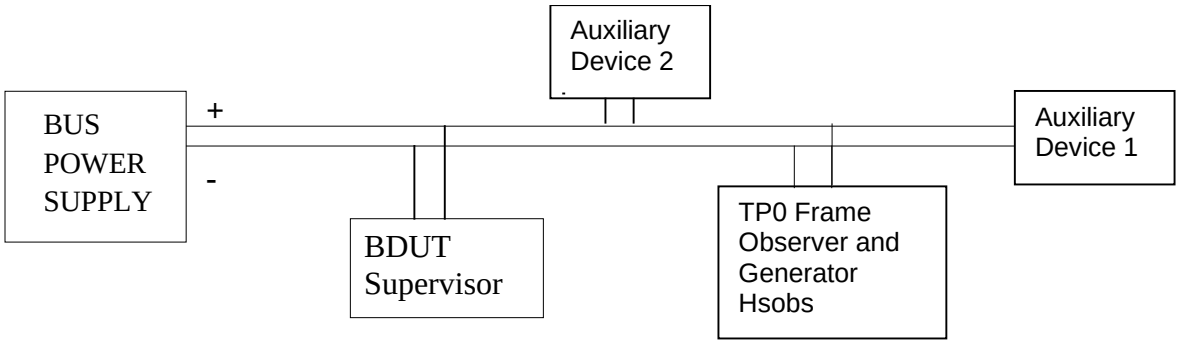

Figure 2 – Wiring diagram

The auxiliary devices are used during some of the tests, as a product to be supervised. When carrying out these tests on an open medium, the following domain address is assumed: e.g. AB82h (or F0F0h).

4.1 CEMO/LTSS/DO

The objective of this test is to validate whether the supervisor complies with the NM_DomainAddress_Read, Write and Scan procedure.

ID	**CEMO/LTSS/DO1**
NAME	Execution of Domain Address Creation Process, Domain Address Distribution Process and Domain Address Acquisition Process
PROP	Attribution of new domain address to a server by the Supervisor (via two mechanisms) and checking the existence of a domain address on the network by the Supervisor
NOTE	It shall be checked in the supplied documentation (PIXIT), how the Domain Address Creation Process, the Domain Address Distribution Process and the Domain Address Acquisition Process can be initiated.
CONFIG	Case A (Creation process - positive): install an auxiliary device with domain address 1234h and Individual Address 1.0.5 Case B (Creation process – negative): remove the above auxiliary device from the network Case C (Acquisition process – positive): reinstall auxiliary device 1 Case D (Distribution process – positive): install auxiliary device 2, which is able to request a domain address with the Supervisor
PROC	• Start up the observer and the generator • CASE A: select at the Supervisor a domain address to be checked (e.g. 1234). Start the DoA Creation Process. Check that the Supervisor initiates a A_DomainAddress_SelectiveRead for the to be checked domain address, indicating an Individual Address from which onward the selective read shall be initiated as well as the check range (e.g. 1.0.0 – range 10). Check that the Supervisor sends this message via broadcast (on DoA 0000h) and that it continues its search on DoA 1235h after it receives a response on DoA 1234. • CASE B: select at the Supervisor a domain address to be checked (e.g. 1234). Start the DoA Creation Process. Check that the Supervisor initiates a

A_DomainAddress_SelectiveRead for the to be checked domain address, indicating an Individual Address from which onward the selective read shall be initiated as well as the check range (e.g. 1.0.0 – range 10). Check that the Supervisor sends this message via broadcast (on DoA 0000h) and that it accepts the selected DoA as a free DoA after the time-out (medium dependant).

- CASE C: select at the Supervisor a domain address to be written (e.g. 1235). Start the DoA Acquisition process. Check that the Supervisor initiates an A_DomainAddress_Write for the set domain address. Check that the Supervisor sends this message via broadcast (on Destination Address and DoA 0000h) and repeats the A_DomainAddrWrite periodically for a maximum time smaller than 10 minutes (check in the PIXIT proforma of the Supervisor).

- CASE D: Initiate at the auxiliary device the A_DomainAddrRead service. Select at the Supervisor a domain address (e.g. 1235) to be written in the auxiliary device requesting a new domain address. Check that the Supervisor initiates an A_DomainAddress_Response for the to be set domain address. Check that the Supervisor sends this message via broadcast (on Destination Address and DoA 0000h).

CONFOR	Result		Y/N
	The Domain Address Management NM Procedures are handled correctly	Satisfactory	

Examples See sequence CEMODOA1.seq

4.2 CEMO/LTSS/SN (mandatory if supported)

The objective of this test is to validate whether the supervisor complies with the NM_SubnetworkAddressWrite procedure .

ID **CEMO/LTSS/SN**

NAME Writing of Subnetwork address

PROP Check whether the Supervisor is able to attribute a new SNA to a device via the NM_SubnetworkAddressWrite procedure

NOTE It shall be checked in the supplied documentation (PIXIT), how the writing of the subnetwork address can be activated (may be disabled by default).

CONFIG The auxiliary device shall have the default SNA address.

PROC

- Start up the observer and the generator.

- Stimulate the supervisor to send an A_Network_Parameter_Write service (IO Type 0, Prop ID 57d, value of new SNA 88h). Check whether the routing counter is set to 0 in the A_Network_Parameter_Write service and the service is sent broadcast

CONFOR	Result		Y/N
	The frame as sent by the Supervisor has the correct format and the correct routing counter	Satisfactory	

Examples

- Set up the frame generator using the CEMOSNS1.SEQ sequence file, 1 cycle. The supervisor receives the new SNA.
- Set up the frame generator using the CEMOSNS2.SEQ sequence file, 1 cycle. The supervisor is the server of the new SNA.
- TPDU : 03 E4 00 00 39 88 (88 is the new SNA value)

4.3 CEMO/LTSS/IA

The objective of this test is to validate whether the supervisor complies with the NM_IndividualAddressWrite and NM_IndividualAddress_SerialNumber_Write.

ID **CEMO/LTSS/IA1**

NAME Writing of Individual Address in programming mode

PROP Setting of Individual Address of devices in programming mode according relevant Network Management Procedure

NOTE It shall be checked in the supplied documentation (PIXIT), how the writing of the Individual Address can be activated (may be disabled by default).

CONFIG Case A (positive): install only one auxiliary device supporting connection oriented communication

Case B (negative): install auxiliary device not supporting connection oriented communication

Case C (negative): install first auxiliary device with Individual Address identical to 1.1.1 and a second auxiliary device with Individual Address not identical to 1.1.1

Case D (positive): install a first auxiliary device with Individual Address 1.1.1

Case E (negative): install first auxiliary device with Individual Address identical to 1.1.1 and a second auxiliary device with the Individual Address 1.1.2 and put both auxiliary devices into programming mode

PROC

- Start up the observer and the generator
- CASE A: Check that the supervisor executes the entire NM_IndividualAddressWrite procedure correctly when writing the Individual Address 1.1.1 (transmission of A_Connect, A_DeviceDescriptor_Read (type0), A_Disconnect, A_IndividualAddress_Read, A_IndividualAddress_Write, A_Connect, A_DeviceDescriptor_Read (type0), A_Restart, A_Disconnect)
- CASE B: (if supported) check that the supervisor aborts the NM_IndividualAddressWrite procedure after receiving a A_Disconnect_ind on a A_Disconnect_req and restarts the operation connectionless point to point.
- CASE C: try to attribute to the second auxiliary device the Individual Address 1.1.1. Check that the supervisor on executing a descriptor type 0 read

(connectionless) establishes that the Individual Address is already assigned in the network and aborts the NM_IndividualAddressWrite procedure

- CASE D: try to attribute to the first auxiliary device the Individual Address 1.1.1, which is already assigned to the device. Check that the supervisor executes the entire NM_InvididualAddressWrite procedure correctly (does not abort it).
- CASE E: try to attribute to the first auxiliary device the Individual Address 1.1.3. Check that the supervisor sends out an A_IndividualAddressRead and aborts the NM_IndividualAddressWrite procedure on receiving two A_IndividualAddress_Responses

CONFOR	Result		Y/N
	The NM_IndividualAddressWrite procedure is handled correctly	Satisfactory	

Examples

- *See CEMOIAS1.seq*

ID	**CEMO/LTSS/IA2**

NAME Writing of Individual Address via serial number

PROP Setting of Individual Address of devices via serial number according relevant Network Management Procedure

NOTE It shall be checked in the supplied documentation (PIXIT), how the writing of the Individual Address can be activated (may be disabled by default).

CONFIG Case A (positive): install an auxiliary device with a known serial number

Case B (negative): install first auxiliary device with Individual Address identical to 1.1.1 and a second auxiliary device with Individual Address not identical to 1.1.1

PROC

- Case A: the Supervisor assigns the Individual Address (e.g. 1.1.1) correctly to the BDUT according to the Network Management Procedure "IndividualAddressSerialNumber_Write and verifies the assigned address via the A_IndividualAddressSerialNumber_Read. Check whether the IndividualAddressSerialNumber_Write is sent broadcast.
- Case B: try to attribute the Individual Address 1.1.1 to the second auxiliary device. Check that the supervisor aborts the IndividualAddressSerialNumber_Write procedure when detecting that the Individual Address 1.1.1 already exists.

CONFOR	Result		Y/N
	The NM_IndividualAddress_SerialNumber_Write is executed correctly	Satisfatory	

Examples

- *See CEMOIAS2.seq*

4.4 CEMO/LTSS/DI

The objective of this test is to validate whether the supervisor correctly verifies the device identification of devices via the DeviceDescriptor-Read request service. It shall be verified in the PICS and PIXIT proforma, which device descriptors (0 to 63) can be read out via the supervisor.

ID	**CEMO/LTSS/DI/1**

NAME Device Descriptor

PROP The Supervisor sends requests upon Device Descriptor types.

NOTE It shall be checked in the supplied documentation (PIXIT) how the reading of device descriptors can be activated (if implemented).

CONFIG

- Case A (positive): install an auxiliary device that reports a device descriptor supported by the Supervisor
- Case B (negative): ensure that the Generator returns a device descriptor not supported by the Supervisor
- Case C (negative): ensure that the Generator returns a device descriptor supported by the Supervisor but with unspecified coding

PROC

- Start up the observer then the generator.
- Case A (positive): the Supervisor sends out a correct A_DeviceDescriptorRead_Service (Individual Address connection oriented or connectionless) requesting a supported device descriptor and correctly handles the (correct) received information
- Case B (negative): the Supervisor sends out a correct A_DeviceDescriptorRead_Service (Individual Address connection oriented or connectionless) requesting a supported device descriptor and ignores a response addressed to it with an unsupported device descriptor.
- Case C (negative): the Supervisor sends out a correct A_DeviceDescriptorRead_Service (Individual Address connection oriented or connectionless) requesting a supported device descriptor and handles correctly any returned supported device descriptor coding which is not specified.

CONFOR

Result		Y/N
The Supervisor executes correctly the A_DeviceDescriptorRead_Service and correctly handles A_DeviceDescriptor_Response_Services.	Satisfactory	

Examples Case A : TPDU sent : . 03 00 (type 0) or 03 02 (type 2),

TPDU received : . 03 40 30 12 (type 0),
or 03 42 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 (type 2).

See sequence CEMODIS1.seq connectionless and CEMODIS2.seq connection oriented, both for case A only.

4.5 CEMO/LTSS/GA

The objective of this test is to validate whether the supervisor correctly handles the NM_GroupAddress_Scan procedure. The NM_GroupAddress_Scan is mandatory for the Group Address space C000h to DFFFh according to the Network Resources (supplement 18).

ID	**CEMO/LTSS/GA**
NAME	Group Address Check
PROP	Verify whether the Supervisor is able to check if a Group Address is in use in the Group Address space C000h to DFFFh
NOTE	It shall be checked in the supplied documentation (PIXIT), how the NM_GroupAddress_Scan procedure can be initiated.
CONFIG	• Case A and B : install an auxiliary device to which Group Address C123h has been assigned
PROC	• Start up the observer and the generator. • Case A (positive): Check that the Supervisor is able to carry out a GroupAddressScan for one Group Address by sending out an A_NetworkParameter_Read.req (IO Type = 1d, Property-ID = 23d, range = 1d, Group Address = C123h) – broadcast. Check whether the Supervisor takes into account that this Group Address is already assigned, when assigning Group Addresses to devices. • Case B (positive) (mandatory if supported): Check that the Supervisor is able to carry out a GroupAddressScan for a range of Group Addresses by sending out an A_NetworkParameter_Read.req (IO Type = 1d, Property-ID = 23d, range = 4d, Group Address = C321h) – broadcast. Check whether the Supervisor waits until the timeout has expired and takes into account that none of the Group Addresses in this range are already assigned, when assigning Group Addresses to devices.

CONFOR

Result		Y/N
The Supervisor complies to the NM_GroupAddress_Scan procedure	Satisfactory	

Examples

Case A :
TPDU of the request (broadcast) is : 03 DA 00 01 17 01 CE 01 (here CE 01 is the GA to check).

TPDU received (Individual Addressing) for same GA : 03 DB 00 01 17 01 CE 01

See sequence CEMOGAS1.seq

4.6 CEMO/LTSS/LM

The objective of this test is to validate whether the supervisor correctly supports the Link Management procedures (adding, deleting and reading links respectively handling of Link_Responses).

ID	**CEMO/LTSS/LM**
NAME	Link Management
PROP	Verify whether the supervisor is able to read, write and delete Group Addresses assigned to Group Objects
NOTE	It shall be checked in the supplied documentation (PIXIT), how the Link Management on the supervisor can be initiated.
CONFIG	- install an auxiliary device 1 supporting GO0 with Group Addresses: C240 (S) GO 255 with Group Address: C241 (S) GO1 with Group Addresses: C250, C251, C252, C253 (S), C254, C255, C256, C257, C258, C259 GO4 with no attributed Group Addresses - install an auxiliary device 2 supporting GO 255 with no attributed Group Addresses For Case D: ensure that the generator is able to generate a A-Link-response with more than 6 Group Addresses For Case E: ensure that the generator is able to generate a A-Link-response on broadcast
PROC	• Start up the observer and the generator. • For all cases, check whether the A-Link messages are sent individual connectionless. In case of A-Link-Read, check that the four bits before the start index are always set to 0 and the start index is always smaller or equal 15. In case of A-Link-Write, check that the 6 bits before the delete flag are always set to zero. • Case A (positive): Stimulate the supervisor to generate a A-Link-Read service on auxiliary device 1 - Group Object 0 with start index 1 and on Group Object 255 with start index 1. Check that the supervisor handles the one returned (sending) Group Address correctly • Case B (positive): Stimulate the supervisor to generate an A-Link-Read service on auxiliary device 1 – Group Object 1 with start index 1. Check that the supervisor handles the six first returned Group Addresses correctly (including the sending one) • Case C (positive): Stimulate the supervisor to generate an A-Link-Read service on auxiliary device 1 – Group Object 1 with start index 7. Check that the supervisor handles the four last returned Group Addresses correctly. • Case D (negative): Stimulate the supervisor to generate an A-Link-Read service on auxiliary device 1 – Group Object 3 with start index 1. Check that the

supervisor rejects a A-Link-Response message with more than 6 Group Addresses.

- Case E (negative): Stimulate the supervisor to generate an A-Link-Read service on auxiliary device 1 – Group Object 3 with start index 1. Check that the supervisor rejects a A-Link-Response message which is sent broadcast
- Case F (negative): Stimulate the supervisor to generate an A-Link-Read service on auxiliary device 1 – Group Object 4 with start index 1. Check that the supervisor properly handles an A-Link-Response with start index 0 and no Group Addresses correctly
- Case G (positive): Stimulate the supervisor to add Group Address C2 70 (sending) to Group Object 4 of Auxiliary device 1. Check that the supervisor handles the response correctly.
- Case H (positive): Stimulate the supervisor to again delete the Group Address C2 70 (sending) of Group Object 4 of auxiliary device 1. Check that the supervisor handles the response correctly.
- Repeat case G and H: stimulate the supervisor to add/delete Group Address C2 70 (sending) to/from Group Object 255 of auxiliary device 2. Check that the supervisor handles the responsecorrectly.

CONFOR	Result		Y/N
	The Supervisor generates the correct A-Link Management services and handles the respective responses correctly.	Satisfactory	

Examples

- See the CEMOLMS1.seq and CEMOLMS2.seq sequence files.
- Case A : TPDU of the request : 03 E5 00 01,
- TPDU of the response: 03 E6 00 51 gg aa (gg aa is the GA),
  5 means that the 5th GA is the sending GA, 1 means that the starting index is equal to 1.

Note : If 00 replaces 51 then it means “negative response”, and no more data.

4.7 CEMO/LTSS/NS (mandatory if supported)

The objective of this test is to validate whether the supervisor correctly supports the Network Management procedures Functional block scan respectively Serial number scan (via the A_NetworkParameter_Read service and a reduced interface object) and handles the responses correctly.

ID	**CEMO/LTSS/NS/1**
NAME	Reduced interface object; serial number scan
PROP	Verify whether the Supervisor is able to carry out a Serial number scan
NOTE	It shall be checked in the supplied documentation (PIXIT), how the Serial number scan can be initiated by the Supervisor. Note: the serial number can be read also by using the A-PropertyValue-Read on property 11d of Device object 0d. This service is mandatory but not checked in this test.

CONFIG Case A: install an auxiliary device 1 with a known serial number

Case B: remove auxiliary device 1

Case C: ensure that the Generator is able to return serial number 000000h when the Supervisor addresses a fictitious auxiliary device 2

PROC

- Start the observer and the generator
- For all cases, check whether the A-NetworkParameter_Read.req are sent broadcast and that the test info is set to 00.
- Case A (positive): Stimulate the supervisor to generate an A-NetworkParameter_Read.req on Auxiliary device 1 – IO type 0d, Prop ID 11d. Check that the supervisor handles the returned serial number correctly
- Case B (negative): Stimulate the supervisor to generate an A-NetworkParameter_Read.req on Auxiliary device 1 – IO type 0d, Prop ID 11d. Check that the supervisor waits until time-out (medium dependant) and assumes that the addressed device is not installed in the network.
- Case C (positive): Stimulate the supervisor to generate an A-NetworkParameter_Read.req on Auxiliary device 2 – IO type 0d, Prop ID 11d. Check that the supervisor handles the returned empty serial number 000000h correctly

CONFOR

Result		Y/N
The supervisor correctly carries out the Serial number scan and handles the responses correctly.	Satisfactory	

Examples

- See the CEMONSS1.SEQ sequence file.
- Case A :
TPDU of the A-NetworkParameter-Read request : 03 DA 00 00 0B 00. (broadcast), response is : 03 DB 00 00 0B 00 (+ 6 bytes).

ID **CEMO/LTSS/NS/2**

NAME Reduced interface object; functional block scan

PROP Verify whether the Supervisor is able to carry out a Functional block scan

NOTE It shall be checked in the supplied documentation (PIXIT), how the Functional block scan on the Supervisor can be initiated.

CONFIG Case A: install an auxiliary device 1 with a reduced interface objects with ID 50d

Case B: remove auxiliary device

PROC

- Start the observer and the generator
- For all cases, check whether the A-NetworkParameter_Read.req are sent

point-to-point connectionless and that the test info is set to 00.

- Case A (positive): Stimulate the supervisor to generate a A-NetworkParameter_Read.req on Auxiliary device 1 – IO type 50d, Prop ID 1d. Check that the supervisor handles the returned response (and its object_index) correctly.
- Case B (negative): Stimulate the supervisor to generate a A-NetworkParameter_Read.req on Auxiliary device 1 – IO type 50d, Prop ID 1d. Check that the supervisor waits until time-out (medium dependant) and assumes that the scanned functionality is not available on the network.

CONFOR	Result		Y/N
	The supervisor correctly carries out the Functional block scan and handles the responses correctly.	Satisfactory	

Examples

- See the CEMONSS2.seq file.

4.8 CEMO/LTSS/ECH-PA

ID **CEMO/LTSS/CH/1**

NAME Channel – connection code – connection rule tests

PROP Verify whether the Supervisor is able to recognize channels of its own application domain (those listed in the PIXIT as supported) and correctly handle channels of other application domains or channels not supported of its own application domain

NOTE It shall be checked in the supplied documentation (PIXIT), for which application domain the supervisor has been designed and which channels it supports.

CONFIG Case A: install an auxiliary device 1 containing channels all belonging to its application domain and supported by the supervisor

Case B (negative): install an auxiliary device 2 containing channels all belonging to another application domain as the one for which it was designed or channels of its own application domain not supported according to the PIXIT.

PROC

- Start the observer and the generator
- Case A: Stimulate the supervisor to read the device descriptor of auxiliary device 1. Check that the supervisor correctly handles the channel information it supports (calculates and assigns the correct Group Addresses in accordance with the read channels and links devices in accordance with the relevant connection rules) – extend of this channel tests still to be defined.
- Case B (negative): Stimulate the supervisor to read the device descriptor of auxiliary device 2. Check that the supervisor either

a) does not react at all

b) signals unknown channels to the installer

c) signals unknown channels to the installer and instructs the installer to assign the

appropriate Group Address and link devices.

CONFOR	Result		Y/N
	The supervisor correctly handles all received channel information as indicated above	Satisfactory	

Examples

- Depends on the application domain supported by the supervisor

ID **CEMO/LTSS/PA (mandatory if supported)**

NAME Reading and Writing of parameters associated to Easy channels by the supervisor (if supported)

PROP Verify whether the Supervisor is able to read and write parameters of easy channels by means of the A_PropertyValue_Read and Write mechanisms

NOTE It shall be checked in the supplied documentation (PIXIT), whether the Supervisor supports reading and writing of Easy channel parameters and which channel codes are supported by the Supervisor.

CONFIG Case A/B (writing - positive): install an auxiliary device 1, which supports an easy channel also supported by the Supervisor. In the underneath example, it is assumed that channel 3 of the auxiliary device supports the Channel Code 102: this channel code (according to the channel code specifications) has two parameters:

- time delay with no bit offset
- pre-warning duration with 8 bit offset

Case C (writing – negative): install a generator, which is able to return a negative A_PropertyValue_Response when the Supervisor tries to access the same parameters as above of the auxiliary device, which is removed from the network.

Case D/E (reading – positive): re-install the same auxiliary device 1

Case F (reading – negative): install a generator, which is able to return a negative A_PropertyValue_Response when the Supervisor tries to access the same parameters as above of the auxiliary device, which is removed from the network.

PROC

- Start the observer and the generator
- For all cases, check whether the A-PropertyValue_Read/Write-services are sent connectionless (or optionally connection-oriented). Verify in case of writing parameter values that the Supervisor always takes into account the specified bit offset (if any).
- Case A: Stimulate the Supervisor to carry out an A_PropertyValue_Write on Obj Type 0 (device object), Prop ID 103 (as this property contains the parameters of channel 3), start index 1, ElementCount=1, data 0Ah (value of parameter = 1 min.). Verify that the Supervisor correctly handles the A_PropertyValue_Response, which is returned by the auxiliary device.
- Case B: Stimulate the Supervisor to carry out an A_PropertyValue_Write on

Obj Type 0 (device object), Prop ID 103 (as this property contains the parameters of channel 3), start index 2, ElementCount=1, data 10h (value of parameter = pre-warning). Verify that the Supervisor correctly handles the A_PropertyValue_Response, which is returned by the auxiliary device.

- Case C: Stimulate the Supervisor to carry out an A_PropertyValue_Write on Obj Type 0 (device object), Prop ID 103, start index 1, ElementCount=1, data 0Ah (value of parameter = 1 min.).Verify that the Supervisor correctly handles the negative A_PropertyValue_Response.

- Case D: Stimulate the Supervisor to carry out an A_PropertyValue_Read on Obj Type 0 (device object), Prop ID 103 (as this property contains the parameters of channel 3), start index 1, ElementCount=1. Verify that the Supervisor correctly handles the A_PropertyValue_Response, which is returned by the auxiliary device.

- Case E: Stimulate the Supervisor to carry out an A_PropertyValue_Read on Obj Type 0 (device object), Prop ID 103 (as this property contains the parameters of channel 3), start index 2, ElementCount=1. Verify that the Supervisor correctly handles the A_PropertyValue_Response, which is returned by the auxiliary device.

- Case F: Stimulate the Supervisor to carry out an A_PropertyValue_Read on Obj Type 0 (device object), Prop ID 103, start index 1, ElementCount=1.Verify that the Supervisor correctly handles the negative A_PropertyValue_Response.

Note : in case of parameter blocks larger than 10 bytes, also check the correct handling of arrays.

CONFOR

Result		Y/N
The Supervisor correctly supports the reading and writing of parameters associated to easy channels and correctly handles the responses	Satisfactory	

Examples

- See the CEMOPAS2.SEQ sequence file.
- TPDU of A_PropertyValue_Write is : 03 D7 00 65 20 01 22 22 ,
- “00” is the n° of “Device Object”,
- “65” is the Property_ID “channel 1 parameter block”,
- “2” from “20” indicates number of elements of the block,
- “1” from “01” indicates start-index,
- 22 22 are the written bytes

5 Appendix for examples

Sequences are for TP0 only.

Note : The part “checksum” of the messages proposed in the sequences is incorrect most of the time. This is so because Hsobs calculates this part of every sent message if needed.

5.1 Description of an «2 PB + 2 LED» application for BIM

This is an example of application used to test a BIM.

The BIM includes the stack software and needs to be programmed with an application, by a programming tool.

Then it has to be connected on an application board supporting the software application.

- This example has two channels each of them managing two objects having at least one link.
- Description of the application:
  - Channels:
    Channel 1 = Top key: one PB function + one LED function (output) [channel code = 0100h]
    Channel 2 = Bottom key: one PB function + one LED function (output) [channel code = 1EFFh]
  - Initial object configurations:
    cc is the connection code. x/y = 1/1.
    N°0: PB, top key, PR = emergency (10),
    Read/Write/Communication/Transmit/Update: Enable, 1Bit, Group Address: [80cch].
    N°1: LED, top key, PR = Normal (11), Read/Write/Communication/Transmit/Update: Enable, 1Bit, Group Address: [82cch].
    N°2: BP, bottom key, PR = High (01), Read/Write/Communication/Transmit/Update: Enable, 1Bit, Group Address: [81cch].
    N°3: LED, bottom key, PR = Normal (11),
    Read/Write/Communication/Transmit/Update: Enable, 1Bit, Group Address: [83cch].
    N°4: FOCI received, Group Address: [F001]. Date.
  - Parameters :
    For channel 1, two values : 11, 12.
    For channel 2, four values : 21, 22, 23, 24.

5.2 Examples of sequence files for HSOBS (LT Reflex device)

In the sequences the printed “IN” and “OUT” show what is sent and expected to be seen by the tool. **CEMOAD1.SEQ**

```
TP0
# LT Reflex
# 2BP-2LED application, x/y = 8/4
# Test 1
# Modification of LED status, addressed to Group Address 9F01h
# IN, LED on (Service A-GroupValue-Write)
3000,i,CD 91 9F 01 01 45 00 81 0A 66
# Expected : LED switches on
# IN, verify the previous write (Service A-GroupValue-read sent by another device)
1000,i,CD 91 9F 01 01 B7 00 00 BB 2E
# OUT, Expected Service A-GroupValue-Response: CD 91 9F 01 01 FF 00 41 0B 77
```

```
# LED off
6000,i,CD 91 9F 01 01 45 00 80 0A 65
# Expected : LED switches off
```

CEMOAD2.SEQ

```
TP0
# LT Reflex, connection-code = 01
# 2BP-2LED application, x/y = 16/1
# IN, LED on (Service A-GroupValue-Write)
3000,i,CD 91 BC 01 01 45 00 81 0A 66
# Expected : no action
# IN, (Service A-GroupValue-read sent by another device)
1000,i,CD 91 BC 01 01 B7 00 00 BB 2E
# OUT, Expected : no response
# 2BP-2LED application, x/y = 16/2
# IN, LED on (Service A-GroupValue-Write)
3000,i,CD 91 BD 01 01 45 00 81 0A 66
# Expected : no action
# IN, (Service A-GroupValue-read sent by another device)
1000,i,CD 91 BD 01 01 B7 00 00 BB 2E
# OUT, Expected : no response
# x/y = 16/3
# IN, LED on (Service A-GroupValue-Write)
3000,i,CD 91 BE 01 01 45 00 81 0A 66
# Expected : LED switches on
# IN, verify the previous write (Service A-GroupValue-read sent by another
device)
1000,i,CD 91 BE 01 01 B7 00 00 BB 2E
# OUT, Expected Service A-GroupValue-Response: CD 91 9F 01 01 FF 00 41 0B 77
# LED off
6000,i,CD 91 BE 01 01 45 00 80 0A 65
# Expected : LED switches off
# x/y = 16/4
# IN, LED on (Service A-GroupValue-Write)
3000,i,CD 91 BF 01 01 45 00 81 0A 66
# Expected : no action
# IN, (Service A-GroupValue-read sent by another device)
1000,i,CD 91 BF 01 01 B7 00 00 BB 2E
# OUT, Expected : no response
```

CEMOSN1.SEQ

```
TP0
# Individual address : 01 FF in reflex mode
# IN, Test of service A_NetworkParameter_Write SNA “88” from 01 45 to all 00 00
12000,i,CD 95 00 00 01 45 03 E4 00 00 39 88 0A 66
# Make the product send a frame to check that it uses the SNA value “88”. Press BP
# OUT, Expected A-GroupValue-Write: CD 91 9F 01 88 FF 00 81 0B 77 (reflex mode)
# Test of service A_NetworkParameter_Write SNA “01” from 01 45 to all 00 00
1000,i,CD 95 00 00 01 45 03 E4 00 00 39 01 0A 66
```

CEMODI1.SEQ

```
TP0
# Device Identification for LT reflex
# DeviceDescriptor-Read service addressed to Individual Address 01 FF
# IN, Descriptor type 0 = Mask version read request
1000,i,CB 91 01 FF 01 45 03 00 0A 65
# OUT, Expected A- DeviceDescriptor-Response: 3012 is the mask for “TP0-BIM”
#      CB 93 01 45 01 FF 03 40 30 12 0B 77
# IN, Descriptor type 2 = Identification
1000,i,CB 91 01 FF 01 45 03 02 0A 65
# OUT, Expected service A- DeviceDescriptor-Response:
#      CB 9F 01 45 01 FF 03 42 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 0A 65
# IN, Descriptor type 1
1000,i,CB 91 01 FF 01 45 03 01 0A 65
# OUT, Expected “error”: CB 91 01 45 01 FF 03 3F 0A 65 for the rest of the test
# Descriptor type 3
1000,i,CB 91 01 FF 01 45 03 03 0A 65
# Descriptor type 4
1000,i,CB 91 01 FF 01 45 03 04 0A 65
# Descriptor type 5
1000,i,CB 91 01 FF 01 45 03 05 0A 65
# Descriptor type 6
1000,i,CB 91 01 FF 01 45 03 06 0A 65
# Descriptor type 7
1000,i,CB 91 01 FF 01 45 03 07 0A 65
# Descriptor type 8
1000,i,CB 91 01 FF 01 45 03 08 0A 65
# Descriptor type 33
1000,i,CB 91 01 FF 01 45 03 33 0A 65
```

CEMODI2.SEQ

```
TP0
# Device Identification for LT reflex
# DeviceDescriptor-Read service addressed to Group Address 9F 01
# IN, Descriptor type 0 = Mask version read request
1000,i,CD 91 9F 01 01 45 03 00 0A 65
# OUT, NO ANSWER expected for the whole test
# Descriptor type 2 = Identification
1000,i,CD 91 9F 01 01 45 03 02 0A 65
# Descriptor type 1
1000,i,CD 91 9F 01 01 45 03 01 0A 65
# No answer expected
# Descriptor type 3
1000,i,CD 91 9F 01 01 45 03 03 0A 65
# Descriptor type 4
1000,i,CD 91 9F 01 01 45 03 04 0A 65
# Descriptor type 5
1000,i,CD 91 9F 01 01 45 03 05 0A 65
# Descriptor type 6
1000,i,CD 91 9F 01 01 45 03 06 0A 65
# Descriptor type 7
1000,i,CD 91 9F 01 01 45 03 07 0A 65
# Descriptor type 8
1000,i,CD 91 9F 01 01 45 03 08 0A 65
# Descriptor type 33
1000,i,CD 91 9F 01 01 45 03 33 0A 65
```

CEMODI5.SEQ

```
TP0
# Connection mode, for LT reflex
# IN, Connection layer 4
1000,i,CB 91 01 FF 01 45 60 80 0A 65
# IN, Descriptor type 0 = Mask version read request and TPCI “Numbered Data
Packet”
1000,i,CB 91 01 FF 01 45 43 00 0A 65
# OUT, Expected A- DeviceDescriptor-Response: CB 93 01 45 01 FF 43 40 30 12 0B
77
# IN, Descriptor type 2
1000,i,CB 91 01 FF 01 45 47 02 0A 65
# OUT, Expected service A- DeviceDescriptor-Response:
#      CB 9F 01 45 01 FF 47 42 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 0A 65
#
# Test with Group Address 9F 01
# IN, Descriptor type 0 = Mask version read request
1000,i,CD 91 9F 01 01 45 4B 00 0A 65
# OUT, No answer expected
# IN, Descriptor type 2
```

```
1000,i,CD 91 9F 01 01 45 4F 02 0A 65
# OUT, No answer expected
# IN, Disconnection layer 4
1000,i,CB 91 01 FF 01 45 60 81 0A 65
```

CEMOGA1.SEQ

```
TP0
#GA check for LT reflex
# IN, Request to all if the GA 9F 01 is used by anybody
1000,i,CD 97 00 00 01 BC 03 DA 00 01 17 01 9F 01 0A 65
# OUT, Expected service A_NetworkScan_Response:
#      CB 97 01 BC 01 FF 03 DB 00 01 17 01 9F 01 0A 65
# IN, Request to the product (Individual Address 01 FF) if it uses the GA 9F
01
1000,i,CB 97 01 FF 01 BC 03 DA 00 01 17 01 9F 01 0A 65
# OUT, Expected service A_NetworkScan_Response:
#      CB 97 01 BC 01 FF 03 DB 00 01 17 01 9F 01 0A 65
#
# IN, Request to all if the GA 66 77 is used by anybody (but nobody uses it)
1000,i,CD 97 00 00 01 BC 03 DA 00 01 17 01 66 77 0A 65
# OUT, No answer expected
# IN, Request to the product (address 01 54) if it uses the GA 66 77
1000,i,CB 97 01 54 01 BC 03 DA 00 01 17 01 66 77 0A 65
# OUT, No answer expected
```

CEMOWN1.SEQ

```
TP0
# Wildcard service
# Group 9F01 write ON (group according to logical tag of the BDUT)
5000,i,CD 91 9F 01 01 FF 00 81 0A 65
# Expected LED switches on
# Group 9F01 write OFF
3000,i,CD 91 9F 01 01 FF 00 80 0A 65
# Expected LED switches off
#
# Group write ON with wildcard (BE01 tested)
5000,i,CD 91 BE 01 01 FF 00 81 0A 65
# Expected LED switches on
# Group write OFF with wildcard
3000,i,CD 91 BE 01 01 FF 00 80 0A 65
# Expected LED switches off
```

CEMOWN2.SEQ

```
TP0
# Wildcard service (9F 01 and BE 01)
# Group write ON (adapt the Group Address to the product under test)
5000,i,CD 91 9F 01 01 45 00 81 0A 65
# No answer expected
# Group write OFF
3000,i,CD 91 9F 01 01 45 00 80 0A 65
# No answer expected
#
# Group write ON with wildcard
5000,i,CD 91 BE 01 01 45 00 81 0A 65
# Expected LED switches on
# Group write OFF with wildcard
3000,i,CD 91 BE 01 01 45 00 80 0A 65
# Expected LED switches off
# Expected service A-GroupValue-Write after pressing a BP:
#      CD 91 BE 02 BE 01 00 81 0B 77
```

CEMOWN3.SEQ

```
TP0
# Wildcard service (Group Address BE 01 according to logical tag, and 9F01
tested)
# Group 9F01 write ON
5000,i,CD 91 9F 01 01 FF 00 81 0A 65
# Expected LED switches on
# Group 9F01 write OFF
3000,i,CD 91 9F 01 01 FF 00 80 0A 65
# Expected LED switches off
#
# Group write ON with wildcard
5000,i,CD 91 BE 01 01 FF 00 81 0A 65
# Expected LED switches on
# Group write OFF with wildcard
3000,i,CD 91 BE 01 01 FF 00 80 0A 65
# Expected LED switches off
```

CEMOCH1.SEQ

```
TP0
# LT reflex
# Channel numbers 1 and 2 (GA = 9E 01 and 9F01)
# verify services A-groupvalue-read, A-groupvalue-write
# IN, Value read request, channel 1
1000,i,CD 91 9E 01 01 45 00 00 0A 65
# OUT, Expected Service A-GroupValue-Response: CD 91 9E 01 01 FF 00 41 (or 40)
0B 77
# IN, Value read request, channel 2
```

```
1000,i,CD 91 9F 01 01 45 00 00 0A 65
# OUT, Expected Service A-GroupValue-Response: CD 91 9F 01 01 FF 00 41 (or 40)
0B 77
# IN, Group write ON, channel 1
5000,i,CD 91 9E 01 01 45 00 81 0A 65
# OUT, Expected LED switches on
# IN, Group write OFF, channel 1
3000,i,CD 91 9E 01 01 FF 00 80 0A 65
# Expected LED switches off
# IN, Group write ON, channel 2
5000,i,CD 91 9F 01 01 43 00 81 0A 65
# Expected LED switches on
# IN, Group write OFF, channel 2
3000,i,CD 91 9F 01 01 43 00 80 0A 65
# Expected LED switches off
# IN, Descriptor type 2, Individual Address of the BDUT
1000,i,CB 91 01 FF 01 45 03 02 0A 65
# OUT, Expected service A- DeviceDescriptor-Response:
#      CB 9F 01 45 01 FF 03 42 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 0A 65
```

CEMOLM1.SEQ

```
TP0
# Link Management A_Link_Read
# [Group Object n° and start-index n° are tried in sequence]
# (responses depend on the application :
#  n° of objects, n° of GA attached per Group Object)
#
# IN, Link read request start-index to check
1000,i,CB 93 01 FF 01 89 03 E5 00 01 0A 65
# OUT, Expected A_Link_Response CB 95 01 89 01 FF 03 E6 00 01 (or 11) .. .. 0A
65
# or OUT, If no association with the Group Object expected A_Link_Response:
#      CB 93 01 89 01 FF 03 E6 00 00 0A 65 (start-index at 0)
# Responses depend upon the pixit of the BDUT
1000,i,CB 93 01 FF 01 89 03 E5 01 01 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 02 01 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 00 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 01 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 02 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 03 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 04 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 05 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 06 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 07 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 08 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 09 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 0A 0A 65
```

```
1000,i,CB 93 01 FF 01 89 03 E5 01 0B 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 0C 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 0D 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 0E 0A 65
1000,i,CB 93 01 FF 01 89 03 E5 01 0F 0A 65
```

CEMOLM2.SEQ

```
TP0
# A_Link_Write service for LT supervised
# For LT reflex Individual Address is 01 FF
#
# IN, Write "C024" in the product to the object 1
1000,i,D9 95 01 FF 10 89 03 E7 01 01 C0 24 08 B3
# OUT, Expected service A_Link_Response : D9 95 10 89 01 FF 03 E6 01 01 C0 24
08 B3
# IN, verify it answers to this new Group Address (ON, then OFF)
1000,i,DD 91 C0 24 10 89 00 80 0B 81
# Expected LED to switch on
1000,i,DD 91 C0 24 10 89 00 81 0B 80
# Expected LED to switch on
# IN, Delete C024 from the product
1000,i,D9 95 01 FF 10 89 03 E7 01 03 C0 24 08 B3
# OUT, Expected service A_Link_Response : D9 93 10 89 01 FF 03 E6 01 00 08 B3
# IN, Verify that it does not react (on)
1000,i,DD 91 C0 24 10 89 00 80 0B 81
# OUT, No answer expected, No expected action on the LED
# IN, Verify that it does not react (off)
1000,i,DD 91 C0 24 10 89 00 81 0B 80
# OUT, No answer expected, No expected action on the LED
# IN, Verify that another group if present (here C016) is still working (ON)
1000,i,DD 91 C0 16 10 89 00 80 0B 81
# Expected LED switches off
# IN, Verify that another group if present (here C016) is still working (OFF)
1000,i,DD 91 C0 16 10 89 00 81 0B 80
# Expected LED switches on
```

CEMOLM3.SEQ

```
TP0
# A_Link_Write service and s flag for sending address
# For LT reflex Individual Address is 01 FF
#
# IN, Write GAs (at least 10) in the product
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 24 08 B3
# OUT, Expected service A_Link_Response : D9 95 02 89 01 FF 03 E6 01 01 C0 24
08 B3
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 25 08 B3
# Expected service A_Link_Response : D9 97 02 89 01 FF 03 E6 01 01 C0 24 C0 25
08 B3
```

```
# IN, Write first GA with s flag set to 1
1000,i,D9 95 01 FF 02 89 03 E7 01 01 C0 26 08 B3
# OUT, Expected : D9 99 02 89 01 FF 03 E6 01 31 C0 24 C0 25 C0 26 08 B3
# IN, Write another GA with s flag set to 1
1000,i,D9 95 01 FF 02 89 03 E7 01 01 C0 27 08 B3
# Verify that the “sending” group is now the last one:
# OUT, Expected : D9 9B 02 89 01 FF 03 E6 01 41 C0 24 C0 25 C0 26 C0 27 08 B3
# IN, other GAs
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 28 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 29 08 B3
# IN, Write a wrong frame containing two GA
1000,i,D9 97 01 FF 02 89 03 E7 01 00 C0 30 C0 31 08 B3
# OUT, Expected : this frame is ignored
# IN, Write other GA (s flag to 0)
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 32 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 33 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 34 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 35 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 36 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 37 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 38 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 01 00 C0 39 08 B3
# 13 GA are already written (Maybe it was not possible to put all in the BDUT)
# IN, Verify that it answers to many of the new GA
# Expected LED switches off then on
1000,i,DD 91 C0 24 01 78 00 80 0B 81
1000,i,DD 91 C0 24 01 78 00 81 0B 80
1000,i,DD 91 C0 25 01 78 00 80 0B 81
1000,i,DD 91 C0 25 01 78 00 81 0B 80
1000,i,DD 91 C0 26 01 78 00 80 0B 81
1000,i,DD 91 C0 26 01 78 00 81 0B 80
1000,i,DD 91 C0 28 01 78 00 80 0B 81
1000,i,DD 91 C0 28 01 78 00 81 0B 80
1000,i,DD 91 C0 33 01 78 00 80 0B 81
1000,i,DD 91 C0 33 01 78 00 81 0B 80
1000,i,DD 91 C0 39 01 78 00 80 0B 81
1000,i,DD 91 C0 39 01 78 00 81 0B 80
#
# The test may be divided in two parts. Here begins the second part.
# IN, Try to read the Group Addresses at index 1
3000,i,D9 93 01 FF 02 89 03 E5 00 01 08 B3
# OUT, Expected service A_Link_Response :
#      D9 9B 02 89 01 FF 03 E6 01 41 C0 24 C0 25 C0 26 C0 27 08 B3
# IN, Try to read the GA at index 4, service A_Link_Read
3000,i,D9 93 01 FF 02 89 03 E5 00 04 08 B3
# OUT, Expected service A_Link_Response : D9 95 02 89 01 FF 03 E6 01 44 C0 27
08 B3
```

```
# IN, Delete the GA C024 from the product
1000,i,D9 95 01 FF 02 89 03 E7 00 02 C0 24 08 B3
# OUT, Expected : D9 9B 02 89 01 FF 03 E6 01 31 C0 25 C0 26 C0 27 C0 28 08 B3
# IN, Delete the GA C028 from the product
1000,i,D9 95 01 FF 02 89 03 E7 00 02 C0 28 08 B3
# OUT, Expected : D9 99 02 89 01 FF 03 E6 01 31 C0 25 C0 26 C0 27 08 B3
# IN, Delete other GA
1000,i,D9 95 01 FF 02 89 03 E7 00 02 C0 25 08 B3
# OUT, Expected : D9 97 02 89 01 FF 03 E6 01 21 C0 26 C0 27 08 B3
# IN, Delete other GA
1000,i,D9 95 01 FF 02 89 03 E7 00 02 C0 26 08 B3
# OUT, Expected : D9 95 02 89 01 FF 03 E6 01 11 C0 27 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 00 02 C0 29 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 00 02 C0 30 08 B3
1000,i,D9 95 01 FF 02 89 03 E7 00 02 C0 31 08 B3
# Verify that it is deleted (no answers, no actions expected)
1000,i,DD 91 C0 24 01 78 00 80 0B 81
1000,i,DD 91 C0 24 01 78 00 81 0B 80
# Verify that other groups are still working (LED switches OFF, then ON)
1000,i,DD 91 C0 27 01 78 00 80 0B 81
1000,i,DD 91 C0 27 01 78 00 81 0B 80
1000,i,DD 91 C0 32 01 78 00 80 0B 81
1000,i,DD 91 C0 32 01 78 00 81 0B 80
1000,i,DD 91 C0 39 01 78 00 80 0B 81
1000,i,DD 91 C0 39 01 78 00 81 0B 80
```

CEMOLM4.SEQ

```
TP0
# Link Management Write after a A-Memory-Write
# LT reflex
# L4 Ack are not said expected, but they may be present in responses (TPCI=C2)
#
# IN, Link_read request start-index
1000,i,CB 93 01 FF 01 89 03 E5 01 01 0A 65
# OUT, Expected : CB 95 01 89 01 FF 03 E6 01 11 C0 27 08 B3
# IN, Connection
1000,i,CB 90 01 FF 01 89 80 B2 64
# IN, A-Memory-Read of 4 bytes, from address 0139h
3000,i,CB 93 01 FF 01 89 02 04 01 39 0A 65
# OUT, Expected A_Memory_Response: CB 97 01 89 01 FF 02 44 01 39 AA BB CC DD
08 B3
# IN, A-Memory-Write of 2 bytes (55 and AA)
1000,i,CB 95 01 FF 01 89 02 82 01 39 55 AA 0A 65
# IN, Disconnection
1000,i,CB 90 01 FF 01 89 81 B2 64
#
# IN, Link_write
```

```
1000,i,CB 95 01 FF 01 CC 03 E7 01 01 C0 24 0A 65
# OUT, No answer expected
# IN, Link_read request start-index
1000,i,CB 93 01 FF 01 89 03 E5 01 01 0A 65
# OUT, No answer expected
# Try to push the button and see the new GA on/off (it will not work)
# IN, Verify that it is deleted (no answers, no actions expected [on])
500,i,CD 91 C0 24 01 CC 00 81 0A 65
# OUT, No answer expected
# IN, Verify that it is deleted (no answers, no actions expected [off])
500,i,CD 91 C0 24 01 CC 00 80 0A 65
# OUT, No answer expected
```

CEMONS1.SEQ

```
TP0
# LT reflex
# Reduced interface object with A_NetworkParameter_Read
# IN, Serial number read request (request coming from 01 AA to 01 FF) (APCI =
03 DA)
1000,i,CB 95 01 FF 01 AA 03 DA 00 00 0B 00 0A 65
# OUT, As this service is optional, maybe no response
# or OUT, Expected service A_NetworkParameter_Response:
#      CB 95 01 AA 01 FF 03 DB 00 00 0B (then serial number on 6 bytes) 0A 65
# A_PropertyValue_Read request (on serial number)
# IN
1000,i,CB 95 01 FF 01 AA 03 D5 00 0B 60 01 0A 65
# OUT, Expected response is :
#      CB 9B 01 AA 01 FF 03 D6 00 0B 60 01 (then serial number on 6 bytes) 0A
65
#
```

CEMONS2.SEQ

```
TP0
# LT reflex
# Reduced interface object with A_NetworkParameter_Read
# IN, Object Type read request (request coming from 01 AA to 01 FF) (APCI = 03
DA)
1000,i,CB 95 01 FF 01 AA 03 DA 00 00 01 00 0A 65
# OUT, As this service is optional, maybe no response
# or OUT, Expected service A_NetworkParameter_Response:
#      CB 95 01 AA 01 FF 03 DB 00 00 01 (then object type on 2 bytes) 0A 65
# A_PropertyValue_Read request of object type
# IN (APCI = 03 D5)
1000,i,CB 95 01 FF 01 AA 03 D5 00 01 20 01 0A 65
# OUT, Expected response is :
#      CB 9B 01 AA 01 FF 03 D6 00 01 20 01 (then object type on 2 bytes) 0A 65
```

CEMONS3.SEQ

```
TP0
# LT reflex
# Reduced interface object is supported, test of A_PropertyDescription_Read
# IN, read request (request coming from 01 AA to 01 FF) (APCI = 03 D8)
1000,i,CB 95 01 FF 01 AA 03 D8 00 0B 01 0A 65
# OUT, no response expected
```

CEMOFO1.SEQ

```
TP0
# FOCIs
# First Foci tried = F0 01
1000,i,CD 95 F0 01 01 5A 64 00 80 12 34 56 0A 65
# Expected answer depends on the FOCI, see also pixit
# First Foci tried = F0 01, new value
1000,i,CD 95 F0 01 01 5A 64 00 80 00 00 00 0A 65
# Expected answer depends on the FOCI, see also pixit
# Second Foci tried = F4 03
1000,i,CD 91 F4 03 01 5A 60 80 0A 65
# No response expected from this unknown FOCI
```

CEMOPA1.SEQ

```
TP0
# LT reflex
#
# Parameters, preliminary = read size of blocks
# IN, PropertyDescriptor size read request on property “Channel parameter”
channel 1
1000,i,CB 94 01 FF 01 AA 03 D8 00 65 00 0A 65
# OUT, Expected A_PropertyDescription_Response:
#      CB 97 01 AA 01 FF 03 D9 00 65 00 03 02 00 0A 65
# IN, PropertyDescriptor size read request on property “Channel parameter”
channel 2
1000,i,CB 94 01 FF 01 AA 03 D8 00 66 00 0A 65
# OUT, Expected A_PropertyDescription_Response:
#      CB 97 01 AA 01 FF 03 D9 00 66 00 03 04 00 0A 65
#
# Parameters, first part = read
# IN, Device Object read request on property “Channel parameter” channel 1
1000,i,CB 95 01 FF 01 AA 03 D5 00 65 20 01 0A 65
# OUT, Expected:CB 97 01 AA 01 FF 03 D6 00 65 20 01 (+ the block “11 12”) 0A
65
# IN, Device Object read request on property “Channel parameter” channel 2
1000,i,CB 95 01 FF 01 AA 03 D5 00 66 40 01 0A 65
# OUT, Expected:CB 99 01 AA 01 FF 03 D6 00 65 40 01 (+ the block “21 22 23
24”) 0A 65
#
# Second part = Write
```

```
# IN, Device Object write property “Channel parameter” channel 1, 2 bytes
1000,i,CB 97 01 FF 01 AA 03 D7 00 65 20 01 22 22 0A 65
# OUT, Expected:CB 97 01 AA 01 FF 03 D6 00 65 20 01 22 22 0A 65
# IN, Device Object write property “Channel parameter” channel 2, 4 bytes
1000,i,CB 99 01 FF 01 AA 03 D7 00 66 40 01 33 33 44 44 0A 65
# OUT, Expected:CB 99 01 AA 01 FF 03 D6 00 65 40 01 33 33 44 44 0A 65
#
# Third part = Partial Write
# IN, Device Object write property “Channel parameter” channel 2, 3 bytes
1000,i,CB 98 01 FF 01 AA 03 D7 00 66 30 01 55 66 77 0A 65
# OUT, Expected:CB 99 01 AA 01 FF 03 D6 00 65 40 01 55 66 77 44 0A 65
```

5.3 Examples of sequence files for HSOBS (LT Supervised device)

In the sequences the printed “IN” and “OUT” show what is sent and expected to be seen by the tool.

CEMOSN2.SEQ

```
TP0
# Individual address : 01 78 in supervised mode
# IN, Test of service A_NetworkParameter_Write SNA “88” from 01 45 to all 00
00
12000,i,CD 95 00 00 01 45 03 E4 00 00 39 88 0A 66
# Make the product send a frame to check that it uses the SNA value “88”.
Press BP
# OUT, Expected A-GroupValue-Write: CD 91 9F 01 88 78 00 81 0B 77
# Test of service A_NetworkParameter_Write SNA “01” from 01 45 to all 00 00
1000,i,CD 95 00 00 01 45 03 E4 00 00 39 01 0A 66
```

CEMODI3.SEQ

```
TP0
# Device Identification for LT supervised
# DeviceDescriptor-Read service addressed to Individual Address 01 78
(x/y=6/7)
# IN, Descriptor type 0 = Mask version read request
1000,i,CB 91 01 78 01 45 03 00 0A 65
# OUT, Expected A- DeviceDescriptor-Response: 3012 is the mask for “TP0-BIM”
#      CB 93 01 45 01 78 03 40 30 12 0B 77
# IN, Descriptor type 2 = Identification
1000,i,CB 91 01 78 01 45 03 02 0A 65
# OUT, Expected service A- DeviceDescriptor-Response:
#      CB 9F 01 45 01 78 03 42 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 0A 65
# IN, Descriptor type 1
1000,i,CB 91 01 78 01 45 03 01 0A 65
# OUT, Expected “error”: CB 91 01 45 01 78 03 3F 0A 65 for the rest of the
test
# Descriptor type 3
1000,i,CB 91 01 78 01 45 03 03 0A 65
# Descriptor type 4
1000,i,CB 91 01 78 01 45 03 04 0A 65
```

```
# Descriptor type 5
1000,i,CB 91 01 78 01 45 03 05 0A 65
# Descriptor type 6
1000,i,CB 91 01 78 01 45 03 06 0A 65
# Descriptor type 7
1000,i,CB 91 01 78 01 45 03 07 0A 65
# Descriptor type 8
1000,i,CB 91 01 78 01 45 03 08 0A 65
# Descriptor type 33
1000,i,CB 91 01 78 01 45 03 33 0A 65
```

CEMODI4.SEQ

```
TP0
# Device Identification for LT supervised
# DeviceDescriptor-Read service addressed to Group Address 9F 01
# IN, Descriptor type 0 = Mask version read request
1000,i,CD 91 9F 01 01 45 03 00 0A 65
# OUT, NO ANSWER expected for this whole test
# Descriptor type 2 = Identification
1000,i,CD 91 9F 01 01 45 03 02 0A 65
# Descriptor type 1
1000,i,CD 91 9F 01 01 45 03 01 0A 65
# Descriptor type 3
1000,i,CD 91 9F 01 01 45 03 03 0A 65
# Descriptor type 4
1000,i,CD 91 9F 01 01 45 03 04 0A 65
# Descriptor type 5
1000,i,CD 91 9F 01 01 45 03 05 0A 65
# Descriptor type 6
1000,i,CD 91 9F 01 01 45 03 06 0A 65
# Descriptor type 7
1000,i,CD 91 9F 01 01 45 03 07 0A 65
# Descriptor type 8
1000,i,CD 91 9F 01 01 45 03 08 0A 65
# Descriptor type 33
1000,i,CD 91 9F 01 01 45 03 33 0A 65
# Descriptor type 2 = Identification verification
1000,i,CD 91 9F 01 01 45 03 02 0A 65
```

CEMODI6.SEQ

```
TP0
# Connection mode, for LT supervised Individual Address 01 78
# IN, Connection layer 4
1000,i,CB 91 01 78 01 45 60 80 0A 65
# IN, Descriptor type 0 = Mask version read request
1000,i,CB 91 01 78 01 45 43 00 0A 65
```

```
# OUT, Expected A- DeviceDescriptor-Response: CB 93 01 45 01 78 43 40 30 12 0B
77
# IN, Descriptor type 2
1000,i,CB 91 01 78 01 45 47 02 0A 65
# OUT, Expected service A- DeviceDescriptor-Response:
#      CB 9F 01 45 01 78 47 42 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 0A 65
#
# Test with Group Address 9F 01
# IN, Descriptor type 0 = Mask version read request
1000,i,CD 91 9F 01 01 45 4B 00 0A 65
# OUT, No answer expected
# IN, Descriptor type 2
1000,i,CD 91 9F 01 01 45 4F 02 0A 65
# OUT, No answer expected
# IN, Disconnection layer 4
1000,i,CB 91 01 78 01 45 60 81 0A 65
```

CEMOGA2.SEQ

```
TP0
#GA check for LT supervised
# IN, Request to all if the GA 9F 01 is used by anybody
1000,i,CD 97 00 00 01 BC 03 DA 00 01 17 01 9F 01 0A 65
# OUT, Expected service A_NetworkScan_Response:
#      CB 97 01 BC 01 78 03 DB 00 01 17 01 9F 01 0A 65
# IN, Request to the product (Individual Address 01 78) if it uses the GA 87
65
1000,i,CB 97 01 78 01 BC 03 DA 00 01 17 01 9F 01 0A 65
# OUT, Expected service A_NetworkScan_Response:
#      CB 97 01 BC 01 78 03 DB 00 01 17 01 9F 01 0A 65
#
# IN, Request to all if the GA 66 77 is used by anybody (but nobody uses it)
1000,i,CD 97 00 00 01 BC 03 DA 00 01 17 01 66 77 0A 65
# OUT, No answer expected
# IN, Request to the product (address 01 78) if it uses the GA 66 77
1000,i,CB 97 01 78 01 BC 03 DA 00 01 17 01 66 77 0A 65
# OUT, No answer expected
```

- TPDU of the response:

CASE B2: 03 E6 00 12 C2 31

CASE C: 03 E6 01 15 C2 44 C2 45 C2 46 C2 47 C2 48 C2 49

CASE D: 03 E6 02 00

CASE E: 03 E6 03 1F C2 6F C2 70 C2 71 C2 72 C2 74 C2 75

CEMOLMD1.SEQ

```
TP0
# Link Management A_Link_Read
# For LT supervised, x/y = 8/9
#Case A
```

```
1000,i,CB 93 01 78 01 89 03 E5 04 01 0A 65
# expected response:
#      CB 95 01 89 01 78 03 E6 04 00 0A 65
#Case B1
1000,i,CB 93 01 78 01 89 03 E5 00 01 0A 65
# expected response:
#      CB 95 01 89 01 78 03 E6 00 11 C2 30 C2 31 0A 65
#Case B2
1000,i,CB 93 01 78 01 89 03 E5 00 02 0A 65
# expected response:
#      CB 95 01 89 01 78 03 E6 00 12 C2 31 0A 65
#Case C
1000,i,CB 93 01 78 01 89 03 E5 01 05 0A 65
# expected response:
#      CB 95 01 89 01 78 03 E6 01 15 C2 44 C2 45 C2 46 C2 47 C2 48 C2 49 0A 65
#Case D
1000,i,CB 93 01 78 01 89 03 E5 02 03 0A 65
# expected response:
#      CB 95 01 89 01 78 03 E6 02 00 0A 65
#Case E
1000,i,CB 93 01 78 01 89 03 E5 03 0F 0A 65
# expected response:
#      CB 95 01 89 01 78 03 E6 03 1F C2 6F C2 70 C2 71 C2 72 C2 74 C2 75 0A 65
The following can also be tried (combinations of start index)
# (not described in the text of the suite of tests)
# [Group Object n° and start-index n° are tried in sequence]
# expected response If no association with the Group Object A_Link_Response:
#      CB 93 01 89 01 78 03 E6 00 00 0A 65 (start-index at 0)
# Responses depend upon the pixit of the BDUT
1000,i,CB 93 01 78 01 89 03 E5 01 00 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 01 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 02 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 03 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 04 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 05 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 06 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 07 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 08 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 09 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 0A 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 0B 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 0C 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 0D 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 0E 0A 65
1000,i,CB 93 01 78 01 89 03 E5 01 0F 0A 65
```

CEMOLMD2.SEQ

```
TP0
# A_Link_Write service for LT supervised. This sequence only checks the
service
# It does not check the given GA themselvesin the BDUT.
# For LT supervised, x/y = 8/9
#
#Case A
1000,i,CB 95 01 78 01 89 03 E7 01 01 C2 40 0A 65
# expected response:
#      CB 95 01 89 01 78 03 E6 01 11 C2 40 0A 65
1000,i,CB 95 01 78 01 89 03 E7 01 00 C2 41 0A 65
# expected response:
#      CB 97 01 89 01 78 03 E6 01 11 C2 40 C2 41 0A 65
1000,i,CB 95 01 78 01 89 03 E7 01 00 C2 42 0A 65
# expected response:
#      CB 99 01 89 01 78 03 E6 01 11 C2 40 C2 41 C2 42 0A 65
1000,i,CB 95 01 78 01 89 03 E7 01 00 C2 43 0A 65
# expected response:
#      CB 9B 01 89 01 78 03 E6 01 11 C2 40 C2 41 C2 42 C2 43 0A 65
1000,i,CB 95 01 78 01 89 03 E7 01 00 C2 44 0A 65
# expected response:
#      CB 9D 01 89 01 78 03 E6 01 11 C2 40 C2 41 C2 42 C2 43 C2 44 0A 65
1000,i,CB 95 01 78 01 89 03 E7 01 00 C2 45 0A 65
# expected response:
#      CB 9F 01 89 01 78 03 E6 01 11 C2 40 C2 41 C2 42 C2 43 C2 44 C2 45 0A 65
1000,i,CB 95 01 78 01 89 03 E7 01 00 C2 46 0A 65
# expected response:
#      CB 9F 01 89 01 78 03 E6 01 13 C2 41 C2 42 C2 43 C2 44 C2 45 C2 46 0A 65

#Case B
1000,i,CB 95 01 78 01 89 03 E7 02 00 C2 41 0A 65
# expected response (C2 41 added to GO2 which does not exist)
#      CB 93 01 89 01 78 03 E6 02 00 0A 65

#Case C
1000,i,CB 95 01 78 01 89 03 E7 00 01 C2 36 0A 65
# expected response: (C2 36 is added with s flag to 1)
#      CB 99 01 89 01 78 03 E6 00 31 C2 34 C2 35 C2 36 0A 65

#Case D
1000,i,CB 95 01 78 01 89 03 E7 00 02 C2 35 0A 65
# expected response: (C2 35 is deleted)
#      CB 97 01 89 01 78 03 E6 00 21 C2 34 C2 36 0A 65

The following can also be tried
# (not described in the text of the suite of tests)
```

```
# IN, Write "C024" in the product to the object 1
1000,i,D9 95 01 78 10 89 03 E7 01 01 C0 24 08 B3
# OUT, Expected service A_Link_Response : D9 95 10 89 01 78 03 E6 01 01 C0 24
08 B3
# IN, verify it answers to this new Group Address (ON, then OFF)
1000,i,DD 91 C0 24 10 89 00 80 0B 81
# Expected LED to switch on
1000,i,DD 91 C0 24 10 89 00 81 0B 80
# Expected LED to switch on
# IN, Delete C024 from the product
1000,i,D9 95 01 78 10 89 03 E7 01 03 C0 24 08 B3
# OUT, Expected service A_Link_Response : D9 93 10 89 01 78 03 E6 01 00 08 B3
# IN, Verify that it does not react (on)
1000,i,DD 91 C0 24 10 89 00 80 0B 81
# OUT, No answer expected, No expected action on the LED
# IN, Verify that it does not react (off)
1000,i,DD 91 C0 24 10 89 00 81 0B 80
# OUT, No answer expected, No expected action on the LED
# IN, Verify that another group if present (here C016) is still working (ON)
1000,i,DD 91 C0 16 10 89 00 80 0B 81
# Expected LED switches off
# IN, Verify that another group if present (here C016) is still working (OFF)
1000,i,DD 91 C0 16 10 89 00 81 0B 80
# Expected LED switches on
```

CEMOLMD3.SEQ

```
TP0
# A_Link_Write service and s flag for sending address
# For LT supervised, x/y = 8/9
#
# (not described in the text of the suite of tests)


# IN, Write GAs (at least 10) in the product
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 24 08 B3
# OUT, Expected service A_Link_Response : D9 95 02 89 01 78 03 E6 01 01 C0 24
08 B3
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 25 08 B3
# Expected service A_Link_Response : D9 97 02 89 01 78 03 E6 01 01 C0 24 C0 25
08 B3
# IN, Write first GA with s flag set to 1
1000,i,D9 95 01 78 02 89 03 E7 01 01 C0 26 08 B3
# OUT, Expected : D9 99 02 89 01 78 03 E6 01 31 C0 24 C0 25 C0 26 08 B3
# IN, Write another GA with s flag set to 1
1000,i,D9 95 01 78 02 89 03 E7 01 01 C0 27 08 B3
# Verify that the “sending” group is now the last one:
# OUT, Expected : D9 9B 02 89 01 78 03 E6 01 41 C0 24 C0 25 C0 26 C0 27 08 B3
# IN, other GAs
```

```
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 28 08 B3
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 29 08 B3
# IN, Write a wrong frame containing two GA
1000,i,D9 97 01 78 02 89 03 E7 01 00 C0 30 C0 31 08 B3
# OUT, Expected : this frame is ignored
# IN, Write other GA (s flag to 0)
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 32 08 B3
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 33 08 B3
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 34 08 B3
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 35 08 B3
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 36 08 B3
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 37 08 B3
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 38 08 B3
1000,i,D9 95 01 78 02 89 03 E7 01 00 C0 39 08 B3
# 13 GA are already written (Maybe it was not possible to put all in the BDUT)
# IN, Verify that it answers to many of the new GA
# Expected LED switches off then on
1000,i,DD 91 C0 24 01 78 00 80 0B 81
1000,i,DD 91 C0 24 01 78 00 81 0B 80
1000,i,DD 91 C0 25 01 78 00 80 0B 81
1000,i,DD 91 C0 25 01 78 00 81 0B 80
1000,i,DD 91 C0 26 01 78 00 80 0B 81
1000,i,DD 91 C0 26 01 78 00 81 0B 80
1000,i,DD 91 C0 28 01 78 00 80 0B 81
1000,i,DD 91 C0 28 01 78 00 81 0B 80
1000,i,DD 91 C0 33 01 78 00 80 0B 81
1000,i,DD 91 C0 33 01 78 00 81 0B 80
1000,i,DD 91 C0 39 01 78 00 80 0B 81
1000,i,DD 91 C0 39 01 78 00 81 0B 80
#
# The test may be divided in two parts. Here begins the second part.
# IN, Try to read the Group Addresses at index 1
3000,i,D9 93 01 78 02 89 03 E5 00 01 08 B3
# OUT, Expected service A_Link_Response :
#      D9 9B 02 89 01 78 03 E6 01 41 C0 24 C0 25 C0 26 C0 27 08 B3
# IN, Try to read the GA at index 4, service A_Link_Read
3000,i,D9 93 01 78 02 89 03 E5 00 04 08 B3
# OUT, Expected service A_Link_Response : D9 95 02 89 01 78 03 E6 01 44 C0 27
08 B3
# IN, Delete the GA C024 from the product
1000,i,D9 95 01 78 02 89 03 E7 00 02 C0 24 08 B3
# OUT, Expected : D9 9B 02 89 01 78 03 E6 01 31 C0 25 C0 26 C0 27 C0 28 08 B3
# IN, Delete the GA C028 from the product
1000,i,D9 95 01 78 02 89 03 E7 00 02 C0 28 08 B3
# OUT, Expected : D9 99 02 89 01 78 03 E6 01 31 C0 25 C0 26 C0 27 08 B3
# IN, Delete other GA
1000,i,D9 95 01 78 02 89 03 E7 00 02 C0 25 08 B3
```

```
# OUT, Expected : D9 97 02 89 01 78 03 E6 01 21 C0 26 C0 27 08 B3
# IN, Delete other GA
1000,i,D9 95 01 78 02 89 03 E7 00 02 C0 26 08 B3
# OUT, Expected : D9 95 02 89 01 78 03 E6 01 11 C0 27 08 B3
1000,i,D9 95 01 78 02 89 03 E7 00 02 C0 29 08 B3
1000,i,D9 95 01 78 02 89 03 E7 00 02 C0 30 08 B3
1000,i,D9 95 01 78 02 89 03 E7 00 02 C0 31 08 B3
# Verify that it is deleted (no answers, no actions expected)
1000,i,DD 91 C0 24 01 78 00 80 0B 81
1000,i,DD 91 C0 24 01 78 00 81 0B 80
# Verify that other groups are still working (LED switches OFF, then ON)
1000,i,DD 91 C0 27 01 78 00 80 0B 81
1000,i,DD 91 C0 27 01 78 00 81 0B 80
1000,i,DD 91 C0 32 01 78 00 80 0B 81
1000,i,DD 91 C0 32 01 78 00 81 0B 80
1000,i,DD 91 C0 39 01 78 00 80 0B 81
1000,i,DD 91 C0 39 01 78 00 81 0B 80
```

CEMOLMD4.SEQ

```
TP0
# Link Management Write after a A-Memory-Write
# For LT supervised, x/y = 8/9
# L4 Ack are not said expected, but they may be present in responses (TPCI=C2)
#
# IN, Link_read request start-index
1000,i,CB 93 01 78 01 89 03 E5 01 01 0A 65
# OUT, Expected : CB 95 01 89 01 78 03 E6 01 11 C0 27 08 B3
# IN, Connection
1000,i,CB 90 01 78 01 89 80 B2 64
# IN, A-Memory-Read of 4 bytes, from address 0139h and TPCI “Numbered Data
Packet”
3000,i,CB 93 01 78 01 89 42 04 01 39 0A 65
# OUT, Expected A_Memory_Response: CB 97 01 89 01 78 42 44 01 39 AA BB CC DD
08 B3
# IN, A-Memory-Write of 2 bytes (55 and AA)
1000,i,CB 95 01 78 01 89 46 82 01 39 55 AA 0A 65
# IN, Disconnection
1000,i,CB 90 01 78 01 89 81 B2 64
#
# IN, Link_write
1000,i,CB 95 01 78 01 CC 03 E7 01 01 C0 24 0A 65
# OUT, No answer expected
# IN, Link_read request start-index
1000,i,CB 93 01 78 01 89 03 E5 01 01 0A 65
# OUT, No answer expected
# Try to push the button and see the new GA on/off (it will not work)
# IN, Verify that it is deleted (no answers, no actions expected [on])
```

```
500,i,CD 91 C0 24 01 CC 00 81 0A 65
# OUT, No answer expected
# IN, Verify that it is deleted (no answers, no actions expected [off])
500,i,CD 91 C0 24 01 CC 00 80 0A 65
# OUT, No answer expected
```

CEMONS4.SEQ

```
TP0
# For LT supervised keep Individual Address at 01 78, and x/y = 8/9
# Reduced interface object with A_NetworkParameter_Read
# IN, Serial number read request (request coming from 01 AA to 01 78) (APCI =
03 DA)
1000,i,CB 95 01 78 01 AA 03 DA 00 00 0B 00 0A 65
# OUT, As this service is optional, maybe no response
# or OUT, Expected service A_NetworkParameter_Response:
#      CB 95 01 AA 01 78 03 DB 00 00 0B (then serial number on 6 bytes) 0A 65


# A_PropertyValue_Read request (on serial number)
# IN
1000,i,CB 95 01 78 01 AA 03 D5 00 0B 60 01 0A 65
# OUT, Expected response is :
#      CB 9B 01 AA 01 78 03 D6 00 0B 60 01 (then serial number on 6 bytes) 0A
65
#
```

CEMONS5.SEQ

```
TP0
# For LT supervised keep Individual Address at 01 78, and x/y = 8/9
# Reduced interface object with A_NetworkParameter_Read
# IN, Object Type read request (request coming from 01 AA to 01 78) (APCI = 03
DA)
1000,i,CB 95 01 78 01 AA 03 DA 00 00 01 00 0A 65
# OUT, As this service is optional, maybe no response
# or OUT, Expected service A_NetworkParameter_Response:
#      CB 95 01 AA 01 78 03 DB 00 00 01 (then object type on 2 bytes) 0A 65


# A_PropertyValue_Read request of object type
# IN (APCI = 03 D5)
1000,i,CB 95 01 78 01 AA 03 D5 00 01 20 01 0A 65
# OUT, Expected response is :
#      CB 9B 01 AA 01 78 03 D6 00 01 20 01 (then object type on 2 bytes) 0A 65
```

CEMONS6.SEQ

```
TP0
# For LT supervised keep Individual Address at 01 78, and x/y = 8/9
# Reduced interface object is supported, test of A_PropertyDescription_Read
# IN, read request (request coming from 01 AA to 01 78) (APCI = 03 D8)
```

```
1000,i,CB 95 01 78 01 AA 03 D8 00 0B 01 0A 65
# OUT, no response expected
```

CEMOFO2.SEQ

```
TP0
# FOCIs
# First Foci tried = F0 01
1000,i,CD 95 F0 01 01 5A 64 00 80 12 34 56 0A 65
# Expected answer depends on the FOCI, see also pixit
# First Foci tried = F0 01, new value
1000,i,CD 95 F0 01 01 5A 64 00 80 00 00 00 0A 65
# Expected answer depends on the FOCI, see also pixit
# Second Foci tried = F4 03
1000,i,CD 91 F4 03 01 5A 60 80 0A 65
# No response expected from this unknown FOCI
```

CEMOPA2.SEQ

```
TP0
# For LT supervised, x/y = 8/9
#
# Parameters, preliminary = read size of blocks
# IN, PropertyDescriptor size read request on property “Channel parameter”
channel 1
1000,i,CB 94 01 78 01 AA 03 D8 00 65 00 0A 65
# OUT, Expected A_PropertyDescription_Response:
#      CB 97 01 AA 01 78 03 D9 00 65 00 03 02 00 0A 65
# IN, PropertyDescriptor size read request on property “Channel parameter”
channel 2
1000,i,CB 94 01 78 01 AA 03 D8 00 66 00 0A 65
# OUT, Expected A_PropertyDescription_Response:
#      CB 97 01 AA 01 78 03 D9 00 66 00 03 04 00 0A 65
#
# Parameters, first part = read
# IN, Device Object read request on property “Channel parameter” channel 1
1000,i,CB 95 01 78 01 AA 03 D5 00 65 20 01 0A 65
# OUT, Expected:CB 97 01 AA 01 78 03 D6 00 65 20 01 (+ the block “11 12”) 0A
65
# IN, Device Object read request on property “Channel parameter” channel 2
1000,i,CB 95 01 78 01 AA 03 D5 00 66 20 01 0A 65
# OUT, Expected:CB 99 01 AA 01 78 03 D6 00 65 40 01 (+ the block “21 22 23
24”) 0A 65
#
# Second part = Write
# IN, Device Object write property “Channel parameter” channel 1, 2 bytes
1000,i,CB 97 01 78 01 AA 03 D7 00 65 20 01 22 22 0A 65
# OUT, Expected:CB 97 01 AA 01 78 03 D6 00 65 20 01 22 22 0A 65
# IN, Device Object write property “Channel parameter” channel 2, 4 bytes
1000,i,CB 99 01 78 01 AA 03 D7 00 66 40 01 33 33 44 44 0A 65
```

```
# OUT, Expected:CB 99 01 AA 01 78 03 D6 00 65 40 01 33 33 44 44 0A 65
#
# Third part = Partial Write
# IN, Device Object write property “Channel parameter” channel 2, 3 bytes
1000,i,CB 98 01 78 01 AA 03 D7 00 66 30 01 55 66 77 0A 65
# OUT, Expected:CB 99 01 AA 01 78 03 D6 00 65 40 01 55 66 77 44 0A 65
```

5.4 Examples of sequence files for HSOBS (Supervisor)

In the sequences the printed “IN” and “OUT” show respectively what is sent and what is expected to be seen by the tool.

CEMODOA1.SEQ

```
PL132, open medium
# Individual address of supervisor is 03 78 in supervised mode
# Case A : creation with response
#    00 00 EF 03 78 00 00 66 03 E3 12 34 03 00 FF 0A 66
# expected response :
1000,i,00 00 EF 03 69 00 00 63 03 E2 12 34 0A 66

# Case B : creation without response
#    00 00 EF 03 78 00 00 66 03 E3 12 34 03 00 FF 0A 66

# Case C : distribution
#    00 00 EF 03 78 00 00 63 03 E0 12 35 0A 66

# Case D : query from auxiliary device
# IN, A_DomainAddress_read
2000,i,00 00 EF 03 FF 00 00 61 03 E1 0A 66
# expected response from supervisor:
#      00 00 EF 03 78 00 00 63 03 E2 12 35 0A 66
```

CEMOSNS1.SEQ

```
TP0
# Individual address of SNA sender is 01 45
# Individual address of supervisor is 01 78 in supervised mode
# IN, Test of service A_NetworkParameter_Write SNA “88” from 01 45 to all 00
00
# Device Object is 00 00 and propertyID is 39 (57d)
8000,i,CD 95 00 00 01 45 03 E4 00 00 39 88 0A 66
# Make the product send a frame to check that it uses the SNA value “88”
#       (e.g. Press a BP)
# OUT, Expected A-GroupValue-Write: CD 91 9F 01 88 78 00 81 0B 77
# Back to previous value
# IN, Test of service A_NetworkParameter_Write SNA “01” from 01 45 to all 00
00
1000,i,CD 95 00 00 01 45 03 E4 00 00 39 01 0A 66
```

CEMOSNS2.SEQ

```
TP0
# Individual address of supervisor is 01 78 in supervised mode
# OUT, Supervisor is the sender of this service A_NetworkParameter_Write SNA
“88”
#    CD 95 00 00 01 78 03 E4 00 00 39 88 0A 66
# Make the supervisor send a frame to check that it uses the SNA value “88”
# OUT, Expected A-GroupValue-Write: CD 91 9F 01 88 78 00 81 0B 77
# Back to previous value
# OUT, Test of service A_NetworkParameter_Write SNA “01”
```

```
#    CD 95 00 00 88 78 03 E4 00 00 39 01 0A 66
```

CEMOIAS1.SEQ

```
TP0
# NM_IndividualAddressWrite procedure
# Connection mode, Individual Address of supervisor 01 78

# OUT, Connection layer 4 from supervisor
#    CB 90 01 FF 01 78 80 0A 65
# OUT, Descriptor type 0 = Mask version read request from supervisor
#    CB 91 01 FF 01 78 07 00 0A 65
# IN, Expected A- DeviceDescriptor-Response from auxiliary device
1000,i,CB 93 01 78 01 FF 07 40 30 12 0B 77
# OUT, Disconnection layer 4 from supervisor
#    CB 90 01 FF 01 78 81 0A 65

#Supervisor : sending of service A_PhysicalAddress_Read
#    CB 91 01 FF 01 78 01 00 0A 65
# expected response
1000,i,CB 93 01 78 01 FF 01 40 01 FF 0A 65
#Supervisor : sending of service A_PhysicalAddress_Write
#    CB 93 01 FF 01 78 00 C0 01 51 A 65

# OUT, Connection layer 4 from supervisor
#    CB 90 01 51 01 78 80 0A 65
# OUT, Descriptor type 0 = Mask version read request from supervisor
#    CB 92 01 51 01 78 07 00 00 0A 65
# IN, Expected A- DeviceDescriptor-Response from auxiliary device
1000,i,CB 93 01 78 01 51 07 40 30 12 0B 77
# OUT, A_Restart
#    CB 91 01 51 01 78 0B 80 0B 77
# OUT, Disconnection layer 4 from supervisor
1000,i,CB 90 01 51 01 78 81 0A 65
```

CEMOIAS2.SEQ

```
TP0
# NM_IndividualAddressSerialNumberWrite procedure
# Individual Address of supervisor 01 78
#Supervisor : sending of service A_IndividualAddressSerialNumber_Write
#    CB 9D 01 FF 01 78 03 DE 30 40 50 60 70 80 01 51 00 00 00 00 0A 65
#Supervisor : sending of service A_ IndividualAddressSerialNumber _Read
#    CD 97 00 00 01 78 03 DC 30 40 50 60 70 80 0A 65
# expected response
1000,i,CB 93 01 78 01 51 03 DD 30 40 50 60 70 80 00 00 00 00 0A 65
```

CEMODIS1.SEQ

```
TP0
# Device Identification done by LT supervisor
# Case A only, connectionless
# DeviceDescriptor-Read service addressed to correct device
# by supervisor at the Individual Address 01 78 (x/y=6/7)
# OUT, Descriptor type 0 = Mask version read request
#    CB 91 01 45 01 78 03 00 0A 65
# IN, Expected A- DeviceDescriptor-Response:
1000,i,CB 93 01 78 01 45 03 40 30 12 0B 77
# OUT, Descriptor type 2 = Identification
#    CB 91 01 45 01 78 03 02 0A 65
# IN, Expected service A- DeviceDescriptor-Response:
1000,i,CB 9F 01 78 01 45 03 42 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 0A 65
```

CEMODIS2.SEQ

```
TP0
# Device Identification done by LT supervisor
# Case A only, connection oriented
# DeviceDescriptor-Read service addressed to correct device
# by supervisor at the Individual Address 01 78 (x/y=6/7)
# OUT, Connection layer 4 from supervisor
#    CB 90 01 45 01 78 80 0A 65
# OUT, Descriptor type 0 = Mask version read request from supervisor
#    CB 91 01 45 01 78 07 00 0A 65
# IN, Expected A- DeviceDescriptor-Response from auxiliary device
1000,i,CB 93 01 78 01 45 07 40 30 12 0B 77
# OUT, Descriptor type 2
#    CB 91 01 45 01 78 03 02 0A 66
# IN, Expected response from auxiliary device
1000,i,CB 9F 01 78 01 45 03 42 00 64 22 02 4C 01 00 AE FF 00 00 00 00 00 0A 66
# OUT, Disconnection layer 4 from supervisor
#    CB 90 01 45 01 78 81 0A 65
```

CEMOGAS1.SEQ

```
TP0
#Case A
#GA check from supervisor for LT supervised correct device
# OUT, Request to all if the GA C1 23 is used by anybody (from
supervisor)[case 1]
#    CD 97 00 00 01 78 03 DA 00 01 17 01 C1 23 0A 65
# OUT, Expected service A_NetworkParameter_Response:
#      CB 97 01 78 01 BC 03 DB 00 01 17 01 C1 23 0A 65
# or OUT, Request (from supervisor) to one product (Individual Address 01
BC)[case 2]
#      if it uses the GA C1 23
#    CB 97 01 BC 01 78 03 DA 00 01 17 01 C1 23 0A 65
```

```
# OUT, Expected service A_NetworkParameter_Response:
#      CB 97 01 78 01 BC 03 DB 00 01 17 01 C1 23 0A 65
#
# OUT, Request (from supervisor) to all if the GA 66 77 is used by anybody
#      (but nobody uses it)
#    CD 97 00 00 01 78 03 DA 00 01 17 01 66 77 0A 65
#      No answer expected
# or OUT, Request (from supervisor) to the product (address 01 BC)
#      if it uses the GA 66 77
#    CB 97 01 BC 01 78 03 DA 00 01 17 01 66 77 0A 65
#      No answer expected


Case B [optional] (range equal to 7F)
# OUT, Request to all if any GA from C1 23 to C1 A3 are used by anybody [case
1]
#    CD 97 00 00 01 78 03 DA 00 01 17 7F C1 23 0A 65
```

CEMOLMS1.SEQ

```
TP0
# Link Management A_Link_Read
# [Group Object n° are tried in sequence]
# (Responses depend on the application :
#     n° of objects, n° of GA attached per Group Object)
# For LT supervised correct device, x/y = 8/9
#
# Case A
# OUT, Link read request start-index to check, from supervisor
#    CB 93 01 AA 01 78 03 E5 00 01 0A 65
# IN, Expected A_Link_Response from correct device
1000,i,CB 95 01 78 01 AA 03 E6 00 11 C2 40 0A 65
# or (case F) IN, if no association with the Group Object expected
A_Link_Response:
1000,i,CB 93 01 78 01 AA 03 E6 00 00 0A 65 (start-index at 0)
# OUT, Link read request, from supervisor
#    CB 93 01 AA 01 78 03 E5 FF 01 0A 65
# IN, Expected A_Link_Response from correct device
1000,i,CB 95 01 78 01 AA 03 E6 FF 11 C2 41 0A 65


# Case B
# OUT, Link read request, from supervisor
#    CB 93 01 AA 01 78 03 E5 01 01 0A 65
# IN, Expected A_Link_Response from correct device
1000,i,CB 9F 01 78 01 AA 03 E6 01 41 C2 50 C2 51 C2 52 C2 53 C2 54 C2 55 0A 65


# Case C
# OUT, Link read request, from supervisor
#    CB 93 01 AA 01 78 03 E5 01 07 0A 65
```

```
# IN, Expected A_Link_Response from correct device
1000,i,CB 9B 01 78 01 AA 03 E6 01 47 C2 56 C2 57 C2 58 C2 59 0A 65
```

CEMOLMS2.SEQ

```
TP0
# Case G
# A_Link_Write service by supervisor for LT supervised correct device
#
# OUT, Supervisor writes "C270" in the correct device to the object 4 with s
flag to 1
#    D9 95 01 89 01 78 03 E7 04 01 C2 70 08 B3
# IN, Expected service A_Link_Response from correct device :
1000,i,D9 95 10 89 01 78 03 E6 04 11 C2 70 08 B3
# OUT, verify it answers to this new Group Address (ON, then OFF)
#
# Case H
# OUT, Supervisor deletes C270 from the product
1000,i,D9 95 01 89 01 78 03 E7 04 03 C2 70 08 B3
# IN, Expected service A_Link_Response :
1000,i,D9 93 10 89 01 78 03 E6 04 00 08 B3
#Case G and H may be repeated
```

CEMONSS1.SEQ

```
TP0
# Reduced interface object with A_NetworkParameter_Read
# For supervisor x/y = 8/9
#
# Case A
# OUT, Serial number read request
#     (request coming from the supervisor 01 78 to 01 AA) (APCI = 03 DA)
#    CB 95 01 AA 01 78 03 DA 00 00 0B 00 0A 65
# As this service is optional in the correct device, maybe no response
# or IN, Expected service A_NetworkParameter_Response:
1000,i,CB 9A 01 78 01 AA 03 DB 00 00 0B (then serial number on 6 bytes) 0A 65
#
# Case C
# OUT, Serial number read request
#     (request coming from the supervisor 01 78 to 01 BB) (APCI = 03 DA)
#    CB 95 01 BB 01 78 03 DA 00 00 0B 00 0A 65
# IN, Expected service A_NetworkParameter_Response with no serial number:
1000,i,CB 9A 01 78 01 BB 03 DB 00 00 0B 00 00 00 00 00 00 0A 65
```

CEMONSS2.SEQ

```
TP0
# Reduced interface object with A_NetworkParameter_Read
# For supervisor x/y = 8/9
```

```
#
# Case A
# OUT, Interface Object Type read request
#     (request coming from the supervisor 01 78 to 01AA) (APCI = 03 DA)
#    CB 95 01 AA 01 78 03 DA 00 32 01 00 0A 65
# This service can also be sent to all (not tested here)
# IN, Expected service A_NetworkParameter_Response: (test info 00, test result
01)
1000,i,CB 96 01 78 01 AA 03 DB 00 32 01 00 01 0A 65
```

CEMOPAS1.SEQ

```
TP0
# For LT supervised correct device, x/y = 8/9
# Parameters
# Preliminary = read size of blocks by the supervisor
# OUT, PropertyDescriptor size read request on property “Channel parameter”
channel 1
#    CB 94 01 AA 01 78 03 D8 00 65 00 0A 65
# IN, Expected A_PropertyDescription_Response:
1000,i,CB 97 01 78 01 AA 03 D9 00 65 00 03 02 00 0A 65
# OUT, PropertyDescriptor size read request on property “Channel parameter”
channel 2
#    CB 94 01 AA 01 78 03 D8 00 66 00 0A 65
# IN, Expected A_PropertyDescription_Response:
1000,i,CB 97 01 78 01 AA 03 D9 00 66 00 03 04 00 0A 65
#
# First part = read by the supervisor
# OUT, Device Object read request on property “Channel parameter” channel 1
#    CB 95 01 AA 01 78 03 D5 00 65 20 01 0A 65
# IN, Expected (block 11 12):
1000,i,CB 97 01 78 01 AA 03 D6 00 65 20 01 11 12 0A 65
# OUT, Device Object read request on property “Channel parameter” channel 2
#    CB 95 01 AA 01 78 03 D5 00 66 20 01 0A 65
# IN, Expected (block 21 22 23 24):
1000,i,CB 99 01 78 01 AA 03 D6 00 65 40 01 21 22 23 24 0A 65
#
# Second part = Write by the supervisor
# OUT, Device Object write property “Channel parameter” channel 1, 2 bytes
#    CB 97 01 AA 01 78 03 D7 00 65 20 01 22 22 0A 65
# IN, Expected:
1000,i,CB 97 01 78 01 AA 03 D6 00 65 20 01 22 22 0A 65
# OUT, Device Object write property “Channel parameter” channel 2, 4 bytes
#    CB 99 01 AA 01 78 03 D7 00 66 40 01 33 33 44 44 0A 65
# IN, Expected:
1000,i,CB 99 01 78 01 AA 03 D6 00 66 40 01 33 33 44 44 0A 65
#
# Third part = Partial Write by the supervisor
```

```
# OUT, Device Object write property “Channel parameter” channel 2, 3 bytes
#    CB 98 01 AA 01 78 03 D7 00 66 30 01 55 66 77 0A 65
# IN, Expected:
1000,i,CB 99 01 78 01 AA 03 D6 00 66 40 01 55 66 77 44 0A 65

# Case A
# OUT, Device Object write property “Channel parameter” channel 3, 1 byte
#    CB 96 01 AA 01 78 03 D7 01 67 10 01 0A 0A 65
# IN, Expected:
1000,i,CB 96 01 78 01 AA 03 D6 01 67 10 01 0A 0A 65

# Case B
# OUT, Device Object write property “Channel parameter” channel 3, 1 byte
#    CB 96 01 AA 01 78 03 D7 01 67 10 02 10 0A 65
# IN, Expected:
1000,i,CB 97 01 78 01 AA 03 D6 01 67 20 01 0A 10 0A 65

# Case C
# OUT, Device Object write property “Channel parameter” channel 3, 1 byte
#    CB 96 01 AA 01 78 03 D7 01 67 10 01 0A 0A 65
# IN, Expected (negative answer):
1000,i,CB 95 01 78 01 AA 03 D6 01 67 10 00 0A 65

# Case D
# OUT, Device Object read property “Channel parameter” channel 3, 1 byte
#    CB 95 01 AA 01 78 03 D5 01 67 10 01 0A 65
# IN, Expected:
1000,i,CB 96 01 78 01 AA 03 D6 01 67 10 01 0A 0A 65

# Case E
# OUT, Device Object read property “Channel parameter” channel 3, 1 byte
#    CB 95 01 AA 01 78 03 D5 01 67 10 02 0A 65
# IN, Expected:
1000,i,CB 96 01 78 01 AA 03 D6 01 67 10 02 10 0A 65

# Case F
# OUT, Device Object read property “Channel parameter” channel 3, 1 byte
#    CB 95 01 AA 01 78 03 D5 01 67 10 01 0A 65
# IN, Expected (negative answer):
1000,i,CB 95 01 78 01 AA 03 D6 01 67 10 00 0A 65
```

5.5 Example of session for HSOBS

Example of configuration of the observer session: CEMODLTR.SES

```
Home Systems Network Observer - version 4.3
```

```
Session name      : C:\HSOBS \CEMODLTR\CEMODLTR.SES
Acquisition source: online
LogFiles          : valid.net (Network level)
Medium            : TP0
Port              : Com1
Buffer mode       : continue when buffer full
------------------------------------------------------------------------------
Filters used      : Filter1 (inactive)
------------------------------------------------------------------------------
Start condition   : immediate
Stop condition    : immediate
After stop        : nothing
TrigOut condition : none
------------------------------------------------------------------------------
Time stamping     : Beginning of frame: relative to RUN
                    pattern: hh:mm:ss ms (precision 100µs)
                    End of frame: relative to RUN
                    pattern:::ss ms (precision 100µs)
Error level       : errors and warnings
------------------------------------------------------------------------------
Generator
---------
Ack transmission  : inhibited
                    waiting time : 18ms
                    MAC addresses: DAh DAl Ack GrpAdr (Y/N)
                                   01  54  00  No
                                   01  AA  00  No
                                   00  03  00  No
                                   00  04  00  No
MAC repetition    : inhibited
                    time out for waiting ACK: 30ms
                    number of repetitions   : 0
LLC repetition    : inhibited
                    number of repetitions if No ACK: 0
                    number of repetitions if NACK: 0
                    time out if No ACK: 0ms
                    time out if NACK: 0ms
FCS generation    : automatic
InterFrame time   : different senders = 22ms
                    same sender = 22ms
Generator sequence: source = file: cemosns1.seq
                    number of cycles = 1
                    time between 2 cycles = 100000
                    Start condition = immediate
------------------------------------------------------------------------------
--
```

5.6 List of equipment and tools used for examples on TP0

List of equipment and tools used	Comments	Quantity
PC	1 equipped with HSOBS.	2
HSOBS V4.3	Frame observation and generation software.	1
CEMODLTR session file and sequence files associated with the validation plan	All these files are found in the HSOBS CEMODLTR directory.	1
HSOBS TP0 acquisition board	HSOBS external acquisition board.	1
BDUT	The device to be tested. It may be a BIM like device; or a product without accessible PEI.	1
BIM Loader	A tool to download applications in a BIM (installed on a PC).	1
«2 PB + 2 LED» module	For BIM or BCU only: an application with PEI type 4, to be connected to the BDUT with a PEI.	1