AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

CDI2SIM

BSM and Device Simulator

User Manual

Authors: Alois Hahn, Tamme Dittrich, Venditti Benjamin

Version 1.10 / 23.01.2025

Version	Date	Change History/Reason	Name
V0.1	2018-08-04	Initial Draft	Alois Hahn
V0.2	2018-08-08	ECB Revision and comments	Harald Deinhammer
V1.0	2018-08-23	Release version	Alois Hahn
V1.1	2018-08-25	refinement and corrections, update Annex Descriptions	Alois Hahn
v1.2	2018-09-06	corrections according feedback from FAT	Alois Hahn
v1.3	2018-11-06	Updated Wireshark CDI2 Powerlink protocol with new expert mode (chapters 12.1.6 and 12.1.7). Updated XML validation (chapter 12.7). Updated flutter detector usage (12.11 Flutter Detector and Analysis). New SW Installation instructions (7.2.4). New front view and cabling (3.3 and 5.3).	Alois Hahn
v1.3A	2018-11-13	Fixed typo at end of 12.1.6, 220 mm instead 200 mm. Improved software installation instructions in 6.2. and 7.2.4.	Alois Hahn
v1.4	2020-01-29	new chapter "CDI Version 2.5 Support". new compact version information added to chapter 7.2.3. new everything.log examples to chapter 9.2.8 new chapter 12.1.8 "Wireshark Installation/Upgrade Hints" new chapter 8.2.10. "Common Interface Protocol"	Alois Hahn
v1.5	2021-07-14	Software Version SW_CDI2SIM_V27_20210714_IDB	Alois Hahn
v1.6	2022-03-15	Software Version SW_CDI2SIM_V27C_20220315, removed 2.5 support, Extra BSMS and DS parameters (9.2.10 and 9.2.11), IDB Sender setup (10.4), IDB Receiver setup (10.5), Update Annex: Description of Test Sets (14)	Alois Hahn
v1.7	2022-03-31	Software Version SW_CDI2SIM_V27CB_20220331, 2.7C/2.7B compatibility (7.4)	Alois Hahn
V1.8	2023-02-07	Version SW_CDI2SIM_V27CB_20221205_x86VB, new installer, stand-alone install (7.2.4, 7.3), x86 Visionbox (7.4), new BSMS_BSM-G01-HRMAX, DS_BSM-G01-HRMAX testcases (17.15, 16.14)	Alois Hahn
V1.9	2024-03-14	Updated 6.4.1. Linux Ubuntu Installation Fixed typo in Section 6.5	Benjamin Venditti

Table of Contents

1. Introduction

This document provides the necessary information to put the CDI2 BSM Simulator (BSMS) and the CDI2 Device Simulator (DS) into operation and to conduct test sequences with pre-stored test sets. Furthermore, the user shall be able to apply the provided tool-set to run the acceptance test procedures for the BSMS/DS and for CDI2 compliant BSMs and devices later.

The document consists of specific chapters for the BSMS and for the DS and of chapters, which are common to both.

Figure 1: BSM Simulator and Device Simulator

2. References

[Ref 1.]
Common Detector Interface 2 (CDI2) Specifications v2.7C, 21 March 2022
https://www.ecb.europa.eu/euro/pdf/CDI2_version_2.7C_Final.pdf

[Ref 2.]
Common Detector Interface 2 (CDI2) Specifications v2.7B, 29 January 2020
https://www.ecb.europa.eu/euro/pdf/CDI2_version_2.7B_Final.pdf

[Ref 3.]
Detector Test Rig User Manual
“6.27 User manual ADAPTOR 2 test transport V3.docx”

[Ref 4.]
Banknote Displacement Detector User Manual
“6.15.2 User manual Banknote displacement detector V2.docx”

[Ref 5.]
Banknote Displacement Detector User Manual (Detector casing)
"6.50 User manual Banknote displacement detector (Detector casing).pdf"

[Ref 6.]
CDI2 Device - PC Implementation User Manual, Version 1.3 / 26.6.2019
"CDI2-Device-PC-Implementation-v13-20190626.pdf"

2.1. Terms and abbreviations

DEV	CDI2 Device (CS, IEU or Detector)
Flutter Detector	Used in this document for *Banknote Displacement Detector*, an external device based on laser triangulation that supports the qualification of the BSM's transport characteristics. It is provided with the DS.
JSON	JavaScript Object Notation Is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication.
Lua	Is a lightweight, multi-paradigm programming language designed primarily for embedded use in applications.
REST	Representational State Transfer Is an architectural style that defines a set of constraints to be used for creating web services.

SS	Simulator Setup
Test Case	Specific configuration on DS and/or BSMS to test aspects of the CDI2 specification
Test Case Parameter	Configuration detail of a Test Case. E.g.: number of banknotes

For additional terms and abbreviations used in this document, please refer to the corresponding chapters in the CDI2 specifications [Ref 1.] as well.

This is the list of abbreviations as used in the CDI2 Specification (excerpt from [Ref 1.]).

AU	Aggregation Unit
BFA	Belt Free Area
BN	Banknote
BNID	Banknote ID
BP	Banknote Present
BSM	Banknote Sorting Machine
BSMS	BSM Simulator
CLE	Colour Linearity Error
CN	Controlled Node
CS	Camera System
DET	Detector
DMB	Detector Machine Bus
DN	Digital numbers
DN rms	DN root mean square
DR	Data Range
DSNU	Dark Signal Non-Uniformity
EEU	External Evaluation Unit
ESP	External Service Port
EU	Evaluation Unit
FS	Full Scale

GVSP	GigE Vision Stream Protocol
IDB	Image Data Bus
IEU	Image Evaluation Unit
IR	Infrared
LP	Line Pairs
MN	Managing Node
MTF	Modulation Transfer Function
MU	Measurement Unit
PL	POWERLINK
PRNU	Photo Response Non-Uniformity
RGB	Red, Green, Blue
SNR	Signal to Noise Ratio
TAP	Timed Air Pulses
TC	Transport Clock
TS	Transport Simulator
TTS	Transport Timing Signals

3. BSM Simulator

3.1. Overview

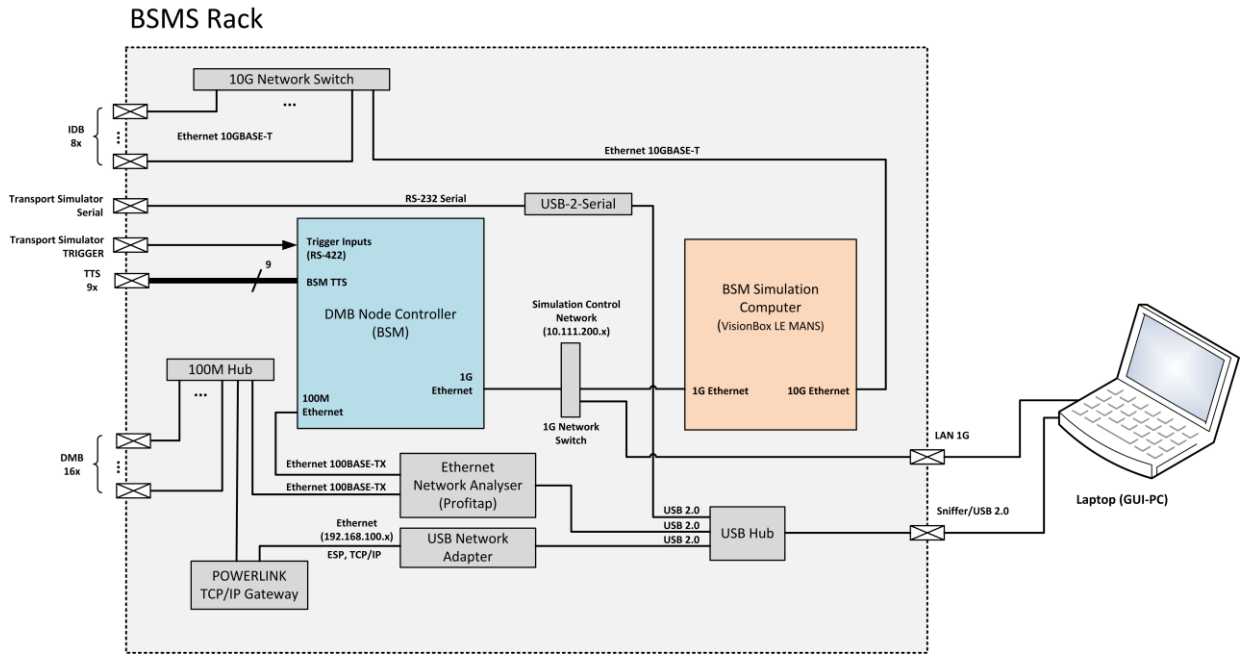


Figure 2: BSMS Rack Components

3.2. Packaging List

- BSMS main cabinet
  includes 1G network switch, 10G network switch, VisionBox Lemans, BSM Node Controller, three 100M 8-port Hubs, Powerlink Gateway, Profitap 100 Network Sniffer
- Power Cords
  2x US (Type B NEMA 5-15) and 2x Europe (Type F, CEE 7/4) version for Laptop and BSMS
- Dell Laptop with UK QWERTY keyboard layout
- TS Trigger Cable, D-sub DE-9 male/DE-9 female
- TS Serial Cable, D-sub DE-9 male/DE-9 female
- TTS cables, 9x10m, 1x20m, D-sub HD DE-15 male/DE-15 male
- DMB cables, RJ-45 patch cord, green, 16x10m
- IDB cables, RJ-45 patch cord, red, 2x10m, 1x20m
- DMB RJ-45 to M12 D-coded plug cables, 1x10m, 1x15m
- IDB RJ-45 to M12 X-coded plug cable, 1x20m
- RJ-45 patch cable, yellow, for BSMS-GUI PC
- USB cable, grey, Type A → Type B
- TS Trigger breakout box, D-sub DE-9 male/DE-9 female

- TS Trigger extension cable 2m, D-sub DE-9 male/DE-9 female
- USB Sticks with disk images (Recover Stick) and installation packages (Data Stick) for full software recovery
- This user manual

3.3. Main Modules and Front View

At normal operation all external connections are made at the rear panel (see 3.4) and the rack doors are closed. Figure 4 shows the position of the main functional units of the BSMS, as seen from the front when the front door has been opened.

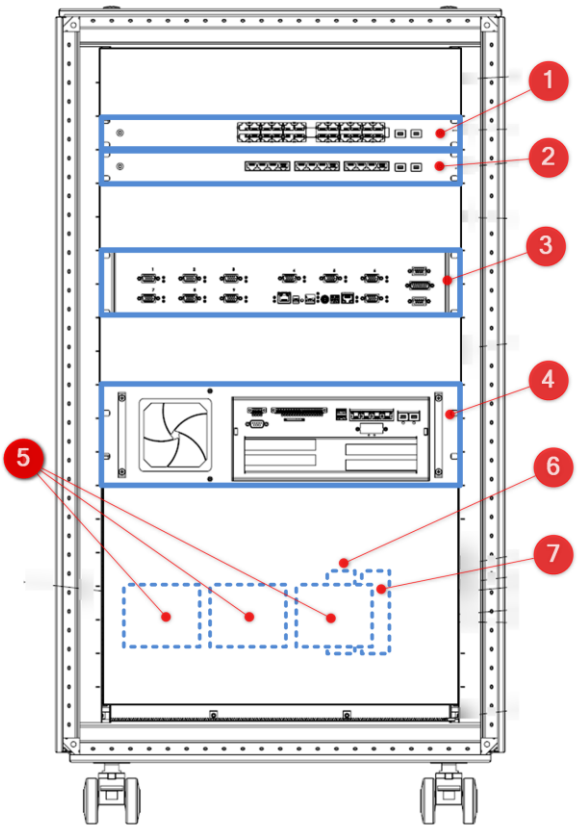


Figure 3: BSMS Front View

(1) Network Switch 1G for Simulation Control Network
NETGEAR ProSAFE GS724Tv4
(2) Network Switch 10G for IDB
NETGEAR ProSAFE XS712Tv2
(3) BSM Node Controller (see 3.6)
(4) BSM Simulation Computer
VisionBox "LeMans", 8-core ARM Cortex-A72
Imago Technologies, www.imago-technologies.com
(5) 8-port HUB for DMB *)
0AC808.9-1, B&R Industrial Automation GmbH, www.br-automation.com
(6) USB Hub *)
(7) POWERLINK TCP/IP Gateway *)
X20HB8815, B&R Industrial Automation GmbH, www.br-automation.com

*) Components (5), (6) and (7) are mounted on a top-hat rail covered by an aluminium plate. They are not visible without removing the cover.

All internal cables of the simulator are factory assembled already. User access to the cables is not needed during normal operation, except for service, repair or special test configurations. An overview about the cabling on the front panel is shown in Figure 4.

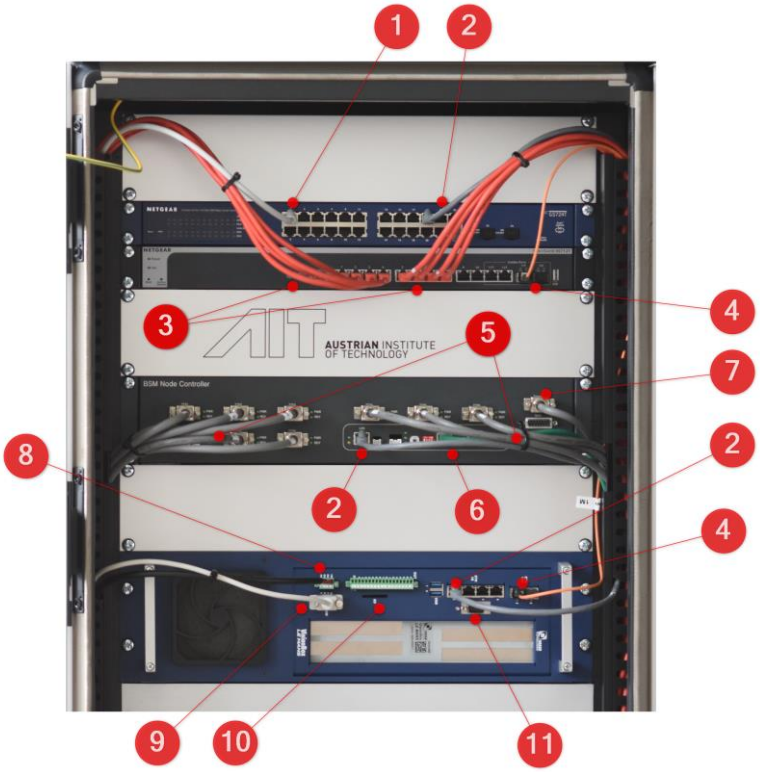


Figure 4: BSMS Front Side Cables

(1) Simulation Control Network, grey, to rear connector PC LAN (see 3.4)
(2) Simulation Control Network, grey, to VisionBox and BSM Node Controller
(3) IDB connections, red, to rear connectors IDB 1-8 (see 3.4)
(4) 10G network connection IDB switch to Visionbox, orange fibre cable
(5) TTS connections, grey, to rear connectors TTS 1-9 (see 3.4)
(6) DMB connection to HUB (see Figure 3/(5))
(7) Trigger connection to Transport Simulator, grey, to rear panel connector TRIG
(8) VisionBox power supply
(9) VisionBox console, grey, rear panel connector CONSOLE (see 3.4)
(10) VisionBox SD card slot
(11) VisionBox external triggers, not used

3.4. Rear Panel Connectors

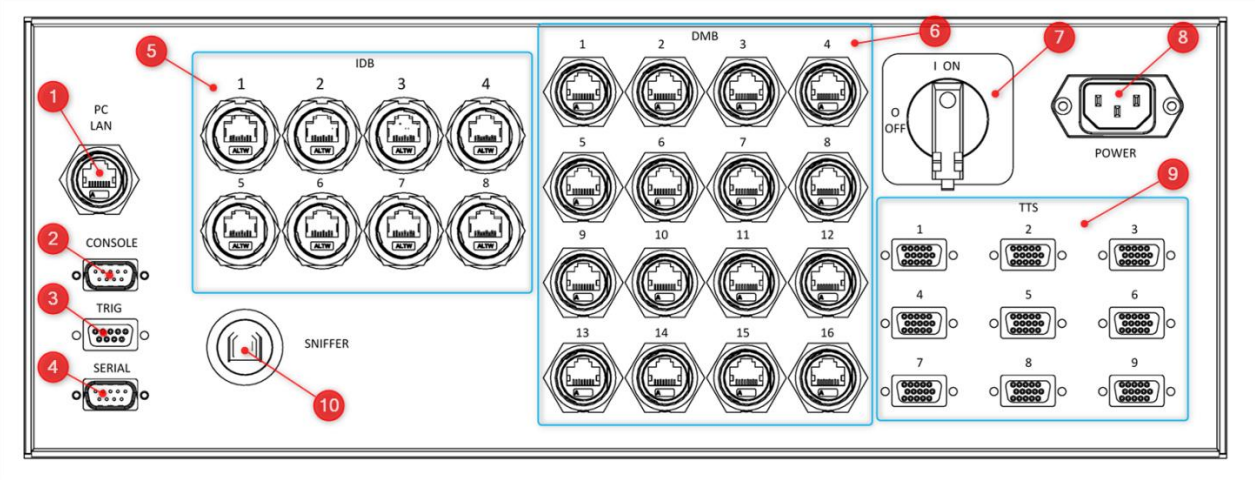


Figure 5: BSMS Rear Panel

(1) PC LAN
Network connection to Laptop

(2) CONSOLE
Console of VisionBox, serial RS-232, for debugging purposes, left unconnected

(3) TRIG
Trigger input for connecting the Transport Simulator [see 3.5.2]

(4) SERIAL
Control interface for the Transport Simulator, serial RS-232 [see 3.5.3]

(5) IDB
10G Network ports for the Image Data Bus

(6) DMB
100M Network ports for the Data Machine Bus, Powerlink industrial field bus

(7) Mains Switch
Power supply master switch

(8) POWER
Main inlet, IEC-60320 C14

(9) TTS
TTS trigger connectors, BSM side [see 3.5.1]

(10) SNIFFER
USB Type B, Laptop connection for three USB devices (via an USB hub), connects to the ProfiTap network sniffer, an USB-to-Serial interface and an USB-2-LAN Network Interface, [see 12.3]

3.5. Connectors and Pinning

3.5.1. Transport Timing Signals (TTS)

The TTS connector complies to [Ref 1.] chapter 5 Transport Timing Signals (TTS).

Pin	Name
1	BP+
6	BP-
3	TC+
8	TC-
9	RESET
14	READY
11	+5V
5	reserved
7	reserved
10	reserved
12	reserved
2	GND
4	GND
13	GND
15	GND

Table 1 : TTS, D-Sub HD 15-pin

3.5.2. Transport Simulator Trigger (TRIG)

Pin	Signal	Direction	Description	Electrical
1 6	TS_TRIGGER_P TS_TRIGGER_N	Input (from TS to BSMS)	Trigger Pulse: One pulse per banknote	RS-422 with 100 Ohm termination at receiver
7 8	TS_CLOCK_P TS_CLOCK_N	Input (from TS to BSMS)	Transport Clock: Clock Signal is synchronous to transport speed.	RS-422 with 100 Ohm termination at receiver
4 9	TS_JAM_P TS_JAM_N	Input (from TS to BSMS)	Not used	RS-422 with 100 Ohm termination at receiver
5	GND	Power	Signal Ground	

Table 2 : Transport Simulator Trigger, D-Sub 9-pin

3.5.3. Transport Simulator Control (SERIAL)

Pin	Signal	Direction	Description	Electrical
2	RxD	Input (from TS to BSMS)		RS-232
3	TxD	Output (from BSMS to TS)		RS-232
5				Ground

Table 3 : TTS, D-Sub HD 15-pin

3.6. BSM Node Controller

The BSM node controller is housed in a 19" rack unit, occupying 2 height units (2U). The device is already cabled inside the rack and user access to its connectors is not required usually.

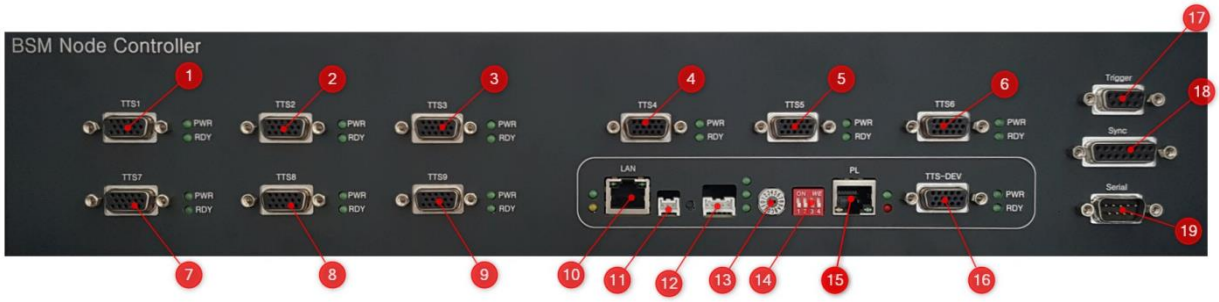


Figure 6: BSM Node Controller Front Panel

(1) BSM side TTS port 1
(2) BSM side TTS port 2
(3) BSM side TTS port 3
(4) BSM side TTS port 4
(5) BSM side TTS port 5
(6) BSM side TTS port 6
(7) BSM side TTS port 7
(8) BSM side TTS port 8
(9) BSM side TTS port 9
(10) LAN 1G, Simulation Control
(11) Serial Console, for debugging purposes
(12) General purpose digital IO, for debugging purposes
(13) Powerlink Node ID, bits [3:0], must be set to "0" on BSM
(14) Configuration Switch
(15) Powerlink Network
(16) Device side TTS, not used at BSM
(17) Trigger input from Transport Simulator
(18) Sync, not used
(19) Serial, not used

4. BSM Simulator Light

4.1. Overview

The BSMS Simulator Light has the same functionality as the original BSM Simulator but for fewer CDI2 devices. It simulates all signals and commands to operate a reduced set of CDI2 devices (1CS and 2 DET or 3 DET). Differences are explained as follows and depictured in Figure 7.

(1) It does not include the 10G Network Switch to connect a CS (Camera System) as well as IEUs (Image Evaluation Unit) to the BSM Simulation Computer. Instead, only a single CS can be connected.
(2) Only a single 10G IDB port is available at the rear end of the BSMS Simulation Computer.
(3) Only 3 TTS ports are provided instead of 9.
(4) Only 8 DMB ports are provided instead of 16.

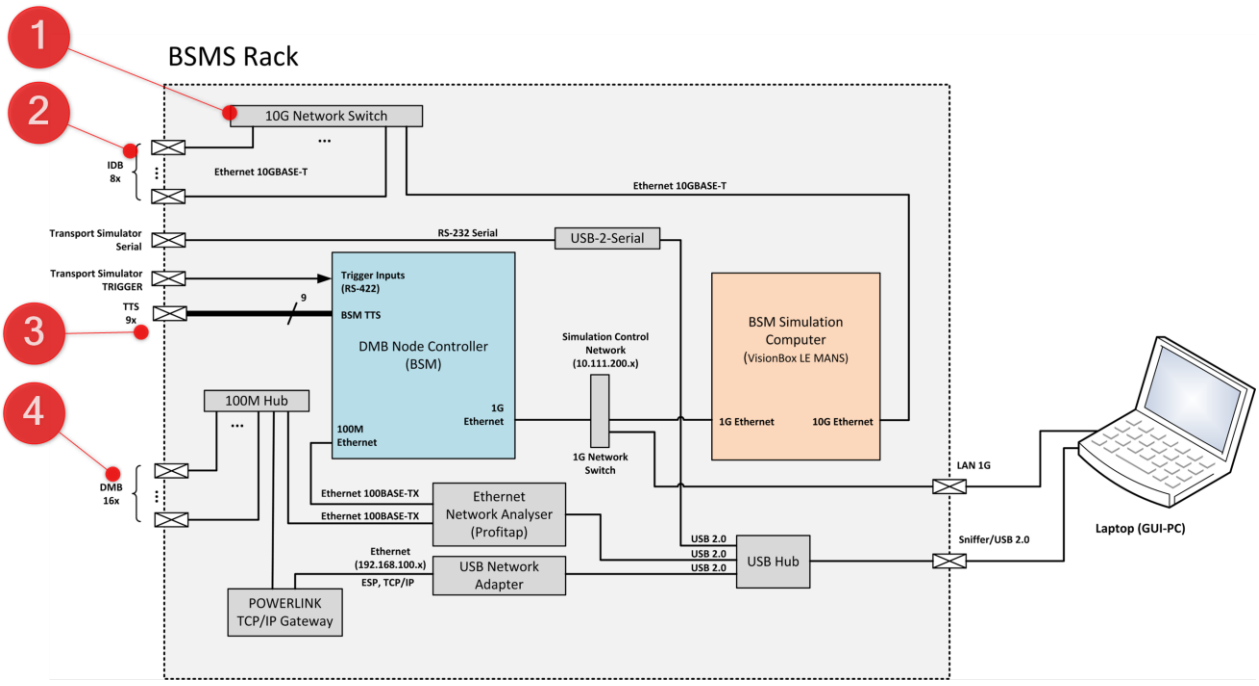


Figure 7: BSMS Light Rack Components / Differences

4.2. Packaging List

- BSMS Light 19” 8HE trolley rack
  - 1G 5-port network switch
  - BSM Simulation Computer
  - BSM Node Controller
  - 100M 8-port Hub
  - Powerlink Gateway
  - USB Hub

- Cable box
  - Power Cords for BSMS 1x US (Type B NEMA 5-15) and 1x Europe (Type F, CEE 7/4)
  - TS Trigger Cable, D-sub DE-9 male/DE-9 female, 10m
  - TS Serial Cable, D-sub DE-9 male/DE-9 female, 10m
  - TTS cables, D-sub HD-15 male/HD-15 male, 1x10m, 1x3m
  - DMB cables, RJ-45 patch cord, green, 2x10m
  - IDB cable, RJ-45 patch cord, red, 1x10m
  - DMB RJ-45 to M12 D-coded plug cable, green, 1x10m
  - IDB RJ-45 to M12 X-coded plug cable, red, 1x20m
  - RJ-45 patch cable for BSMS-GUI PC, yellow, 3m
  - TS Trigger extension cable, D-sub DE-9 male/DE-9 female, 2m
  - TTS Trigger extension cable incl. gender-changer, D-sub HD-15 male/HD-15 male, 3m
  - Flutter detector trigger cable, 10m
  - USB extension cable, 10m

- Suitcase
  - Profitap 100 Network Sniffer incl. 1x USB cable, Type A/ B & 1x 0,25m patch cable green
  - Digital Scope Unit (USB)
  - TS Trigger breakout box, D-sub DE-9 male/DE-9 female
  - TTS Trigger breakout box, D-sub HD-15 male/HD-15 female
  - USB to serial cable
  - USB network interface
  - USB Stick with installation packages for software recovery
  - This user manual

- Dell Laptop (GUI PC for Testing)

4.3. Main Modules and Front View

Figure 2 shows the position of the main functional units of the BSMS Light, as seen from the front when the front lid has been removed.

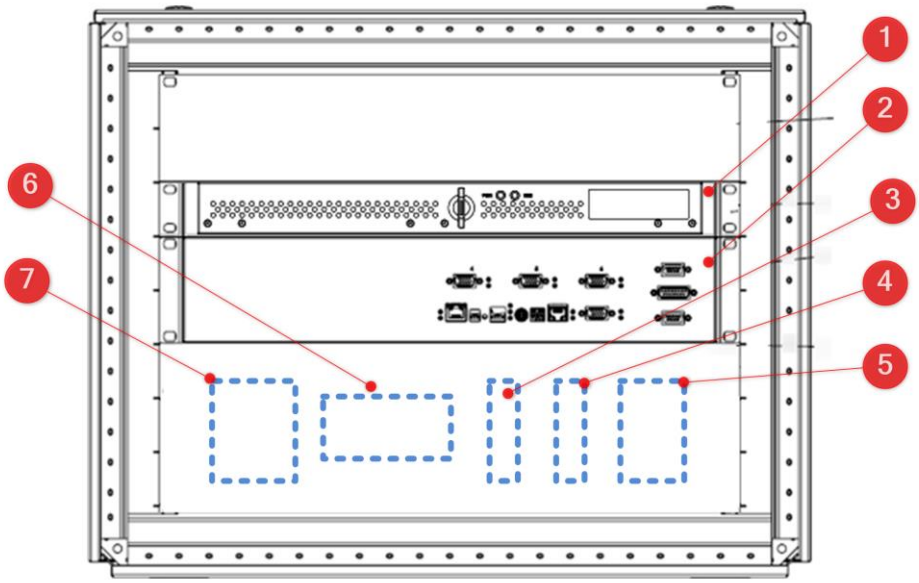


Figure 8: BSMS Light Front View

(1) BSM Simulation Computer
Kontron KISS 1U Short V3 CFL
(2) BSM Node Controller
(3) Network Switch 1G for Simulation Control Network*)
RS Pro, Compact Ethernet, 5 Port Gigabit Switch, 189-0537
(4) USB Hub *)
(5) POWERLINK TCP/IP Gateway *)
X20HB8815, B&R Industrial Automation GmbH, www.br-automation.com
(6) 8-port HUB for DMB *)
0AC808.9-1, B&R Industrial Automation GmbH, www.br-automation.com
(7) 24V Power Supply *)
0PS1040.0, B&R Industrial Automation GmbH, www.br-automation.com

*) Components (3-7) are mounted on a top hat section rail.

All internal cables of the simulator are factory assembled already. An overview about the cabling on the front panel is shown in Figure 4.

Figure 9: BSMS Light Front Side Cables

(1) Simulation Control Network, grey, to BSM Simulation Computer
(2) Simulation Control Network, grey, to BSM Node Controller
(3) Chained 24V Power Supply to peripherals on top hat section rail
(4) TTS connector
(5) Trigger connector to Transport Simulator
(6) DMB connection, green to BSM node Controller
(7) DMB connection, green, to Powerlink Gateway

4.4. Rear Panel Connectors

Figure 10: BSMS Rear Panel

(1) POWER
Main inlet, IEC-60320 C14
(2) Mains Switch
Power supply master switch
(3) Maintenance door, open to access 10G IDB port of BSM Simulation Computer

5. Device Simulator

5.1. Overview

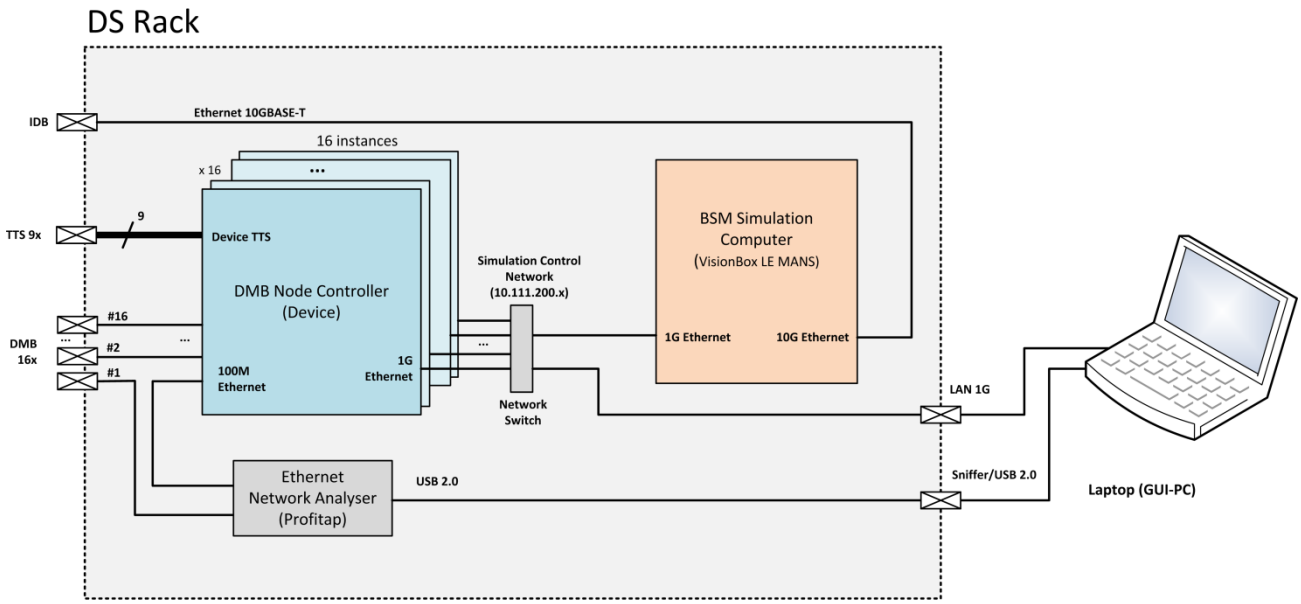


Figure 11: DS Rack Components

5.2. Packaging List

- DS main cabinet
  includes 1G network switch, VisionBox Lemans, 4 x Device Node Controller, Profitap 100 Network Sniffer
- Power cords
  2x US (Typ B NEMA 5-15) and 2x Europe (Typ F, CEE 7/4) version for Laptop and DS
- Dell Laptop with UK QWERTY keyboard layout
- DMB RJ-45 to M12 D-coded connector cables, 9x2m
- IDB RJ-45 to M12 X-coded connector cables, 1x1m
- RJ-45 patch cable, yellow, for BSMS-GUI PC
- USB cable, grey, Type A → Type B
- USB active extension cable, 10m (for Flutter Detector)
- Transport case including
  - Picoscope USB scope (including 2 probes and USB cable)
  - TTS breakout box, D-sub HD DE-15 male/DE-15 female
  - TTS extension cable 2m, D-sub HD DE-15 male/DE-15 female
  - USB Sticks with disk images (Recover Stick) and installation packages (Data Stick) for full software recovery
  - This user Manual

- Shipping box including
  - Flutter Detector in CS casing
  - Flutter Detector in DET casing
  - CDI2 CS empty casings
  - CDI2 DET empty casings
  - CDI1 Device Simulator case (if applicable)

5.3. Main Modules and Front View

At normal operation all external connections are made at the rear panel (see 5.4) and the rack doors are closed. Figure 12 shows the position of the main functional units of the DS, as seen from the front when the front door has been opened.

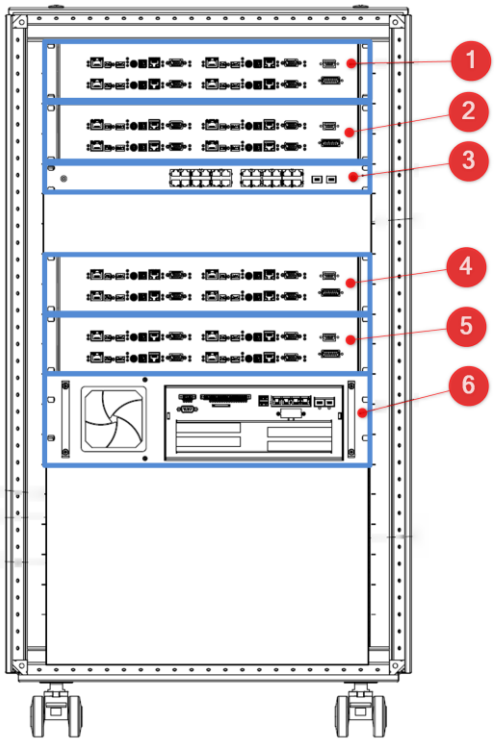


Figure 12: DS Front View

(1) DS Node Controller (see 5.5), CN 1-4
(2) DS Node Controller (see 5.5), CN 5-8
(3) Network Switch 1G for Simulation Control Network
NETGEAR ProSAFE GS724Tv4
(4) DS Node Controller (see 5.5), CN 9-12
(5) DS Node Controller (see 5.5), CN 13-16
(6) DS Simulation Computer
VisionBox "LeMans", 8-core ARM Cortex-A72
Imago Technologies, www.imago-technologies.com

All internal cables of the simulator are factory assembled already. User access to the cables is not needed during normal operation, except for service, repair or special test configurations. An overview about the cabling on the front panel is shown in Figure 13.

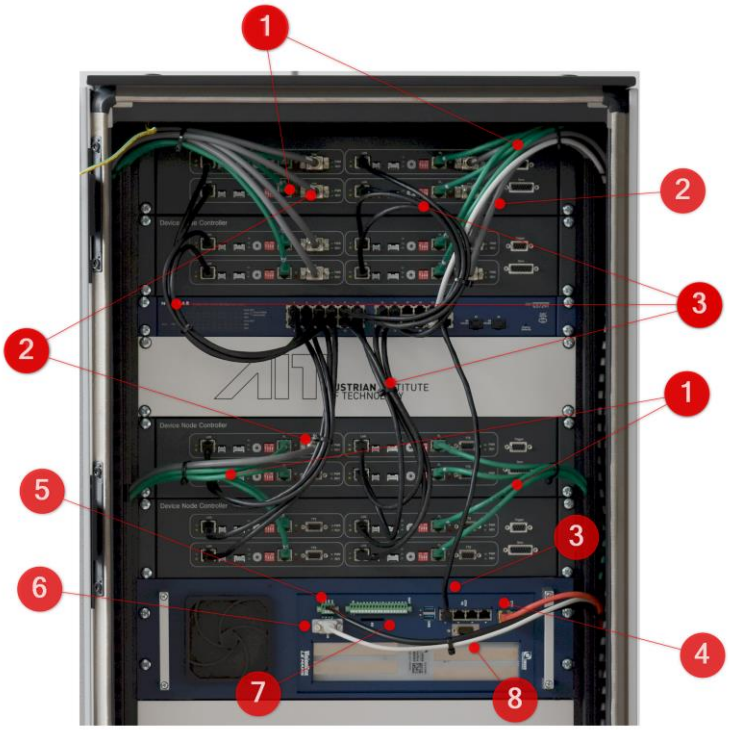


Figure 13: DS Front Side Cables

(1) DMB connections, green, to rear connectors DMB 1-16 (see 5.4)
(2) TTS connections, grey, to rear connectors TTS 1-9 (see 5.4)
(3) Simulation Control Network, black, to VisionBox and DS Node Controllers
(4) IDB connection, red, to rear connectors IDB 1 (see 5.4)
(5) VisionBox power supply
(6) VisionBox console, grey, rear panel connector CONSOLE (see 5.4)
(7) VisionBox SD card slot
(8) VisionBox external triggers, not used

5.4. Rear Panel Connectors

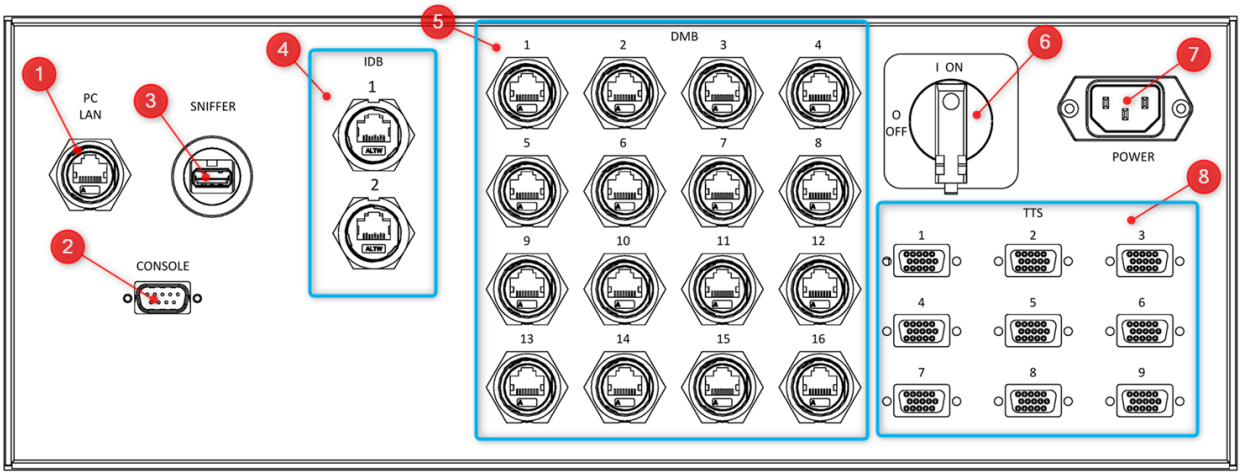


Figure 14: DS Rear Panel

(1) PC LAN
Network connection to Laptop

(2) CONSOLE
Console of VisionBox, serial RS-232, for debugging purposes, left unconnected

(3) SNIFFER
USB Type B, Laptop connection for ProfiTap network sniffer

(4) IDB
10G Network ports for the Image Data Bus, port 1 is connected, port 2 is unused

(5) DMB
100M Network ports for the Data Machine Bus, Powerlink industrial field bus

(6) Mains Switch
Power supply master switch

(7) POWER
Main inlet, IEC-60320 C14

(8) TTS
TTS trigger connectors, Device side [see 5.4.1]

5.4.1. Transport Timing Signals (TTS)

The TTS connector complies to [Ref 1.] chapter 5 Transport Timing Signals (TTS).

Pin	Name
1	BP+
6	BP-
3	TC+
8	TC-
9	RESET
14	READY
11	+5V
5	reserved
7	reserved
10	reserved
12	reserved
2	GND
4	GND
13	GND
15	GND

Table 4 : TTS, D-Sub HD 15-pin

5.5. Device Node Controller

The Device node controller is housed in a 19" rack unit, occupying 2 height units (2U). One Device node controller incorporates four identical nodes (Node n, Node n+1, Node n+2 and Node n+3), each independent from each other. Four such 19" units are installed in the DS rack, providing 16 device nodes at all.

The Device node controllers are cabled inside the rack and user access to its connectors is usually not required.

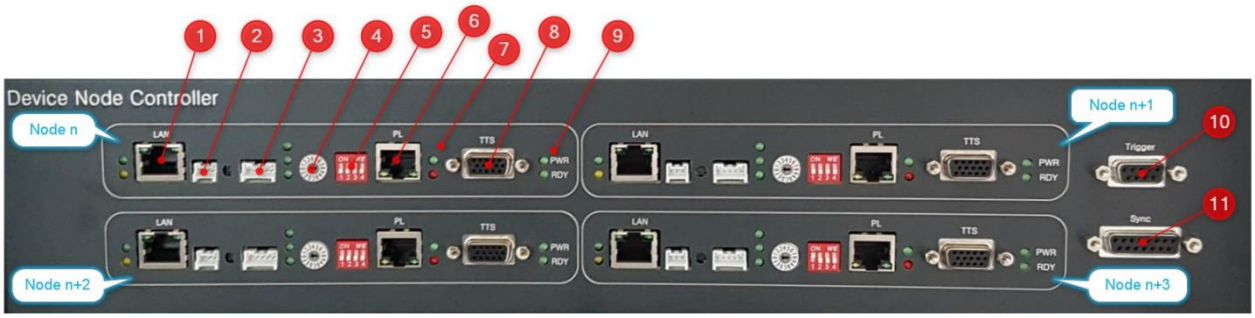


Figure 15: Device Node Controller Front Panel

(1) LAN 1G, Simulation Control
(2) Serial Console, for debugging purposes
(3) General purpose digital IO, for debugging purposes
(4) Powerlink Node ID, bits [3:0]
"0" ... Node ID = 1
"1" ... Node ID = 2
...
"F" ... Node ID = 16
(5) Configuration Switch
(6) Powerlink Network
(7) Powerlink LEDs
(8) Device side TTS
(9) TTS Ready (RDY) and Power (PWR) LED
(10) Trigger, not used
(11) Sync, not used

6. GUI-PC Laptop

Each simulator is delivered with a laptop to interact with the simulation instance itself and to conduct testing procedures. The laptop has Microsoft Windows 10 installed and provides the tools listed in Section 11.4.

The main feature used is the internet browser to access the Test Controller GUI [9.2].

Figure 16 shows a typical DS desktop, with icons for the most relevant tools and programs.

Figure 16: GUI PC Desktop

(1) PuTTY tools [12.4]
(2) WinSCP [12.2]
(3) LibreOffice
(4) WireShark [12.1]
(5) Transport Simulator Tool [12.6]
(6) Flutter Detector Tool [12.11]
(7) PicoScope [12.5]
(8) Firefox browser
(9) Notepad++ text editor
(10) Edge internet browser
(11) Shortcut to *C:\CDI2*
(12) Shortcut to *C:\CDI2-Installation*
(13) Shortcut to *C:\Acceptance Tests*
(14) Restart test controller [11.2]

6.1. CDI2 Folders

CDI2 specific files are stored in following folders.

C:\CDI2-Installation

This folder contains all installation packages and tool sets for both the Windows PC and the Linux computers. This folder is provided as part of the delivery and shall be treated read-only. Installation of any component will use files/packages from this folder.

Recover scripts for Windows

Windows\002_InstallCDI2-SW.bat

This batch script can be used to recover the CDI2 specific windows installation on the GUI-PC, but keeping windows untouched. It installs the required programs and recreates the *C:\CDI2* folder.

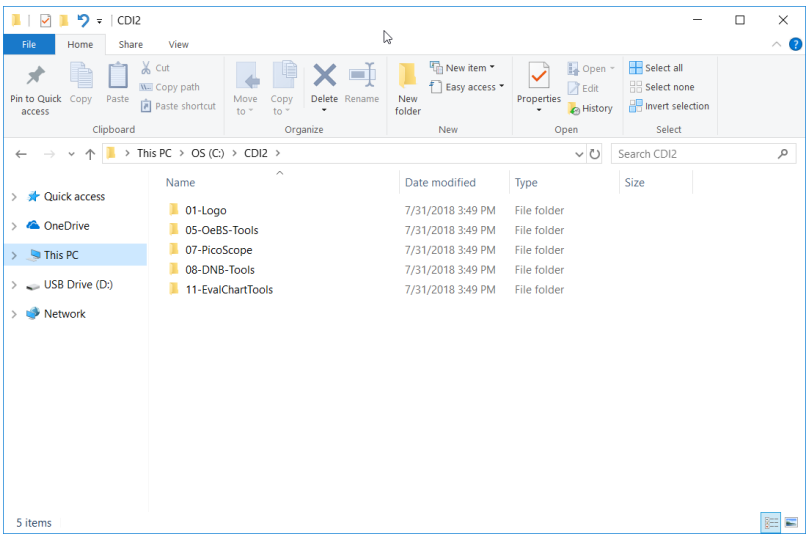
Windows\003_InstallCDI2-plugins.bat

This batch script can be used to update the Wireshark CDI2 plugins automatically.

C:\CDI2

This folder contains the tools for the Windows PC.

01-Logo	... Images for desktop background
05-LibreOffice-Tools	... Tools for flutter analyses, image quality analyses and generator for DS testcases
07-PicoScope	... Settings files for the PicoScope USB scope
08-DNB-Tools	... Control tool for the transport simulator and the flutter detector tool
11-EvalChartTools	... Tools for analysing testchart images



6.2. CDI2 Software and USB Sticks

A set of two USB Sticks is provided for the recovery of the Windows based GUI-PC. The "Recover Stick" is bootable and contains a full disk image, whereas the "Data Stick" contains all files to recover the CDI2 specific installation in folders *C:\CDI2-Installation* and *C:\CDI2*.

For an update of the simulator software on the Linux computers in the BSMS/DS rack refer to the instructions in chapter 7.2.4.

The "Recover Stick" contains a full disk image and it shall be used in emergency cases only, e.g. if the windows system is broken or when a disk is to be replaced. To recover the disk image, first boot from the "Recover Stick", follow the screen menu to start the Acronis True Image tool. Use F12 at boot to select the USB stick as boot device (UEFI BOOT: UEFI: SanDisk, Partition 1). Select the provided True Image backup file from same stick, select target partitions on disk C: and start the restore process. Please note, that this procedure deletes all files on the GUI-PC. If possible, do not recover all partitions, try to recover only the windows partition first.

The "Data Stick" contains following folders

CDI2-Development

contains development data, not needed for normal operation

CDI2-Installation

all files needed to install the CDI2 specific programs on the GUI-PC and on the BSMS/DS racks

CDI2-Manuals

contains user manuals

To recover or restore the CDI2 specific installation follow these steps.

1. backup and remove existing folders *C:\CDI2-Installation*, *C:\CDI2-Manuals* and *C:\CDI2 (if applicable)*
2. copy folder *CDI2-Installation* to *C:\CDI2-Installation*
3. copy folder *CDI2-Manuals* to *C:\CDI2-Manuals*
4. navigate to *C:\CDI2-Installation\Windows*
5. close open applications, if applicable, e.g. WinSCP, PicoScope
6. double-click and *run 002_InstallCDI2-SW.bat*
   Follow the given instructions, use "spacebar" to continue when "Press any key to continue ..." is displayed. See Figure 17, Figure 18 and Figure 19.
   The installation of windows programs (e.g. WinSCP, PicoScope, Firefox) opens the tool specific dialogs. Guide the installation of each tool specifically, default settings are sufficient in most cases.
7. Check if *C:\CDI2* has been created.
8. Reboot computer
9. navigate to *C:\CDI2-Installation\Windows*
10. double-click and run *003_InstallCDI2-plugins.bat*
11. Follow the given instructions, use "spacebar" to continue when "Press any key to continue ..." is displayed.
12. check if tool installation was successful
    refer to chapters 12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 12.8 and 12.11

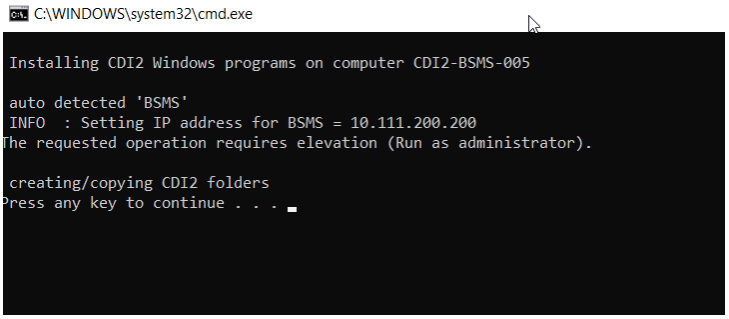


Figure 17: Starting *002_InstallCDI2-SW.bat*

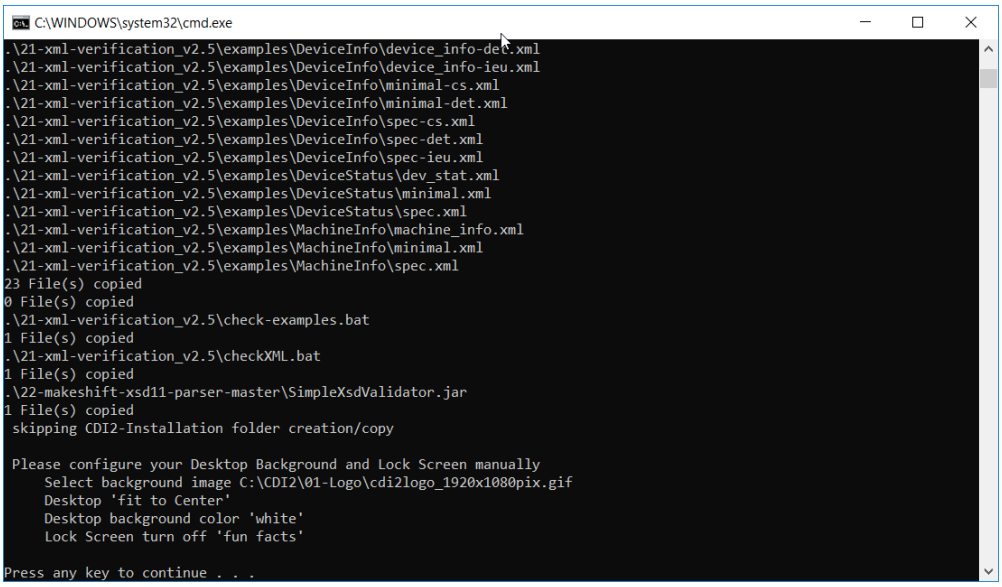


Figure 18: *002_InstallCDI2-SW.bat (continued 1)*

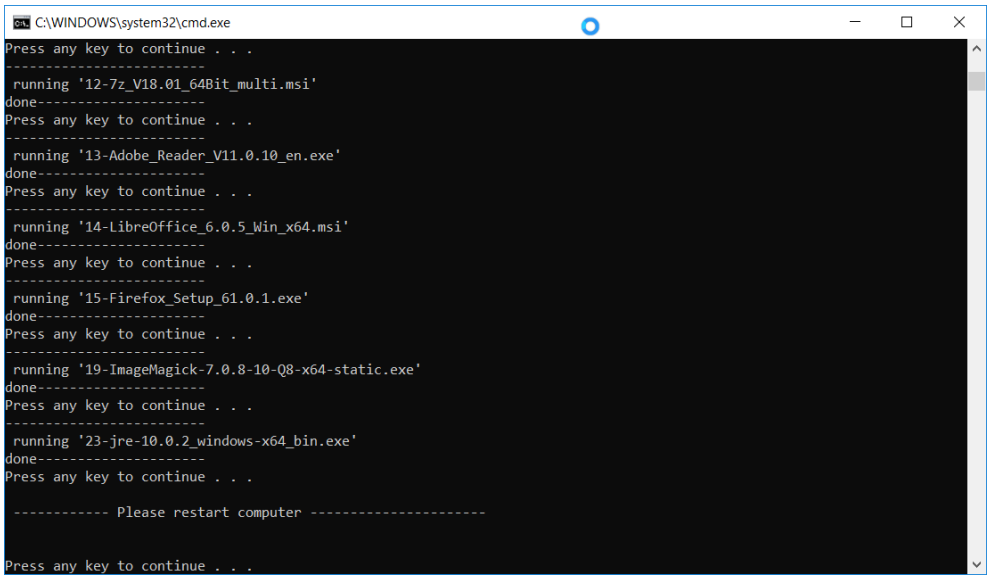


Figure 19: *002_InstallCDI2-SW.bat completed*

7. Putting it into Operation

7.1. Cable Connections

Connect the Transport Simulator, BSMS and DS as shown in Figure 20 using the supplied cables. The standard cables for DMB, TTS and IDB are 10m in length. Refer to chapters 3.3 and 5.3 for the details on rear panel connectors as well.

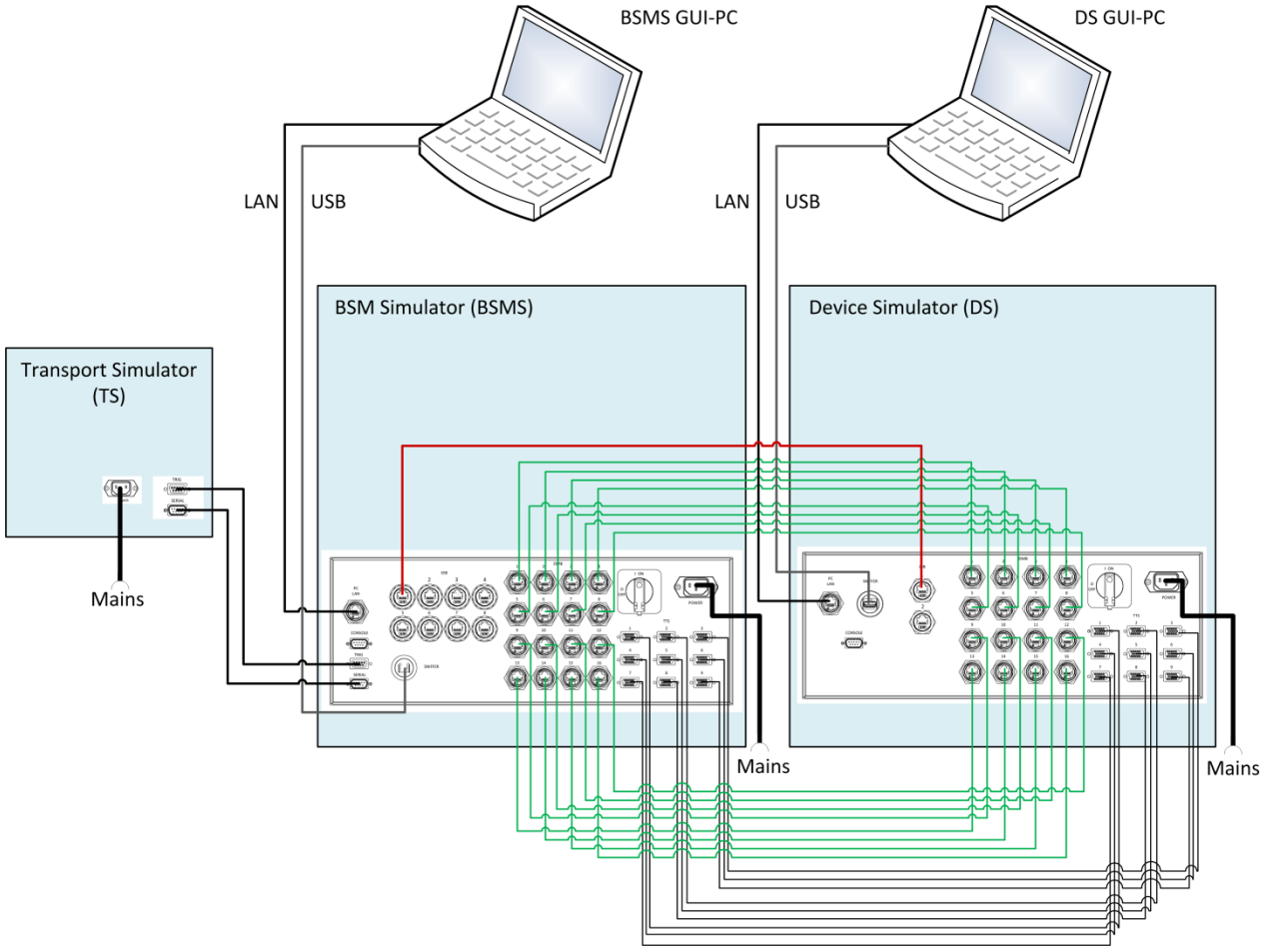


Figure 20: Connection diagram

For the DMB and TTS connections a 1:1 scheme is used. This means, that each BSMS connector has a corresponding connector on the DS side. BSMS/DMB[1-16] are connected to DS/DMB[1-16] and BSMS/TTS[1-9] are connected to DS/TTS[1-9].

7.2. Startup and Basic Operation

Use this procedure for basic startup from power off:

1. Switch on the power supply of TS, BSMS and DS, make sure the Laptops are starting as well. Wait for about 2 minutes boot time.
2. BSMS: wait until the browser can connect to the simulator start page (10.111.200.70). Refer to chapter 9.2 for details.
3. DS: wait until browser can connect to the simulator start page (10.111.200.71). Refer to chapter 9.2 for details.

Test Transport Simulator Control

1. Make sure the TS control on the control panel mounted at the TS is set to “Online”
2. Start the transport simulator tool [refer to 12.6], select proper serial port, select desired speed, e.g. 4.0 m/s, start transport and stop transport. The transport simulator shall follow the issued commands.

   For detailed instructions for using the TS refer to [Ref 3.]

7.2.1. Running a Test Case

Typically running a test consists of the following steps:

1. Select a test set
2. (optional) Change parameters
3. Activate
4. Start
5. Monitor Output
6. (optional) Stop
7. Download Results

Use Abort if test does not respond to Stop.

7.2.2. Shutdown

To perform a controlled shutdown, select the test set “BSMS-999 Shutdown” or “DS-999 Shutdown”, activate and start it. The status output will be updated for every device that has started the shutdown sequence. The GUI will become unreachable as soon as the Simulation Computer initializes its shutdown sequence. Wait at least two minutes after starting the test set. Then switch the mains switch to the off position.

7.2.3. Retrieve Version Information

To retrieve all software versions installed on the various components select, activate and start the test set "BSM-003 Get Version Infos" or "DS-003 Get Version Infos".

At completion of the testcase the status message shows the version string of the detected software, e.g. BSM Version = SW_CDI2SIM_V27_20200129 (Figure 21).

Detailed version information can be downloaded. Open the download archive and open folder "*IDB_BSM/log*" on the BSMS or "*IDB_DEV/log*" on the DS.

It contains files "***versions-summary.txt***" and ***"versions.log".***

"versions-summary.txt"

Shows the version information in an easy-to-read compact form (Figure 22).

"***versions.log***"

Contains the raw information of the package managers, with sections for each component with a list of all installed CDI2 related packages. This file is mainly provided for debug purposes.

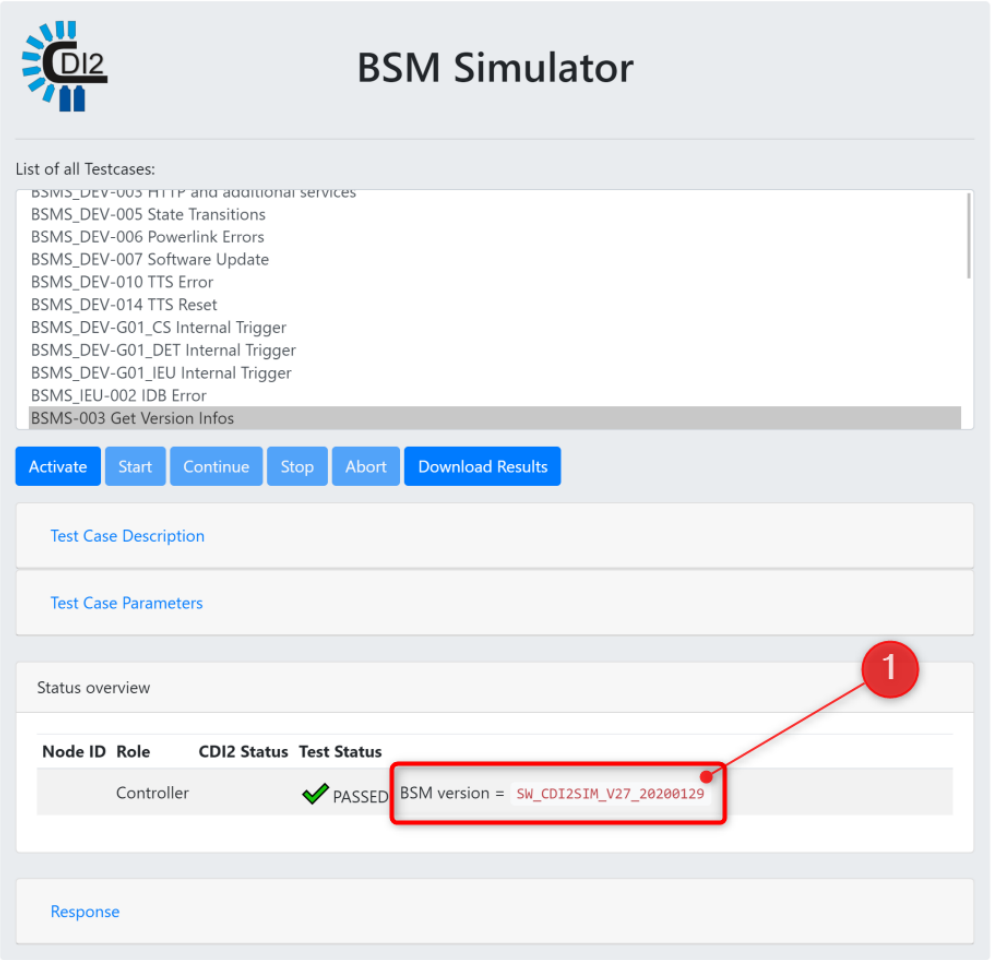


Figure 21: Version message on testcase GUI

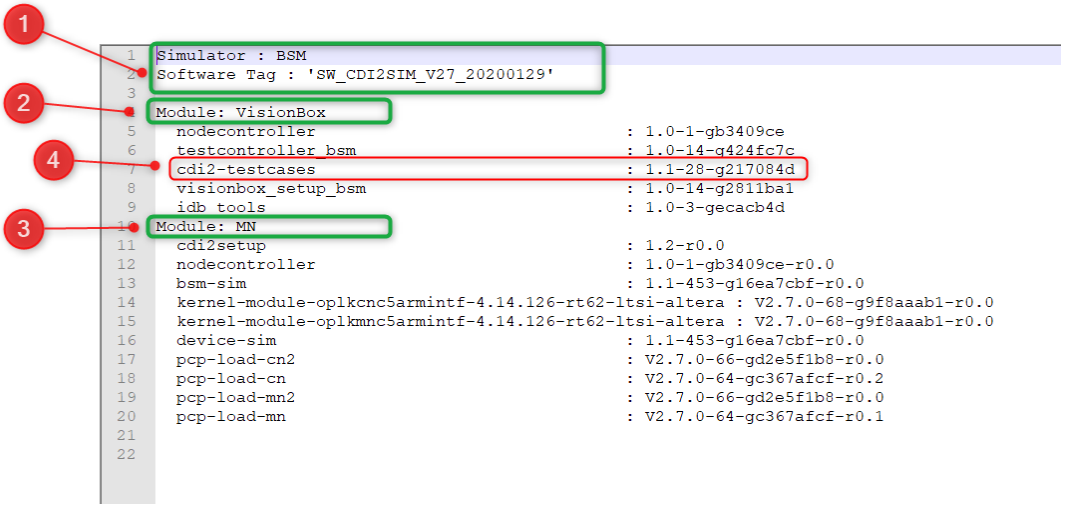


Figure 22: Compact version information version-summary.txt example

(1) Component information with software tag string
(2) Module name
(3) Package name and version
(4) Module name

7.2.4. Simulator Software Update

The software installation of the simulators consists of a Windows system on the GUI-PC and the actual simulation software of the Linux nodes in the BSMS/DS racks.

This chapter describes to update of the Linux nodes. For an update or recovery of the Windows based GUI-PC refer to chapter 6.2.

The software installations on the Linux nodes use standard package managers *dpkg* and *opkg*. *dpkg* is used on the VisonBox and *opkg* on the node controllers. Thus, a specific software version consists of a set of *.deb* (for *dpkg*) and *.ipk* (for *opkg*) packages. When a software update is supplied, the desired software packages are provided and a detailed description of how to install the packages is included.

For the installed system and for future updates, above packages are packed together with a self-extracting installer into a single installer file. So, the user can run the full installation by issuing a single command (see 7.3 for a detailed installer description).

A software delivery package contains a *CDI2-Installation**Linux* folder with all installation files, including a self-extracting installer to update the system with a single command.

Self-extracting installer

.\UpdateScripts\cdi2-installer-<version_tag>.run

Node Controllers

.\EnclustraPackages	*... contains all software packages (.ipk)*
.\EnclustraImage	*... SD card image for initial installation*

VisionBox

.\VisionBoxPackages	*... contains all software packages (.deb)*
.\VisionBoxImage	*... files for initial installation*

VisionBox x86

.\ x86_64_packages	*... contains all software packages (.deb)*
.\x86_64_Ubuntu_Installation	*... files for initial Ubuntu installation*

Use the following steps to install a specific software release on the BSMS/DS racks. The described procedure can be used to recover the pre-installed version or to install future updates. If desired, the update procedure can be used to switch between older and newer versions and vice-versa as well.

Important Note

User specific simulator settings (e.g. tc-correction value, chapter 7.8) are kept in the global file */usr/share/cdi2/testcontroller/**default-params.json***. The software update process keeps this file and does not overwrite it. Nevertheless, it is recommended to make a backup of this file before running a software update, e.g. download it with WinSCP to the GUI PC.

Execute the self-extracting installer on the BSM/DS VisionBox simulation computer as shown in the following examples.

Full installation on VisionBox platform

```
./cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run -- --BSM
./cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run -- --DEV
```

On an x86 platform the sudo is needed to provide root privileges to the installer

```
sudo ./cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run -- --BSM
sudo ./cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run -- --DEV
```

Please see 7.3 for a detailed installer description.

Software Update Step-by-Step Instructions

In following, step-by-step instructions to update a simulator are given. As an example, the update of a BSMS is described, with the execution of the *cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run* update package on the BSMS VisionBox computer (10.111.200.70). The update of a DS is equivalent, except that the update package is executed on the DS VisionBox (10.111.200.71). See chapter 8 for login information details.

Step 1: Connect to the VisionBox

Use WinSCP (chapter 12.2) to connect to the BSM VisionBox. On the login screen (Figure 23) select the BSM VisionBox site (1) and do a Login (2).

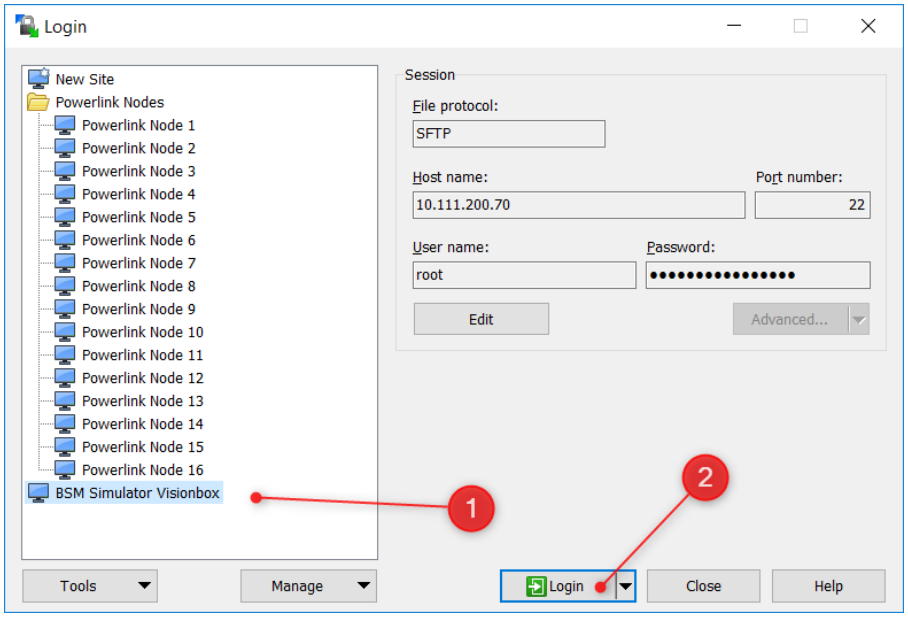


Figure 23: WinSCP login screen

Step 2: Upload the installation file

Figure 24 shows the WinSCP screen after successful connection to the VisionBox computer.

Navigate to the local folder (1) and to the remote folder */root*/tmp (2). Upload the respective installer file *cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run* (3) to the remote folder (4). Use the right-click command or drag&drop to upload the file.

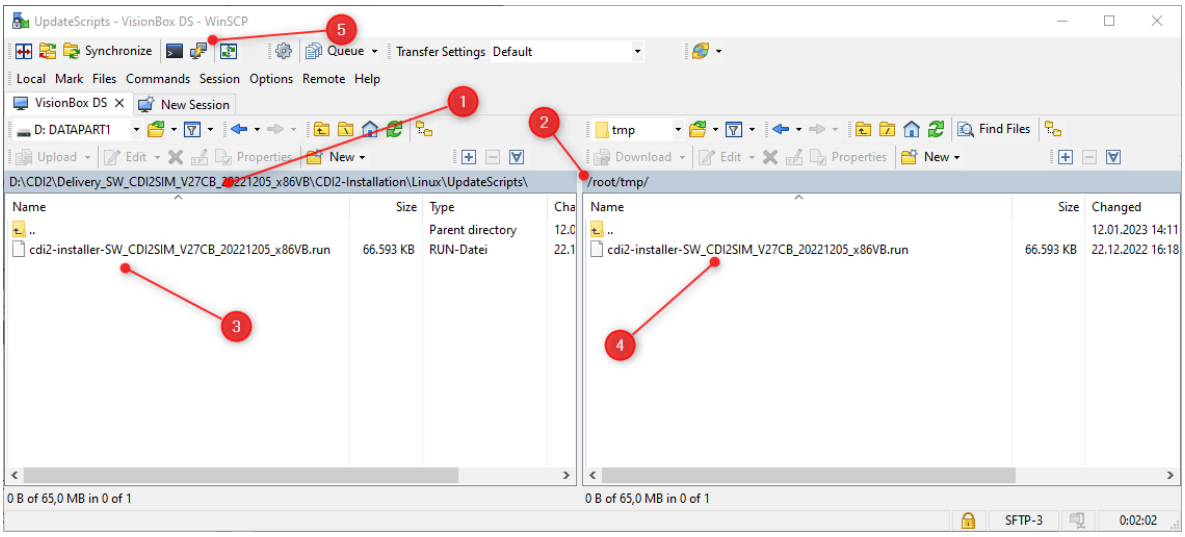


Figure 24: Upload Installer File

Step 3: Open console and start the installer script

Open an ssh console on the Visionbox, either use WinSCP (5) or open a standalone PuTTY console.

Watch the new PuTTY window and enter password to complete login. Then use the following shell commands to run the software update.

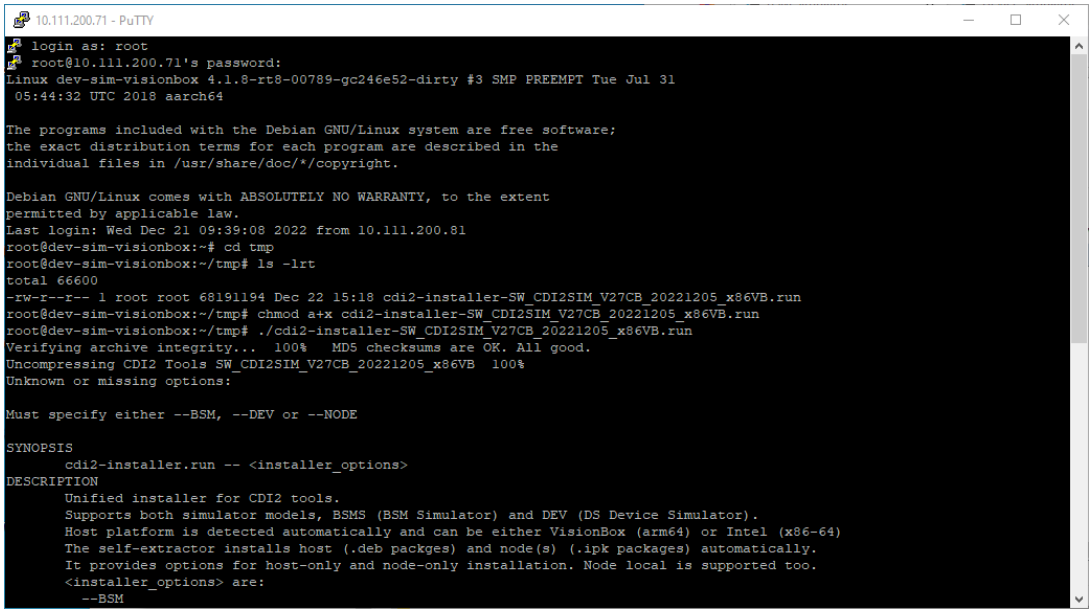


Figure 25: PuTTY shell

List files in directory, make sure that *cdi2-installer-<sw_tag>.run* is listed

```
ls -lrt
```

Make *cdi2-installer-<sw_tag>.run* executable

```
chmod a+x cdi2-installer-<sw_tag>.run
```

Check the integrity of the archive

```
./cdi2-installer-<sw_tag>.run --check
```

Run the installation on an BSM

```
./cdi2-installer-<sw_tag>.run -- --BSM | tee bsm.run.log
```

The installer determines the platform and asks for confirmation. Once confirmed, watch the installation progress for successful completion (see Figure 26). The update process will need about 5 minutes. At the end of the installation process a reboot of all nodes is requested. Thus, a "connection closed" alert is asserted by PuTTY. All output text during the install processes is stored in *bsm.run.log* also. Download the file to the GUI-PC (e.g. with WinSCP) and keep it for reference.

After reboot the new software is up and running. Check the installed software versions according 7.2.3 and double-check the global settings in */usr/share/cdi2/testcontroller/default-params.json* (see chapter 7.8)

```
root@bsm-sim-visionbox:~/tmp# ./cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run
-- --BSM
Verifying archive integrity...  100%   MD5 checksums are OK. All good.
Uncompressing CDI2 Tools SW_CDI2SIM_V27CB_20221205_x86VB  100%
Platform/architecture=arm64
Hostname=bsm-sim-visionbox
   Static hostname: bsm-sim-visionbox
         Icon name: computer
        Machine ID: f72ca990d2864af5a8dcb7b59e36e8a5
           Boot ID: eeaa9211accb4085ab527c5f2e276c08
  Operating System: Debian GNU/Linux 9 (stretch)
            Kernel: Linux 4.1.8-rt8-00789-gc246e52-dirty
      Architecture: arm64
Do you wish to start installation [y/N]? y
Checking install files .....
Starting package installation .....
DEB installation .....
...
```

Figure 26: Installation progress

7.3. Unified Self-Extracting Installer

The install script is a self-extracting file made with the *makeself* tool (see https://makeself.io). It contains all required .ipk and .deb packages and an installation script. It extracts the packages to a temporary folder and runs the installation script with the proper package manager commands to complete the installation automatically.

7.3.1. Makeself Options

makeself options are useful for checking the integrity of the archive or extract individual package files for debugging purposes.

Examples:

`./cdi2-installer-<sw_tag>.run --check`
Checks the integrity of the archive.

`./cdi2-installer-<sw_tag>.run --info`
Provide version infos.

`./cdi2-installer-<sw_tag>.run --target ./packages --noexec`
Extracts the archive into to a target directory, e.g. *./packages*, but does install anything. The option is used to extract the contained .ipk and .deb packages into single files.

`./cdi2-installer-<sw_tag>.run --help`
Print the *makeself* help message.

7.3.2. Installer Options

The unified installer is suited for BSMS and DS. The installer is to be started on a BSMS/DS VisionBox computer. It supports two platforms, VisionBox (arm64) and x86 PC with Ubuntu. It installs the local packages first and continues to update the DMB nodes (mn, cn00, cn01, …) remotely. Commandline options can be used to parametrize the installation process, e.g. exclude DMB node controllers from installation or prevent reboot.

If executed on a DMB node controller, the installer can be directed to install just .ipk packages (see 0).

Please note that installer options are preceded by an extra `'--'`.

Running the installer without options will print a help text.

```
cdi2@bsm-sim-visionbox:~$ ./cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run
Verifying archive integrity...  100%   MD5 checksums are OK. All good.
Uncompressing CDI2 Tools SW_CDI2SIM_V27CB_20221205_x86VB  100%
Unknown or missing options:

Must specify either --BSM, --DEV or --NODE

SYNOPSIS
       cdi2-installer.run -- <installer_options>
DESCRIPTION
```

```
       Unified installer for CDI2 tools.
       Supports both simulator models, BSMS (BSM Simulator) and DEV (DS Device Simulator).
       Host platform is detected automatically and can be either VisionBox (arm64) or Intel (x86-
64)
       The self-extractor installs host (.deb packges) and node(s) (.ipk packages) automatically.
       It provides options for host-only and node-only installation. Node local is supported too.
       <installer_options> are:
         --BSM
              Install BSM Simulator (!!! select platform carefully !!!!)
         --DEV
              Install Device Simulator (!!! select platform carefully !!!!)
         --host-only
              Install host packages only (.deb)
         --node-only
              Install node(s) packages only (.ipk)
         --NODE
              Install on node locally (.ipk), installer shall be started on a node (e.g. cn00)
         --no-reboot
              Do not reboot host after .deb installations
         --help
              show this help
MAKESELF
      The installer is built with the makeself tool. Use makeself options for more specific
functions.
      Examples:
          obtain makeself help
             cdi2-installer.run --help
          list version info
             cdi2-installer.run --info
          extract packaged files into a folder, without installing
             cdi2-installer.run --noexec --target cdi2-packages
EXAMPLES
       ./cdi2-installer.run -- --BSM
       ./cdi2-installer.run -- --DEV
       ./cdi2-installer.run -- --DEV --host-only --no-reboot
       ./cdi2-installer.run -- --DEV --node-only
       ./cdi2-installer.run -- --NODE
```

The default setting performs a full installation. Options `--host-only` and `--node-only` can be used for selective installation on either host (VisionBox) or nodes (DMB node controller). Use `--no-reboot` to prevent host rebooting at the end of the installation.

For debugging and testing purposes you can unpack installer using following options:

```
./cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run --noexec --target cdi2-packages
```

This command unpacks the content into target folder *cdi2-packages*. Refer to *cdi2-packages /extract_unified.sh*, which is the actual installation script. Use it to see examples for specific installation commands.

7.3.3. Stand-Alone DMB Node Controller Installation

The unified installer can be instructed to perform a local install on a DMB node controller, instead running it on the BSMS/DS VisionBox computer. This allows software installation on stand-alone node controllers, including Enclustra EB1 boards (see 0), without the need for a BSMS/DS VisionBox computer.

Use the ftp tool to copy the installer to the node and run the installer with the `--NODE` option. It will install the .ipk packages locally and reboot. Please note that the update will need about 5 minutes usually.

If installed on an Enclustra EB1, the Node ID must be configured manually, as the EB board is missing the rotary switch hardware. Create a new file named NODE_ID with the desired number in the home directory.

e.g. set Node ID = 2 (Node 3)

```
cd
echo "2" > NODE_ID
```

7.3.4. Setup Stand-Alone Mode

A DMB node controller can be configured to run in stand-alone mode, e.g. if operated as a stand- alone detector, without a simulator attached to it. In this case the node can be configured for automatic device-sim startup, using a systemd service.

Setup a systemd service named *device-sim* to run *device-simulator.sh*:

Create file /lib/systemd/system/**device-sim.service**

```
#-------------------------------------------------------------------------------
#
# device-sim.service
#
# Systemd service unit configuration for the CDI2SIM device simulator
#
# (c)2018 AIT
#-------------------------------------------------------------------------------
[Unit]
Description=CDI2 Device Simulator
After=network.target

[Service]
Type=simple
User=root
WorkingDirectory=/home/root/
ExecStart=/home/root/device-simulator.sh
ExecStop=/usr/bin/pkill -9 device-sim
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Create an autostart script for the device simulator, `device-simulator.sh`

```
#!/bin/bash

die() { echo "$@" 1>&2 ; exit 1; }

if [ -z "$1" ]; then
        source /etc/cdi2/system.conf || die "failed to open cdi2 configuration"
        NODEID=$CDI2_NODEID
else
        NODEID=$1
fi

NODEID=$((NODEID + 1))

TCDIR=/home/root/cdi2-testcases/DS_BSM-G01
cd $TCDIR

INFO=$(cat META.json | grep device_infos | sed 's/"\(.*\)",/\1/')
SCRIPT=$(cat META.json | grep script | sed 's/.*: "\(.*\)",/\1/')
MAC=$(cat META.json | grep 00:60 | sed 's/"\(.*\)",/\1/')
```

```
TTS_SUPPORT=0

device-sim -v9 --node-id $NODEID \
  --rt-cpu-affinity 1 \
  --result-dir /var/tmp \
  --log-dir /var/tmp \
  --config config_device.xml \
  --device-info $INFO \
  --test-file $SCRIPT \
  --mac-address $MAC \
  -t$TTS_SUPPORT
```

Set file executable

```
chmod a+x device-simulator.sh
```

Above `device-simulator.sh` runs a testcase in a certain folder (see variable TCDIR).

Create the desired testcase in */home/root/cdi2-testcases/DS_BSM-G01*. The folder shall be formatted the same way as created by the testcontroller in folder */var/tmp*, when a testcase is activated. Easiest way, run the desired testcase from the GUI on another simulator and copy */var/tmp/ DS_BSM-G01.*

```
root@cn00:~# ls /var/tmp/DS_BSM-G01
config_device.xml  device_sorting_test.lua  META.json     ui_parameters.lua
device_infos       log                      RESULTS.json
```

Alternately, adapt `device-simulator.sh` to your specific needs.

Finally, start/reload/enable the services and reboot.

```
systemctl daemon-reload
systemctl start device-sim
systemctl status device-sim
# enable your service (if not enabled already)
systemctl enable device-sim

# node reboot
root@cn00:~# reboot
```

7.3.5. Common Installation Issues

Issue (A) – device-sim fails to start

If *device-sim* fails to start on a DMB node controller, this can be caused by a missing kernel module *oplkcnc5armintf*. Use lsmod to check if *oplkcnc5armintf* present.

```
root@cn00:/var/tmp# lsmod
Module                  Size  Used by
oplkcnc5armintf        46788  0
atsha204_i2c            7394  0
root@cn00:/var/tmp#
```

If not present, most commonly the *cdi2setup* service failed. Use following command the check and re-enable cdi2setup.

```
root@cn00:/var/tmp# systemctl status cdi2setup
● cdi2setup.service - CDI2 Simulator System Configuration
   Loaded: loaded (/lib/systemd/system/cdi2setup.service; enabled; vendor preset: enabled)
   Active: active (exited) since Mon 2022-05-16 07:33:44 UTC; 1 weeks 0 days ago
  Process: 1270 ExecStart=/usr/sbin/cdi2setup.sh (code=exited, status=0/SUCCESS)
 Main PID: 1270 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/cdi2setup.service
```

Enable cdi2setup service, if not enabled

```
root@cn00:/var/tmp# systemctl enable cdi2setup
```

Reboot node

```
root@cn00:/var/tmp# reboot
```

Issue (B) – update to earlier version fails

Switching back to an early version (e.g. SW_CDI2SIM_V27_20200129) can fail, due to a file system is mounted read-only. This is caused by a later introduced feature to check the file system at boot time.

In such a case the system will not operate correctly, nor it can be updated to a newer version.

The situation can be fixed by remounting the file system correctly.

Use script `fix-ro-mount.sh` (provided in *folder Delivery_SW_CDI2SIM_V27CB_20221205_x86VB\CDI2-Installation\Linux\UpdateScripts*) or create with following content,

```
#!/bin/sh

echo "fix ro mount issue after re-installation of SW_CDI2SIM_V27_20200129."

mount -o remount,rw tmpfs /sys/fs/cgroup
mount -o remount,rw /dev/sda1 /

systemctl restart testcontroller
systemctl restart nodecontroller

cat <<EOX > /etc/fstab
/dev/sda1            /                    auto       defaults              1  1
proc                 /proc                proc       defaults              0  0
EOX
```

```
cat /etc/fstab
```

Run script

```
chmod a+x fix-ro-mount.sh
./fix-ro-mount.sh
```

Reboot node

```
reboot
```

7.4. Intel x86 VisionBox Hardware Replacement

Starting with software version **SW_CDI2SIM_V27CB_20221205_x86VB**, the BSM/DEV simulation computer “VisonBox LE MANS” may be substituted by an “Intel x86 PC” platform.

The installation on the new platform is fully compatible (e.g. unified installer, use same IP addresses), thus allowing a 1:1 hardware replacement. Please note, that only one platform, either “VisionBox LE MAN” or “Intel x86 PC” may be active, at the same time.

The Intel x86 PC shall be the preferred hardware for future developments due to following features:

- Use of commercially available standard computer
- Use of commercially available 10G network cards
- Use standard Linux Ubuntu software OS
- IDB sender/receiver performance up the maximum line rate of 10 Gbps
- DDR main memory > 4GB (e.g. use 64GB to store more banknote images)

An upgrade of existing simulator hardware to Intel x86 PC is recommend, if IDB date rate > 6Gbps shall be utilized.

The software setup consists of two steps

1. first install and setup Ubuntu
2. install CDI2 software

7.4.1. Linux Ubuntu Installation

Install Ubuntu 22.04.1 LTS from a bootable USB stick as described in the tutorial. Reference copies of the *ISO installation media* and *balenaEtcher* program used to create a bootable USB stick have been preserved in folder `x86_64_Ubuntu_Installation`.

It is recommended to select UTC as neutral timezone during installation. Ubuntu discourages direct root access, thus a user `cdi2` (with password `cdi2`) must be created to allow administrative access to the machine.

After the base system has been installed, prepare the `cdi2` user for remote SSH access by creating a RSA key pair in `~/.ssh`:

```
$ ssh-keygen
```

If your site policy allows internet access during installation, install additional SW needed for administration and CDI2 requirements:

```
$ cd x86_64_Ubuntu_Installation
$ sudo ./offlineinstall.sh --install-online
```

For offline installation, reference copies of these debian packages and their dependencies have been stored in folder `x86_64_Ubuntu_Installation/packages.`

```
$ cd x86_64_Ubuntu_Installation
$ sudo ./offlineinstall.sh --install
```

7.4.2. CDI2 Installation

for common CDI2 software setup follow instructions in chapter 7.2.4

Please note, in Ubuntu the sudo command is needed to provide root privileges to the installer

Example:

```
sudo ./cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run -- --BSM
sudo ./cdi2-installer-SW_CDI2SIM_V27CB_20221205_x86VB.run -- --DEV
```

7.4.3. Qualified Hardware

Software tools generally support Intel x86_64 PC platforms running Ubuntu OS. The tools have been installed and tested on following platforms:

- Option (1) DELL Precision 3930
- Option (2) Kontron KISS 1U Short V3 CFL (recommend)

Network Adapters for 10Gbps IDB

Network adapters for 10GbE can be either use SFP+ or 10GbE BASE-T RJ-45. For compatibility reasons, the BSMS can be equipped with SFP+ and the DS with 10GbE BASE-T adapters.

As CDI2 specifies 10GbE BASE-T for IDB, both computers may use 10GbE BASE-T network cards as well. In this case the BSMS will use a RJ-45 port on the 10G network switch instead SPF+.

General Hardware Requirements

- Intel i7-9700E (8 Core / 2.6GHz) or better
- PCIe 3.0 (8GT/s)
- Intel X550 or X710 LAN Card 10GbE BASE-T

Hardware Recommendation

- Use Kontron KISS 1U Short V3 CFL
- Use 10GbE BASE-T network cards for both BSMS and DS, so that both use the same hardware configuration.

Option (1)

DELL Precision 3930 Rack XCTO Base

- Intel Core i7-9700,(8 Core, 12MB Cache, 3.0Ghz, 4.8 Ghz Turbo w/UHD Graphics 630)
- 64 GB, 2 x 32 GB, DDR4-UDIMM-Arbeitsspeicher no ECC
- SATA-SSD-Festplatte (Klasse 20), 2,5'', 512 GB
- Dell Intel X710-T2L Dual Port 10GbE BASE-T Adapter, PCIe Full Height https://www.dell.com/en-us/shop/dell-intel-x710-t2l-dual-port-10gbe-base-t-adapter-pcie-full-height-customer-install/apd/540-bcsc/networking

Important:

Unfortunately, the mechanical dimensions of the DELL Precision 3930 (max. depth) exceed the size of the simulator racks. Regular mounting is not possible w/o keeping the back door open.

From the mechanical perspective, the Kontron KISS 1U Short V3 CFL is the preferred option.

Option (2)

Kontron KISS 1U Short V3 CFL

https://www.kontron.com/en/products/kiss-1u-short-v3-cfl/p155747

- 1HE Rackmount PC, Front silver
- Intel i7-9700E (8 Core / 2.6GHz) with Cooler
- 64GB DIMM DDR4
- 1x SSD 1TB
- 1TB SSD M.2 2280
- Intel X550 LAN Card 10GbE BASE-T (2x 10 GBits/sec.)

Figure 27: Kontron KISS Configuration

7.4.4. VisonBox LE MANS substitution

A simulator, BSMS or DS, can be easily upgraded to an x86 platform. The VisionBox LE MAN can be directly substited by an x86 system, e.g. Kontron KISS. As network interfaces use the same IP settings, just connect the network cables according Figure 28.

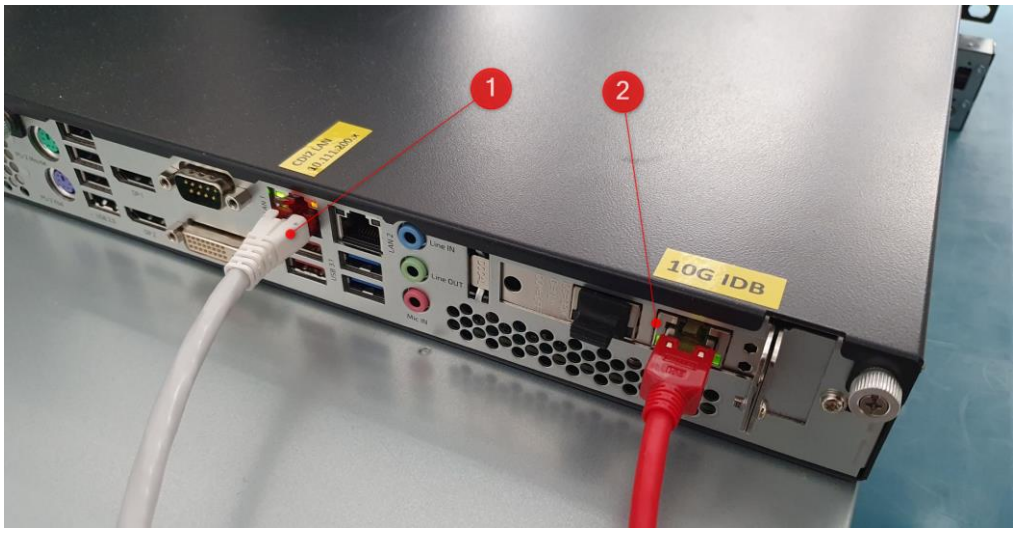


Figure 28: Kontron KISS Network Connections

(1) Simulation Network (10.111.200.xxx)
(2) IDB Network 10G

7.5. CDI Version 2.7C/2.7B Compatibility

The latest CDI2 Version 2.7C [Ref 1.] introduced additional XML attributes, making it incompatible to the previous CDI2 Version 2.7B [Ref 2.] in this respect.

New attributes in *device_info.xml*, *device_config.xml* and <*storage_file*>.*xml* are swUpdateTimeout, gvspPixelFormat and gvspPayloadSize.

The simulator supports Version 2.7C by default but provides an option to support CDI2 Version 2.7B compatibility.

How to instruct the simulator to run in 2.7B compatibility mode

Command line parameter for 2.7B compatibility

bsm-sim / device-sim

--cdi-version-switchover 2.7
Instructs the simulator to run in 2.7B compatibility mode. The simulator will ignore XML attributes swUpdateTimeout, gvspPixelFormat and gvspPayloadSize when reading XML content and will not generate them when writing XML content. Affected files are *device_info.xml, device_config.xml* and <*storage_file*>.*xml.*

Manual edit of *default-params.json* file

The 2.7B compatibility mode of the simulators is activated by means of command line parameters. As the default parameters are stored in the *default-params.json* file, this file can be modified to activate the 2.7B compatibility mode.

The following string pair must be added to each instance in *default-params.json* to supply the proper command line option to the simulator modules (bsm-sim and device-sim).

"--cdi-version-switchover", "2.7"

Navigate to this folder on the BSMS/DS VisionBox with WinSCP (chapter 12.2) or any favourite tool.

/usr/share/cdi2/testcontroller

Make a backup copy of the *default-params.json* and replace it by the *default-params-27b.json* file or edit file accordingly.

After changing the *default-params.json* the Test Controller needs to be restarted, as described in 11.2.

Example of standard *default-params.json* (2.7C mode):

```
{
  "BSM_SIM" :       [ "--rt-cpu-affinity", "1", "--tc-correction", "0" ],
  "DMB_DEVICE_01" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_02" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_03" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_04" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_05" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_06" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_07" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_08" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_09" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_10" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_11" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_12" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_13" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_14" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_15" : [ "--rt-cpu-affinity", "1" ],
  "DMB_DEVICE_16" : [ "--rt-cpu-affinity", "1" ]
}
```

Example of *default-params.json* for 2.7B mode:

```
{
  "BSM_SIM" :       [ "--rt-cpu-affinity", "1", "--tc-correction", "0", "--cdi-version-switchover",
"2.7" ],
  "DMB_DEVICE_01" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_02" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_03" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_04" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_05" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_06" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_07" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_08" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_09" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_10" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_11" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_12" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_13" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_14" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_15" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ],
  "DMB_DEVICE_16" : [ "--rt-cpu-affinity", "1", "--cdi-version-switchover", "2.7" ]
}
```

7.6. Simulated Sorting Test

Use this procedure on BSMS and DS to simulate sorting of 1,000 banknotes, using the internal trigger source.

1. BSMS: Select test set "BSMS_BSM-G01 Sorting", set Test Case Parameters "Trigger Source" to *Internal*, "Manual Mode" to *False* and "Number of Banknotes" to 1,000, activate and start the test set
2. DS: Select, activate and start test set "DS_BSM-G01 Sorting"
3. BSMS: Wait until sorting of 1,000 banknotes has been completed
   Check: After the test, the BSMS shows test result "PASSED"
   Check: After the test, the DS shows test result "PASSED"

7.7. Actual Sorting Test with Transport Simulator

Use this procedure on BSMS and DS to perform sorting of 100 banknotes, using the transport simulator as external trigger source.

1. Insert a blank sheet into the transport of the TS
2. BSMS: Select test set "BSMS_BSM-G01 Sorting", set Test Case Parameters "Trigger Source" to *External*, "Manual Mode" to *True* and "Number of Banknotes" to 100, activate and start the test set
3. DS: Select, activate and start test set "DS_BSM-G01 Sorting"
4. BSMS: Use TS control tool [refer 12.6] to set speed to 3.0 m/s and start transport. Wait until TS has reached nominal speed
5. BSMS: Press "Continue" to start sorting and wait until sorting of 100 banknotes has been completed. Use TS control tool [refer 12.6] to stop transport.
   Check: After the test, the BSMS shows test result "PASSED"

7.8. Speed Compensation of the Transport Simulator

The transport simulator clock output may deviate from the actual speed of the banknote due to slippage. The BSMS incorporates a digital PLL (phase locked loop) circuit to compensate this speed deviation. The default setting of the BSMS is 1:1, which means the clock of the transport simulator is taken as source for the CDI2 transport clock without speed change.

The BSMS offers an option to store a speed compensation factor globally, so that it is applied to any test automatically. Setting of the factor is usually required only once, but the factor is specific for any TS and might vary in time also.

To apply speed compensation, in the first step use the instructions in Acceptance Test Procedure "BSMS - TS transport speed and TC accuracy (BSMS-005)" to find the desired correction factor.

Once the correction factor is known, the BSMS is configured to use this factor permanently for all tests.

To change the factor, perform the following steps:

1. Connect to the BSMS VisionBox using WinSCP [see 12.2]
2. Navigate to the folder */usr/share/cdi2/testcontroller/*
3. Right click on the file *default-params.json*
4. Click *Edit*
5. In the opened file change the value after *–tc-correction* to the desired correction.
   E. g. to slow down the TC output of the BSMS by 4.2% the line should look as follows:

   "BSM_SIM": ["--rt-cpu-affinity", "1", "--tc-correction", "**-4.2**"]

   The correction value is limited to the range of -10% to +10%.
6. Restart test controller to make above settings active [see 11.2]

8. IP Addresses and Login Information

To access the components of the simulator system it might be needed to login on a component. This section provides details on the credentials and connection options to achieve this.

8.1. Login Credentials

Device	**Username**	**Password**
GUI PC	CDI2	cdi2
VisionBox	root	vision
VisionBox x86 Ubuntu	cdi2	cdi2
Node Controller	root	<no password> (empty password string)
NETGEAR ProSAFE GS724T (1G)	<no username>	password
NETGEAR ProSAFE XS712T (10G)	<no username>	password

8.2. IP Address Plan

Name	IP	BSMS	DS	Port/Service
Device Node Controller 1	10.111.200.50		☒	22:ssh, 80:http, 3000:http
Device Node Controller 2	10.111.200.51		☒	22:ssh, 80:http, 3000:http
Device Node Controller 3	10.111.200.52		☒	22:ssh, 80:http, 3000:http
Device Node Controller 4	10.111.200.53		☒	22:ssh, 80:http, 3000:http
Device Node Controller 5	10.111.200.54		☒	22:ssh, 80:http, 3000:http
Device Node Controller 6	10.111.200.55		☒	22:ssh, 80:http, 3000:http
Device Node Controller 7	10.111.200.56		☒	22:ssh, 80:http, 3000:http
Device Node Controller 8	10.111.200.57		☒	22:ssh, 80:http, 3000:http
Device Node Controller 9	10.111.200.58		☒	22:ssh, 80:http, 3000:http
Device Node Controller 10	10.111.200.59		☒	22:ssh, 80:http, 3000:http
Device Node Controller 11	10.111.200.60		☒	22:ssh, 80:http, 3000:http
Device Node Controller 12	10.111.200.61		☒	22:ssh, 80:http, 3000:http
Device Node Controller 13	10.111.200.62		☒	22:ssh, 80:http, 3000:http
Device Node Controller 14	10.111.200.63		☒	22:ssh, 80:http, 3000:http
Device Node Controller 15	10.111.200.64		☒	22:ssh, 80:http, 3000:http
Device Node Controller 16	10.111.200.65		☒	22:ssh, 80:http, 3000:http
BSM Node Controller	10.111.200.66	☒		22:ssh, 80:http, 3000:http
VisionBox BSMS	10.111.200.70	☒		22:ssh, 80:http, 3000:http
VisionBox DS	10.111.200.71		☒	22:ssh, 80:http, 3000:http
BSMS Laptop	10.111.200.200	☒		
DS Laptop	10.111.200.201		☒	

9. Simulator Software

The simulation software is built from a set of components which are described in this section.

9.1. Software Component Overview

The simulation software can be split up into three layers. [1] A web browser which provides the user interface, [2] a Test Controller which controls multiple components and [3] a set of Node Controllers of which each controls a single component (e.g. a DMB node or an IDB sender). These components perform the stimulation and check the system under test. The figure below illustrates a general setup with the connections between these parts.

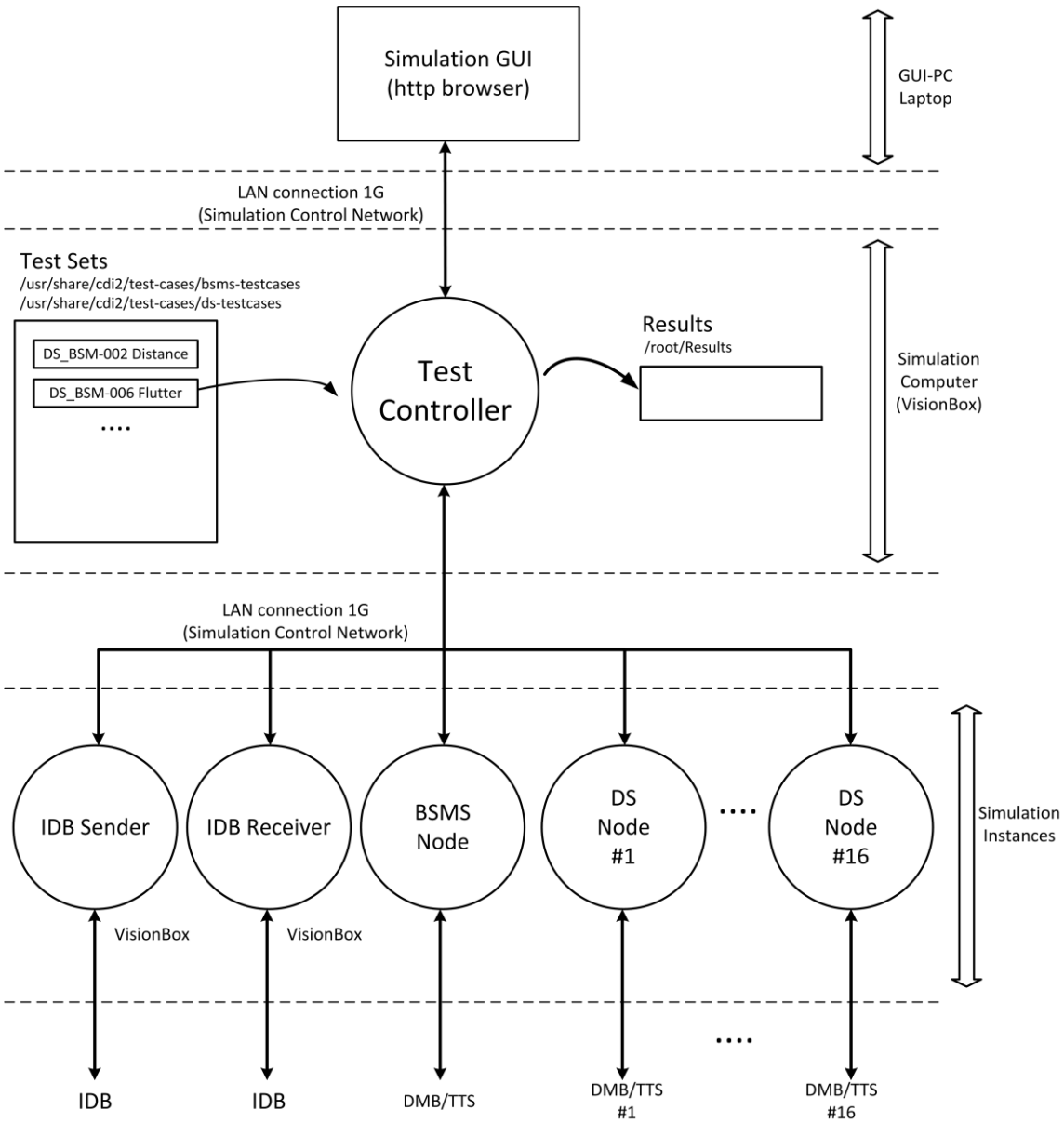


Figure 29: General software architecture for BSMS and DS

Test Controller

The Test Controller provides the user interface on one side and controls a set of Node Controllers on the other. It has access to a collection of test sets. There is one Test Controller instance per simulator (BSMS and DS). The Test Controller runs on the Simulation Computer (VisionBox) and can be accessed via HTTP on port 80.

Node Controller

Each Node Controller is responsible for running one program for the simulation (e.g. bsm-sim). It unpacks needed files sent by the Test Controller, starts and monitors given programs and packages all results after a test run is finished. The Node Controller is controlled via a REST-API over HTTP on port 3000. For DMB and TTS simulation a Node Controller instance runs on every node. For IDB simulation a Node Controller instance runs on the Simulation Computer (VisionBox).

DMB/TTS Simulators (bsm-sim, device-sim)

This component allows the simulation of a single CDI2 BSM or a single CDI2 Device. It can be controlled via Lua scripts which are stored in the Test Set repository and provided via the Test - and Node Controller. These simulators run on the custom Node Controller Hardware.

IDB Sender/Receiver (idbfsender, idbfreceiver)

These components allow the simulation of IDB functionalities. The idbfsender sends images from stored image files to the IDB. It offers the possibility to insert controlled errors on the IDB to test the response of a BSM and Devices. The idbfreceiver receives and checks image streams from the IDB and stores the last 1,000 banknotes received to disk. The IDB tools are executed on the Simulation Computer (VisionBox) and are started by the Node Controller.

SSH Server

For service and maintenance all devices are accessible via SSH on port 22. To access all the Node Controller boards in a DS, the VisionBox offers a terminal multiplexer which can be started on the command line using the command "tmuxinator cdi2-dev".

9.2. Graphical User Interface (GUI)

The CDI2 Simulator uses a web based GUI. Any state of the art web browser can be used, but the Edge Browser on Windows 10 is recommended.

The browser runs on the BSMS and DS GUI Laptop and it connects to the web server of the respective VisionBox, which provides proper content to get interactive access to the simulation.

BSMS URL = http://10.111.200.70

DS URL = http://10.111.200.71

The start page of the browser is set properly for BSMS and DS on the accompanying laptops, so that the connection to the simulator user interface is established automatically at start-up.

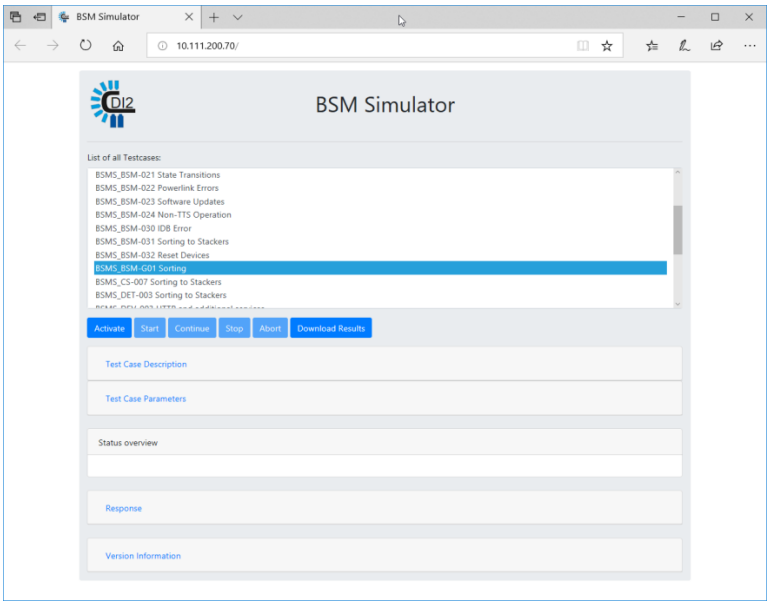


Figure 30: BSMS start page

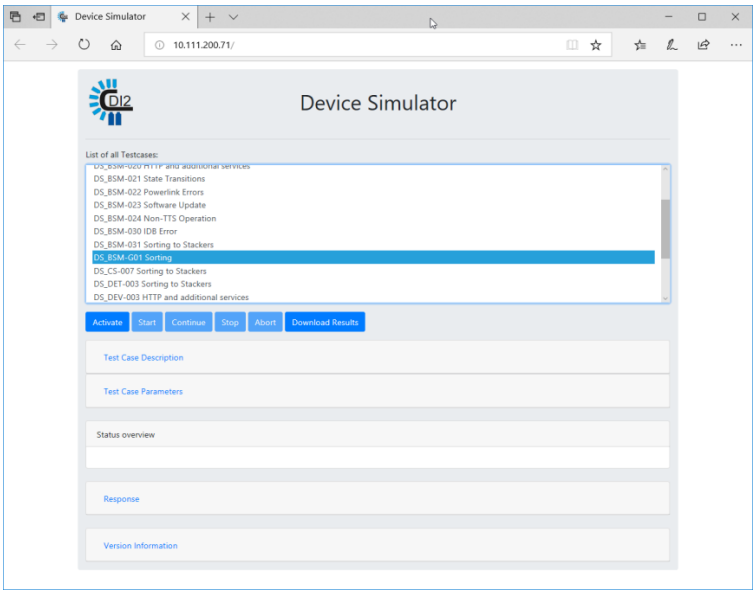


Figure 31: DS start page

9.2.1. Panels

The user interface provides a list of different panels.

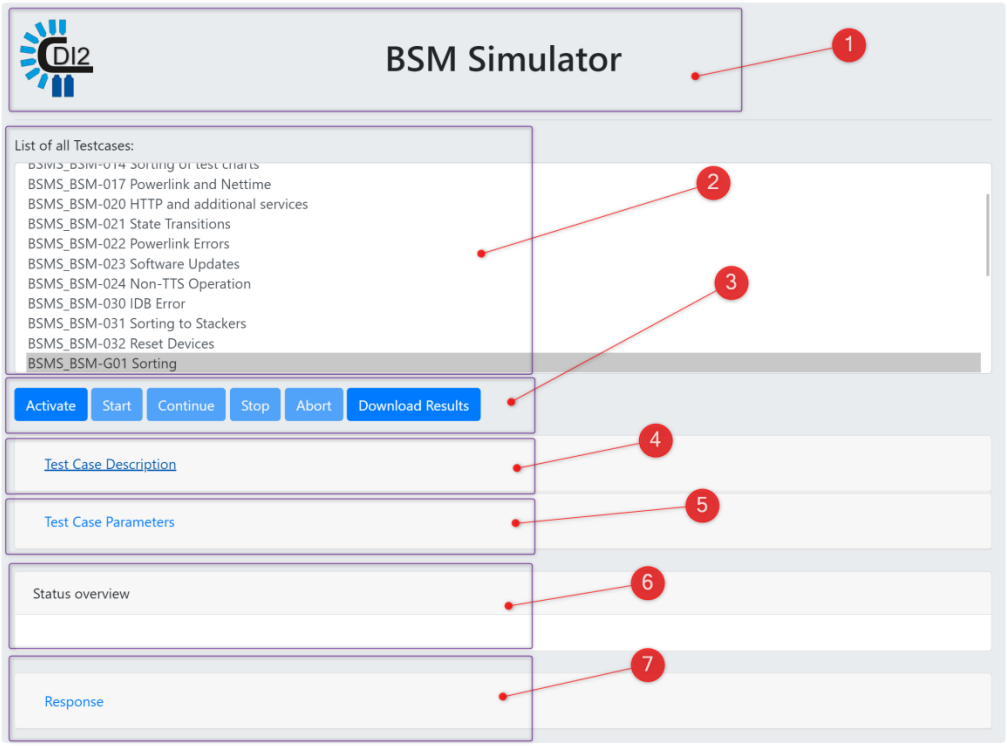


Figure 32: Panels

(1) Header
Indicates to which instance this interface belongs to (either "BSM Simulator" or "Device Simulator"). The animation of the logo indicates a successful update of the test status while a test case is running.

(2) List of Testcases
Lists all available test cases. Test cases can be selected by clicking the test case name.

(3) Buttons
[see 9.2.3]

(4) Test Case Description
The steps that are performed by the test case are described in this panel. It can be shown or hidden by clicking on "Test Case Descriptions".

(5) Test Case Parameters
The parameters offer a way to control the execution of the test case. Parameters must be entered before the test case is activated. Hovering over the name of a parameter will show a short description of the parameter. For detailed descriptions of the parameters refer to the test case description. The panel can be shown or hidden by clicking on "Test Case Parameters" [see 9.2.5]

(6) Status Overview
This panel provides an overview of all the running components in the test case [see 9.2.6].

(7) Response

This panel provides detailed responses from the simulation devices. The data is formatted as JSON and parts of it can be collapsed by clicking "-" or expanded by clicking "+". While a test case is running, the data is refreshed every second which resets the view of this data. To select and analyse this field, it is recommended to let the simulation finish or to stop it first.

The panel can be shown or hidden by clicking on "Response".

9.2.2. Test Case Description

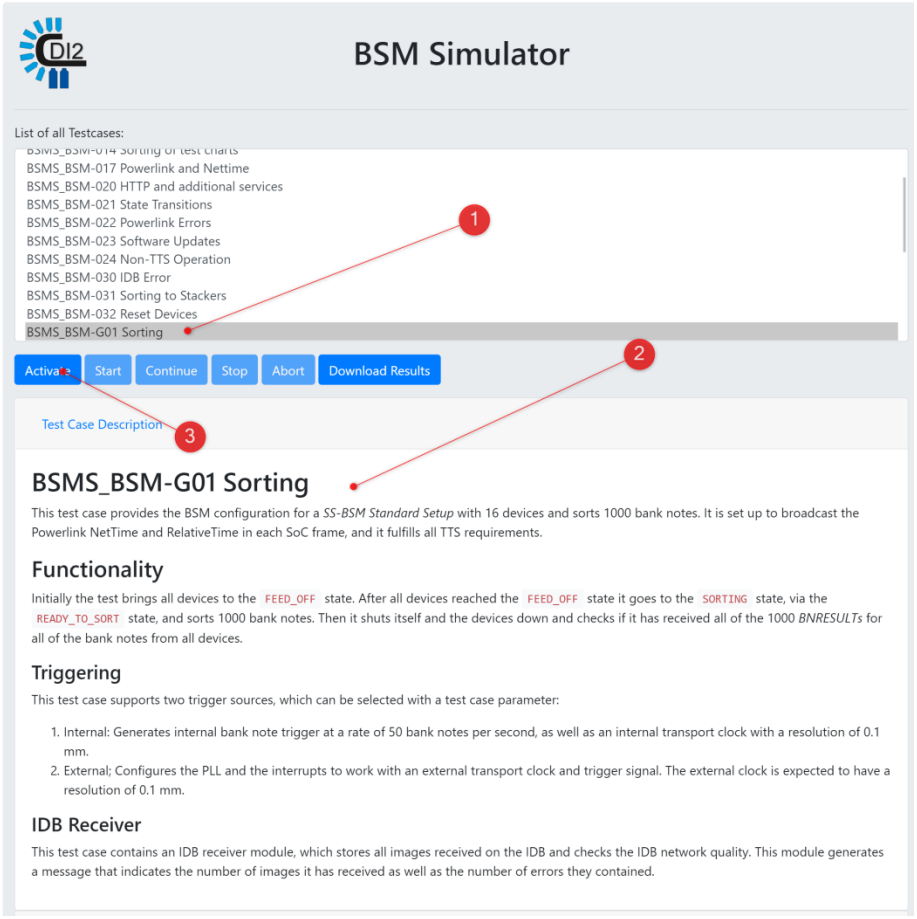


Figure 33: Test case description

When selecting a testcase (1), a description of the test set is shown (2). The "Activate" button (3) is enabled with which the test sets can be loaded to the simulator nodes on activation.

9.2.3. Buttons

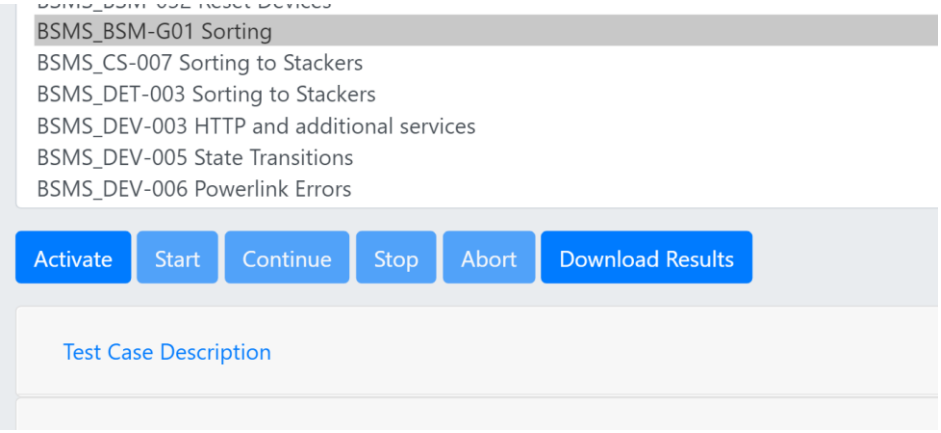


Figure 34: Button

- **Activate**
  Clicking this button transfers all necessary files, scripts and the test case parameters to the nodes being part of the test case. When finished, the Start button becomes available.
- **Start**
  Clicking this button starts the test cases on the nodes and enables the monitoring in the Status Overview and Response panels.
- **Continue**
  If one of the nodes is in the waiting state, this button becomes active. The reaction to pressing this button is dependent on the executed test case. Usually information on the needed interaction is presented in the Instructions panel.
- **Stop**
  This button causes a graceful shutdown of all the test components. Afterwards, logs and status reports will be available.
- **Abort**
  This button causes all components to immediately abort the operation. No logs or reports will be saved.
- **Download Results**
  After the test case is finished, either because the test case finished or the Stop button was pressed, clicking this button will download an archive containing all logs, status reports and images recorded by the test case. The filename of the downloaded file consists of the test case name and the current date and time. See 9.2.8 for further details.

9.2.4. Instructions

If a test case needs user interaction it will display instructions in this blue panel. The content is dependent on the test case.

9.2.5. Test Case Parameters

The parameters offer a way to control the execution of the test case. Parameters must be entered before the test case is activated. Hovering over the name of a parameter will show a short description of the parameter.

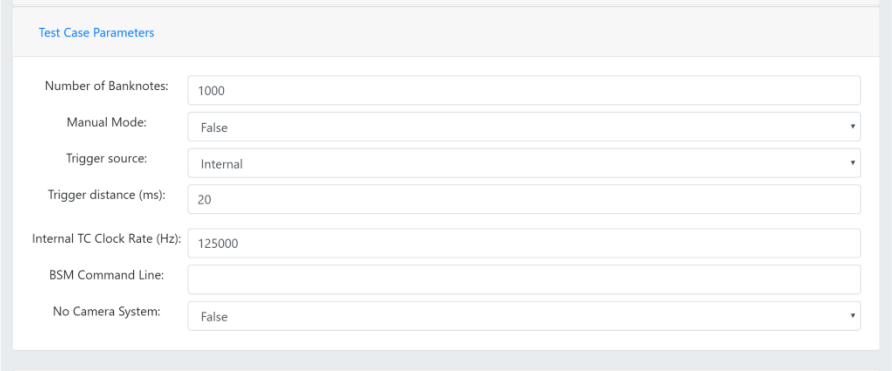


Figure 35: Test case parameters example

In the following, commonly used parameters are explained.

Common

Number of Banknotes

The number of banknotes to sort (for the BSMS), or the number of banknotes to expect (for the DS).

BSM

Manual Mode

When true the BSMS stops during start up at BS_FEED_OFF and waits for the user to press Continue.

Trigger source

Either use the TS (External) or the internal (Internal) trigger and clock generation.

Trigger distance (ms)

When Trigger source is Internal: This value specifies the time between the leading edges of successive simulated banknotes (e.g. use 20 ms to set a sorting rate of 50 banknotes per second). When the trigger source is external this parameter is ignored.

Internal TC Clock Rate (Hz)

Clock rate of the transport clock when in Internal trigger mode. The TC is counting tenths of millimetres so a value of 125000 results in a transport speed of 12.5 m/s. When the trigger source is external the value is ignored.

No Camera System

When set to true the BSMS will ignore the note recognition information sent by the camera system and instead provides simulated BNINFO packets.

BSM Command Line

Additional command line parameters are given to the BSM simulator. Relevant arguments are:

--device-selection

A comma separated list of node IDs which the BSMS should use.

--mechanical-slots

A comma separated list of distances from the photo barrier for each mechanical slot in mm.

--mech-slot-assign
: A comma separated list of slot assignments per device. E. g. "--mech-slot-assign 1,2,0" means device 1 and 2 are placed in mechanical slots 1 and 2 respectively and device 3 is an IEU without a mechanical slot.

--ignore-bn-recognition
: If given the BSMS will ignore the BN recognition of the CS and will instead send its own recognition data.

-v, --verbosity
: Sets the level for messages in the log files, where -v-2 leads to the minimal number of messages, -v0 includes only errors, warnings and info messages and -v9 leads to the highest number of messages. For long time tests a value of -v5 is recommended.
For detailed tests -v9 can be used.

The usage of parameters is described in the acceptance protocol if needed for a test case.

Device

Run Forever

When set true the device does not abort when DS_ERROR is reached. Instead it accepts a new startup sequence of the BSM and continues normal operation. This option is useful when the BSM has to be restarted repeatedly. *Run Forever* is most useful with a general testcase like *DS_BSM-G01*.

Fitness Results

A comma separated list of values (FIT, UNFIT, REJECT, SPECIAL1, SPECIAL2, SPECIAL3, SPECIAL4 or SPECIAL5) that is returned as result code for each banknote. When the end of the list is reached the next result will start again with the first entry.

Device Command Line

Additional command line parameters given to the device simulator. Relevant arguments are:

--number-of-segments
: The number of segments this device will send in its results.

--bp-offset
: The BP offset the device will need in mm.

--tc-resolution
: The TTS TC resolution of the device in mm/clock.

--tts-support
: If 0 the device will not use TTS, if 1 the device will use TTS.

-v, --verbosity
: Sets the level for messages in the log files, where -v-2 leads to the minimal number of messages, -v0 includes only errors, warnings and info messages and -v9 leads to the highest number of messages. For long time tests a value of -v5 is recommended.
For detailed tests -v9 can be used.

The usage of parameters is described in the acceptance protocol if needed for a test case.

9.2.6. Status Overview

This panel provides an overview of all the running components in the test case. It lists the Powerlink node id, the role of the device (e.g. BSM, Camera System or IEU), the current CDI2 state of the simulated device and a small textual description of the test state. The test state includes one of five icons to indicate the general state:

- ▶ the test is running
- ▮▮ the test is waiting for the user to interact with the simulation or to press Continue
- ✓ the test is finished and passed
- ✗ the test is finished and failed

9.2.7. Response

This panel provides detailed responses from the simulation devices. The data is formatted as JSON and parts of it can be collapsed by clicking "-" or expanded by clicking "+". While a test case is running, the data is refreshed every second which resets the view of this data. To select and analyse this field it is recommended to let the simulation finish or to stop it first.

The panel can be shown or hidden by clicking on "Response".

Common fields in the Response are:

- bsm_cdi2_states: A list of all states the BSM has been in during the test case run.
- cdi2_states: A list of all states this simulated device has been in during the test case.
- errors: A list of errors recorded by the device.
- history_errors: A list of Powerlink errors recorded by the device.
- node_id: The Powerlink node id of the device.
- powerlink_states: A list of all Powerlink states the device has been in.
- role: The role of the device in the test case (e.g. BSM, Camera System or IEU).
- status: One of RUNNING, WAITING, PASSED and ERROR indicating the test case status.

The BSMS provides these additional fields:

- nodes: A list with all Powerlink nodes that were detected by the BSM, with all recorded state transitions and errors recorded for each of them.
- system_states: A list of all combined states the CDI2 system has been in.

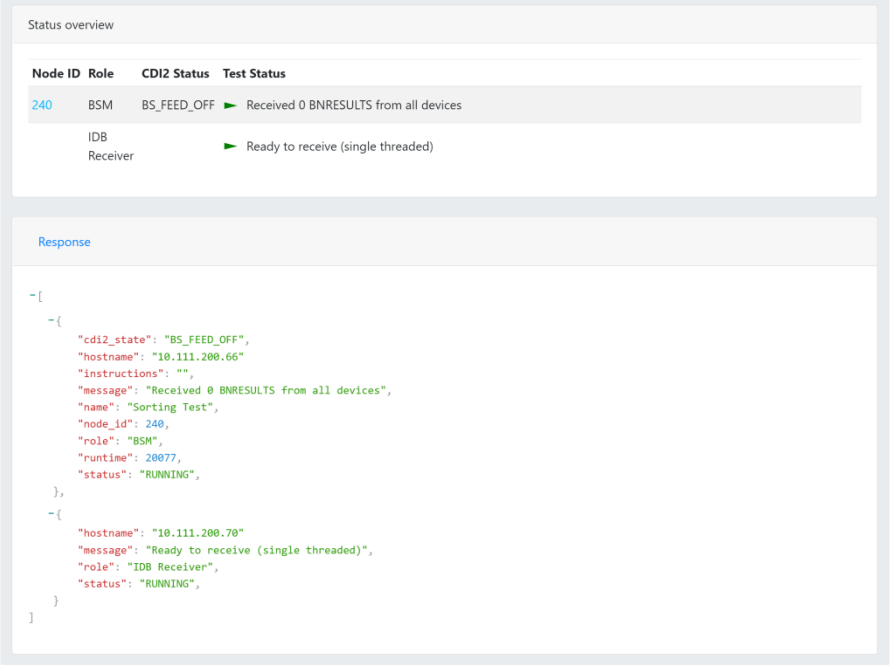


Figure 36: Response example - while test is running

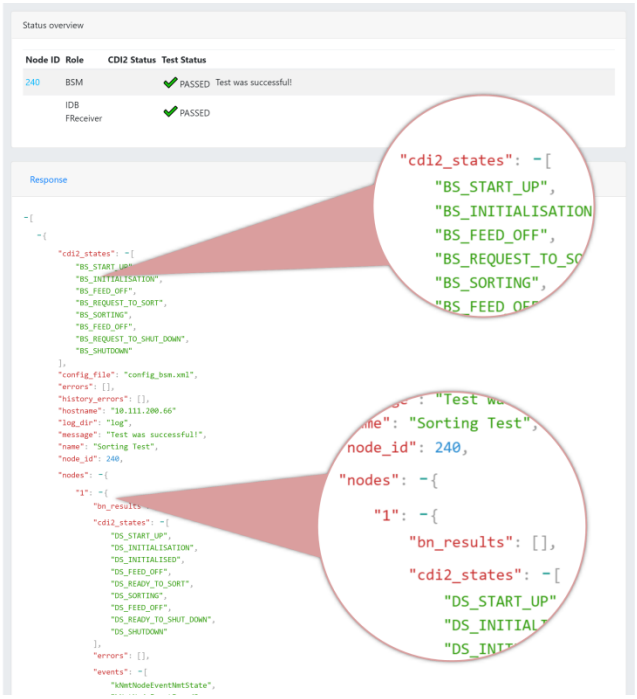


Figure 37: Response example - when test has finished

9.2.8. Download Results (download results archive)

After the test case is finished, either because the test case has finished or the Stop button was pressed, clicking the Download button will download an archive containing all logs, status reports and images recorded by the test case. The filename of the downloaded file consists of the test case name and the current date and time.

Example of a result archive file:

BSMS_BSM-G01_2018-08-01_06-55-52.tar.xz

To open the result archive, the 7-Zip program is recommended.

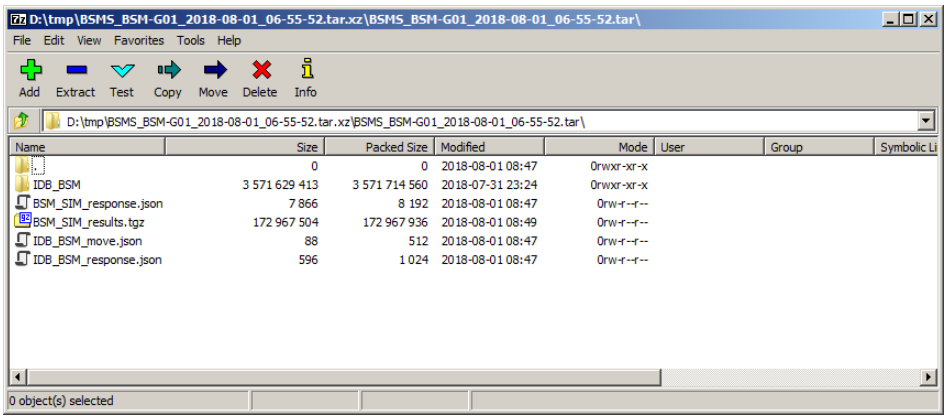


Figure 38: 7-Zip example

In the following the content of the result archive is described.

For every used component the result archive contains one file and one archive *(<node_name>_response.json and <node_name>_result.tgz*). The JSON file contains a quick overview of the test result while the tgz-archive contains all logs and reports that were created during the test run.

For the BSM the tgz-archive contains the CDI2 data store as a compressed XML file, a compressed log file with detailed information on what happened during the test run and a folder for each device containing the received *device_info.xml* and the written *device_config.xml*.

For Devices the tgz-archive contains the *device_config.xml* written by the BSM, the *machine_info.xml* written by the BSM and a compressed log file containing detailed information about the test run.

For the IDB receiver a folder exists containing the images received correctly (*pictures*) and images received containing error (*pictures_ERR*).

For the IDB sender the tgz-archive contains the images sent during the test.

When the version information test case is run the archive contains a file called *versions.log* which contains information about the installed software versions.

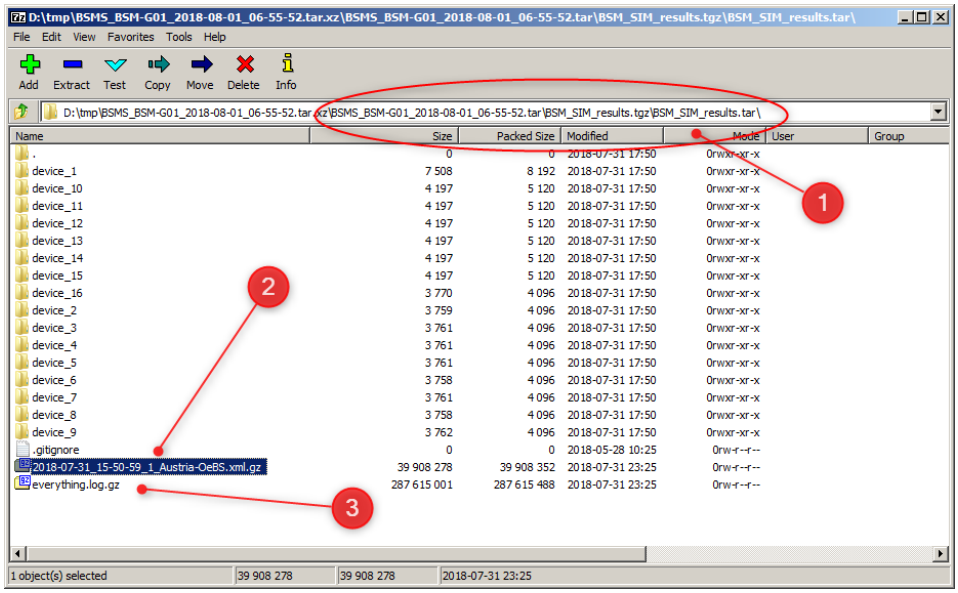


Figure 39: Result archive

(1) path to *BSM_SIM_results*

(2) **Data-Storage file**

The BSMs stores the received banknote results in this file. The file format complies with the XML storage file format as specified in [Ref 1.] 9.1.1 Aggregated Data.

(3) **everything.log.gz**

A gzip compressed logfile generated by the simulator with all events during a simulation run. Mainly used for debugging when the simulation returned an unexpected result (see examples in Figure 40, Figure 41 and Figure 42).

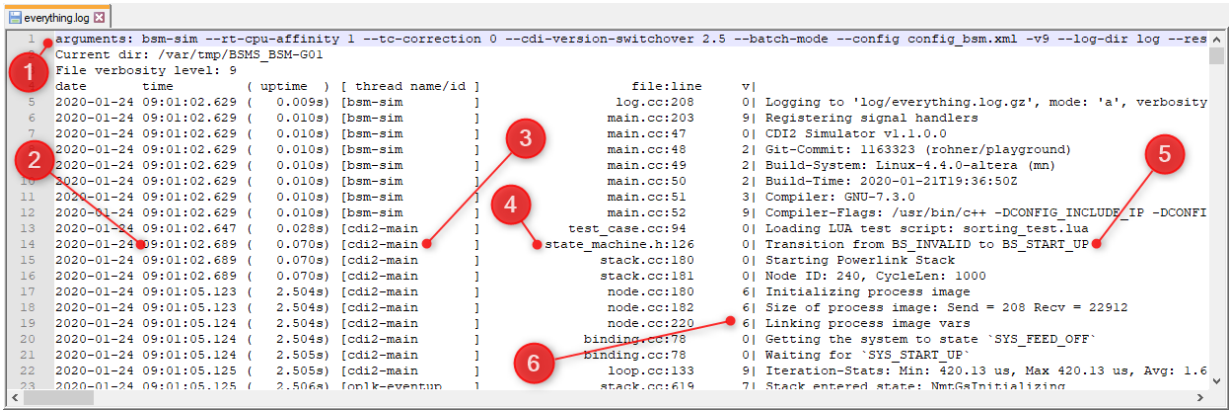


Figure 40: BSMS everything.log example 1 (start)

The example in Figure 40 shows initial messages at startup of the simulator.

(1) Simulator calling string with actual parameters
(2) Date/time
(3) Thread name/id
(4) File:line
(5) Log message
(6) Log level of messages

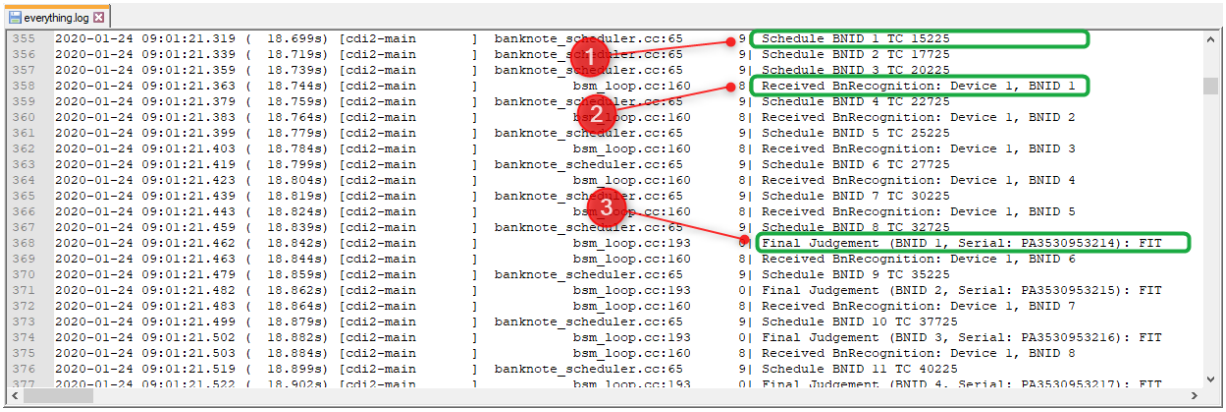


Figure 41: BSM everything.log example 2 (sorting)

The example in Figure 41 shows log messages during normal sorting.

(1) Banknote with BNID=1 is scheduled.
(2) BnRecognition for BNID=1 received from Device 1 (which is the CS).
(3) Final Judgment for BNID=1. Message is issued when BnResults for BNID=1 has been received from all devices.

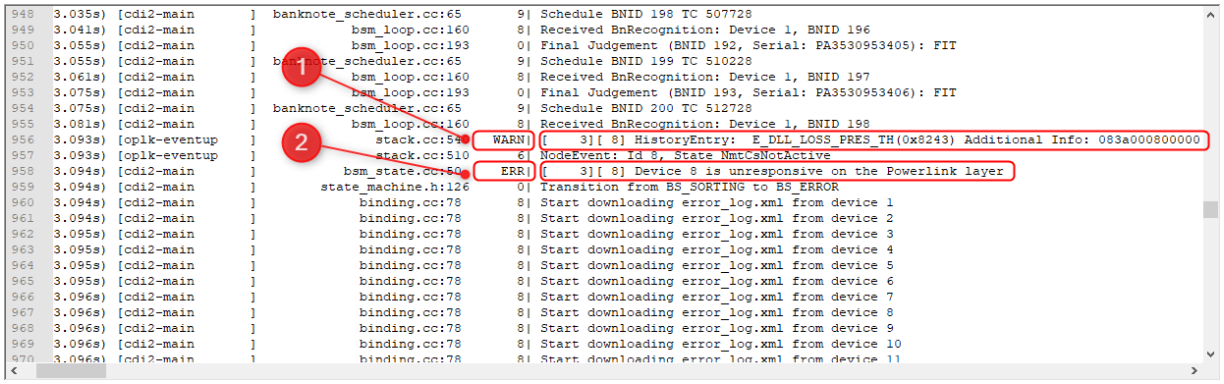


Figure 42: BSM everything.log example 3 (error)

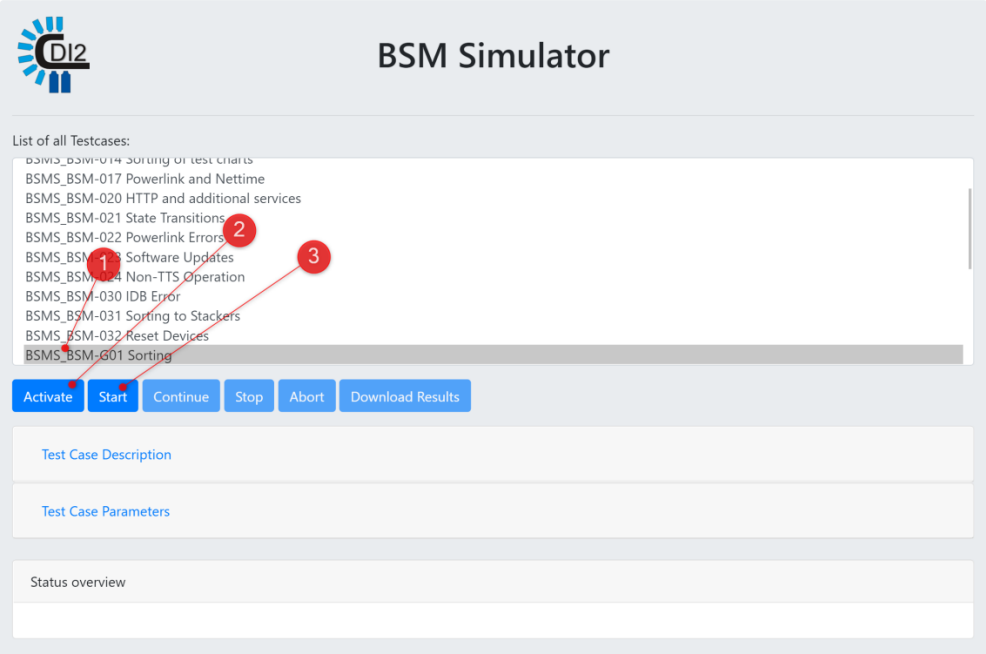
The example in Figure 42 shows a typical warning and error when a Device stops operation during sorting.

(1) Warning, E_DLL_LOSS_PRES_TH, Node 8 did not respond with PRes on Powerlink layer
(2) Error, Device 8 does not respond on the Powerlink layer

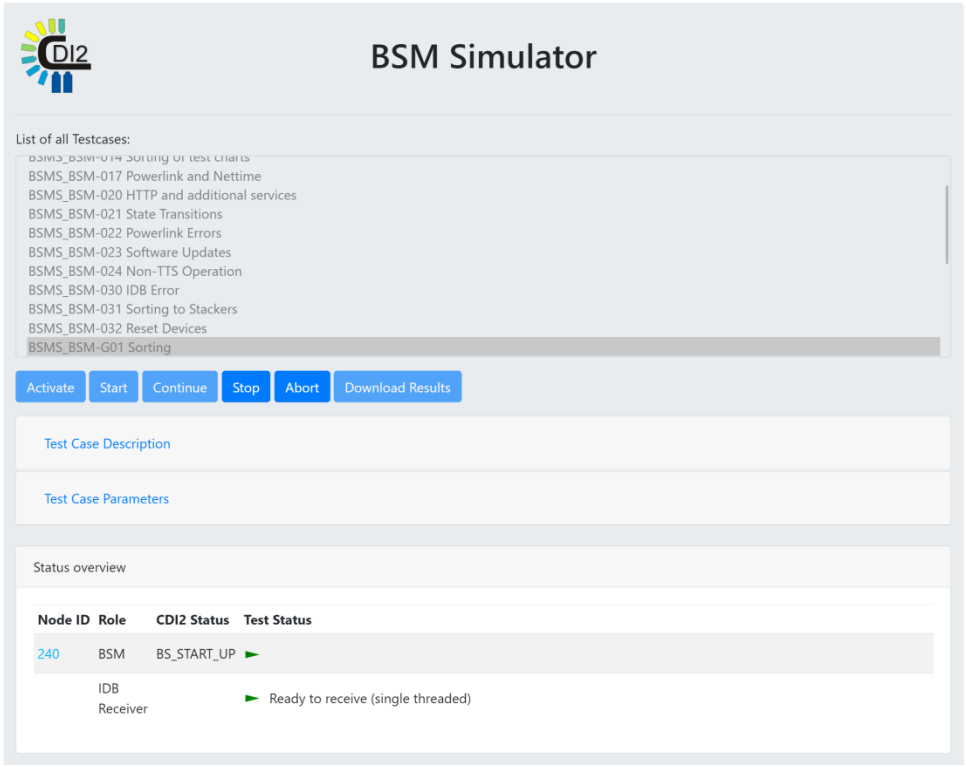
9.2.9. Example Procedure

To illustrate the execution of a test case, this section shows a step by step example of the process for the general sorting test (BSM-G01).

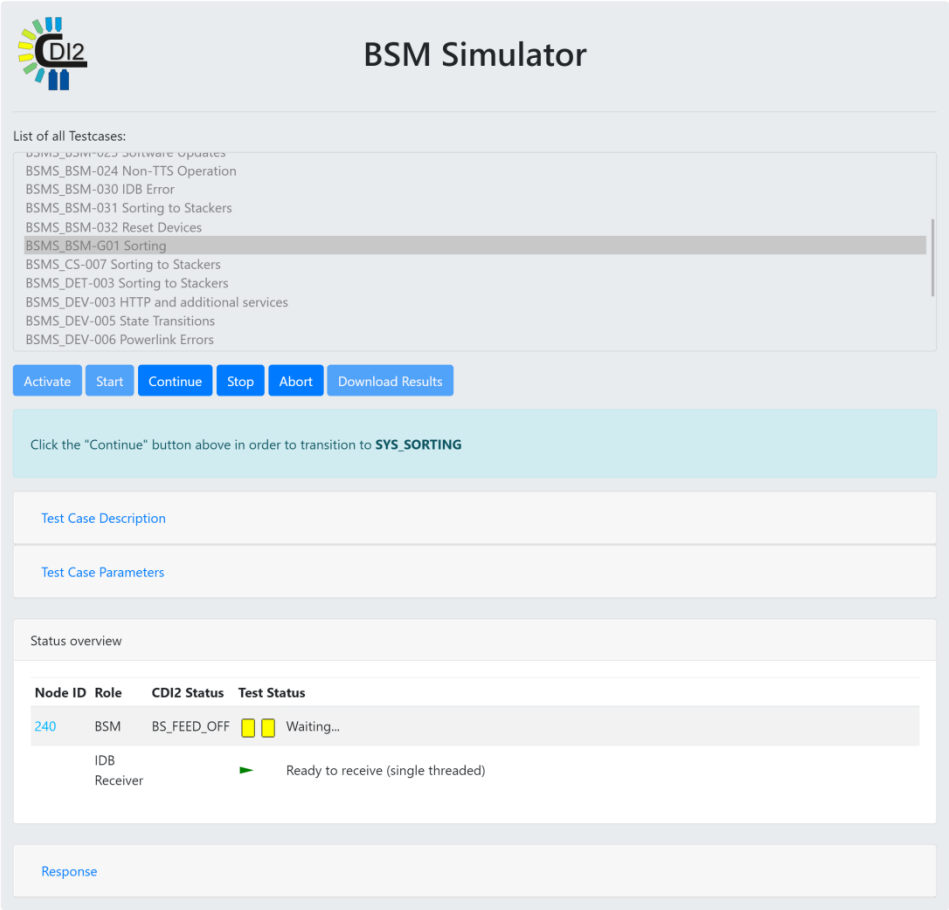
(1) Select a test set
(2) Click activate, and wait until Start becomes available
(3) Click start



The test set will now have been copied to all used components and have been started.

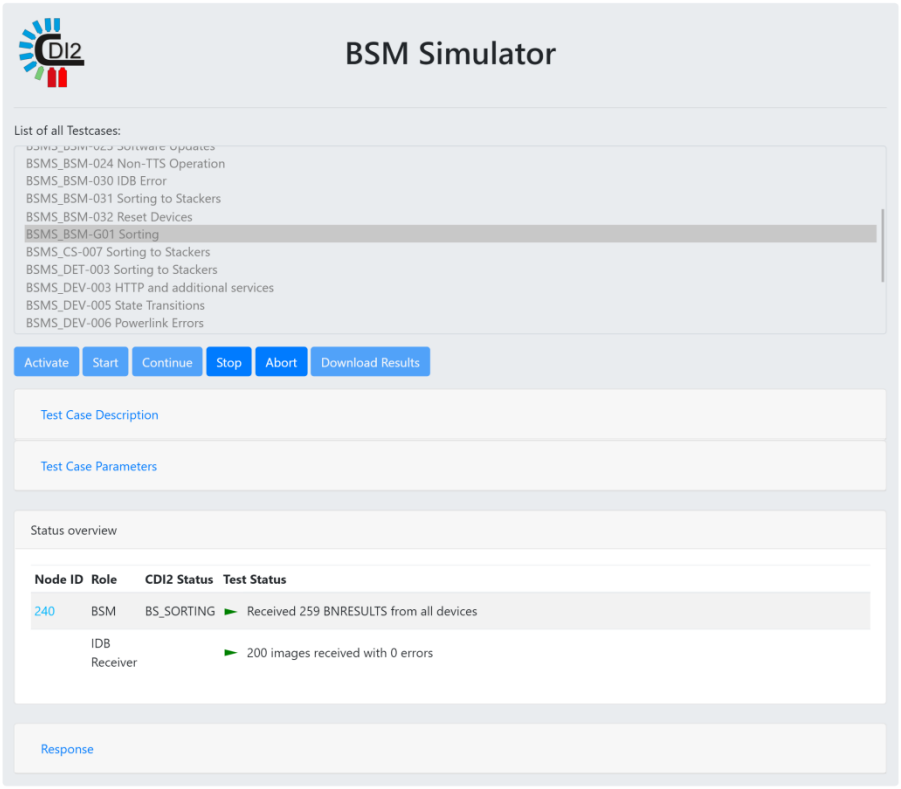


The started components now report their status in the status overview. The logo is animated to indicate that the information is updated. The stop and abort buttons are active to indicate that the simulation can be stopped.

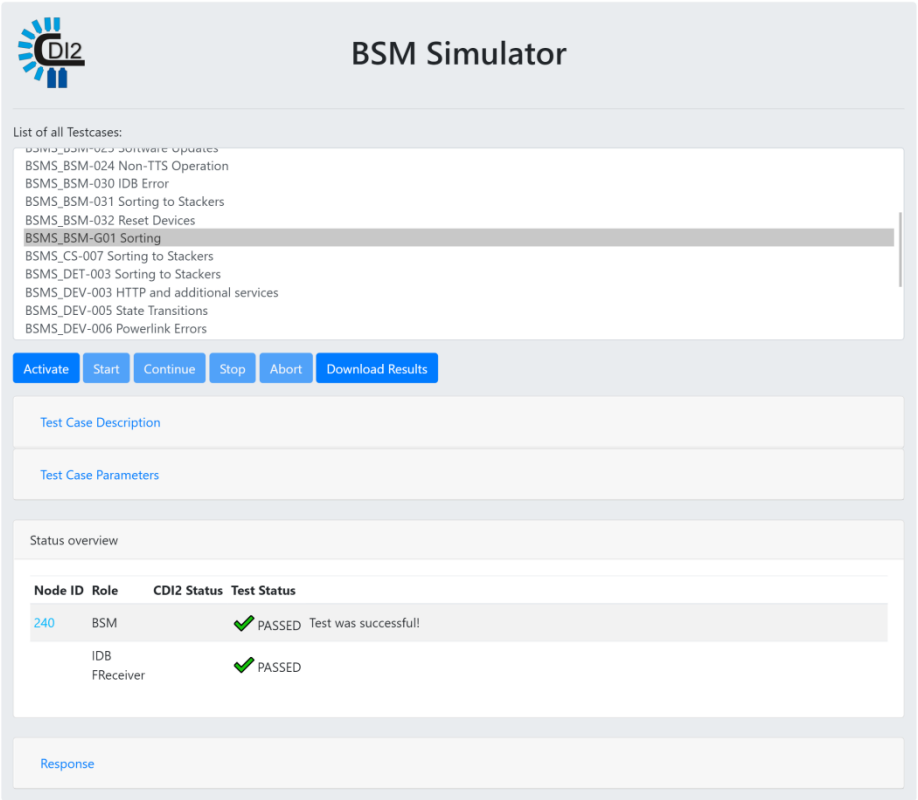


By default, if the sorting test reaches BS_FEED_OFF it waits for the user to click Continue before it transitions to the sorting state. The instructions are now shown in the blue box. At the same time the status overview of the BSMS reports that it is waiting for user interaction. Clicking continue, which became available, will cause the simulation to proceed.

Usually waiting is needed to allow the user to prepare parts of the simulation e.g. start the TS.



The simulator now starts to sort 1,000 banknotes and the IDB receiver component receives images sent on the IDB.



After 1,000 banknotes were sorted and the simulator did not detect any errors the test set is marked as PASSED.

Now the Download Results button becomes available and allows to download all log and result files.

9.2.10. Extra BSMS Parameters

The following BSMS parameters and features are available since version SW_CDI2SIM_V27C_20220315.

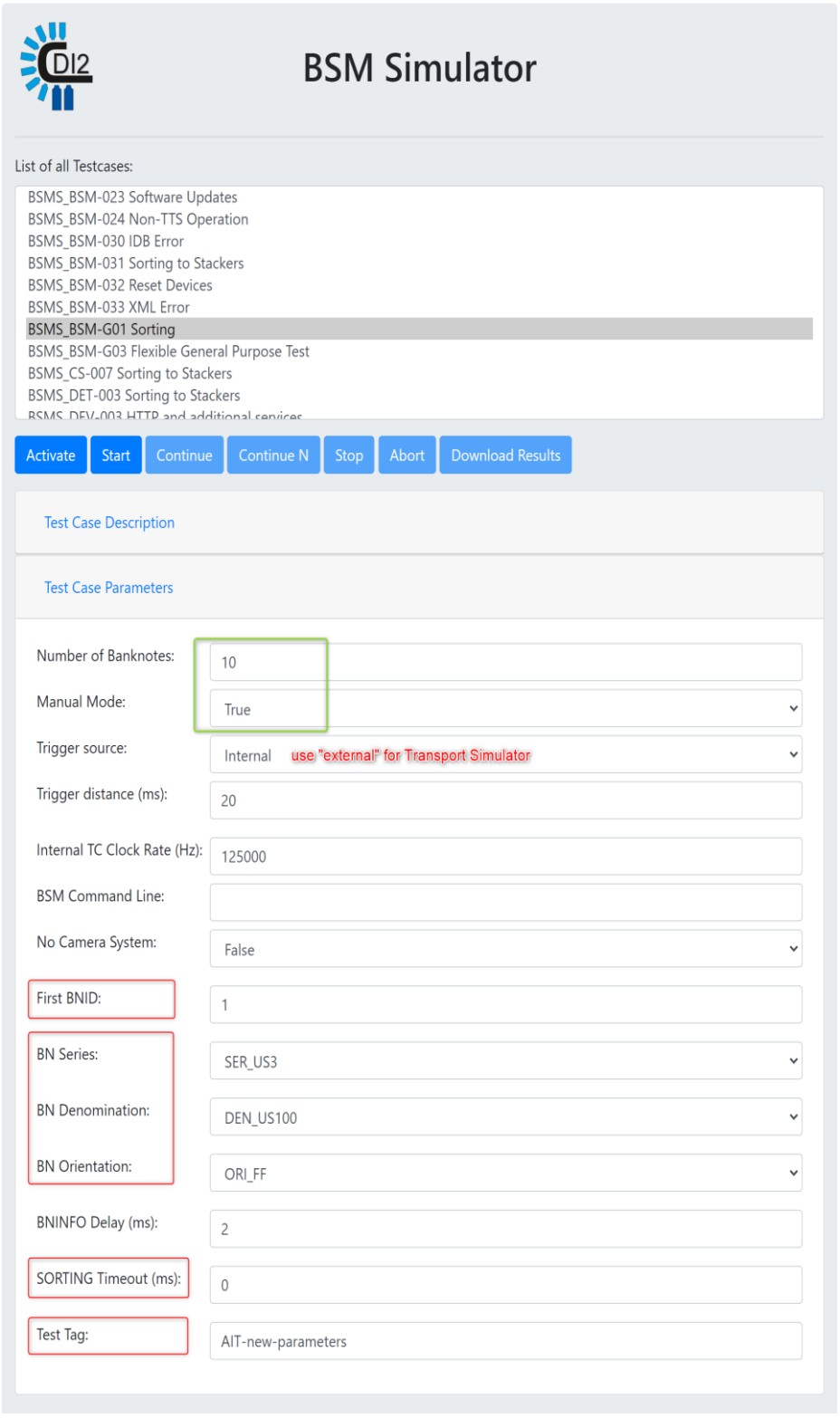
Parameter	**Description**
Sorting Timeout	GUI parameter "*SORTING Timeout (ms)"* Wait this many milliseconds for expected BNRESULT messages before aborting the running test. Set to 0 to wait forever Allow specifying the timeout (in ms) for when the machine would error out after it enters a sorting state and stops receiving results during sorting. Set to 0 to disable the timeout (wait forever). Of course, the testcase can be stopped or aborted still.
BN ID Start	GUI parameter *"First BNID"* Starting point of banknote identifications allocated during this test run
Repeat Note/Trigger Counter	A GUI button "Continue N" (additionally to "Continue") provides a means to repeat a sorting sequence without the need to reload the testcase. This is useful in manual mode, e.g. when tests with the transport simulator are done. The test case starts and waits for operator input before doing the sort run. The operator can prepare the transport simulator and presses either "Continue" or "Continue N" when the transport simulator has spun up. "Continue" works unchanged and will do a single sorting run and terminates. Whereas the new "Continue N" sorts the desired number of banknotes ("Number of Banknotes" parameter) and returns to the waiting state again. The operator may repeat the test run many times. For termination the "Finish" Button is pressed.
Denom/Series/Orientation	GUI parameters for denom/series/orientation *) Parameters are taken for BNINFO when "No Camera System" is set true. *"BN Series"* BN series sent out in BNINFO *"BN Denomination"* BN denomination sent out in BNINFO *"BN Orientation"* BN orientation sent out in BNINFO
Short Self-Test Cycle (BN)	Sort this many BNs between short self-tests. Set to 0 for no short self-tests. The BSMS transitions from BS_SORTING to BS_REQUEST_TO_SORT

	to allow detectors to run a short self-test.
Intensive Self-Test Cycle (BN)	Sort this many BNs between intensive self-tests. Set to 0 for no intensive self-tests. The BSMS transitions from BS_SORTING to BS_FEED_OFF to allow detectors to run an intensive self-test. The Intensive Self-Test Cycle interval shall be longer than the Short Self-Test Cycle interval.
Test Tag	GUI parameter *"Test Tag"* Operator-specified textual metadata that is stored along with the BSM results package. The tag is added to the filename of the download package, to the filename of the BSM_SIM_results.tgz and is added to META.json. e.g. tag = "AIT-new-parameters" BSMS_BSM-G01_**AIT-new-parameters**_2020-12-17_09-30-30.tar.xz BSM_SIM_results_**AIT-new-parameters**.tgz META.json: `"test_tag": "AIT-new-parameters"`

Table 5: Extra BSMS parameters

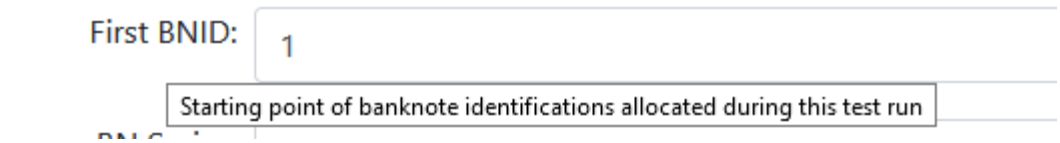
See 17.16.2 for a complete list of parameters or see description of BSMS_BSM-G01 Sorting in the GUI.

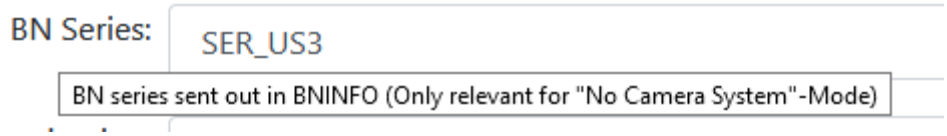
The following screenshots show the GUI with new parameters and an example how a test can be repeated with "Continue N". For below example DS_BSM-G01 Sorting was used at the Device Simulator side.

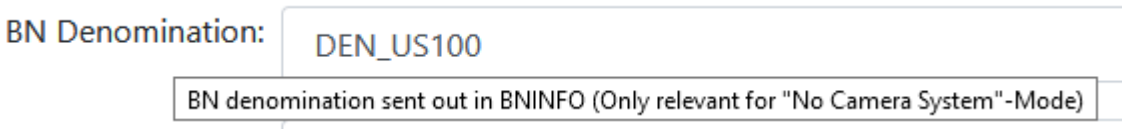


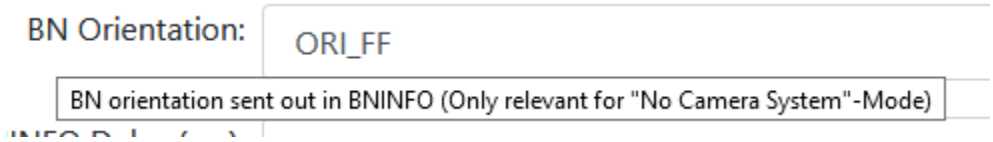
Note:

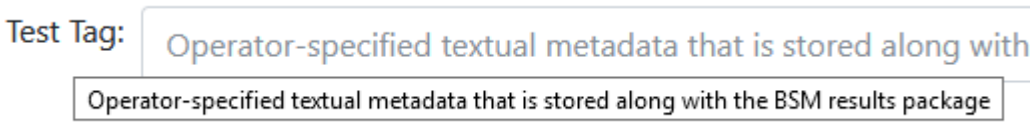
Use "Mouse-Over" to get additional parameter info.

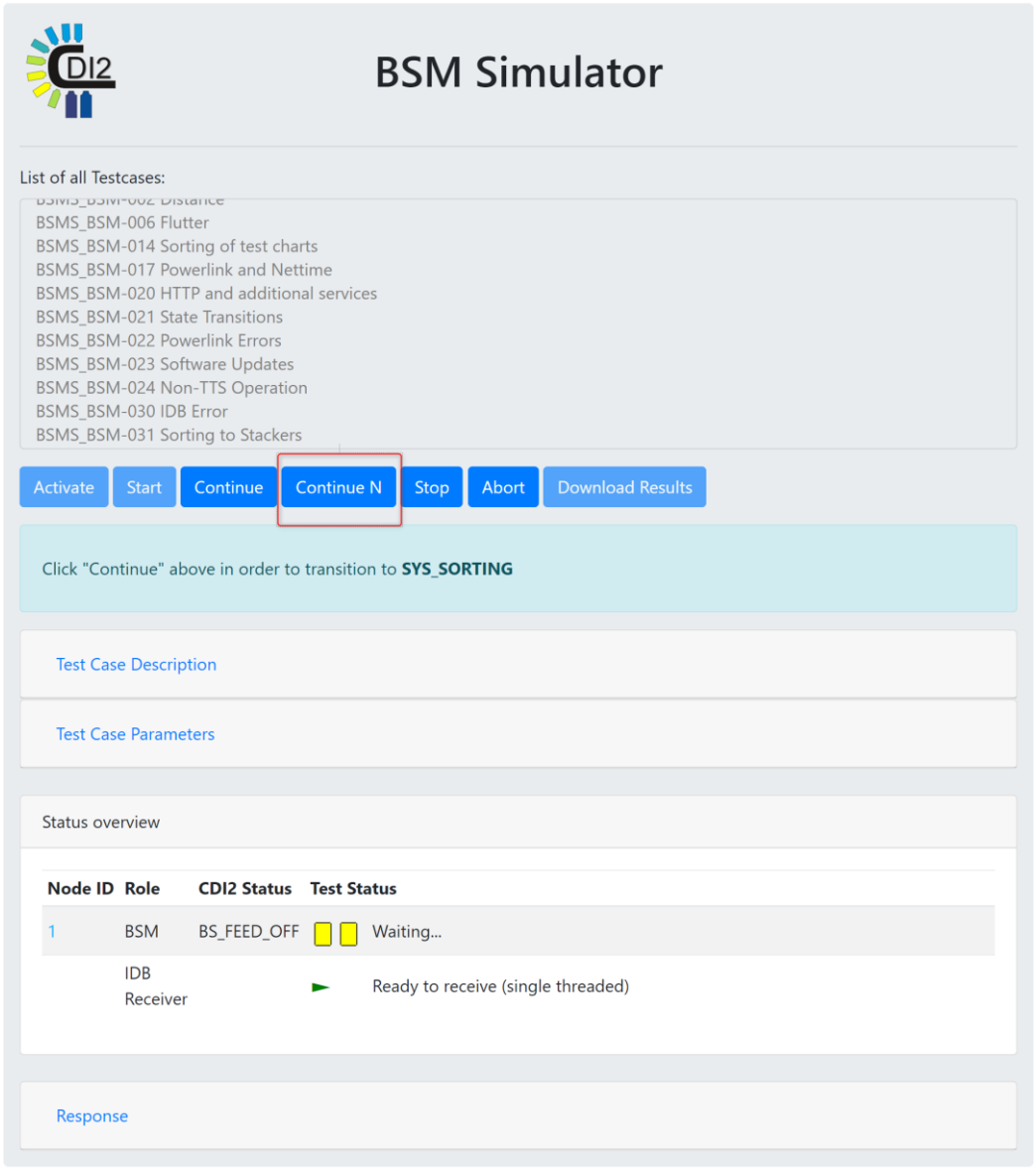


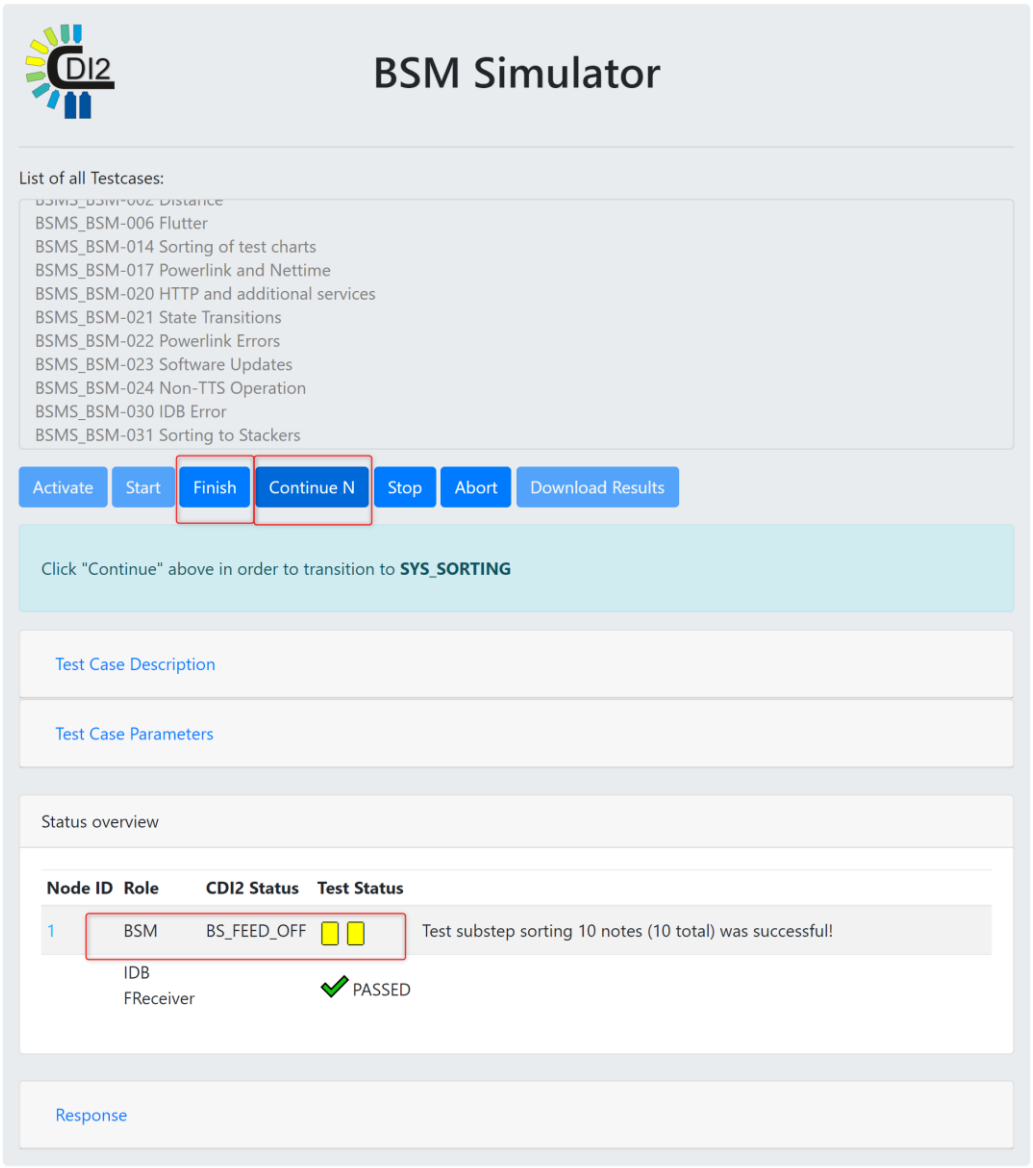


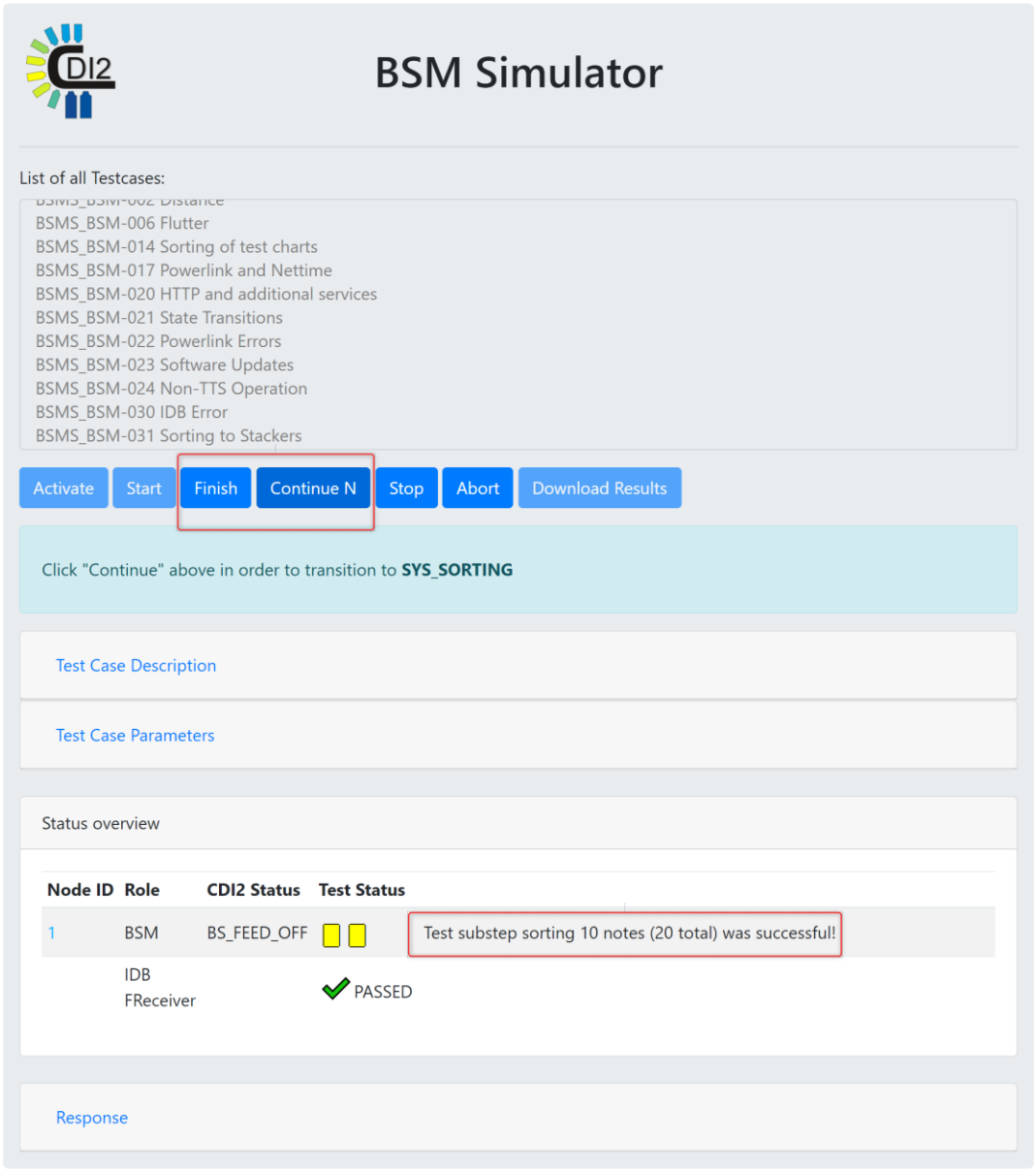


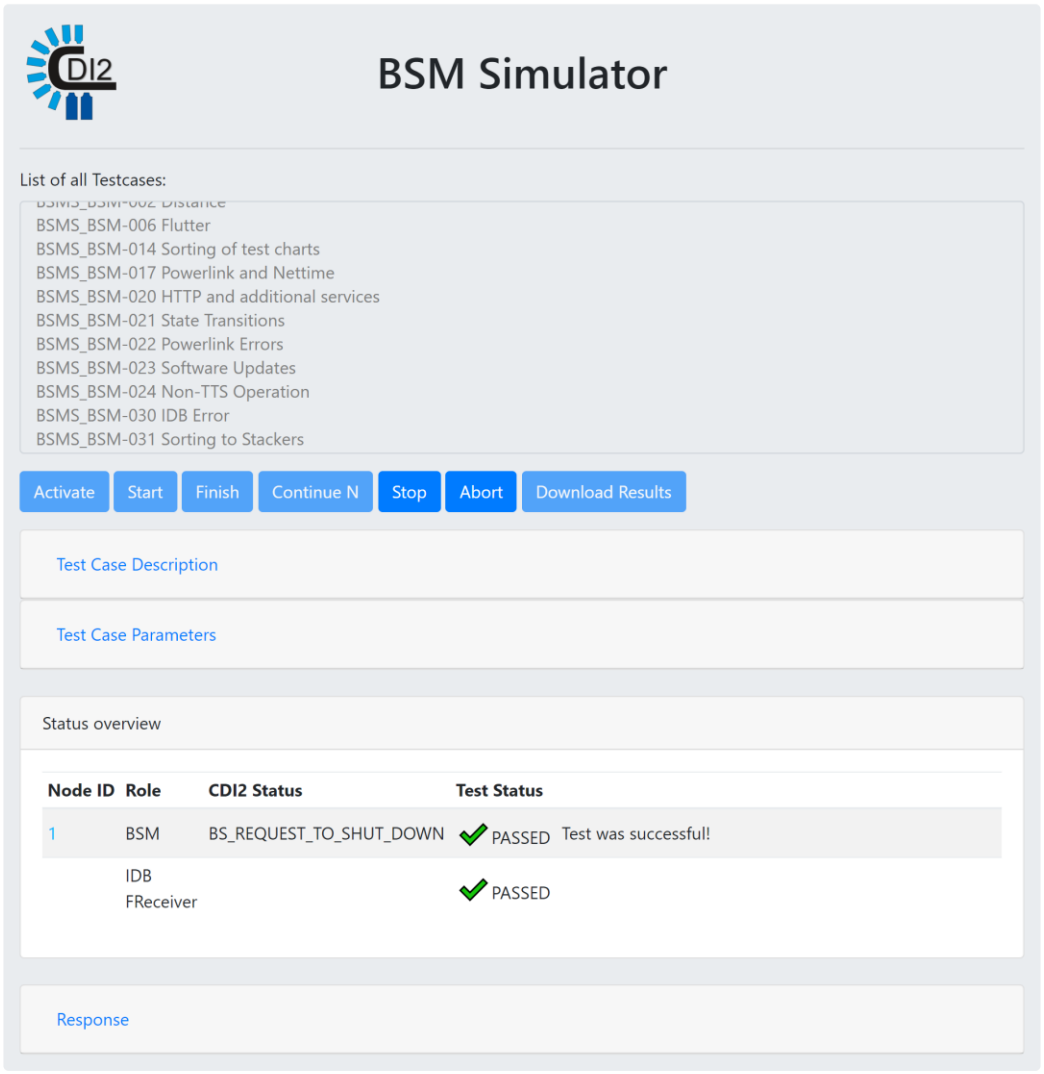





BSM Simulator
List of all Testcases:
BSMS_BSM-006 Flutter
BSMS_BSM-014 Sorting of test charts
BSMS_BSM-017 Powerlink and Nettime
BSMS_BSM-020 HTTP and additional services
BSMS_BSM-021 State Transitions
BSMS_BSM-022 Powerlink Errors
BSMS_BSM-023 Software Updates
BSMS_BSM-024 Non-TTS Operation
BSMS_BSM-030 IDB Error
BSMS_BSM-031 Sorting to Stackers
Activate
Start
Continue
Continue N
Stop
Abort
Download Results
Click "Continue" above in order to transition to SYS_SORTING
Test Case Description
Test Case Parameters
Status overview
Node ID
Role
CDI2 Status
Test Status
1
BSM
BS_FEED_OFF
Waiting...
IDB Receiver
Ready to receive (single threaded)
Response


BSM Simulator
List of all Testcases:
BSMS_BSM-006 Flutter
BSMS_BSM-014 Sorting of test charts
BSMS_BSM-017 Powerlink and Nettime
BSMS_BSM-020 HTTP and additional services
BSMS_BSM-021 State Transitions
BSMS_BSM-022 Powerlink Errors
BSMS_BSM-023 Software Updates
BSMS_BSM-024 Non-TTS Operation
BSMS_BSM-030 IDB Error
BSMS_BSM-031 Sorting to Stackers
Activate
Start
Finish
Continue N
Stop
Abort
Download Results
Click "Continue" above in order to transition to SYS_SORTING
Test Case Description
Test Case Parameters
Status overview
Node ID
Role
CDI2 Status
Test Status
1
BSM
BS_FEED_OFF
Test substep sorting 10 notes (10 total) was successful!
IDB
FReceiver
PASSED
Response


BSM Simulator
List of all Testcases:
BSMS_BSM-006 Flutter
BSMS_BSM-014 Sorting of test charts
BSMS_BSM-017 Powerlink and Nettime
BSMS_BSM-020 HTTP and additional services
BSMS_BSM-021 State Transitions
BSMS_BSM-022 Powerlink Errors
BSMS_BSM-023 Software Updates
BSMS_BSM-024 Non-TTS Operation
BSMS_BSM-030 IDB Error
BSMS_BSM-031 Sorting to Stackers
Activate
Start
Finish
Continue N
Stop
Abort
Download Results
Click "Continue" above in order to transition to SYS_SORTING
Test Case Description
Test Case Parameters
Status overview
Node ID
Role
CDI2 Status
Test Status
1
BSM
BS_FEED_OFF
Test substep sorting 10 notes (20 total) was successful!
IDB FReceiver
PASSED
Response



9.2.11. Extra DS Parameters

The following DS parameters and features are available since version SW_CDI2SIM_V27C_20220315.

Parameter	Description
Short Self-Test Duration (ms)	Time in milliseconds the devices will need to perform a short self-test, including system state transition times (10ms). If specified as min:max, a random value between min and max is chosen.
Intensive Self-Test Duration (ms)	Time in milliseconds the devices will need to perform an intensive self-test, including system state transition times (10ms). If specified as min:max, a random value between min and max is chosen.
Stall Short Self-Test (duration:cycle)	Let a Short Self-Test take a certain duration (ms) at a specific self-test cycle (cycle counting starts at 0). This is done once and overrides the Short Self-Test Duration parameter.

Stall Intensive Self-Test (duration:cycle)	Let an Intensive Self-Test take a certain duration (ms) at a specific self-test cycle (cycle counting starts at 0). This is done once and overrides the Intensive Self-Test Duration parameter.
Set Maintenance State at Short Self-Test (state@cycle)	Set maintenance status flags at a specific Short Self-Test cycle (cycle counting starts at 0). The state code may range from 0 to 15. The maintenance status flags are reset to 0 at the next, either short or intensive self-test.
Set Maintenance State at Intensive Self-Test (state@cycle)	Set maintenance status flags at a specific Intensive Self-Test cycle (cycle counting starts at 0). The state code may range from 0 to 15. The maintenance status flags are reset to 0 at the next, either short or intensive self-test.
Test Tag	GUI parameter *"Test Tag"* Operator-specified textual metadata that is stored along with the BSM results package. The tag is added to the filename of the download package, to the filename of the BSM_SIM_results.tgz and is added to META.json.

Table 6: Extra DS parameters

See 16.17.3 for a complete list of parameters or see description of DS_BSM-G01 Sorting in the GUI.

9.2.12. Common Interface Protocol

The device simulator nodes support the common interface protocol, as it was introduced with the "CDI2 Device - PC Implementation" [Ref 6.]. It allows an external computer to inject banknote results over the TCP/IP network connection and override the internal BnResults generated by the testcase.

The network interface of each device node provides a TCP port at 4000. At this port following commands and messages are available:

- messages originated by the CDI2 BSM
  (banknote announcement, banknote info and banknote trigger)
- messages to be transferred to the CDI2 BSM
  (banknote result)
- configuration commands for CDI2 Device
- error messages of the CDI2 Device

TCP/IP is used as the communication protocol. Basically, the communication uses a simple text-oriented bidirectional Lua shell, which can be accessed on TCP port 4000 of the CDI2 Device Interface Box.

Refer to the appropriate chapters in [Ref 6.] for protocol details and how to use it.

10. Service and Maintenance Screen (DS)

The device simulator provides a Service and Maintenance screen for each simulated node. It provides live status information about the simulated node. The presented information is updated in a regular manner and can be used for supervision and debugging purposes.

On the device simulator this screen is always available, as long as the simulation instance is active. Actually, the same screen is provided via the Powerlink network also, which makes it accessible via the BSM laptop as well.

Refer to chapter 8.2 for the list of IP addresses. For convenience, the device simulator GUI provides direct links on the Status Overview panel for each of the simulated nodes [Figure 43]. Clicking the link opens a new browser window with the Service and Maintenance Screen of the desired node [Figure 44].

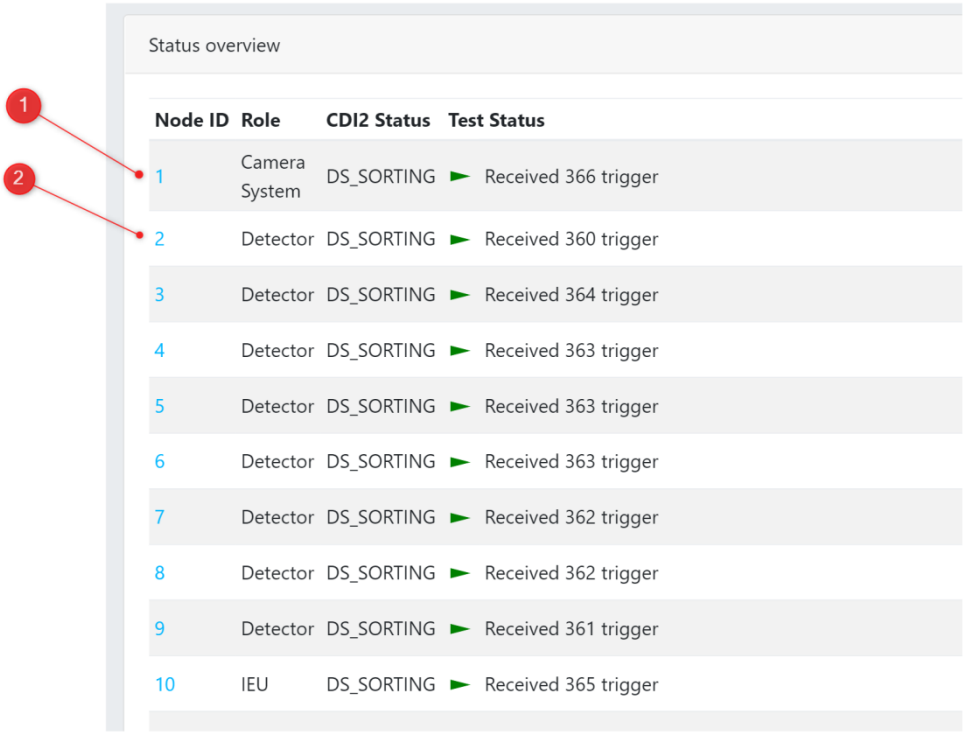


Figure 43: Status overview with links to Service and Maintenance screens

(1) Link to open a new window with Service and Maintenance screen of Node 1
(2) Link to open a new window with Service and Maintenance screen of Node 2

10.1. Main and Status Page

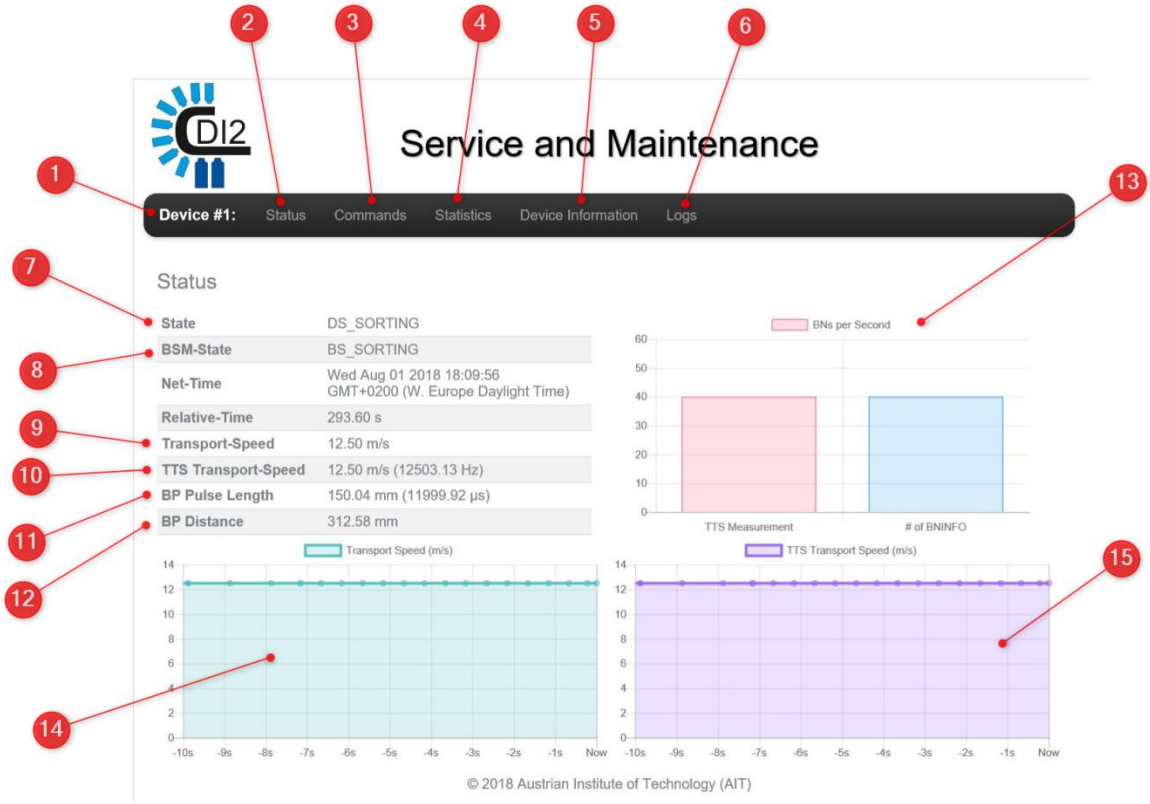


Figure 44: Service and Maintenance screen

The start page provides menu tabs for Status, Commands, Statistics, Device Information and Logs. The default screen opens the Status tab, which displays an overview about the operational status of the Device, with data about the received signals from the DMB and the TTS port. The components of the screen have the following meaning:

(1) The device Powerlink ID
(2) View Status page
(3) View Commands page
(4) View Statistics page
(5) View Device Information page
(6) View Logs page
(7) CDI2 state of the simulated device
(8) CDI2 state of the BSM
(9) Current transport speed as measured from DMB messages
(10) Current transport speed as measured from TTS
(11) Length of the TTS BP pulse
(12) Sampled distance between two banknotes
(13) Graph displaying the banknote rate on the TTS compared to the DMB
(14) Graph displaying the transport speed according to the DMB over time
(15) Graph displaying the transport speed according to TTS over time

10.2. Device Information Page

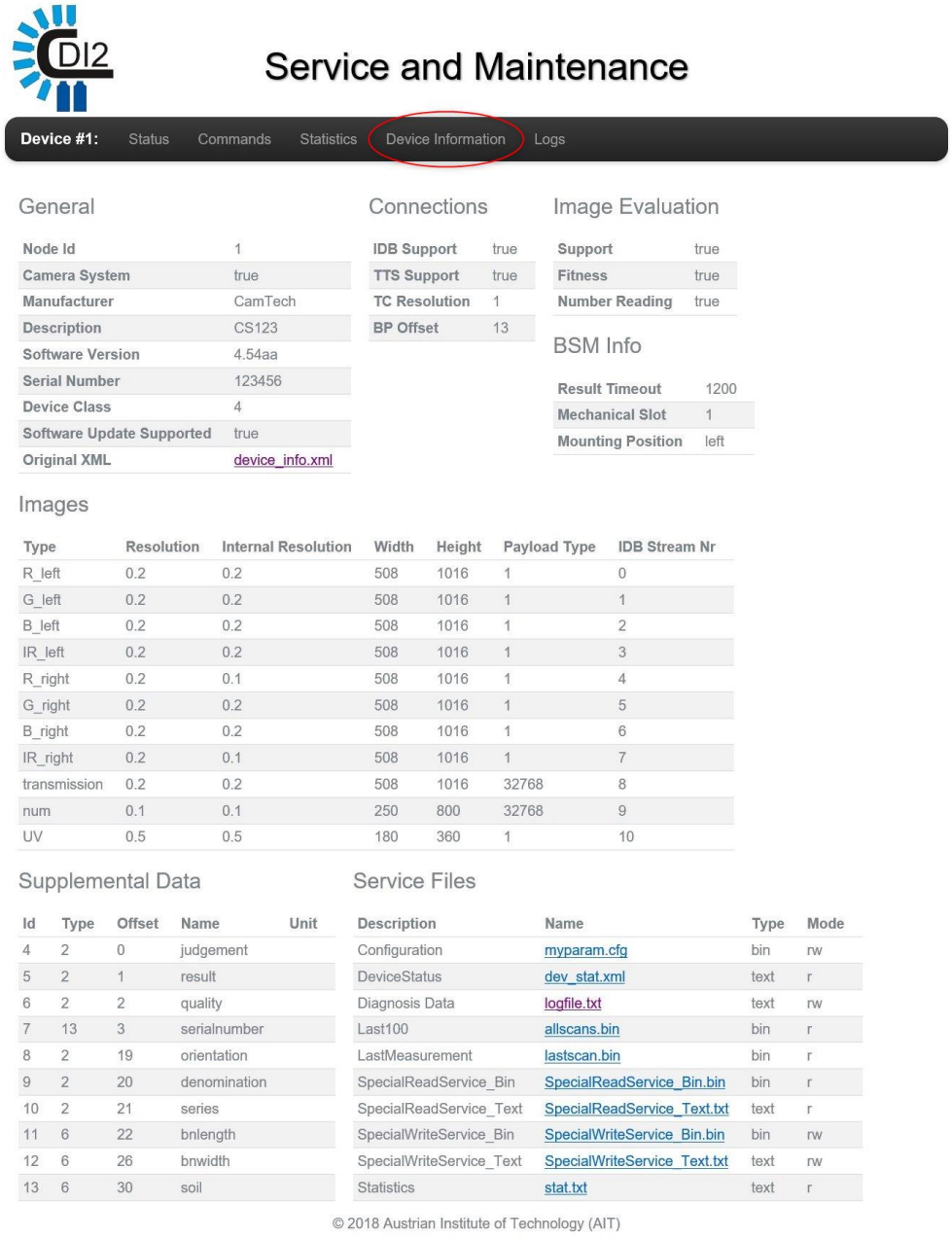


Figure 45: Device Info page

The device information screen provides an overview of the contents of the *device_info.xml* of this device. The service files displayed on the screen can be used to access the additional services, however it is recommended to use WinSCP instead, since it also allows writing of files.

10.3. Logs Page

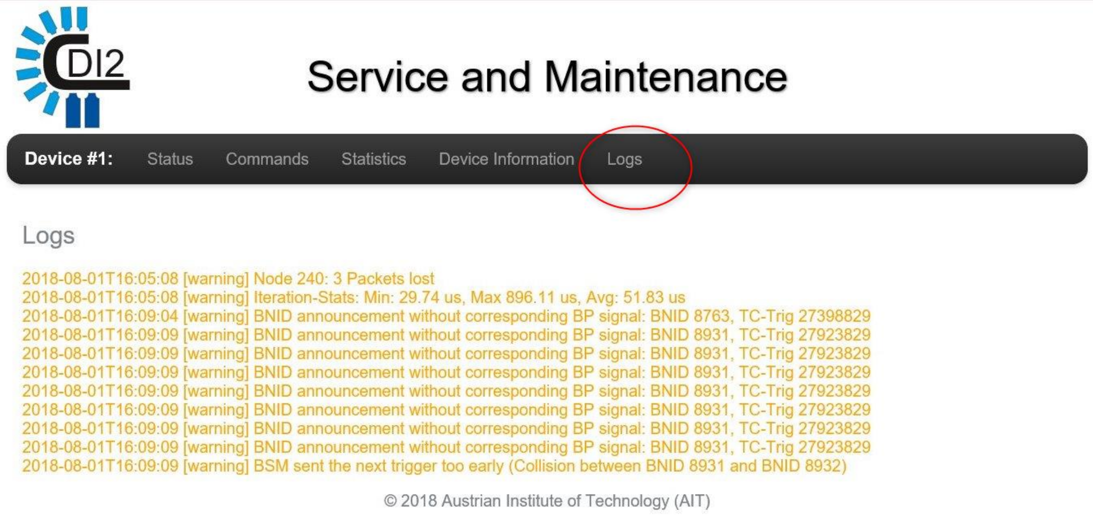


Figure 46: Logs page

The logs screen presents all error and warning messages generated by the simulator instance. Warnings are displayed in yellow and errors in red.

11. Testsets

In order to support testing of CDI2 compliance of a specific BSM or Device, a number of test cases are pre-installed on the Simulation Computer. Further, the user may modify existing test sets or create his own as well.

Alternatively, a convenient tool to generate new sorting tests for the DS is provided by means of the test case generator [11.3].

11.1. Files and Testset Layout

All testsets are stored at the BSM simulator or Device Simulator VisionBox at a pre-defined location. The GUI webserver is configured to retrieve the list of testcases from here.

For the BSM simulator

/usr/share/cdi2/cdi2-testcases/bsm-simulator

For the Device Simulator

/usr/share/cdi2/cdi2-testcases/device-simulator

11.1.1. Files

description.md

Contains a description of the test case. It describes the function, parameters and actions of the testcase. The text is written using the Markdown formatting syntax.

testcaseinfo.json

Describes the test case components to the Test Controller. It contains the list and parameters for all components and nodes taking part in the test case, the description of the parameters that can be used with the test case and default values for these parameters. A testcase can inherit information from another test case using the field *InheritFrom* which points to one or more folders of other testcases.

Meta files

Each component is supplied with a MetaFile which is named in the *testcaseinfo.json*. It contains information on which program to run, which parameters to run the program with and paths to specify where logs and results are to be stored.

Lua Script

Usually DMB nodes are supplied with a Lua script which describes how the simulated node will behave.

pictures/ *folder and* ***idbImgDirList.txt***

When an IDB sender is part of the setup, one or more folders with images are supplied along the testcase. The folder is usually called *pictures/*, but the list of folders is specified in the file *idbImgDirList.txt*. The images are stored in pgm format with one channel being stored in one file.

11.1.2. Activation and Start

When Activate is pressed, the following steps are performed:

- The parameters are written to a file called *ui_parameters.lua* which is needed later.
- For every component:
  - All files listed in *testcaseinfo.json* for this node are added to a .tar.xz archive.
  - The previously generated *ui_parameters.lua* is added to the archive.
  - The MetaFile listed in *testcaseinfo.json* is added as *META.json* to the archive.
  - The archive is uploaded to the component and stored on there in the */var/tmp/<TestCaseName>* folder.
  - The Node Controller running on the component unpacks and checks the archive contents.

When Start is pressed, the Node Controller on the component starts the program listed in META.json. The working directory of the program is within the */var/tmp/<TestCaseName>* folder.

11.1.3. Stop, Abort and Download

At the end of a test run, either when it finishes by itself or when Stop is pressed, all results are gathered and moved to this location.

/root/cdi2/Results

Upon Download Results the content of the *Results* folder is packed into a single file, the result archive. The file is then downloaded to the computer where the browser GUI is located. The filename is generated automatically using the testcase name and the actual date/time string. Note that upon Abort no result archive is generated.

Example of a result archive file:

BSMS_BSM-G01_2018-08-01_06-55-52.tar.xz

To open the result archive the 7-Zip program is recommended.

11.2. How to install a new testcase

All testcases are stored on the Simulation Computer in the folder */usr/share/cdi2/cdi2-testcases*.

New testcases can be added in the corresponding sub folders (BSMS or DS) via WinSCP. After a new testcase has been added the Test Controller needs to be restarted. Either by rebooting the Simulation Computer or by running the following shell command:

```
# systemctl restart testcontroller
```

For convenience, above command can be issued on the GUI-PC using this batch script as well:

On the BSMS

C:\CDI2\20_RestartTestController_BSMS.bat

On the DS

C:\CDI2\20_RestartTestController_DS.bat

11.3. Test Case Generator for the DS

An alternate way to generate a new sorting test case for the DS is to use a test case generator.

The generator is based on an OpenDocument spreadsheet located in

C:\CDI2\05-LibreOffice-Tools\TestCaseGenerator_V1_2.ods.

It provides an interface to configure each of the 16 device simulator nodes as a CS, Detector or an IEU, and whether the node should use TTS. For nodes with TTS the transport clock resolution and the BP offset can be configured.

Once the testcase has been generated by the tool it can be installed on the DS. For this purpose, the output folder must be copied to the VisionBox into folder [see 11.2]

/usr/share/cdi2/cdi2-testcases/device-simulator.

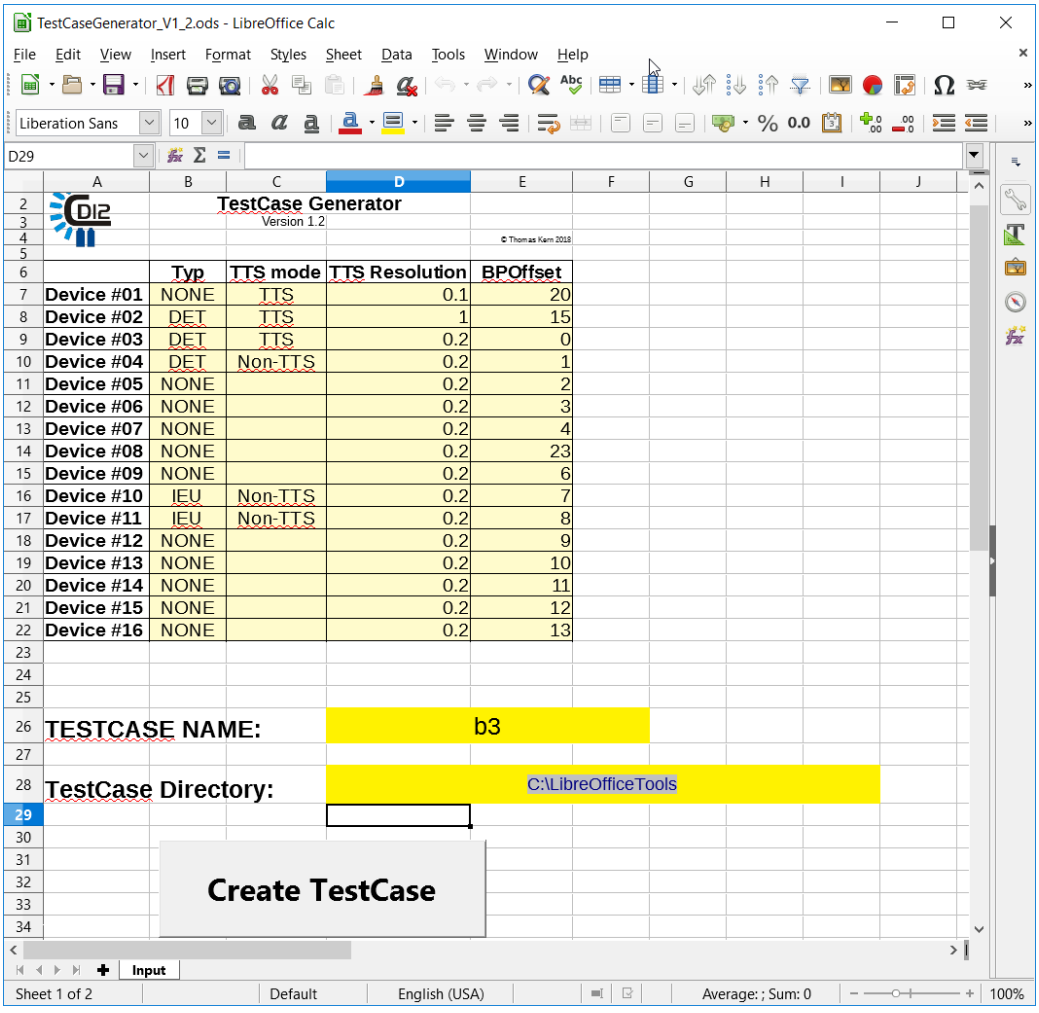


Figure 47: Test Case Generator

11.4. IDB Sender Setup

11.4.1. Testcase Settings

The IDB Sender is used for simulating a CS in the device simulator. It is an extra simulation instance (refer 9.1 also) and has to be listed in the `testcaseinfo.json` of the testcase. Most testcases inherit from the general testcase DS_BSM-G01. Thus, DS_BSM-G01 contains an IDB Sender setup, as shown in the example below. Its metafile points to `idb_sender_run_config.json` with parameters for the IDB Sender executable.

```
root@dev-sim-visionbox:/usr/share/cdi2/cdi2-testcases/device-simulator/DS_BSM-
G01# cat testcaseinfo.json

{
  "Name": "DS_BSM-G01 Sorting",
  "LogDir": "log",
  "Data": [
    { "RemoteDir": "DS_BSM-G01",
      "MetaFile" : "cn1_run_config.json",
      "TestFiles": ["device_infos",
                    "log",
                    "device_sorting_test.lua",
                    "cs_sorting_test.lua",
                    "config_device.xml"],
      "DeviceID" : "DMB_DEVICE_01" },
...
    { "RemoteDir": "DS_BSM-G01",
      "MetaFile" : "idb_sender_run_config.json",
      "TestFiles": ["log",
                    "pictures",
                    "idbImgDirList.txt"],
      "DeviceID" : "IDB_DEV" }
```

```
root@dev-sim-visionbox:/usr/share/cdi2/cdi2-testcases/device-simulator/DS_BSM-
G01# cat idb_sender_run_config.json
{
    "name"        : "DS_BSM-G01 Sorting",
    "description" : "Integration Test",
    "script"      : null,
    "args"        : [ "--imglist",     "./idbImgDirList.txt",
                      "--imgformat",   "512x1024",
                      "--multicastip", "239.205.18.1",
                      "--count",       "-1",
                      "--triggermode", "MC.224.0.0.100:5000"],
    "simulator"   : "/usr/bin/idbfsender"
}
root@dev-sim-visionbox:/usr/share/cdi2/cdi2-testcases/device-simulator/DS_BSM-
G01#
```

11.4.2. Image List Format and Stream Data Files

The *idbfsender* scans all directories listed in the image list file for stream data files. The stream data file must be PGM formated, 8-bit monochrome, images with following filename naming convention.

<*stream_number*>_<*tag*>.pgm

stream_number
This number is the stream number. The number may use leading zeros and denotes the stream number which shall be used for the image. Image streams must include 0, 1, 2, 3, 4, 5, 6 and 7 for mandatory streams. Optional streams continue with 8 and above.

tag
Unique text to mark all files belonging to the same banknote. e.g. images with tag=BN0001 belong to one banknote, whereas images with tag=BN0002 belong to another banknote.

Example *idbImgDirList.txt:*

```
# comment line
#
# image set 1
./pictures-set1
# image set 2
./pictures-set2
```

Example file names:

```
pictures-set1/00_BN001.pgm
pictures-set1/01_BN001.pgm
pictures-set1/02_BN001.pgm
pictures-set1/03_BN001.pgm
pictures-set1/04_BN001.pgm
pictures-set1/05_BN001.pgm
pictures-set1/06_BN001.pgm

pictures-set1/00_BN002.pgm
pictures-set1/01_BN002.pgm
pictures-set1/02_BN002.pgm
pictures-set1/03_BN002.pgm
pictures-set1/04_BN002.pgm
pictures-set1/05_BN002.pgm
pictures-set1/06_BN002.pgm
```

Note that the tag field needs to be unique within the same folder only. Thus, it is permitted to reuse the same tag within other directories.

Example reusing tag=BN001 in pictures-set2

```
pictures-set2/00_BN001.pgm
pictures-set2/01_BN001.pgm
```

```
pictures-set3/02_BN001.pgm
pictures-set2/03_BN001.pgm
pictures-set2/04_BN001.pgm
pictures-set2/05_BN001.pgm
pictures-set2/06_BN001.pgm
```

11.4.3. How to run Optional Images

A testset for sending optional streams requires a proper image set with a matching *device_info.xml* file.

Testcases "BSMS_BSM-G01-HR Sorting" and "DS_BSM-G01-HR Sorting" are provided as reference examples for using optional streams. Please refer to 11.6 for details.

11.4.4. Debugging Hints

For debug purposes the *idbfsender* writes rotating log files, named *idb_<n>.log*, starting with *idb_0.log*. The files are created at the working directory of the *idbfsender* (e.g. */var/tmp/<testcase_name>* during test and are included in the download package. Please note that the */var/tmp/<testcase_name>* is moved to */root/Results/CurrentTestRun* after completion of the testcase and */var/tmp/<testcase_name>* might not exist after a successful test.

If a testcase has not completed for unknown reasons, see files in /var/tmp/<testcase_name> or */root/Results/CurrentTestRun*.

Example of an idbfsender *idb_0.log:*

```
2021-07-06T07:17:16.617348Z [INFO]: {"message":"Parsing options","role":"IDB
Sender","status":"RUNNING","version":"Jul  2 2021, 18:20:13"}
2021-07-06T07:17:16.617599Z [INFO]: {"message":"Scanning image directories","role":"IDB
Sender","status":"RUNNING"}
2021-07-06T07:17:16.617653Z [INFO]: reading --imglist ./idbImgDirList.txt
2021-07-06T07:17:16.617772Z [INFO]: scanning image directory ./pictures/BN00001
2021-07-06T07:17:16.618374Z [INFO]: scanning image directory ./pictures/BN00002
2021-07-06T07:17:16.618643Z [INFO]: scanning image directory ./pictures/BN00003
2021-07-06T07:17:16.618904Z [INFO]: scanning image directory ./pictures/BN00004
2021-07-06T07:17:16.619183Z [INFO]: scanning image directory ./pictures/BN00005
2021-07-06T07:17:16.619452Z [INFO]: scanning image directory ./pictures/BN00006
2021-07-06T07:17:16.619720Z [INFO]: scanning image directory ./pictures/BN00007
2021-07-06T07:17:16.619989Z [INFO]: scanning image directory ./pictures/BN00008
2021-07-06T07:17:16.620262Z [INFO]: scanning image directory ./pictures/BN00009
2021-07-06T07:17:16.620530Z [INFO]: scanning image directory ./pictures/BN00010
2021-07-06T07:17:16.620788Z [INFO]: 10 transmit image sets found
2021-07-06T07:17:16.620806Z [INFO]: 200 transmit channel images found
2021-07-06T07:17:16.620820Z [INFO]: {"message":"Mmap(2)ing image sets","role":"IDB
Sender","status":"RUNNING"}
2021-07-06T07:17:16.621048Z [INFO]: ./pictures/BN00001/00_BN0001.pgm: 488x920
2021-07-06T07:17:16.621225Z [INFO]: ./pictures/BN00001/01_BN0001.pgm: 488x920
2021-07-06T07:17:16.621387Z [INFO]: ./pictures/BN00001/02_BN0001.pgm: 488x920
2021-07-06T07:17:16.621558Z [INFO]: ./pictures/BN00001/03_BN0001.pgm: 488x920
2021-07-06T07:17:16.621721Z [INFO]: ./pictures/BN00001/04_BN0001.pgm: 488x920
2021-07-06T07:17:16.621883Z [INFO]: ./pictures/BN00001/05_BN0001.pgm: 488x920
2021-07-06T07:17:16.622047Z [INFO]: ./pictures/BN00001/06_BN0001.pgm: 488x920
2021-07-06T07:17:16.622208Z [INFO]: ./pictures/BN00001/07_BN0001.pgm: 488x920
2021-07-06T07:17:16.622369Z [INFO]: ./pictures/BN00001/08_BN0001.pgm: 488x920
2021-07-06T07:17:16.622530Z [INFO]: ./pictures/BN00001/09_BN0001.pgm: 488x920
2021-07-06T07:17:16.622700Z [INFO]: ./pictures/BN00001/10_BN0001.pgm: 488x920
2021-07-06T07:17:16.622862Z [INFO]: ./pictures/BN00001/11_BN0001.pgm: 488x920
2021-07-06T07:17:16.623018Z [INFO]: ./pictures/BN00001/12_BN0001.pgm: 488x920
2021-07-06T07:17:16.623177Z [INFO]: ./pictures/BN00001/13_BN0001.pgm: 488x920
2021-07-06T07:17:16.623335Z [INFO]: ./pictures/BN00001/14_BN0001.pgm: 488x920
2021-07-06T07:17:16.623843Z [INFO]: ./pictures/BN00001/15_BN0001.pgm: 976x1840
2021-07-06T07:17:16.624347Z [INFO]: ./pictures/BN00001/16_BN0001.pgm: 976x1840
2021-07-06T07:17:16.624855Z [INFO]: ./pictures/BN00001/17_BN0001.pgm: 976x1840
2021-07-06T07:17:16.624901Z [INFO]: ./pictures/BN00001/18_BN0001.pgm: 976x16
2021-07-06T07:17:16.624946Z [INFO]: ./pictures/BN00001/19_BN0001.pgm: 976x16
```

```
2021-07-06T07:17:16.625108Z [INFO]: ./pictures/BN00002/00_BN0002.pgm: 488x920
2021-07-06T07:17:16.625263Z [INFO]: ./pictures/BN00002/01_BN0002.pgm: 488x920
2021-07-06T07:17:16.625423Z [INFO]: ./pictures/BN00002/02_BN0002.pgm: 488x920
2021-07-06T07:17:16.625585Z [INFO]: ./pictures/BN00002/03_BN0002.pgm: 488x920
2021-07-06T07:17:16.625745Z [INFO]: ./pictures/BN00002/04_BN0002.pgm: 488x920
2021-07-06T07:17:16.625906Z [INFO]: ./pictures/BN00002/05_BN0002.pgm: 488x920
....
2021-07-06T07:17:16.661109Z [INFO]: ./pictures/BN00010/18_BN0010.pgm: 976x16
2021-07-06T07:17:16.661153Z [INFO]: ./pictures/BN00010/19_BN0010.pgm: 976x16
2021-07-06T07:17:16.661174Z [INFO]: {"message":"Setup thread pool, trigger, and TX
tasks","role":"IDB Sender","status":"RUNNING"}
2021-07-06T07:17:16.661349Z [INFO]: TX thread pool configured, 1+2 TX threads
2021-07-06T07:17:16.666133Z [INFO]: {"message":"Setup TX paket mmap ring","role":"IDB
Sender","status":"RUNNING"}
2021-07-06T07:17:16.717681Z [INFO]: TX packet ring: frame size 12288, block size 12288, blocks
128, frames 128
2021-07-06T07:17:16.777886Z [INFO]: configured TX packet ring on ni4, MTU 9000, slots 128
2021-07-06T07:17:16.777925Z [INFO]: {"message":"Ready to send","role":"IDB
Sender","status":"RUNNING"}
2021-07-06T07:17:16.778644Z [INFO]: starting --triggermode MC, TID ffff977a91e0
2021-07-06T07:17:27.779437Z [INFO]: {"failures":0,"message":"100 BNs sent with 0
errors","role":"IDB Sender","sentBNs":107,"sentBlocks":2135,"status":"RUNNING"}
2021-07-06T07:17:29.779708Z [INFO]: {"failures":0,"message":"200 BNs sent with 0
errors","role":"IDB Sender","sentBNs":207,"sentBlocks":4140,"status":"RUNNING"}
2021-07-06T07:17:31.779971Z [INFO]: {"failures":0,"message":"300 BNs sent with 0
errors","role":"IDB Sender","sentBNs":307,"sentBlocks":6140,"status":"RUNNING"}
2021-07-06T07:17:33.780222Z [INFO]: {"failures":0,"message":"400 BNs sent with 0
errors","role":"IDB Sender","sentBNs":407,"sentBlocks":8140,"status":"RUNNING"}
2021-07-06T07:17:35.780445Z [INFO]: {"failures":0,"message":"500 BNs sent with 0
errors","role":"IDB Sender","sentBNs":507,"sentBlocks":10136,"status":"RUNNING"}
2021-07-06T07:17:37.780697Z [INFO]: {"failures":0,"message":"600 BNs sent with 0
errors","role":"IDB Sender","sentBNs":608,"sentBlocks":12136,"status":"RUNNING"}
2021-07-06T07:17:39.780949Z [INFO]: {"failures":0,"message":"700 BNs sent with 0
errors","role":"IDB Sender","sentBNs":708,"sentBlocks":14136,"status":"RUNNING"}
2021-07-06T07:17:41.781209Z [INFO]: {"failures":0,"message":"800 BNs sent with 0
errors","role":"IDB Sender","sentBNs":808,"sentBlocks":16142,"status":"RUNNING"}
2021-07-06T07:17:43.781439Z [INFO]: {"failures":0,"message":"900 BNs sent with 0
errors","role":"IDB Sender","sentBNs":908,"sentBlocks":18141,"status":"RUNNING"}
2021-07-06T07:17:45.720892Z [INFO]: GVSP LEADER 20000, PAYLOAD 1399000, TRAILER 20000, blocks
20000, bytes 12243250000
2021-07-06T07:17:45.720934Z [INFO]: shutting down...
```

idbfsender parameter list

Parameter	Description
`-h [ --help ]`	produce help message
`-v [ --verbose ]`	verbose logging
`-l [ --imglist ] arg`	text document listing all image directories to scan for stream data
`-f [ --imgformat ] arg`	expected image format: This parameter is obsolete and is ignored. The idbfsender takes the images size from the stream data files instead. By this means each stream may use its own size.
`-i [ --multicastip ] arg`	multicast IP address of image streams use 239.205.18.1 for CDI2 IDB
`-c [ --count ] arg`	-1: send BNs until termination, 0: send BNs of --imglist once, n: send n BNs
`-t [ --triggermode ] arg (=FREE)`	one of MC.:, DIG., or FREE
`-r [ --rate ] arg (=0)`	send --rate BN per seconds, 0 for best effort
`-b [ --bnidstart ] arg (=1)`	start with BNID --bnidstart
`--seq`	check BNID sequence
`--log-dir arg (=./LOG/idbfsender)`	set logging directory
`--result-dir arg (=./RESULT/idbfsender)`	set directory for RESULTS.json file
`--errRATE arg (=0)`	inject errors every --errRATE BN, 0: no error injection
`--errNUMBER arg (=0)`	stop error injection after --errNUMBER errors, 0: never stop error injection
`--errCHANNEL`	drop a complete image channel
`--errLEADER`	drop a LEADER packet
`--errPAYLOAD`	drop a PAYLOAD packet
`--errTRAILER`	drop a TRAILER packet
`--errORDER`	wrong sequence number (GVSP packet_id)
`-u [ --mtu ] arg (=9000)`	MTU to use for TX interface
`-s [ --txSlots ] arg (=128)`	number of TX slots in packet ring
`-x [ --xmtthreads ] arg (=2)`	number of TX threads
`--maxthroughput arg (=0)`	egress traffic shaping max. throughput [Mbps]
`--maxburst arg (=0)`	egress traffic shaping max. burst size
`-p [ --rtPrio ] arg (=10)`	real-time priority

`--maxStall arg (=10)`	maximum stall time [ms] avoiding overlapping BNs

Table 7: IDB sender parameters

11.5. IDB Receiver Setup

11.5.1. Testcase Settings

The IDB Receiver is used in the BSM simulator to receive image data coming from a CS. It is an extra simulation instance (refer 9.1 also) and has to be listed in the `testcaseinfo.json` of the testcase. Most testcases inherit from the general testcase BSMS_BSM-G01. Thus, BSMS_BSM-G01 contains an IDB Receiver setup, as shown in the example below. Its metafile points to `idb_receiver_run_config.json` with parameters for the IDB Receiver executable `idbfreceiver`.

```
root@bsm-sim-visionbox:/usr/share/cdi2/cdi2-testcases/bsm-simulator/BSMS_BSM-
G01# cat testcaseinfo.json
{
  "Name": "BSMS_BSM-G01 Sorting",
  "LogDir": "log",
  "Data": [
    { "RemoteDir": "BSMS_BSM-G01",
      "MetaFile" : "mn_run_config.json",
      "TestFiles": ["config_bsm.xml",
                    "mn_run_config.json",
                    "sorting_test.lua",
                    "log"],
      "DeviceID" : "BSM_SIM" },

    { "RemoteDir": "BSMS_BSM-G01",
      "MetaFile" : "idb_receiver_run_config.json",
      "TestFiles": ["log",
                    "idb_receiver_run_config.json"],
      "DeviceID" : "IDB_BSM" }
  ],
....
```

```
root@bsm-sim-visionbox:/usr/share/cdi2/cdi2-testcases/bsm-simulator/BSMS_BSM-
G01# cat idb_receiver_run_config.json
{
    "name"        : "BSMS_BSM-G01 Sorting",
    "description" : "Integration Test",
    "script"      : null,
    "args"        : [ "-f", "10",
                      "/usr/bin/idbfreceiver",
                      "--destination", "./pictures",
                      "--multicastip", "239.205.18.1" ],
    "simulator"   : "chrt"
}
```

11.5.2. Received Image File Format

The received stream images are stored in the *./pictures* folder. The image files are PGM formatted, 8-bit monochrome, with following filename naming convention.

<*stream_number*>_BN<*bnid*>.pgm

stream_number
This number is the stream number

bnid
This field is the left zero-padded banknote id of the received banknote. Images received with detected errors are saved as <*stream_number* >_BN<bnid>_ERR.pgm files, the PGM header comment shows detailed information.

Example:

02_BN000000036.pgm
received image from stream 02 with BNID=36

Important note:

In contrast to the previous version, the IDB receiver stores all images in the same folder and does not use extra folders for each banknote. Having all images files in one folder makes file handling easier, e.g. receive files can be used by the IDB sender directly.

11.5.3. How to run Optional Images

The IDB receiver allocates receive buffers in main memory in advance. As the main memory is a limited resource, the parameters `-streams`, `-depth` and `-hr` are used to control the amount of required memory. `-streams` sets the number of streams to be received, `-depth` sets the maximum number of banknotes kept in the ring buffer and `-hr` sets the maximum image size (see Table 9 for complete parameter list).

Parameter	Mandatory streams only (default)	Mandatory and 12 optional streams
`--streams`	8	20
`--depth`	1000	100
`--hr`	not set (allocate size 512x1024)	set (allocate size 1024x2048)

Table 8: IDB receiver parameters for optional streams

Examples of `idb_receiver_run_config.json`:

Mandatory streams only (default)

```
{
    "name"        : "BSMS_BSM-G01 Sorting",
    "description" : "Integration Test",
    "script"      : null,
    "args"        : [ "-f", "10",
                      "/usr/bin/idbfreceiver",
                      "--destination", "./pictures",
```

```
                    "--depth", "1000",
                    "--streams", "8",
                    "--multicastip", "239.205.18.1" ],
    "simulator"   : "chrt"
}
```

Mandatory and optional streams

```
{
    "name"        : "BSMS_BSM-G01 Sorting",
    "description" : "Integration Test",
    "script"      : null,
    "args"        : [ "-f", "10",
                      "/usr/bin/idbfreceiver",
                      "--destination", "./pictures",
                      "--depth", "100",
                      "--streams", "20",
                      "--hr",
                      "--multicastip", "239.205.18.1" ],
    "simulator"   : "chrt"
}
```

Testcases "BSMS_BSM-G01-HR Sorting" and "DS_BSM-G01-HR Sorting" are provided as reference examples for using optional streams. Please refer to 11.6 for more details.

11.5.4. Debugging Hints

For debug purposes the *idbfreceiver* writes rotating log files *idb_<n>.log, staring with idb_0.log*. The file is created at the working directory of the *idbfreceiver* (e.g. */var/tmp/<testcase_name>* during test and it is included in the download package. Please note that the */var/tmp/<testcase_name>* is moved to */root/Results/CurrentTestRun* after completion of the testcase and */var/tmp/<testcase_name>* does not exist after a successful test.

If a testcase has not completed for unknown reasons, see files in /var/tmp/<testcase_name> or */root/Results/CurrentTestRun*.

Example of an *idbfreceiver idb_0.log:*

```
2021-07-06T11:02:28.891173Z [INFO]: {"message":"Parsing options","role":"IDB
Receiver","status":"RUNNING","version":"Jul  2 2021, 18:22:17"}
2021-07-06T11:02:30.658184Z [INFO]: {"message":"Image stream reassembly buffer configured for 20
image streams/BN, 100 BN save depth","role":"IDB Receiver","status":"RUNNING"}
2021-07-06T11:02:30.658290Z [INFO]: {"message":"Setup RX paket mmap ring","role":"IDB
Receiver","status":"RUNNING"}
2021-07-06T11:02:30.704064Z [INFO]: RX packet ring: frame size 12288, block size 12288, blocks
2048, frames 2048
2021-07-06T11:02:30.745652Z [INFO]: configured RX packet ring on ni4, MTU 9000, slots 2048
2021-07-06T11:02:30.745701Z [INFO]: {"message":"Ready to receive","role":"IDB
Receiver","status":"RUNNING"}
2021-07-06T11:02:30.745759Z [INFO]: RX packet ring: ring processing thread tied to CPU 7
2021-07-06T11:02:44.746847Z [INFO]: {"Gbps_MAC":"0.803249","failed":0,"message":"100 BNs received
with 0 errors","rcvdBNIDs":114,"role":"IDB Receiver","status":"RUNNING"}
2021-07-06T11:02:46.747181Z [INFO]: {"Gbps_MAC":"4.868041","failed":0,"message":"200 BNs received
with 0 errors","rcvdBNIDs":214,"role":"IDB Receiver","status":"RUNNING"}
2021-07-06T11:02:48.748089Z [INFO]: {"Gbps_MAC":"4.918769","failed":0,"message":"300 BNs received
with 0 errors","rcvdBNIDs":314,"role":"IDB Receiver","status":"RUNNING"}
2021-07-06T11:02:50.748415Z [INFO]: {"Gbps_MAC":"4.871095","failed":0,"message":"400 BNs received
with 0 errors","rcvdBNIDs":414,"role":"IDB Receiver","status":"RUNNING"}
```

```
2021-07-06T11:02:52.748874Z [INFO]: {"Gbps MAC":"4.922154","failed":0,"message":"500 BNs received
with 0 errors","rcvdBNIDs":514,"role":"IDB Receiver","status":"RUNNING"}
2021-07-06T11:02:54.749482Z [INFO]: {"Gbps_MAC":"4.874641","failed":0,"message":"600 BNs received
with 0 errors","rcvdBNIDs":614,"role":"IDB Receiver","status":"RUNNING"}
2021-07-06T11:02:56.750383Z [INFO]: {"Gbps_MAC":"4.929154","failed":0,"message":"700 BNs received
with 0 errors","rcvdBNIDs":715,"role":"IDB Receiver","status":"RUNNING"}
2021-07-06T11:02:58.750679Z [INFO]: {"Gbps_MAC":"4.896574","failed":0,"message":"800 BNs received
with 0 errors","rcvdBNIDs":815,"role":"IDB Receiver","status":"RUNNING"}
2021-07-06T11:03:00.750966Z [INFO]: {"Gbps_MAC":"4.896598","failed":0,"message":"900 BNs received
with 0 errors","rcvdBNIDs":915,"role":"IDB Receiver","status":"RUNNING"}
2021-07-06T11:03:02.476114Z [INFO]: shutting down...
2021-07-06T11:03:02.476348Z [INFO]: GVSP LEADER 20000, PAYLOAD 1399000, TRAILER 20000, blocks
20000, bytes 12243250909
2021-07-06T11:03:02.476374Z [INFO]: saving reassembled images...
2021-07-06T11:03:02.476404Z [INFO]: create directory for reassembled images: "./pictures"
```

idbfreceiver parameter list

Parameter	Description
-h [--help]	produce help message
-v [--verbose]	verbose logging
-d [--destination] arg	destination directory collecting received images
-i [--multicastip] arg	multicast IP address of image streams use 239.205.18.1 for CDI2 IDB
-c [--count] arg	0: receive BN until termination N: terminate after receiving N BNs
-s [--streams] arg (=8)	number of streams/images per BN
--depth arg (=1000)	max. number of BN to save at termination
--hr	allocate standard (512x1024) or HR (1024x2048) image buffers
--cks	calculate naive per-image checksum to detect single bit errors
--seq	check BNID sequence
--log-dir arg (=./LOG/idbfreceiver)	set logging directory
--result-dir arg (=./RESULT/idbreceiver)	set directory for RESULTS.json file
-u [--mtu] arg (=9000)	MTU to use for RX interface
-f [--rxSlots] arg (=2048)	number of RX slots in packet ring
-x [--rcvthreads] arg (=2)	number of RX threads

Table 9: IDB receiver parameters

11.6. IDB Reference Testcase for Optional Streams

A reference test showing the use of optional streams is provided. It consists of

"BSMS_BSM-G01-HR Sorting" and "DS_BSM-G01-HR Sorting" for BSMS and DS respectively. The testcases are extended versions of the generic testcases "BSMS_BSM-G01-HR Sorting" and "DS_BSM-G01 Sorting".

11.6.1. DS_BSM-G01-HR Sorting

"DS_BSM-G01-HR Sorting" implements an IDB sender, as described in 11.4.

The testcase files are located at this folder /usr/share/cdi2/cdi2-testcases/device-simulator/DS_BSM-G01-HR.

Hint: Testcase files can be extracted from the Debian package (cdi2-testcases-xxx.deb) also (with e.g. 7-zip)

`pictures-hr/` contains an image set for 20 streams

```
pictures-hr/00_BN0001.pgm : 488x920
pictures-hr/01_BN0001.pgm : 488x920
pictures-hr/02_BN0001.pgm : 488x920
pictures-hr/03_BN0001.pgm : 488x920
pictures-hr/04_BN0001.pgm : 488x920
pictures-hr/05_BN0001.pgm : 488x920
pictures-hr/06_BN0001.pgm : 488x920
pictures-hr/07_BN0001.pgm : 488x920
pictures-hr/08_BN0001.pgm : 488x920
pictures-hr/09_BN0001.pgm : 488x920
pictures-hr/10_BN0001.pgm : 488x920
pictures-hr/11_BN0001.pgm : 488x920
pictures-hr/12_BN0001.pgm : 488x920
pictures-hr/13_BN0001.pgm : 488x920
pictures-hr/14_BN0001.pgm : 488x920
pictures-hr/15_BN0001.pgm : 976x1840
pictures-hr/16_BN0001.pgm : 976x1840
pictures-hr/17_BN0001.pgm : 976x1840
pictures-hr/18_BN0001.pgm : 976x16
pictures-hr/19_BN0001.pgm : 976x16
```

Please note that images have different images size. Also note the mandatory image size of the mandatory streams 0-7 is 488x920 in this example, in contrast to DS_BSM-G01 having 512x1024.

`pictures-hr-10/` contains an annotated image set for 10 banknotes. The images contain an imprinted string with banknote number and stream number.

Device description should match and shall list 20 streams.

Example of `device_infos/device_info_camera_hr.xml`

```
<Images>
    <!--mandatory streams -->
```

```
        <Image type="R_left" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="0" gvspPayloadType="1" />
        <Image type="G_left" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="1" gvspPayloadType="1" />
        <Image type="B_left" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="2" gvspPayloadType="1" />
        <Image type="IR_left" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="3" gvspPayloadType="1" />
        <Image type="R_right" resolution="0.2" internalResolution="0.1"
            height="920" width="488" idbStreamNr="4" gvspPayloadType="1" />
        <Image type="G_right" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="5" gvspPayloadType="1" />
        <Image type="B_right" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="6" gvspPayloadType="1" />
        <Image type="IR_right" resolution="0.2" internalResolution="0.1"
            height="920" width="488" idbStreamNr="7" gvspPayloadType="1" />
        <!--optional streams -->
        <Image type="TR_right" resolution="0.2" internalResolution="0.1"
            height="920" width="488" idbStreamNr="8" gvspPayloadType="1" />
        <Image type="GR_R_left" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="9" gvspPayloadType="1" />
        <Image type="GR_G_left" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="10" gvspPayloadType="1" />
        <Image type="GR_B_left" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="11" gvspPayloadType="1" />
        <Image type="GR_R_right" resolution="0.2" internalResolution="0.1"
            height="920" width="488" idbStreamNr="12" gvspPayloadType="1" />
        <Image type="GR_G_right" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="13" gvspPayloadType="1" />
        <Image type="GR_B_right" resolution="0.2" internalResolution="0.2"
            height="920" width="488" idbStreamNr="14" gvspPayloadType="1" />
        <Image type="HR_left" resolution="0.1" internalResolution="0.1"
            height="1840" width="976" idbStreamNr="15" gvspPayloadType="1" />
        <Image type="HR_right" resolution="0.1" internalResolution="0.1"
            height="1840" width="976" idbStreamNr="16" gvspPayloadType="1" />
        <Image type="HR_TR_left" resolution="0.1" internalResolution="0.1"
            height="1840" width="976" idbStreamNr="17" gvspPayloadType="1" />
        <Image type="BADPIX_left" resolution="0.1" internalResolution="0.1"
            height="16" width="976" idbStreamNr="18" gvspPayloadType="1" />
        <Image type="BADPIX_right" resolution="0.1" internalResolution="0.1"
            height="16" width="976" idbStreamNr="19" gvspPayloadType="1" />
    </Images>
```

Important Note:

All required files and directories of the testcase must be listed as “TestFiles” in *testcaseinfo.json*. Otherwise the files are not transferred to the testcase working directory, e.g. "**`pictures-hr-10`**" must be listed for above example.

testcaseinfo.json:

```
{ "RemoteDir": "DS_BSM-G01-HR",
  "MetaFile" : "idb_sender_run_config.json",
  "TestFiles": ["log",
                "pictures-hr-10",
                "idbImgDirList.txt"],
  "DeviceID" : "IDB_DEV" }
```

idbImgDirList.txt:

```
# banknote with mandatory and hr streams
#./pictures
# 10 annotated banknotes with mandatory and hr streams
./pictures-hr-10
```

idbsender_run_config.json:

```
{
```

```
        "name"        : "DS_BSM-G01-HR Sorting",
        "description" : "Integration Test",
        "script"      : null,
        "args"        : [ "--imglist",     "./idbImgDirList.txt",
                          "--multicastip", "239.205.18.1",
                          "--count",       "-1",
                          "--triggermode", "MC.224.0.0.100:5000"],
        "simulator"   : "/usr/bin/idbfsender"
```

11.6.2. BSMS_BSM-G01-HR Sorting

"BSMS_BSM-G01-HR Sorting" implements an IDB receiver, as described in 11.5.

The testcase files are located at this folder /usr/share/cdi2/cdi2-testcases/bsm-simulator/BSMS_BSM-G01-HR

Hint: Testcase files can be extracted from the Debian package (cdi2-testcases-xxx.deb) also (with e.g. 7-zip).

testcaseinfo.json:

```
    { "RemoteDir": "BSMS_BSM-G01-HR",
      "MetaFile" : "idb_receiver_run_config.json",
      "TestFiles": ["log",
                    "idb_receiver_run_config.json"],
      "DeviceID" : "IDB_BSM" }
```

idb_receiver_run_config.json:

```
    {
        "name"        : "BSMS_BSM-G01-HR Sorting",
        "description" : "Integration Test",
        "script"      : null,
        "args"        : [ "-f", "10",
                          "/usr/bin/idbfreceiver",
                          "--destination", "./pictures",
                          "--depth", "100",
                          "--streams", "20",
                          "--hr",
                          "--multicastip", "239.205.18.1" ],
        "simulator"   : "chrt"
    }
```

12. Tools

The simulator setup is delivered with some hardware and software tools. Their basic usage is described in this section. For detailed information and manuals refer to the referenced sources.

12.1. Wireshark

Wireshark is a network analyser which is used in the testcases to view and inspect the messages on the DMB. For this task it is equipped with some custom tools which are already installed on the GUI PC and can be found in the *CDI2-Installation* folder in "*03-CDI2-Wireshark-Plugins*".

Detailed documentation and manuals can be found at: https://www.wireshark.org/

For capturing the Powerlink network traffic with the required time resolution, the BSMS and the DS are equipped with a dedicated network logger, a ProfiShark 100M device. It is connected via USB to the GUI-PC (Sniffer Connector on the rear panel). The ProfiShark device installs as a network interface and Wireshark can use it as capture device. ProfiShark is capable to timestamp the captured network packets with a resolution < 100 ns. This feature is required to analyse and judge the quality of the Powerlink communication.

Note that the *profishark.dll* plugin, provided with the ProfiShark device, has to be installed when Wireshark logs originating from ProfiShark have to be analysed.

Refer to the manufacturer for further details

ProfiShark 100M, C1AP-100

http://www.profitap.com

For manual Wireshark installation or upgrade to a newer version refer to chapter 12.1.8.

12.1.1. General Usage and Capture

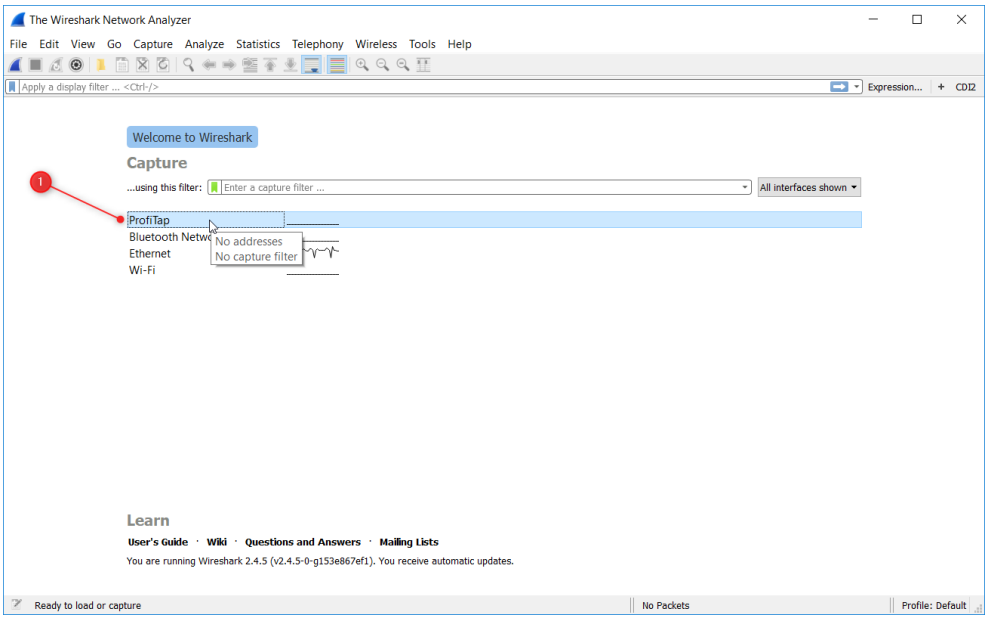


Figure 48: Select Profishark network card at startup

To start a capture open Wireshark and double click the ProfiTap interface (1).

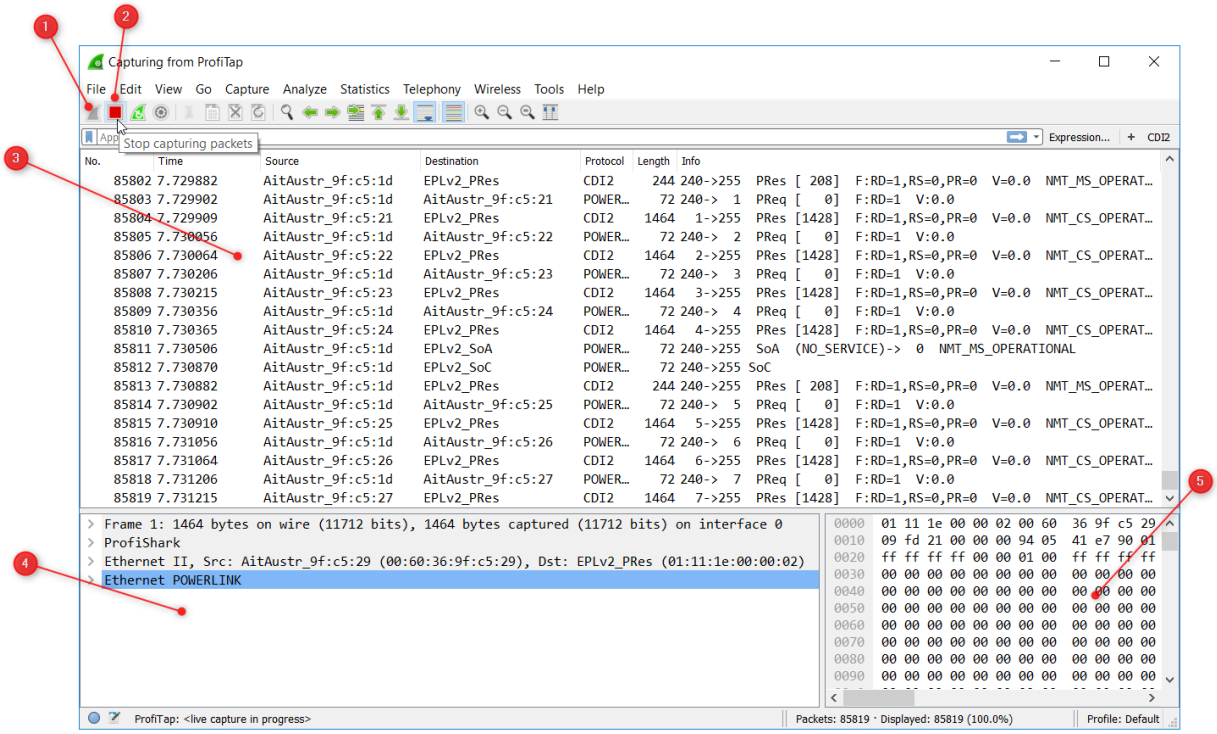


Figure 49: Wireshark main screen

A running capture can be stopped using the stop button (2). A new capture can then be started by clicking Start (1).

The captured packets are shown in overview (3) and the selected packet is shown in detail (4) and (5). (4) Shows the dissected package with all parsed protocol layers, including CDI2 fields. (5) shows the raw bytes of the packet.

12.1.2. Powerlink Timing

Another tool provided for CDI2 is the “Powerlink Timing” statistics. It can be started by clicking “Statistics” (1) -> “Powerlink Timing…” (2).

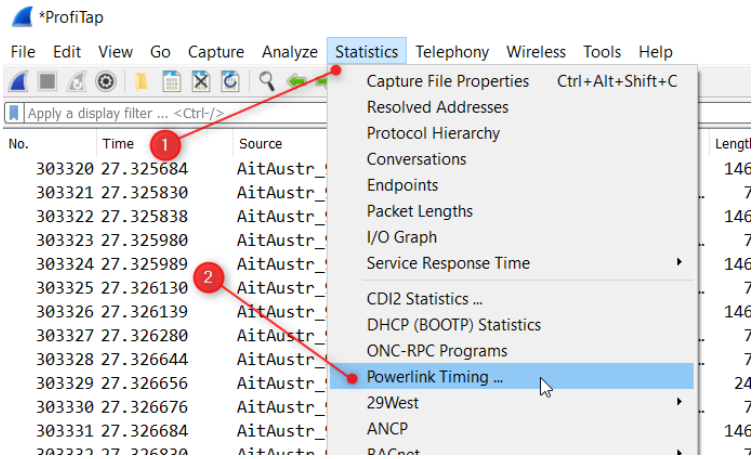


Figure 50: Open Powerlink timing window

The Powerlink Timing statistics provides a statistical summary of time delays involved in the Powerlink communication.

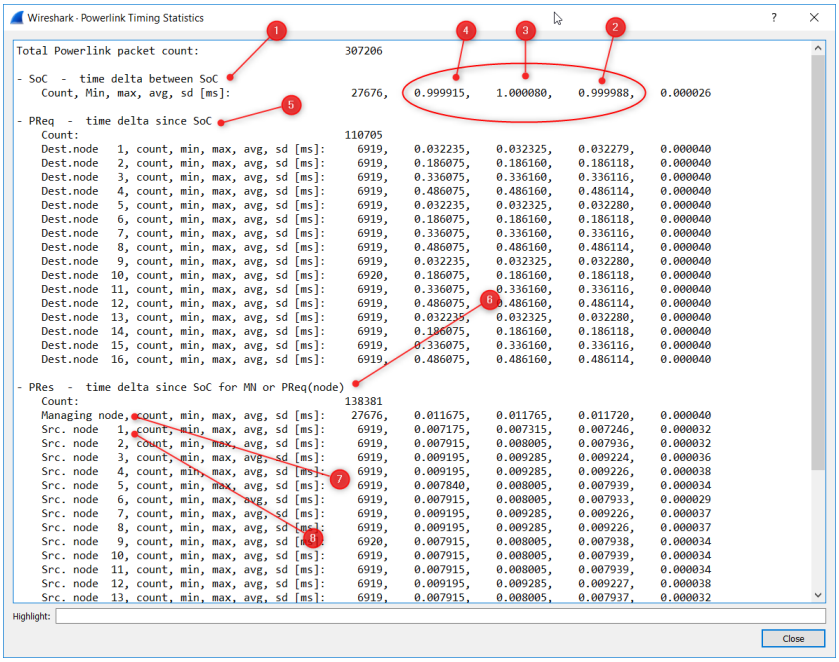


Figure 51: Powerlink Timing Window

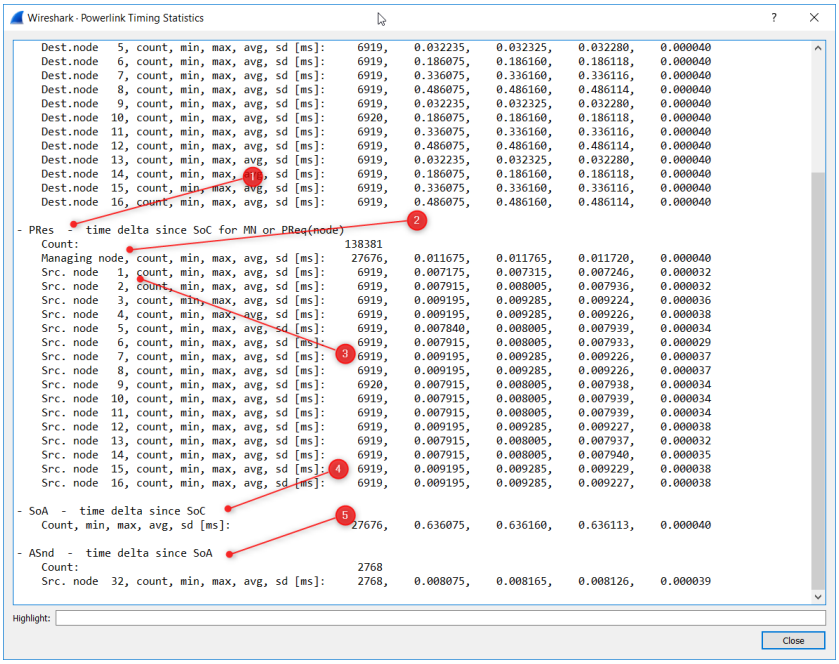


Figure 52: Powerlink timing window (cont.)

12.1.3. CDI2 Statistics

Similar to the Powerlink timing, a tool for timing statistics in the CDI2 protocol is provided. It can be started by clicking “Statistics” (1) -> “CDI2 Statistics…” (2).

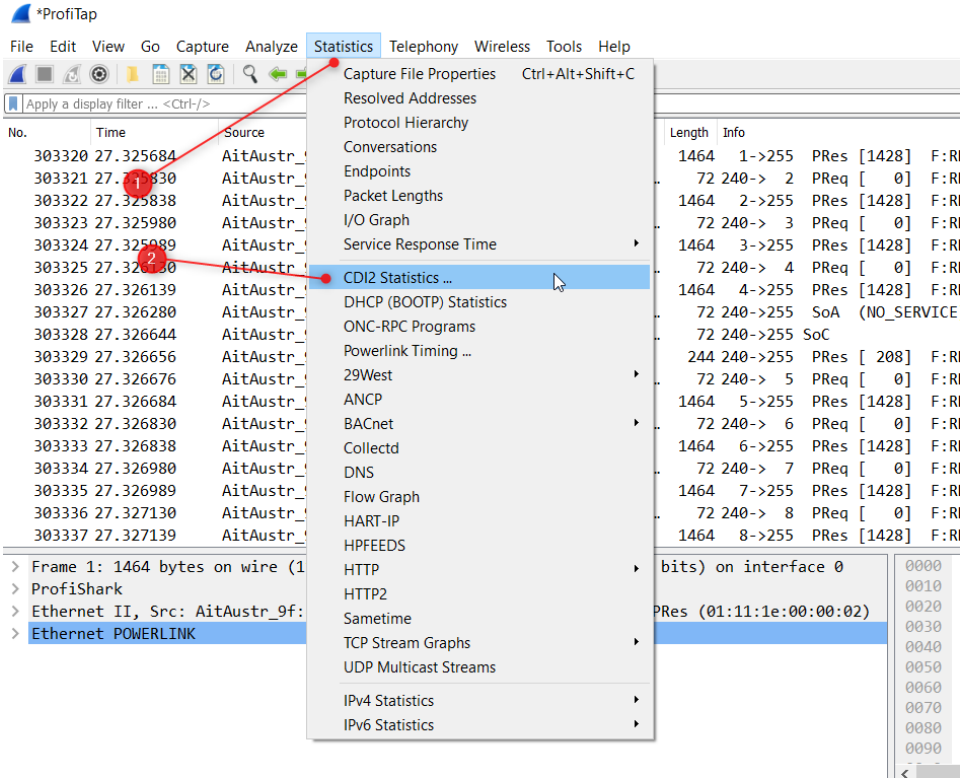


Figure 53: CDI2 Statistics window

12.1.4. Expert Info

Wireshark is able to detect errors in dissected packets. The CDI2 Wireshark plugins extend this capability also to the CDI2 protocol.

To see an overview of all messages that describe these errors, the “Expert Information” dialog is used. It can be opened by clicking “Analyse” (1) -> “Expert Information” (2). For large captures the loading time can be quite long, the status indictor (3) shows whether the loading is completed.

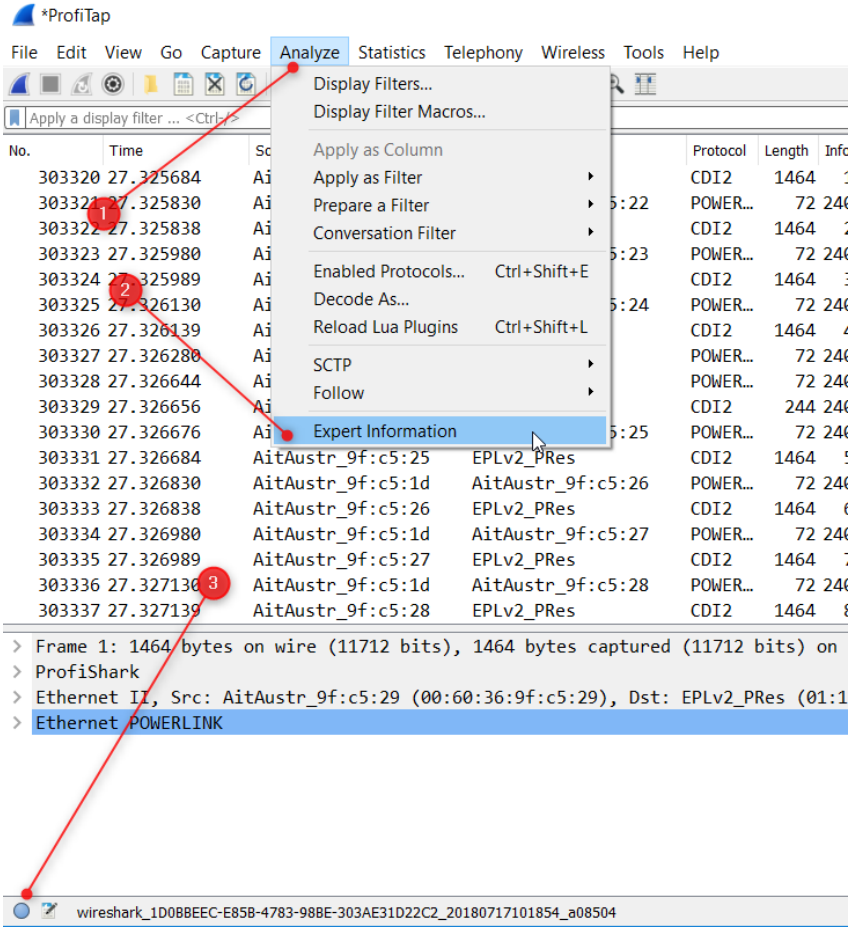


Figure 54: Expert Information

12.1.5. Apply filter expressions

Wireshark offers display filters to select specific packets and to exclude frames without interest. A standard filter equation for CDI2 is preinstalled and can be activated with the button "CDI2" (1). The filter selects active CDI2 packets with valid BNIDs (BNID not equal 0xFFFFFFFF). It can be used to track banknote results in sorting mode. Furthermore, the colouring of frames is used to indicate the type and content of packets. An example screen is shown in Figure 56. Due to the applied CDI2 filter equation, only frames with a valid BNID are shown. The colouring of the frames indicates the type of frame.

(dark blue)	BSMINFO frame with at least one trigger
(light blue)	BSMINFO frame with banknote recognition
(yellow)	frame with valid BNRESULT
(orange)	frame with valid BNRECOGNITION (coming from CS usually)

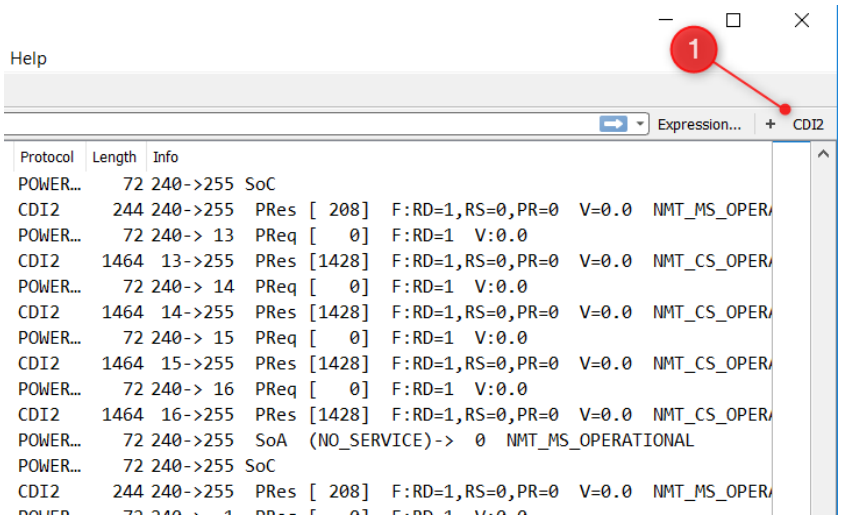


Figure 55: CDI2 filter

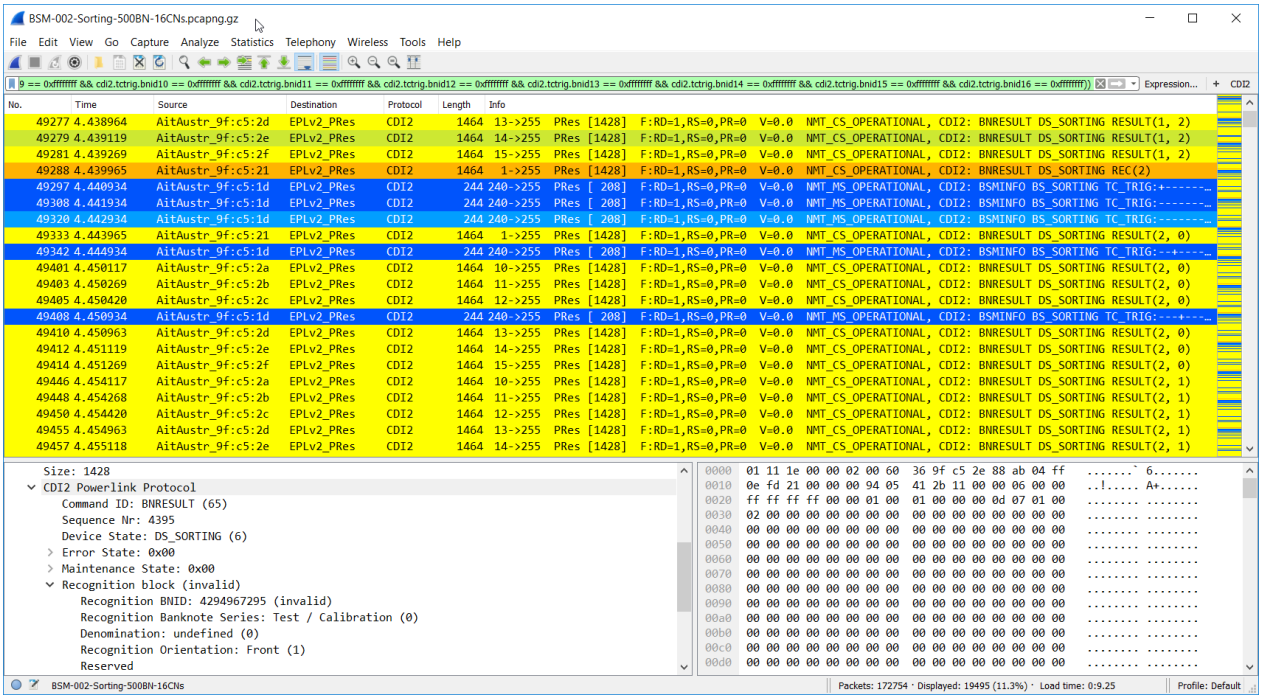


Figure 56: CDI2 filter and colouring of frames

12.1.6. CDI2 Protocol Settings

The CDI2 dissector plugin accepts parameters via the Wireshark protocol setting dialog,

Edit->Preferences->Protocols->CDI2. The same dialog can be opened alternatively by a right-click on a captured CDI2 frame and selecting *Protocol Preferences -> Open CDI2 Powerlink Protocol preferences.*

Essentially, this dialog gives user access to the limits used by the timing checker for the Expert Info [see 12.1.4]. Figure 57 shows a *CDI2 Powerlink Protocol preferences* dialog example.

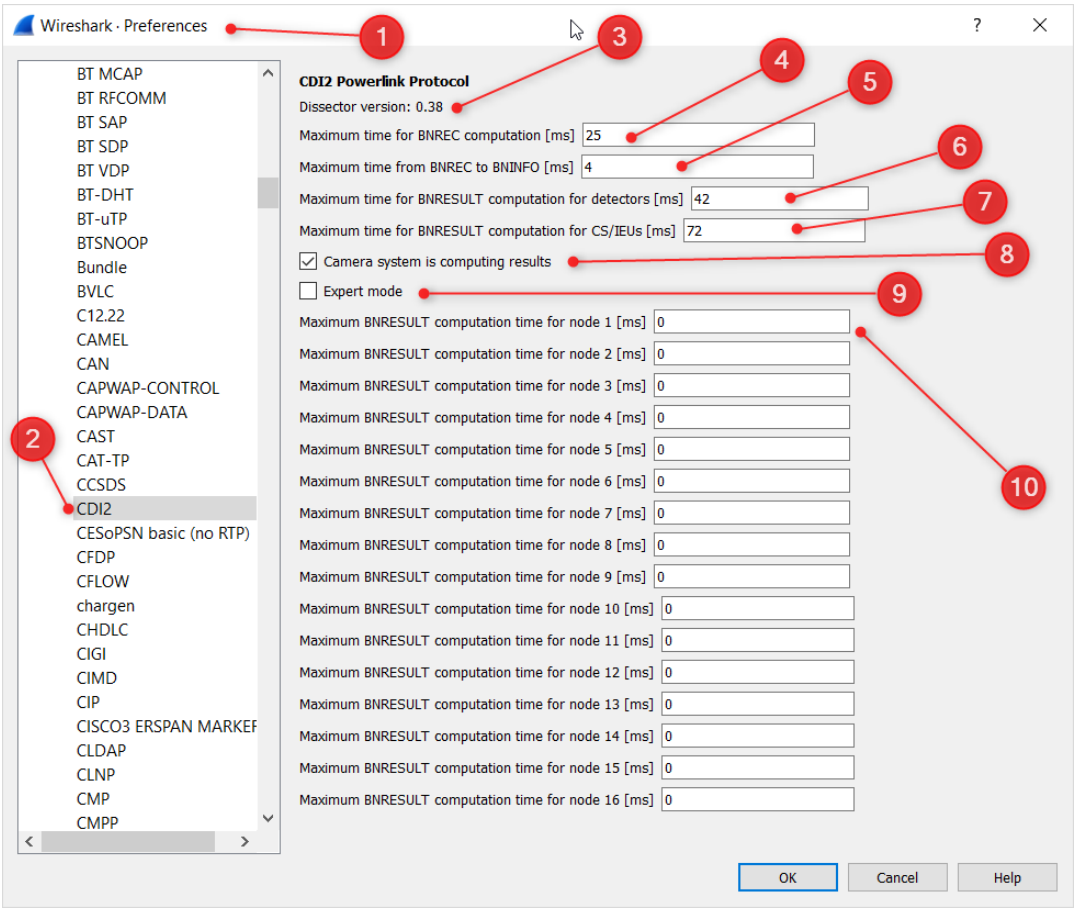


Figure 57: CDI2 Powerlink Protocol settings

(1) Wireshark Preferences dialog
(2) Select CDI2 Protocol
(3) Dissector version info [see 12.1.7 also]
(4) Maximum time for the CS to send the BNRECOGNITION packet (default: 25 ms)
(5) Maximum time for the BSM to send the received BNRECGNITION in the BSMINFO packet (default: 4 ms)
(6) Maximum time for a Detector to send its BNRESULT after trigger (default: 42 ms)
(7) Maximum time for a CS or IEU to den its BNRESULT after trigger (default: 72 ms)
(8) When checked, the CS is expected to send BNSRESULT (default: checked)
(9) Expert mode: when checked, the maximum BNRESULT time can be set for each node individually (default: unchecked)
(10) Maximum BNRESULT time for each node. Use 42 ms for a Detector and 72 ms for a CS or IEU. A value of 0 ms disables the timing check for a node.

Due to a fundamental limitation of the Wireshark dissector implementation, the CDI2 dissector may fail to determine the CDI2 type (CS, IEU, DET) of a controlled node from the network packet log for the first few (up to 5) banknotes. This may lead to inadvertent warnings for those banknotes. To overcome this effect, the expert mode settings are provided. The user can set individual maximum BNRESULT time values for each of the nodes 1 to 16 and the automatic detection of the CDI2 type (CS, IEU, DET) is ignored in this case. As a default the user shall enter 42 ms for a Detector and 72 ms for a CS or IEU. A value of 0 ms disables the timing check for a node.

An example with expert mode enabled is shown in Figure 58. Assuming node 1 is a CS, nodes 2 to 9 being Detectors and nodes 10 to 16 being IEUs.

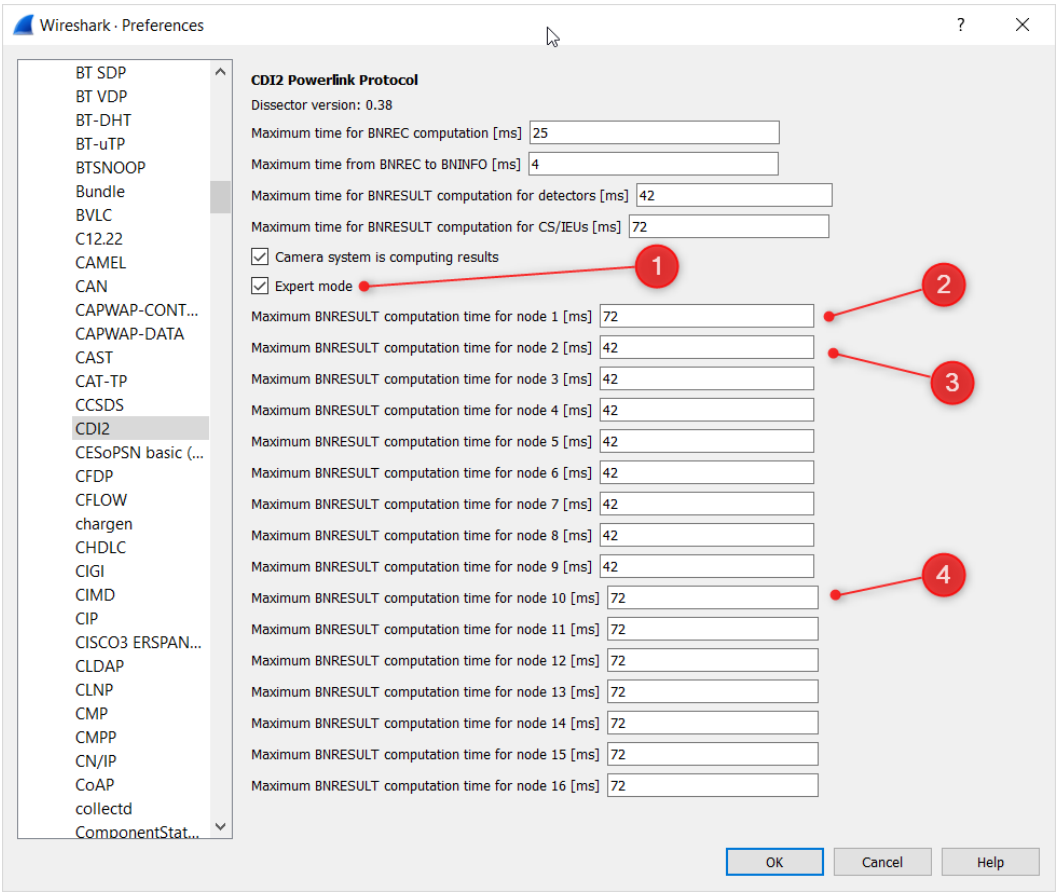


Figure 58: CDI2 Powerlink Protocol settings - Expert mode

(1) Expert mode: when checked, the maximum BNRESULT time can be set for each node individually
(2) Maximum BNRESULT computation time for node 1. Use 72 ms for a CS.
(3) Maximum BNRESULT computation time for node 2. Use 42 ms for a Detector.
(4) Maximum BNRESULT computation time for node 10. Use 72 ms for an IEU.

Note that the BNRECOGNITION and BNRESULT delays for the timing checks include the banknote capture time. Thus, the actual values depend on assumptions for the banknote length and the transport speed. Above examples assume 200 mm banknote length and 12.5 m/s banknote transport speed. Further it is assumed that the node does not request a none-zero trigger offset in its *device_info.xml* (bpOffset=0).

Following calculations apply accordingly.

BNRECOGNITION time of a CS:

220mm capture length / 12.5m/s = 17.6ms + 3ms Recognition + 4ms Powerlink Transmission => 24.6ms

BNRESULT time of a CS or IEU:

220mm capture length / 12.5m/s = 17.6ms + 5ms image transmission + 45ms Calculation + 4ms Powerlink Transmission => 71.6ms

BNRESULT time of a Detector:

220mm capture length / 12.5m/s = 17.6ms + 20ms Calculation + 4ms Powerlink Transmission => 41.6ms

12.1.7. Version Information

The Wireshark version used for the GUI PC is Version 2.4.5 (v2.4.5-0-g153e867ef1) or later. It can be checked by going to Help -> About Wireshark.

Using the “About Wireshark” dialog also allows checking of the installed plugins via the “Plugins” tap. New dissectors or updates to existing ones can be installed to the “Personal Plugins” folder listed in the “Folders” tab.

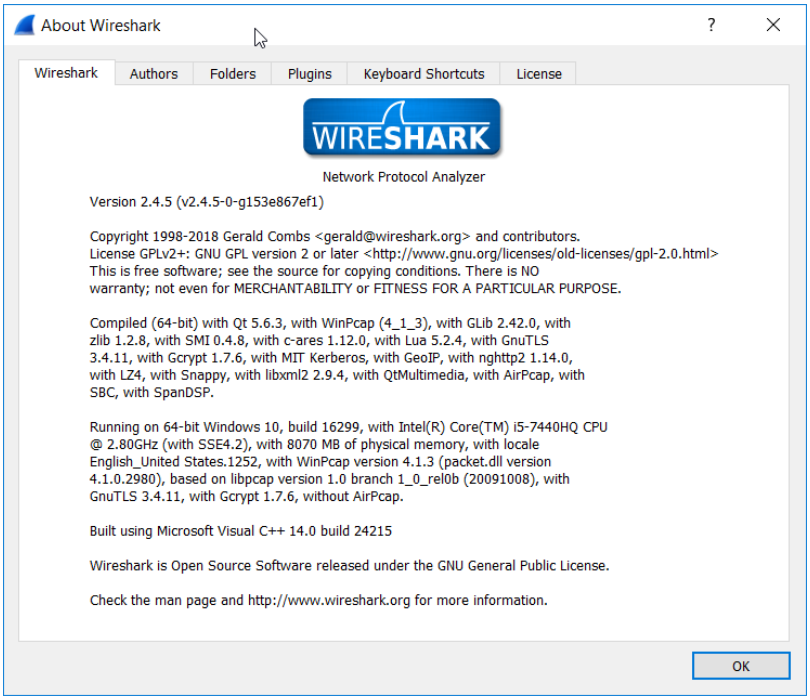


Figure 59: About Wireshark

In the Folders Tab the Wireshark tool lists the actual folders for Plugins. An example is shown in Figure 60.

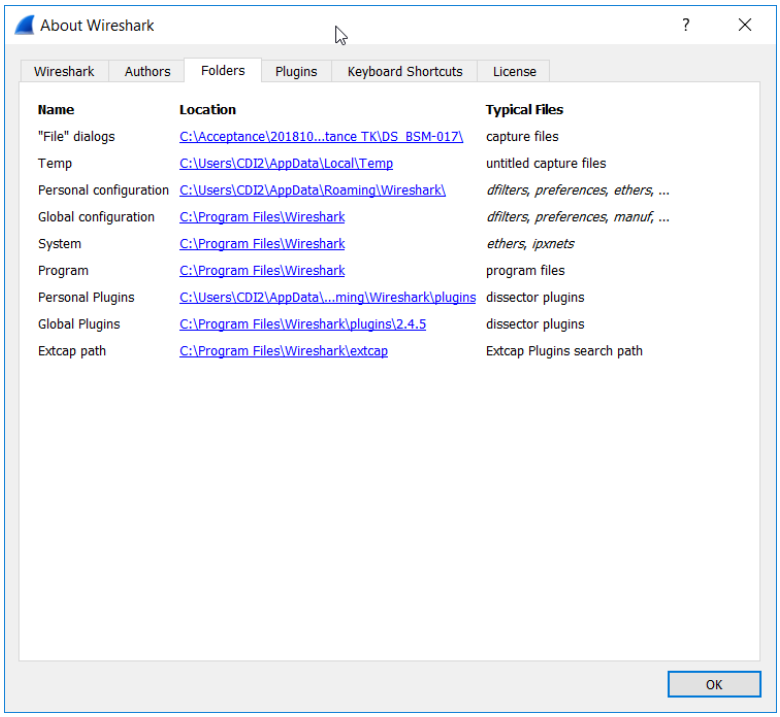


Figure 60: About Folders

In the Plugins Tab the Wireshark tool lists the installed plugins and their versions. The versions of the CDI2 plugins can be checked here. An example is shown in Figure 61.

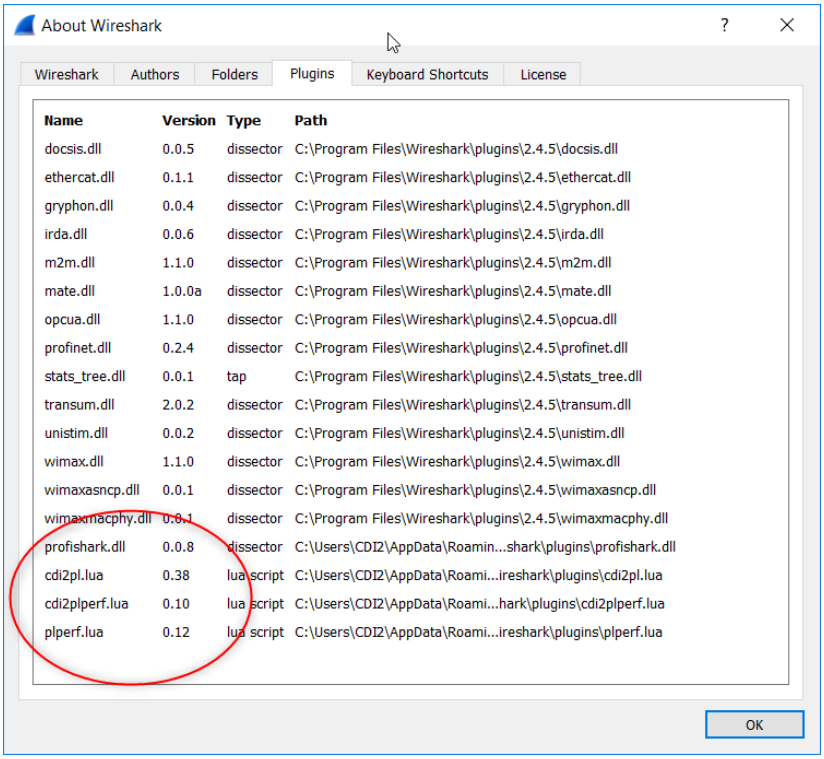


Figure 61: About Plugins

12.1.8. Wireshark Installation/Upgrade Hints

Wireshark installation or upgrading it to a newer version than the installed 2.4.5 requires some precautions.

- The CDI2 LUA plugins *cdi2pl.lua*, *cdi2perf.lua* and *perf.lua* can be used on newer Wireshark versions as well, e.g. for Wireshark 3.2.1.
- The ProfiShark binary plugin (*profishark.dll*) must match to desired Wireshark version and computer platform.
- The plugin folder structure of Wireshark has been changed slightly since version 2.4.5.

  LUA plugins folder *%APPDATA%\Wireshark\plugins*:

  cdi2pl.lua, cdi2perf.lua and *perf.lua*

  Binary plugins folder *%APPDATA%\Wireshark\plugins\3.2\epan*

  profishark.dll

- File locations

  CDI2 LUA plugins

  C:\CDI2-Installation\Windows\03-CDI2-Wireshark-Plugins

  ProfiShark vendor binary plugins for windows platforms

  C:\ CDI2-Installation\Windows\03-CDI2-Wireshark-Plugins\Dissector Wireshark\dist
  e.g. .\wireshark_3.2\amd64\profishark.dll for Wireshark 3.2 on 64-bit platform.

  ProfiShark vendor driver package for all platforms

  C:\CDI2-Installation\Windows\03-CDI2-Wireshark-Plugins\PROFISHARK_USBKEY_89.zip

Important note:

Following Wireshark protocol settings are required for proper handling of the ProfiShark ns-timestamps. Otherwise CDI2 plugins will show imprecise values.

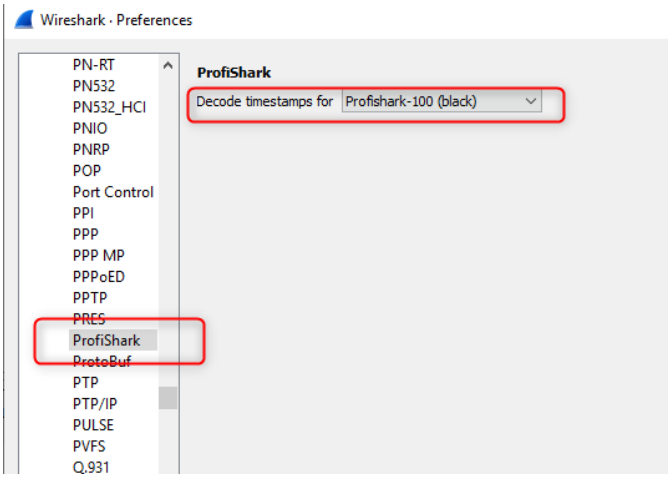


Figure 62: ProfiShark Protocol Settings

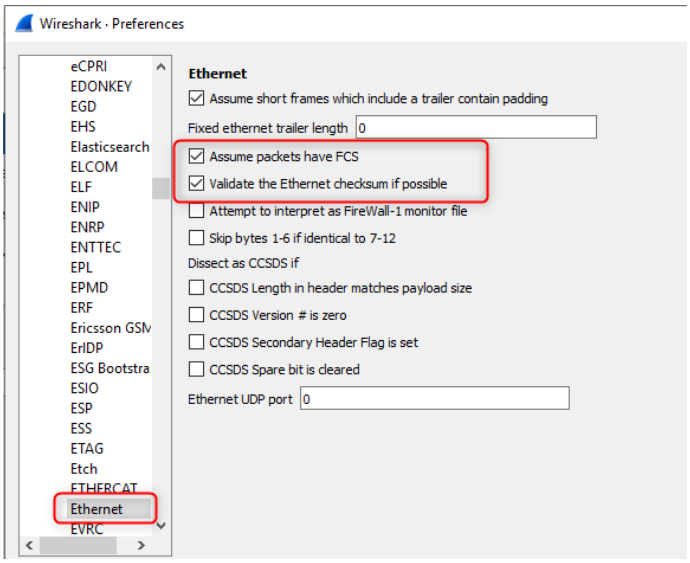


Figure 63: Ethernet Protocol Settings

Installation of coloring rules

View->Coloring Rules->Import...

C:\CDI2-Installation\Windows\03-CDI2-Wireshark-Plugins\CDI2_alt.col_rules

Uncheck Broadcast and System Event (Figure 64).

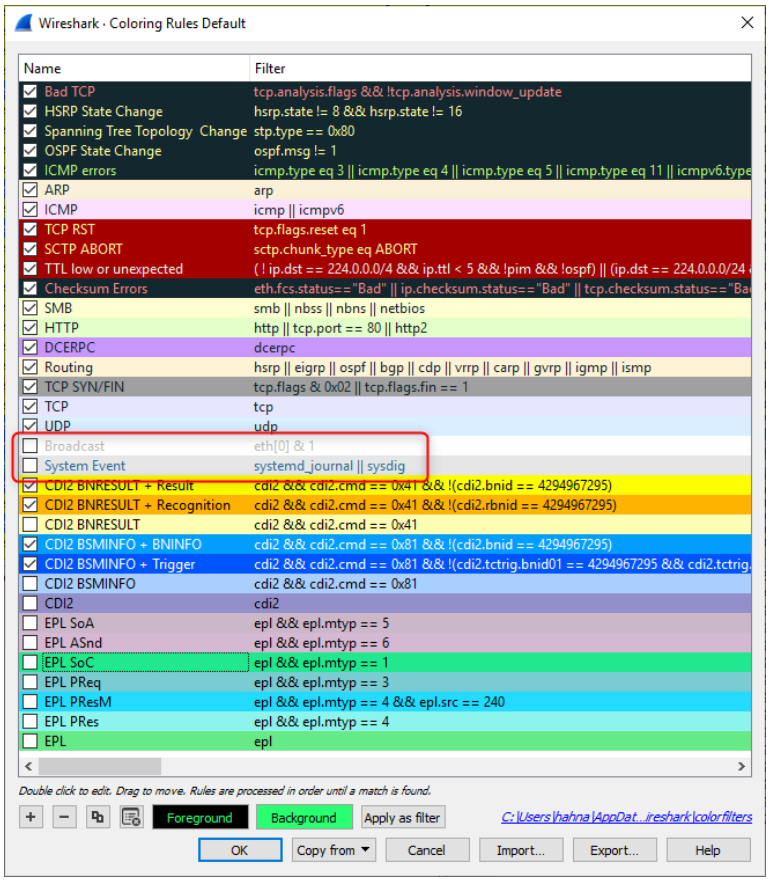


Figure 64: Coloring Rules Settings

12.2. WinSCP

WinSCP is a popular sFTP and FTP client for Microsoft Windows. It copies files between a local computer and remote servers using FTP, FTPS, SCP, SFTP, WebDAV or S3 file transfer protocols.

Refer to https://winscp.net for detailed information.

WinSCP is used to transfer files over the simulator network connections. This can be used to test the FTP interface of a CDI2 device or to connect to the Simulation Computer (VisionBox) to access test cases and results.

12.2.1. Connect

A set of connection options is preconfigured on the GUI PCs.

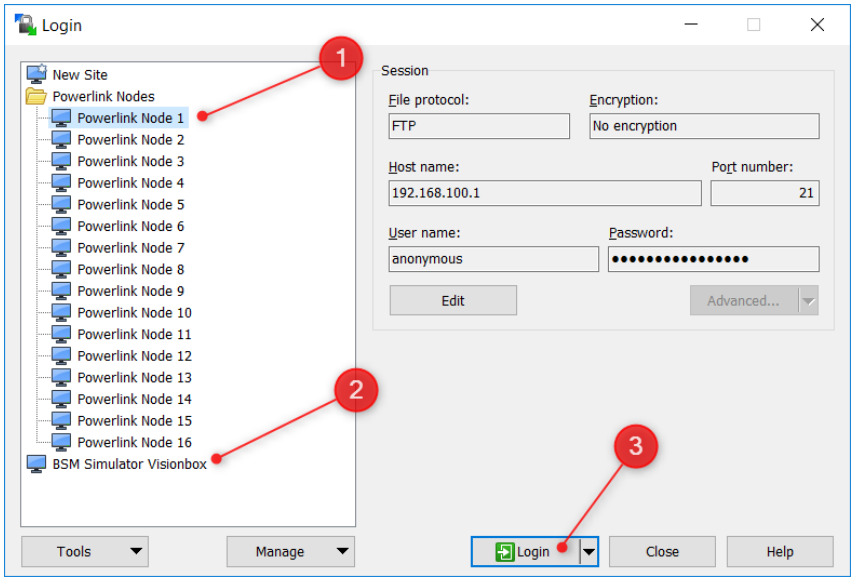


Figure 65: WinSCP connect

(1) Connection pre-set for connecting to the Powerlink node with id 1.
(2) Connection pre-set for a connection to the Simulation Computer.
(3) To open the selected connection, click Login.

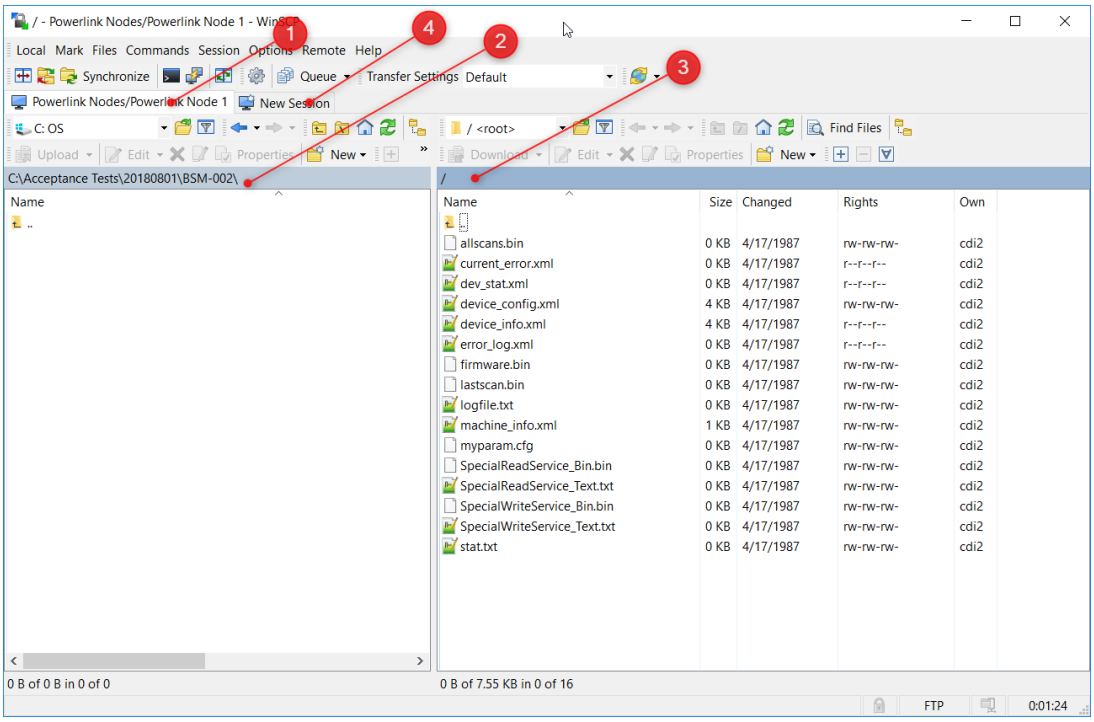


Figure 66: WinSCP main

After the connection is established, WinSCP shows the connection as a tab (1). The local file system is shown on the left pane (2) and the remote filesystem in the right pane (3). It is possible to open additional connections by clicking “New Session” (4).

12.2.2. Edit and Download

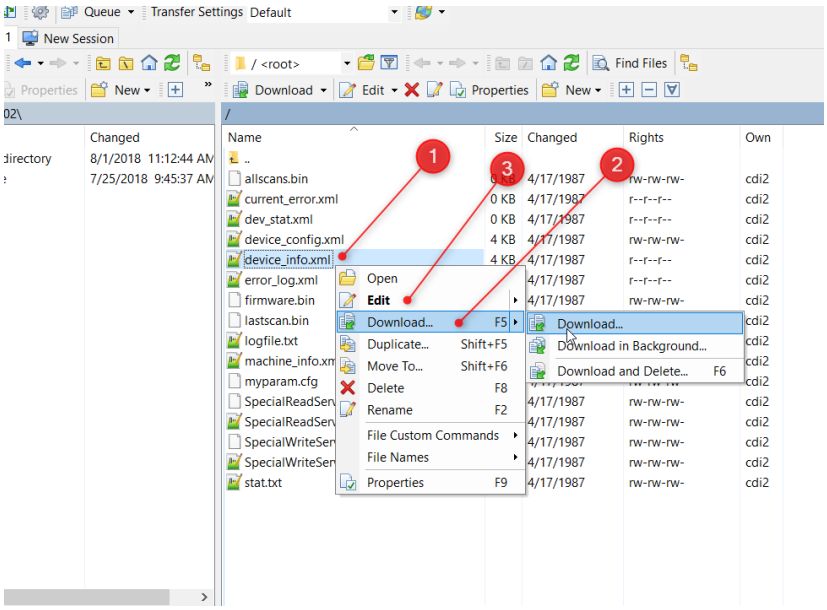


Figure 67: WinSCP Edit and Download

To save a file to the GUI PC, right click the file in the right pane (1), then select “Download…” (2) and “Download…”. It is also possible to edit the file (3), when the file is saved in the editor it is automatically uploaded.

12.2.3. Upload

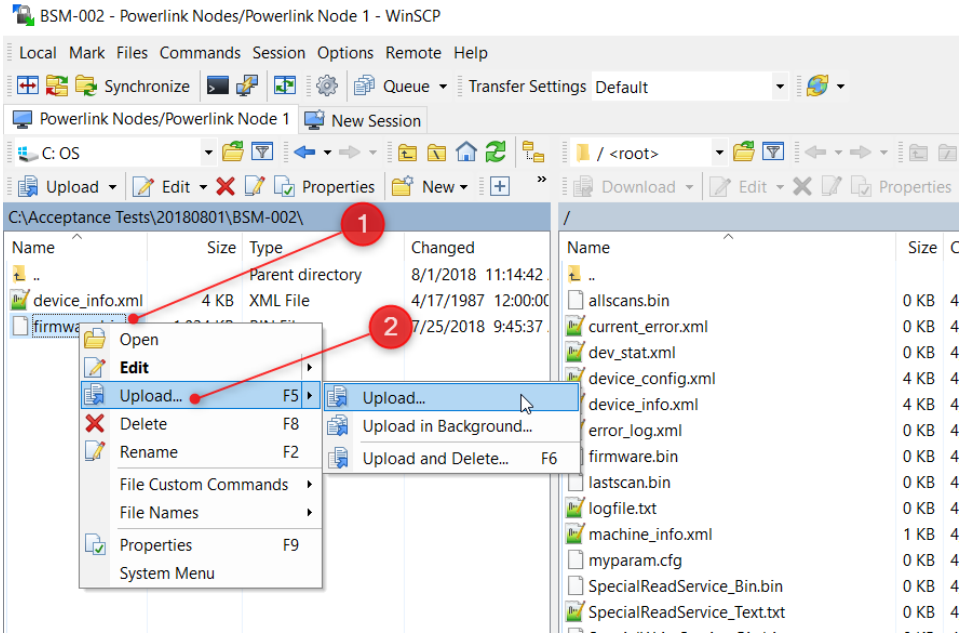


Figure 68: WinSCP Upload

To upload a file (e.g. firmware.bin), first navigate to the containing folder on the left pane. Right click the file which should be uploaded (1), select “Upload…” (2), and then “Upload…”.

12.2.4. New Session

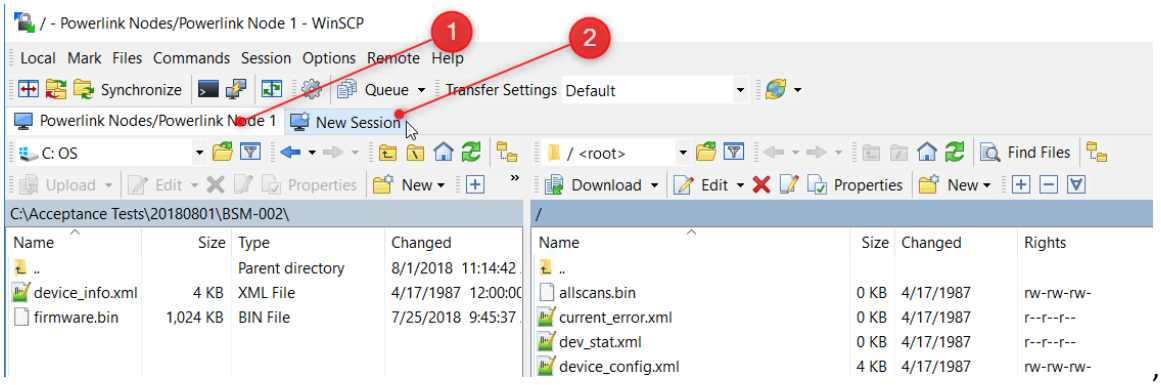


Figure 69: WinSCP New Session

Connections are managed in tabs (1) to create a new connection click “New Session” (2).

12.3. USB Devices

The device manager of Windows 10 provides useful information about the installed USB devices. It can be used to check if the USB devices of the simulator have been detected properly and to identify the proper COM port number of serial devices.

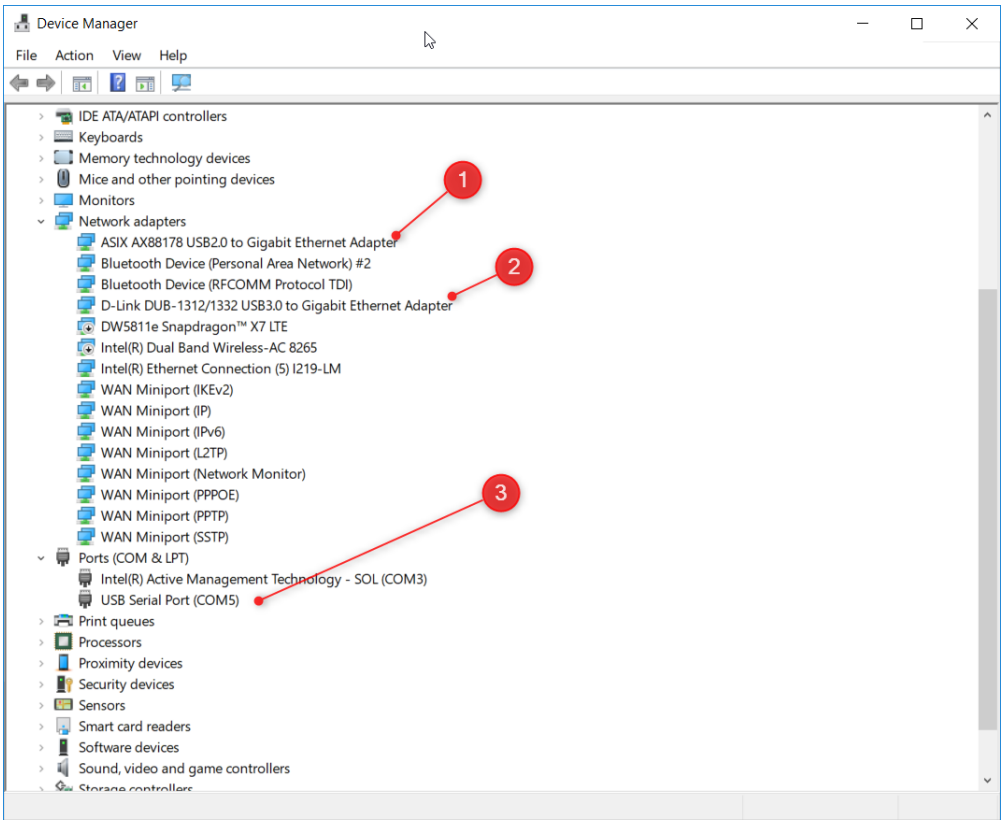


Figure 70: List of detected USB devices (BSMS)

(1) ProfiShark Network Analyser (shown as ASIX AX88178)
This USB device shall be present on the BSMS and on the DS.

(2) USB Network Adapter for Powerlink Gateway (shown as D-Link DUB-1312/1332)
This USB device shall be present on the BSMS only.

(3) USB-to-Serial Converter (shown as USB Serial Port) used for remote control of the transport simulator. The COM number displayed here must be selected in the Transport Simulator Tool [12.6], e.g. COM5.
This USB device shall be present on the BSMS only.

If a USB device is not detected properly the device manager will not list the desired USB device. Figure 71 shows an example of missing USB devices. In case of a missing USB device, try to plug-out the USB cable, wait some seconds, plug-in the USB cable again and watch if the device is added.

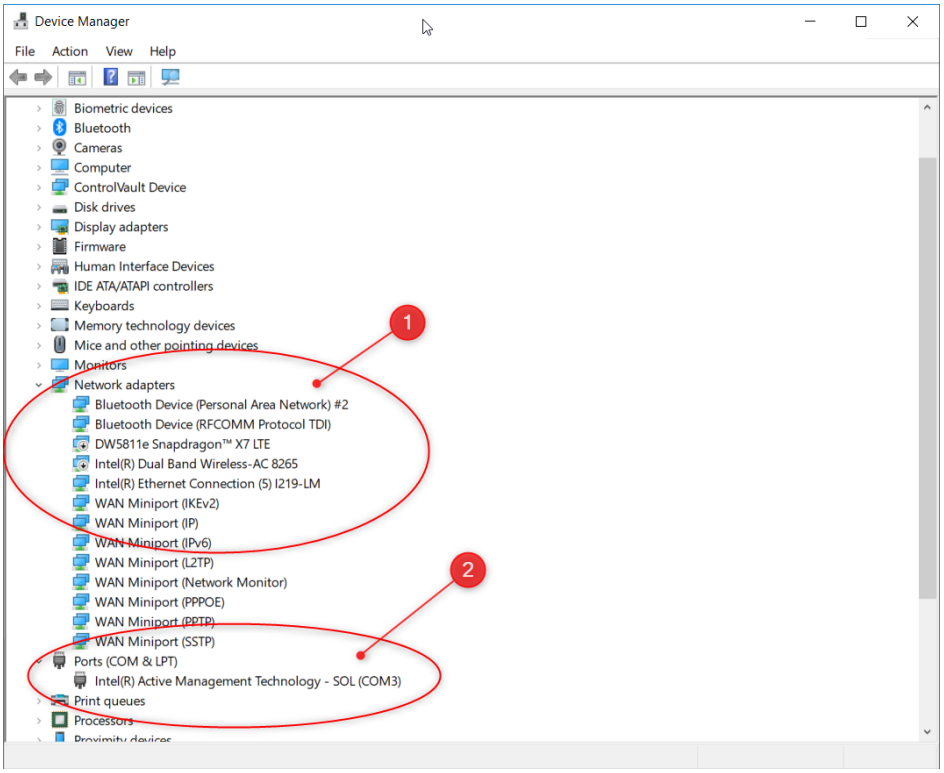


Figure 71: Missing USB components

(1) USB network adapters not detected (ASIX AX88178 nor D-Link DUB-1312/1332)
(2) USB-to-Serial Converter (shown as USB Serial Port) not detected

12.4. PuTTY

PuTTY is a terminal program to establish a console connection via SSH or a serial line. SSH is commonly used to open a console via the network, using the IP address of the node. The connection via the serial line is used when no network connection is available; this is mainly for debugging and installation purposes.

PuTTY is not needed for conducting testcases, but it is an important tool for debugging and when a software update of simulator components is needed (see 7.2.4).

PuTTY Download and Documentation

https://www.putty.org/

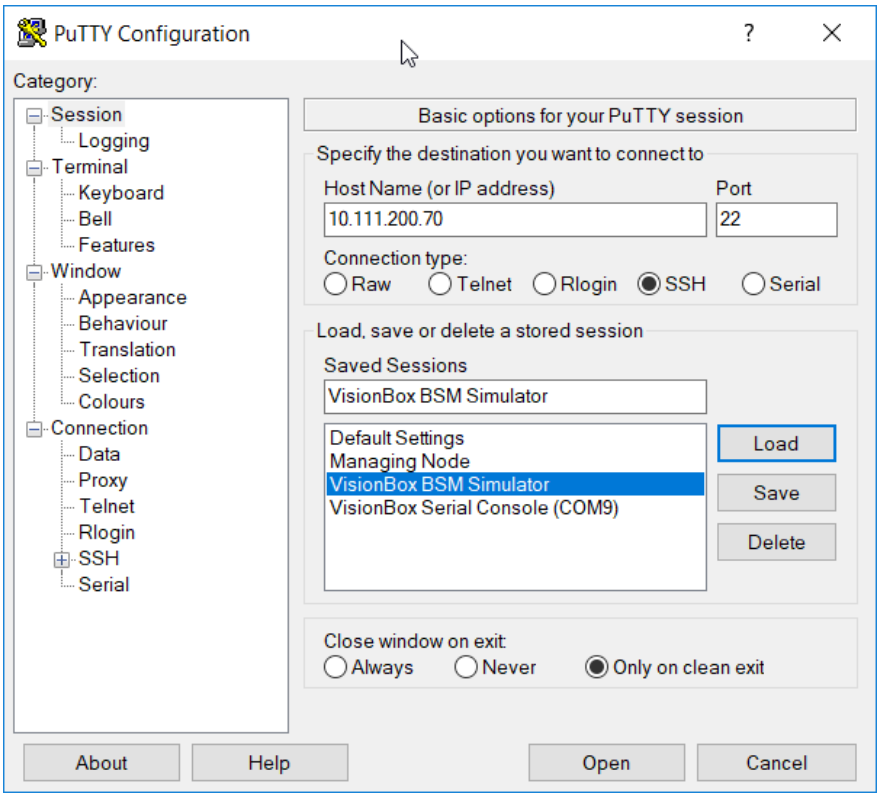


Figure 72: PuTTY connections and settings

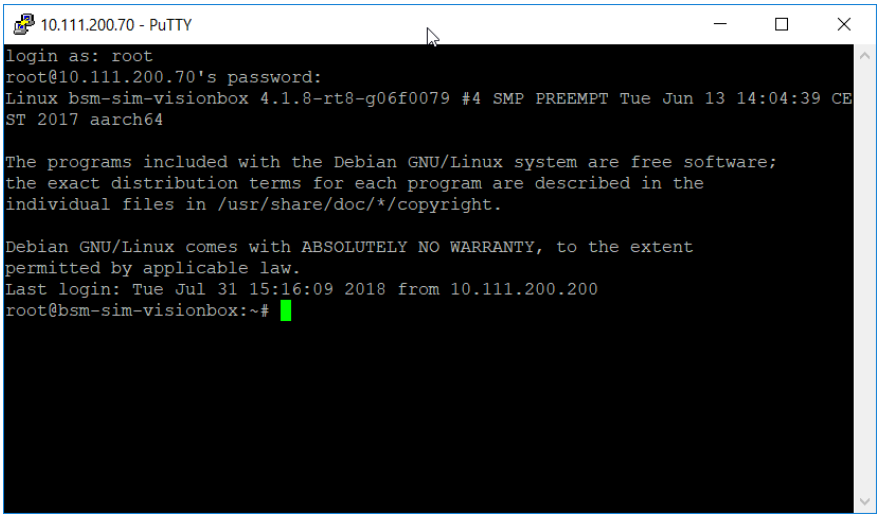


Figure 73: PuTTY command line

12.5. PicoScope

Pico Technology, PICOSCOPE 2204A
https://www.picotech.com

12.5.1. Hardware Setup

The PicoScope unit is connected to the GUI-PC via an USB connection. Connect the measuring probes as depicted in [Figure 74].

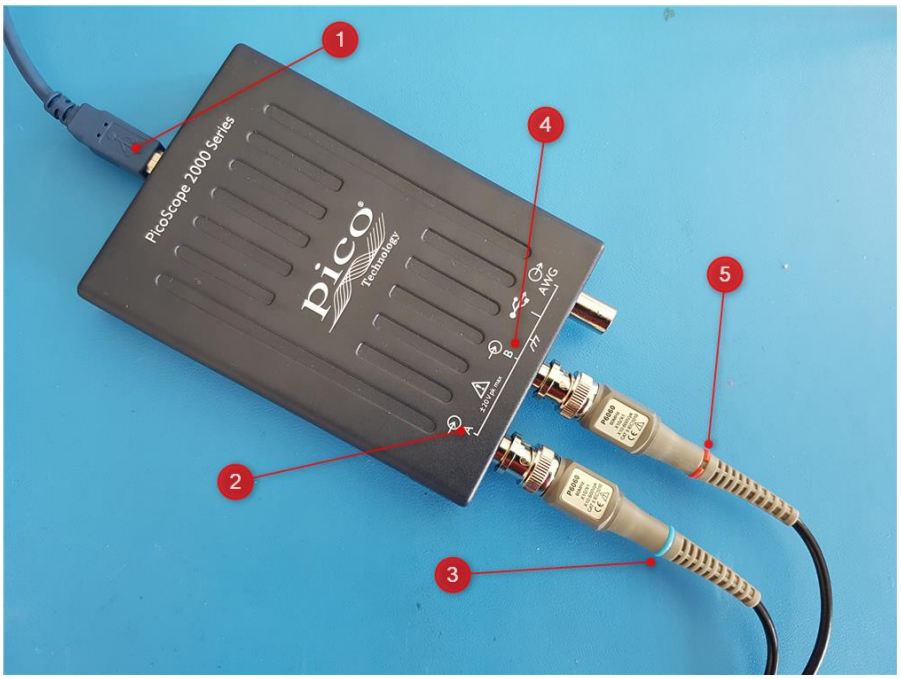


Figure 74: Picoscope connections

(1) USB cable
(2) Probe A input
(3) use blue colour marker for Probe A
(4) Probe B input
(5) use red colour marker for Probe B

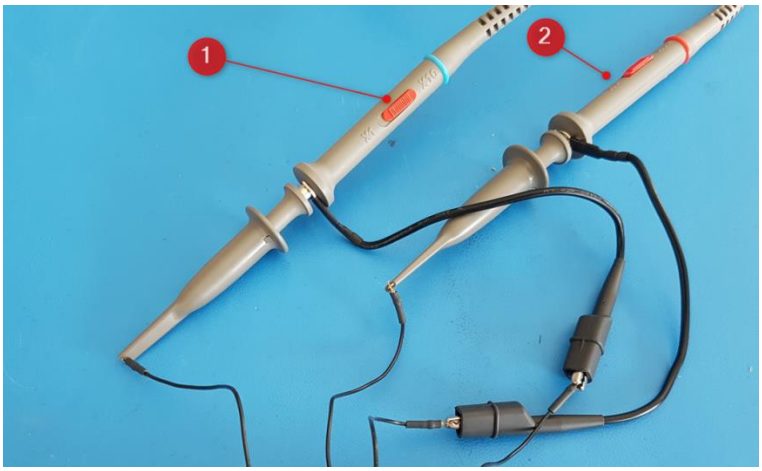


Figure 75: Picoscope probes

(1) and (2): The attenuation switches on both probes shall be set to x10 (10:1)

Figure 76 shows an example of how to connect to the signals of a breakout box.

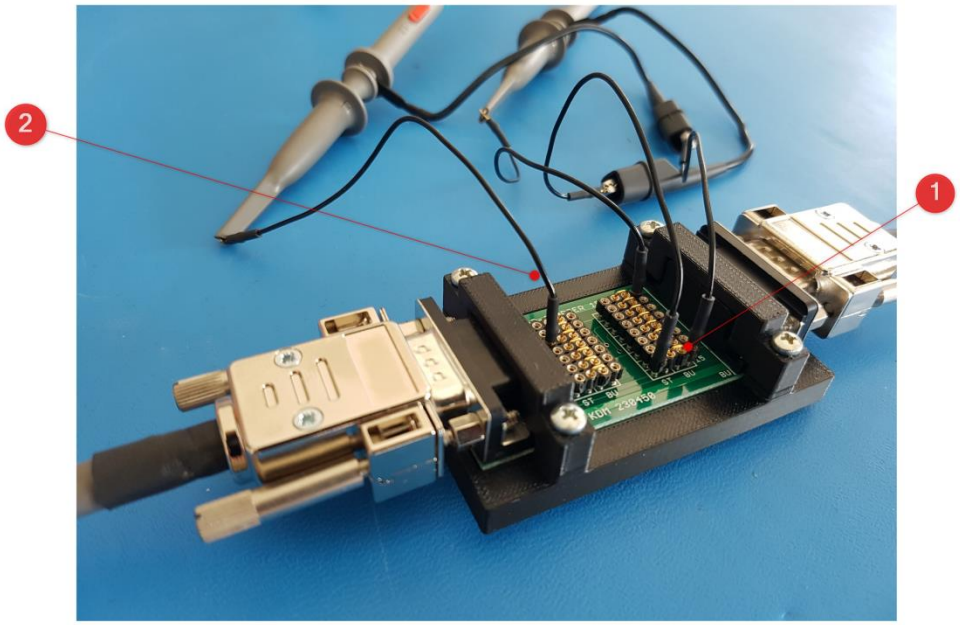


Figure 76: Picoscope connection example

(1) Ground connections
(2) Signal tap

12.5.2. PicoScope Tool

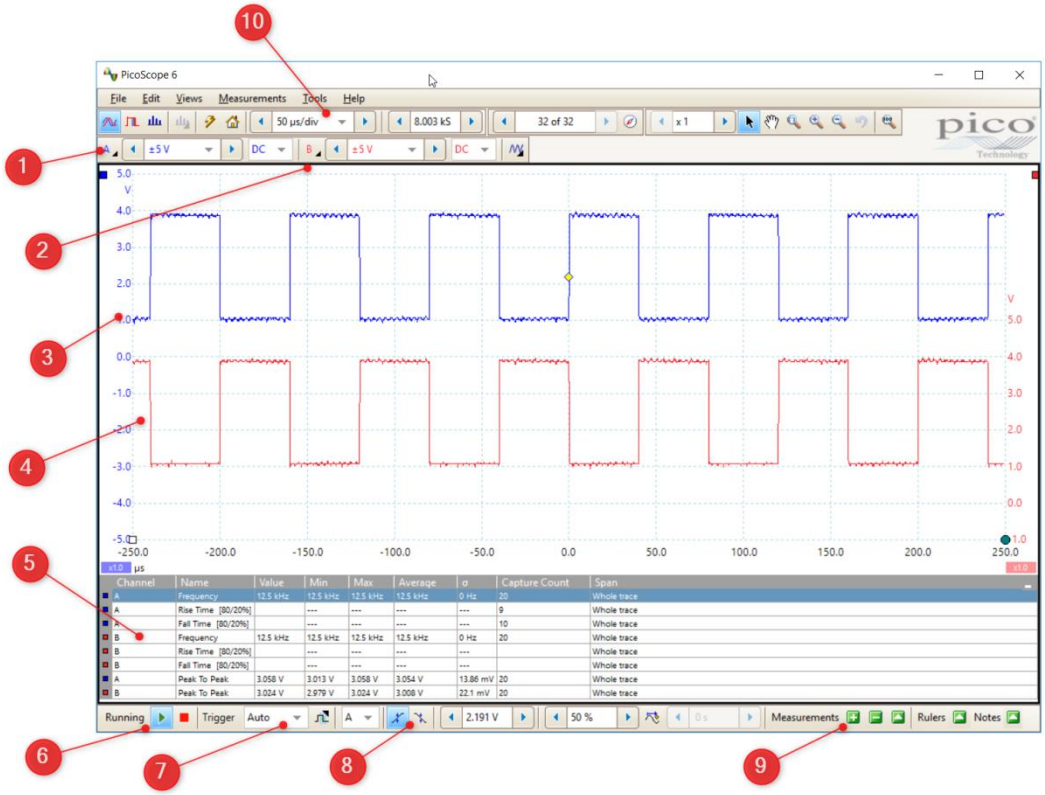


Figure 77: Picoscope overview

(1) Channel A (blue) Settings
Y Scale = +/- 5V, DC Coupling, Probe Setting = 1:10

(2) Channel B (red) Settings
Y Scale = +/- 5V, DC Coupling, Probe Setting = 1:10

(3) Trace A (blue)

(4) Trace B (red)

(5) Measurements Windows

(6) Trigger Status and Control
Running/Stopped

(7) Trigger Mode
Auto ... for periodic signals, e.g. a clock signal
Single ... for single events, e.g. a reset signal

(8) Trigger Edge
select rising or falling edge for trigger

(9) Add/Remove Measurements
add new measurements to measurements window

(10) Time scale

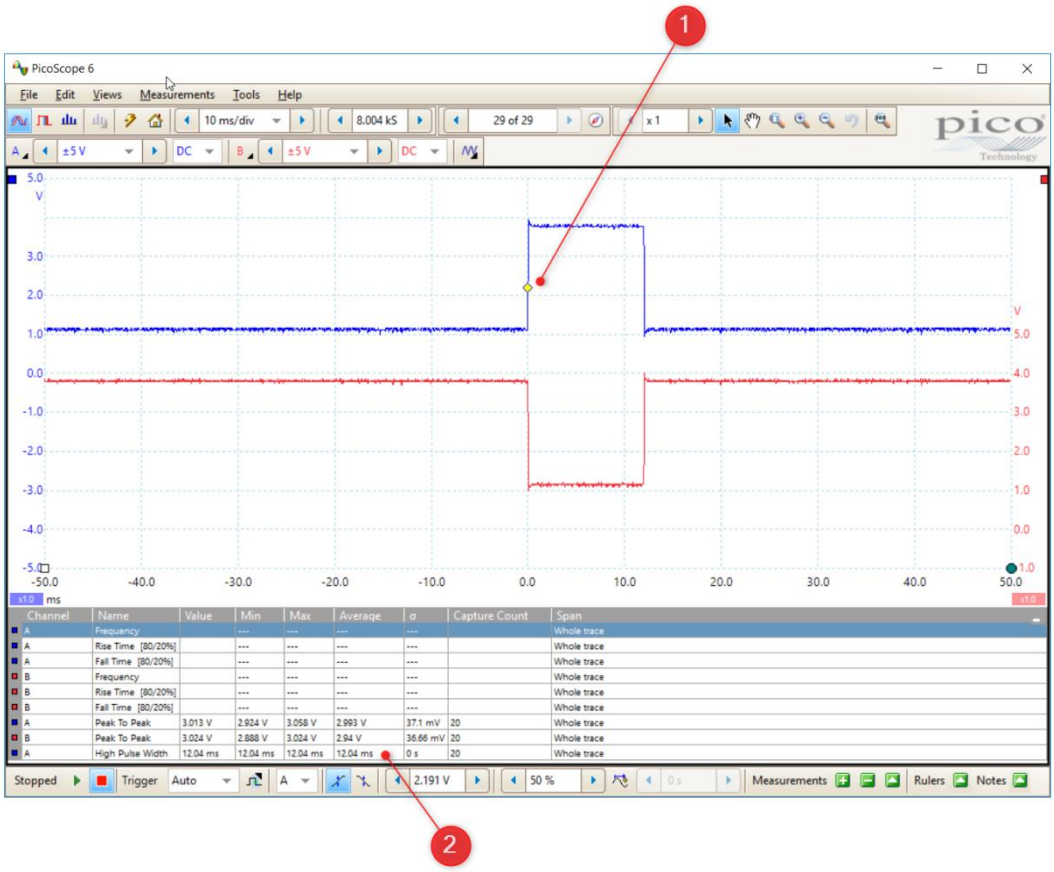


Figure 78: Picoscope pulse width measurement example

(1) Trigger Marker
Indicates the trigger position and threshold, the marker can be moved to change the trigger threshold

(2) Pulse Width Readout
The actual pulse width is measured, display shows actual, min, max and average values.

12.6. Transport Simulator Tool

The transport simulator, also known as "Detector Test Rig" [Ref 3.], can be remote controlled by a software tool on the GUI-PC. The program *Adaptor2_SerialSimulator.exe* is available in folder *C:\CDI2\08-DNB-Tools.*

The tool uses a serial connection to the transport simulator to send transport on/off and speed control commands as specified in chapter 5.2.1 of [Ref 3.]

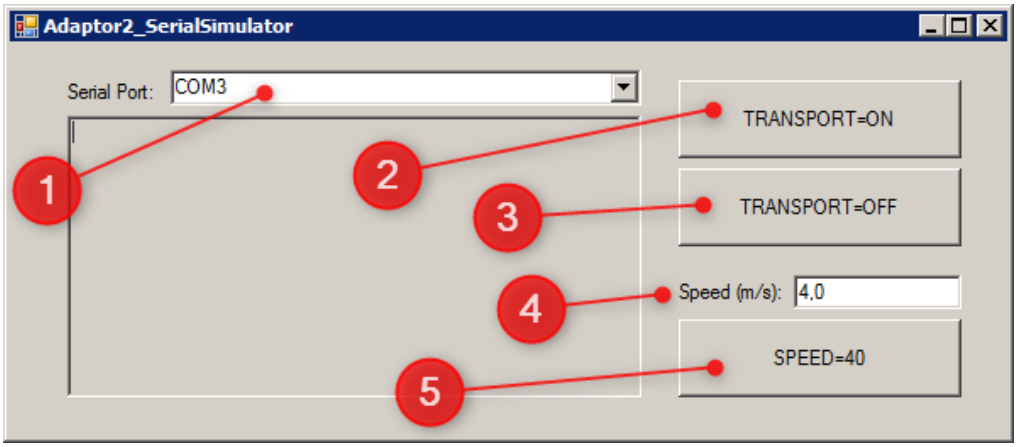


Figure 79: TS control

(1) Serial port selection
refer to [12.3] USB-to-Serial Converter also
(2) Start transport
(3) Stop transport
(4) Select speed in m/s units
The speed can be set with a resolution of 0.1 m/s
(5) Send speed select command to TS.
The selected speed is applied at the next Start transport command
The SPEED button shows the raw value which is sent on the serial line, e.g. 40 for 4.0m/s

12.7. XML validation

A set of XML schema files is provided to support offline validation of XML files. The validation can be done by any commonly available tool which supports XML Schema 1.1, e.g. ALTOVA XMLSpy. Generally, the validation procedure requires two files, the XML file to be validated (.xml) and an XML schema file (.xsd). On the GUI PC the validation of XML files can be done with pre-configured batch scripts that call the command-line tool *SimpleXsdValidator.jar*. One test script, *check-examples.bat*, is configured to run XML validation on a set of example XML files provided together with the schema files. Another batch script, *checkXML.bat*, is setup to run the validation on user supplied XML files.

12.7.1. Schema and Example Files

Schema files and examples can be found in folder

C:\CDI2\21-xml-verification

XML schema files

C:\CDI2\21-xml-verification\schema

DeviceInfo.xsd	... schema for device_info.xml and device_config.xml
MachineInfo.xsd	... schema for machine_info.xml
CdiErrorLog.xsd	... schema for error_log.xml
DeviceStatus.xsd	... schema for DeviceStatus dev_stat.xml
CdiStorage.xsd	... schema for XML storage file

Valid XML example files folder

C:\CDI2\21-xml-verification\examples

DeviceInfo	*... XML in this folder to be validated with DeviceInfo.xsd*
MachineInfo	*... XML in this folder to be validated with MachineInfo.xsd*
CdiErrorLog	*... XML in this folder to be validated with CdiErrorLog.xsd*
DeviceStatus	*... XML in this folder to be validated with DeviceStatus.xsd*
CdiStorage	*... XML in this folder to be validated with CdiStorage.xsd*

User XML files

C:\CDI2\21-xml-verification\xml_to_test

DeviceInfo	*... XML in this folder to be validated with DeviceInfo.xsd*
MachineInfo	*... XML in this folder to be validated with MachineInfo.xsd*
CdiErrorLog	*... XML in this folder to be validated with CdiErrorLog.xsd*
DeviceStatus	*... XML in this folder to be validated with DeviceStatus.xsd*
CdiStorage	*... XML in this folder to be validated with CdiStorage.xsd*

Validation tool and script

C:\CDI2\21-xml-verification

check-examples.bat	*... batch script, runs validation of all XML in the examples folder*
checkXML.bat	*... batch script, runs validation of all XML in the xml_to_test folder*
SimpleXsdValidator.jar	*... XML validation tool, takes .xml and .xsd and runs validation*

12.7.2. Use of XML Validation Scripts

The validation scripts *check_example.bat* and *checkXML.bat* are started with a double-click to the .bat file. On activation a new window with the validation output is shown. Figure 80 shows the XML validation

folder with the .bat scripts and Figure 81 shows a successful validation of the example files with *check_example.bat*.

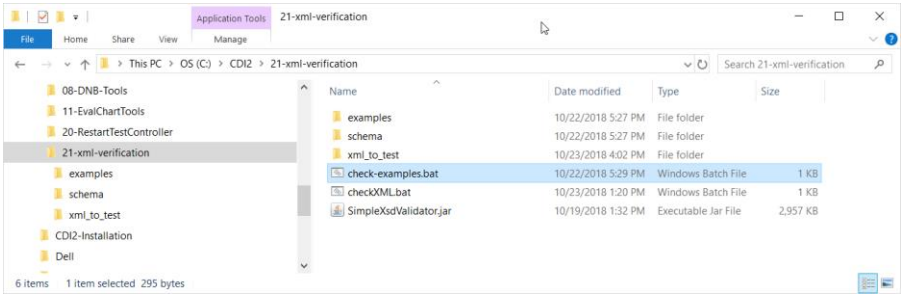


Figure 80: XML validation folder

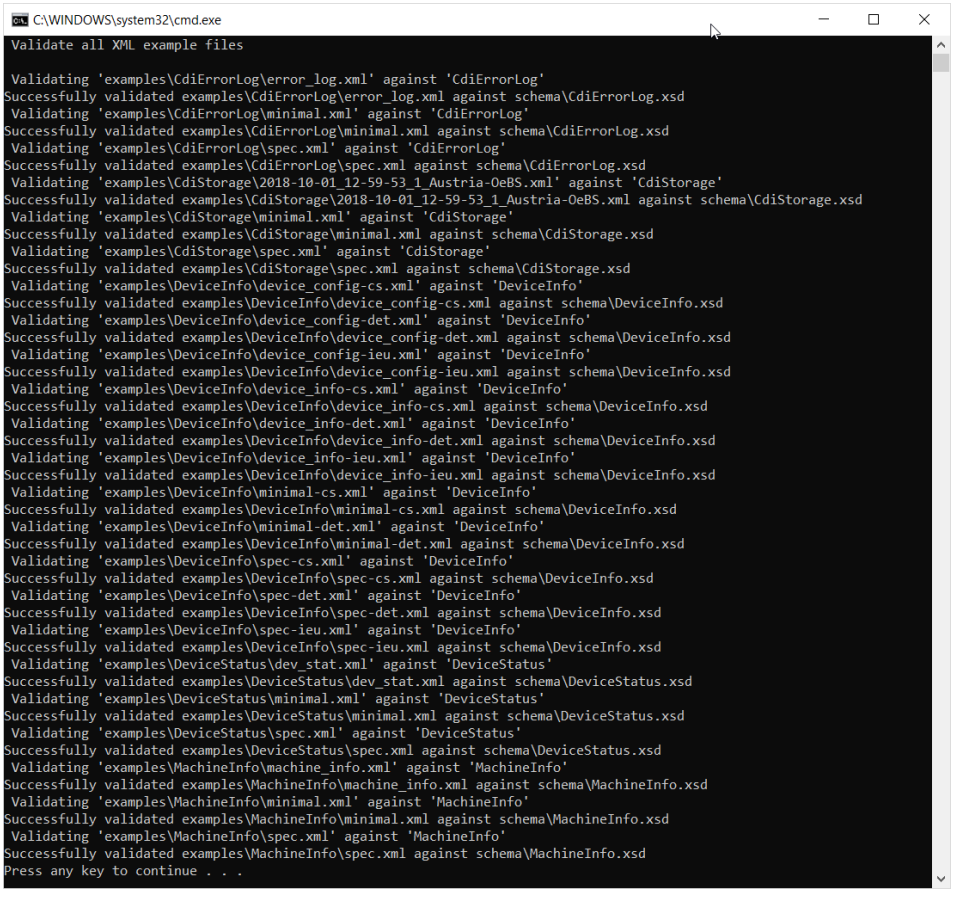


Figure 81: Output of a successful validation of the examples folder

The following example shows how to validate a user supplied *device_info.xml* file. First, the XML file is placed in the appropriate folder. In this case the XML must conform to the *DeviceInfo.xsd* schema and must be placed into *xml_to_test**DeviceInfo* therefore (see Figure 82). The use *checkXML.bat* to validate all files in *xml_to_test*. Figure 83 shows the output of a successful validation and Figure 84 the output of a failing validation. In the shown case the attribute 'NodeI**D**' was invalid because the correct spelling is 'NodeI**d**'.

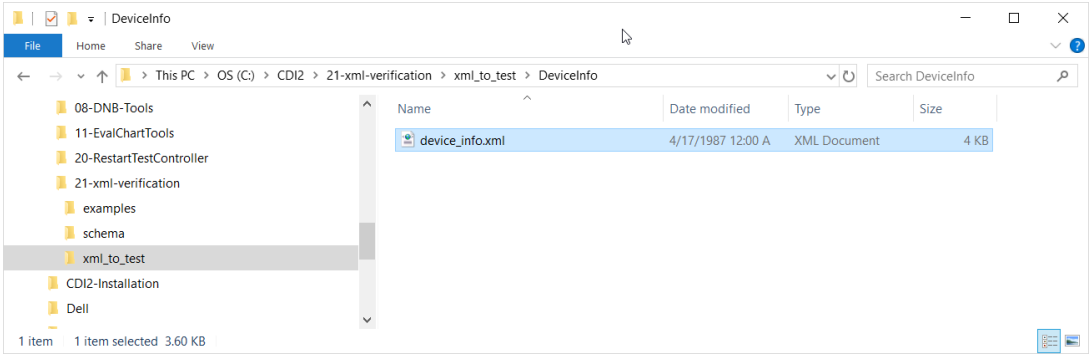


Figure 82: device_info.xml to validate

Figure 83: Output of successful validation

Figure 84: Output of failing validation

12.8. Chart Evaluation and Image Quality Tools

The GUI PC has a set of tools installed to verify the image quality of a camera system. The tools can be found in the folder

C:\CDI2\11-EvalChartTools

along with their documentation and manuals.

EvalChart tools

Programs\Testchart_A\EvalChart_A.exe
Programs\Testchart_B\EvalChart_B.exe
Programs\Testchart_C\EvalChart_C.exe
Programs\Testchart_P\EvalChart_P.exe

EvalChart documentation

Descriptions\EvalChart_A Description.pdf
Descriptions\EvalChart_B Description.pdf
Descriptions\EvalChart_C Description.pdf
Descriptions\EvalChart_P Description.pdf

For convenient summarisation und judgement of the results generated by the EvalChart tools, an OpenDocument spreadsheet and support scripts are provided.

C:\CDI2\05-LibreOffice-Tools\Image Quality
ImageQuality_V1_2.ods
EvalChartA_merge&run.bat
EvalChartB_merge&run.bat
EvalChartC_merge&run.bat

The tools are used for acceptance test "Image quality tools (CS-003)". Follow the instructions described for that test.

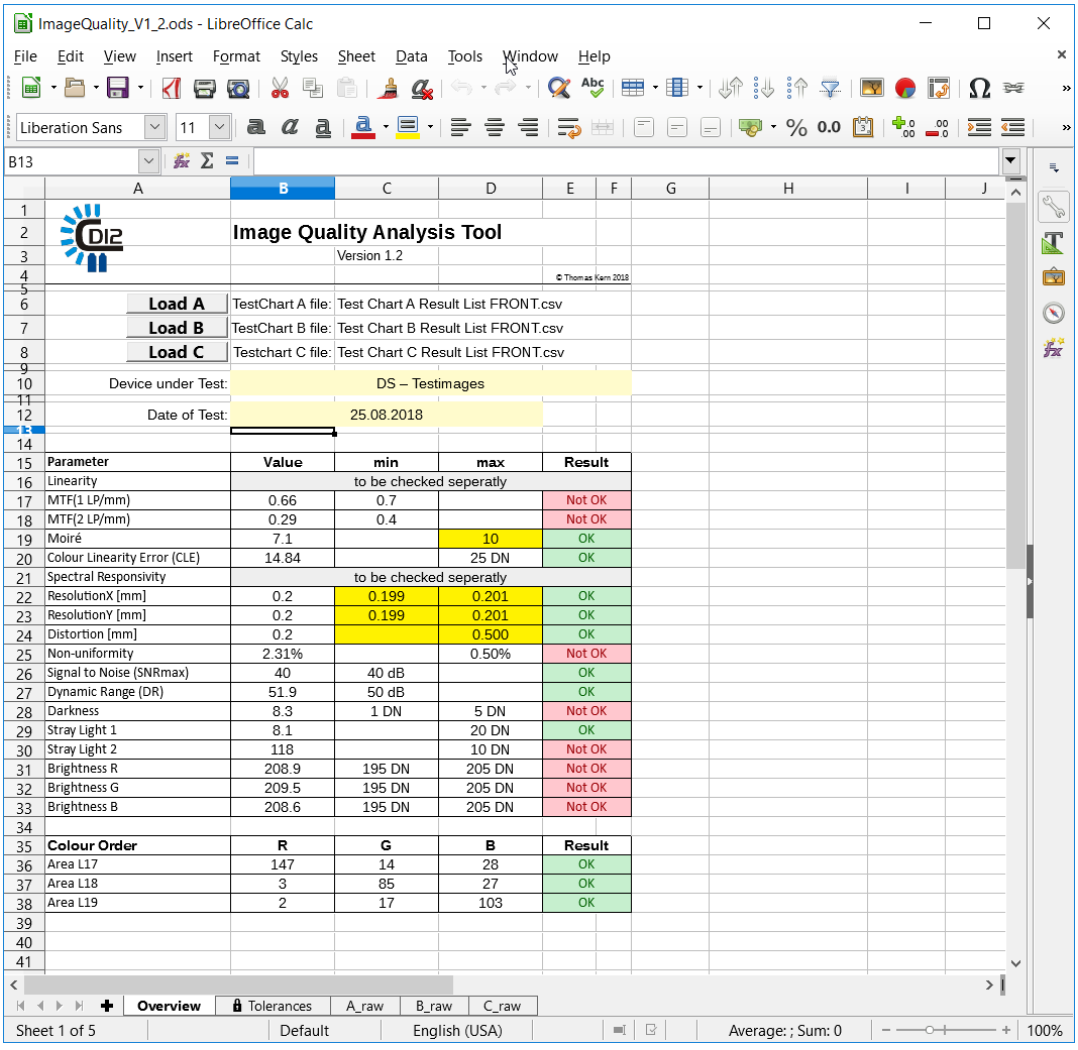


Figure 85: Image Quality Analysis Tool

12.9. TTS Breakout Box

To access the electrical signals of the TTS, a breakout box is supplied together with the PicoScope. It can be plugged between a BSM and a device's TTS connection using the two D-Sub connectors. The patch panel then offers a way to measure or disconnect any single connection of the TTS. To measure a signal with the PicoScope, it is recommended to connect one of the supplied patch cables to the probe on one side and to the patch panel on the other. The TTS breakout box is required for acceptance tests "BSMS - Electrical specification of TTS signals (BSM-032)" and " DS - correct reaction to TTS Reset length (DEV-014)".

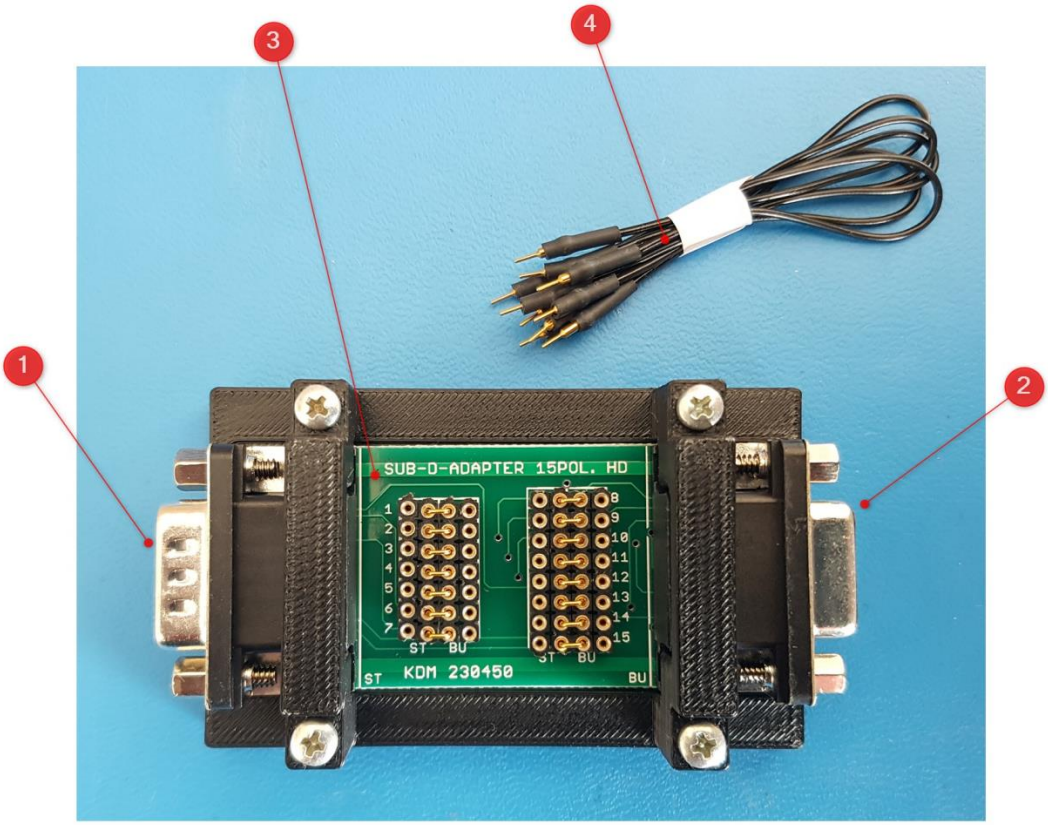


Figure 86: TTS Breakout Box

(1) D-Sub HD DE-15 male
(2) D-Sub HD DE-15 female
(3) Patch panel
(4) Patch cables

12.10. TS Trigger Breakout Box

The TS Trigger Box can be used to tap the clock and trigger signals sent by the TS to the BSMS. It is initially used to calibrate the TC rate (see 7.8 and acceptance test "BSMS - TS transport speed and TC accuracy (BSMS-005)") and to troubleshoot connection problems between the TS and the BS.

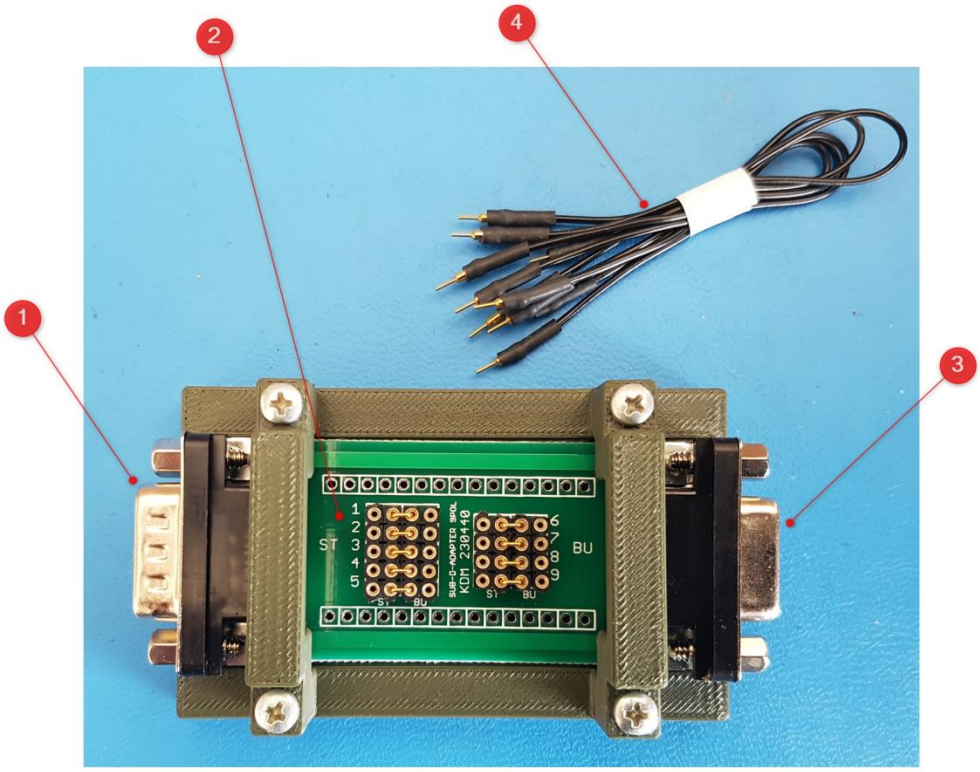


Figure 87: TS Trigger Breakout Box

(1) D-sub DE-9 male
(2) Patch panel
(3) D-sub DE-9 female
(4) Patch cables

12.11. Flutter Detector and Analysis

A set of tools is provided to gather and summarise the measured data samples of the flutter detector [Ref 4.].

The flutter detector tool is used to gather and display the measurement data during actual testing. Finally, after capturing has been completed the raw data is saved into a file for later import into the Analyses tool for more in-depth analyses.

12.11.1. Flutter Detector Capture Tool (Banknote Displacement Detector Tool)

Tool location:

C:\CDI2\05-DNB-Tools\BDD_SerialSimulator.exe

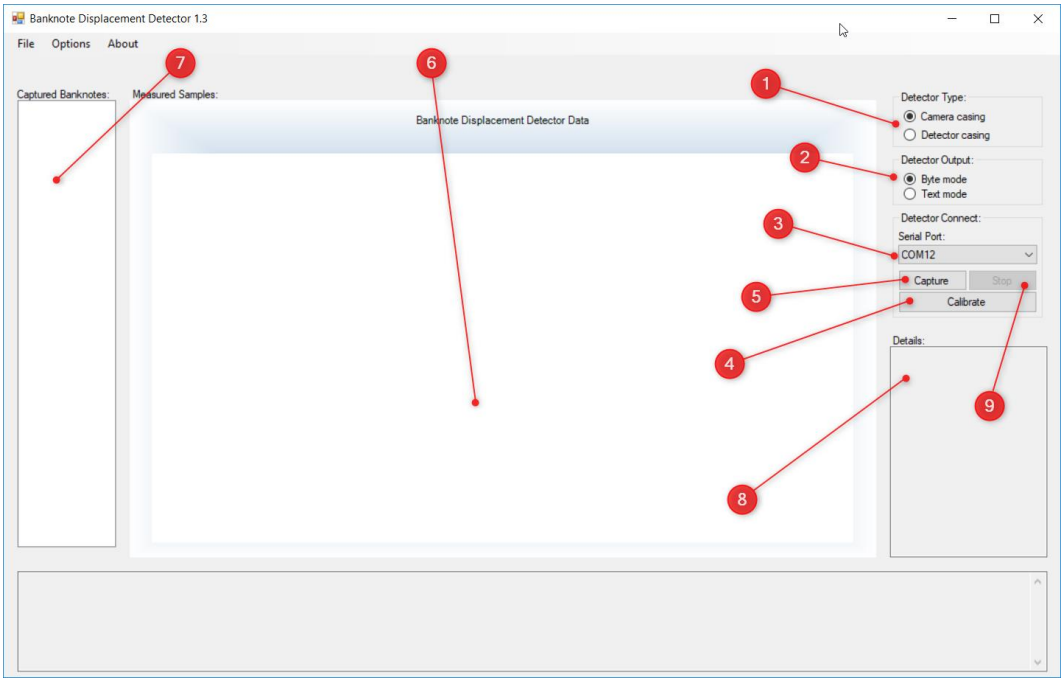


Figure 88: Flutter Tool - Start Screen

(1) Detector Type: select type of Flutter Detector, either Camera System casing or Detector casing. Must set correctly at start-up before performing any other operation.
(2) Detector Output: select Byte mode always
(3) COM port of Flutter Detector, select COM ports as detected by Windows. If not known, use the Device Manager to identify the port number [see 12.3]
(4) Start calibration: must be done before a Capture command is issued. Make sure that the banknote transport is switched off and no banknote is in the laser path during calibration. Calibration needs about 1 sec.
(5) Capture Data: enable capture mode, measurements are taken when Flutter Detector is triggered
(6) Banknote displacement view of selected banknote
(7) List of captured banknotes
(8) Details of selected banknote
(9) Stop: disable capture mode

Depending on the kind of flutter detector, either in CS or in Detector casing, different tool settings are required. The following order is recommended during startup.

1. Select type of Flutter Detector, see Figure 88 (1)
2. Select COM port, Figure 88 (3)
3. Start calibration, Figure 88 (4)
4. Capture Data, Figure 88 (5)

Figure 89 shows the main screen in Camera casing mode and Figure 90 how it looks like in Detector casing mode.

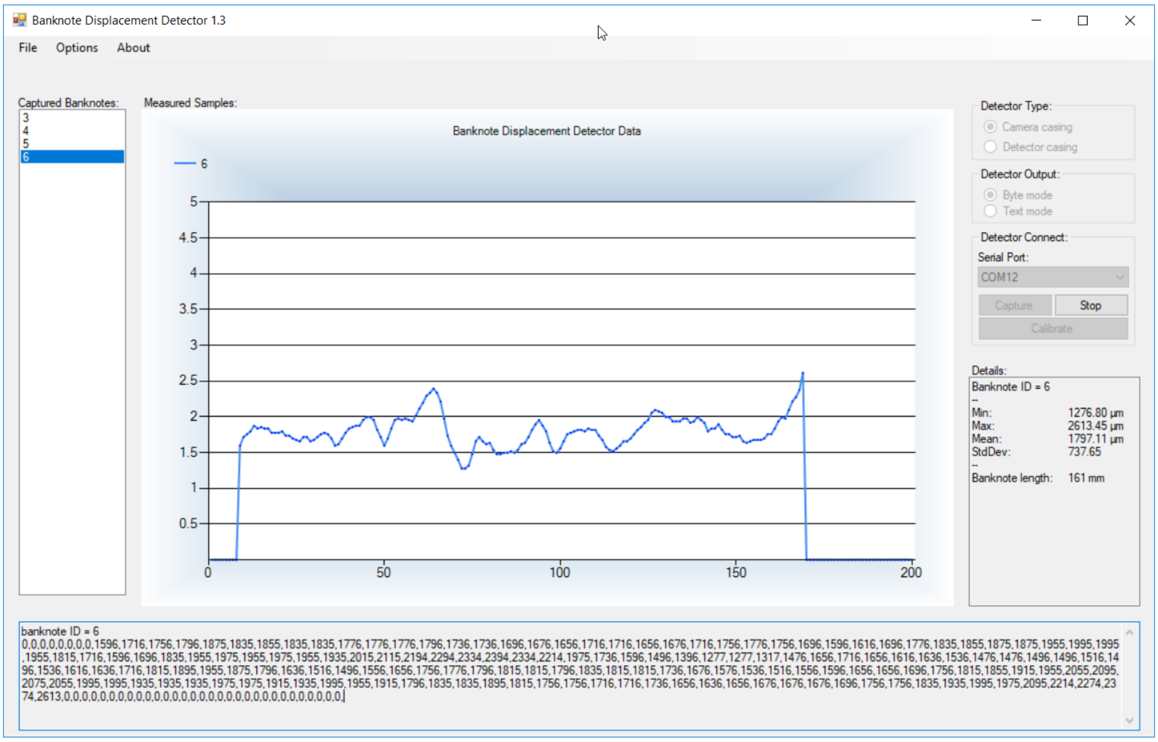


Figure 89: Capture Mode - Camera System casing

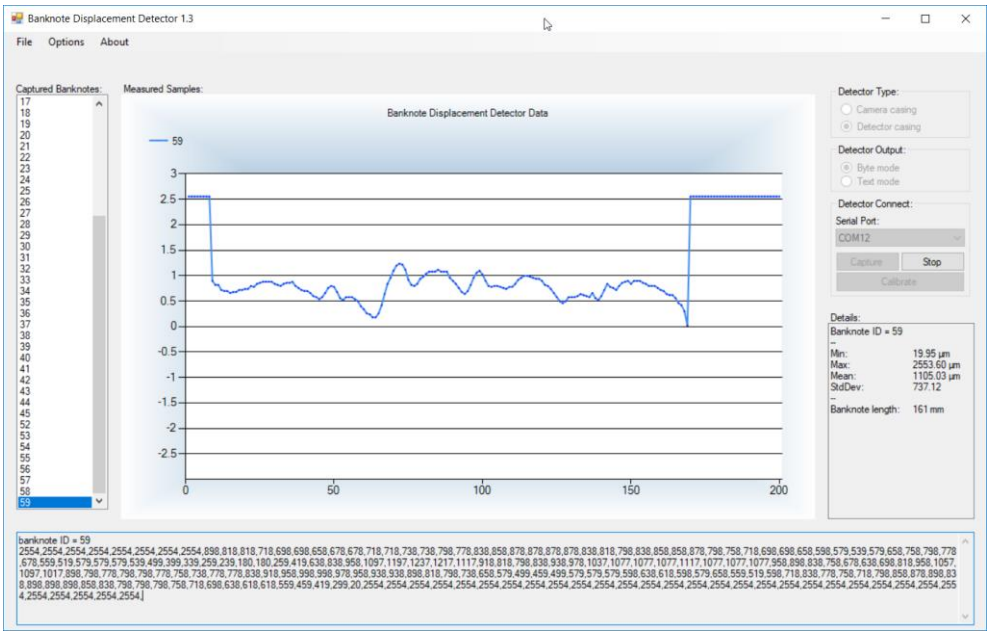


Figure 90: Capture Mode - Detector casing

The Banknote displacement view can be altered in the *Options* menu using *Graph Single Display* (Figure 91). If checked, it displays a single banknote, otherwise an overlay of banknotes is shown (see Figure 92).

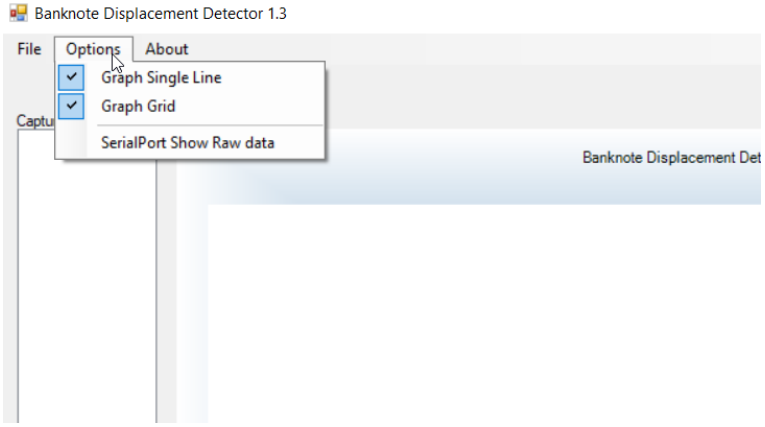


Figure 91: Options

Figure 92: Banknote overlay display

When capturing is completed, measurement data can be saved to a file for later analyses in an external tool by menu option *File->Save All...* (Figure 93).

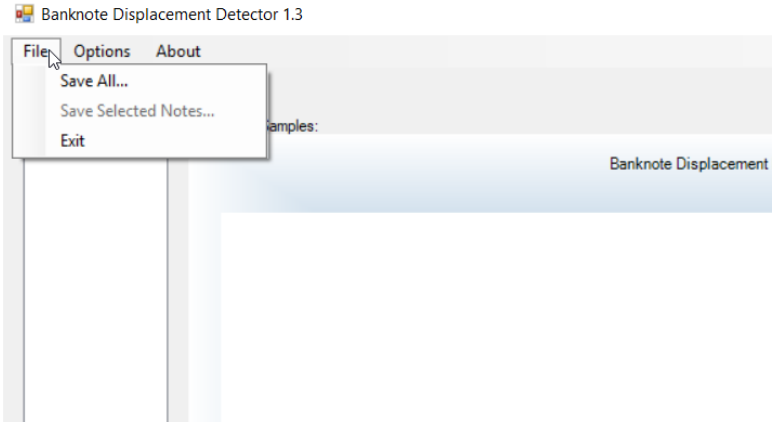


Figure 93: File Options

Version information about the tool is displayed with menu option About (Figure 94)

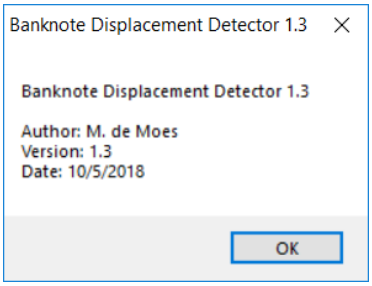


Figure 94: About Version Info

12.11.2. Flutter Analyses

For the evaluation of the measurements recorded by the flutter detector an OpenDocument spreadsheet is available. It can import the saved data and provides further analyses and chart views of the data.

C:\CDI2\05-LibreOffice-Tools\Flutter_analysis V1_1.ods

The tool is opened with *LibreOffice Calc* by a double-click to the .ods file. Use the "Load Data" button to import a previously captured data set from the flutter detector tool (chapter 12.11.1). Figure 95 and Figure 96 show the main display areas.

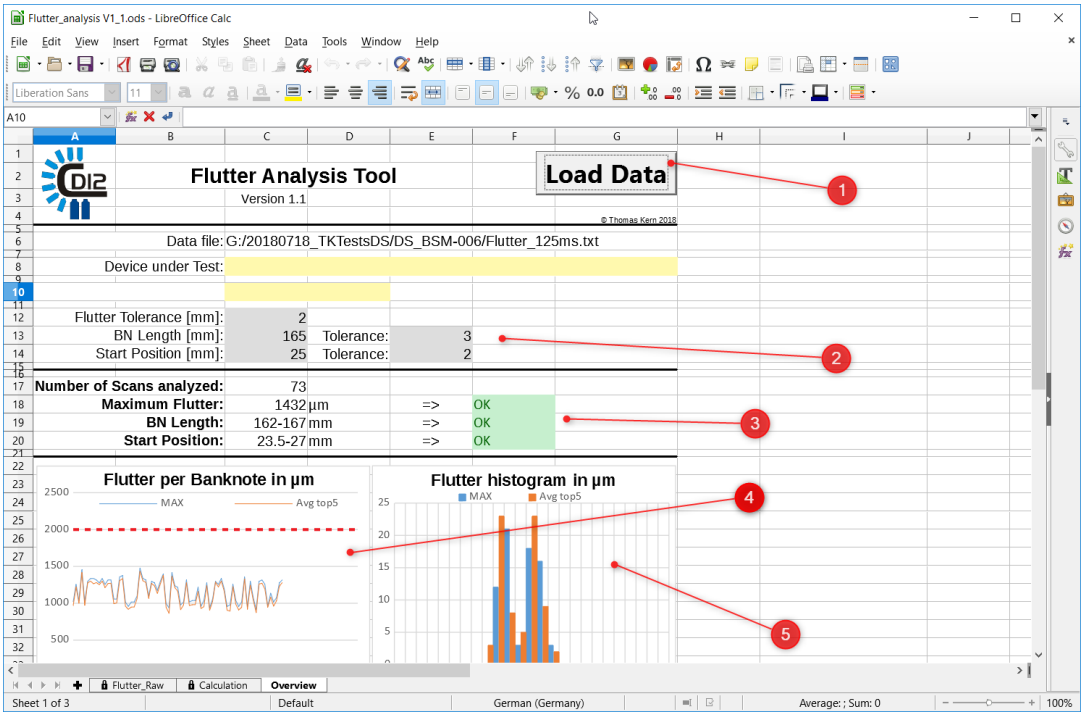


Figure 95: Flutter Analyses Tool

(1) Load Data: Import a previously captured data set from the flutter detector tool (chapter 11.11.1).
(2) Main Settings for limit checking
(3) Analyses Overview with number of scanned banknotes and checking results
(4) Flutter per Banknote: Shows the maximum and average top 5 values
(5) Flutter histogram: Shows a histogram of the maximum and average top 5 values

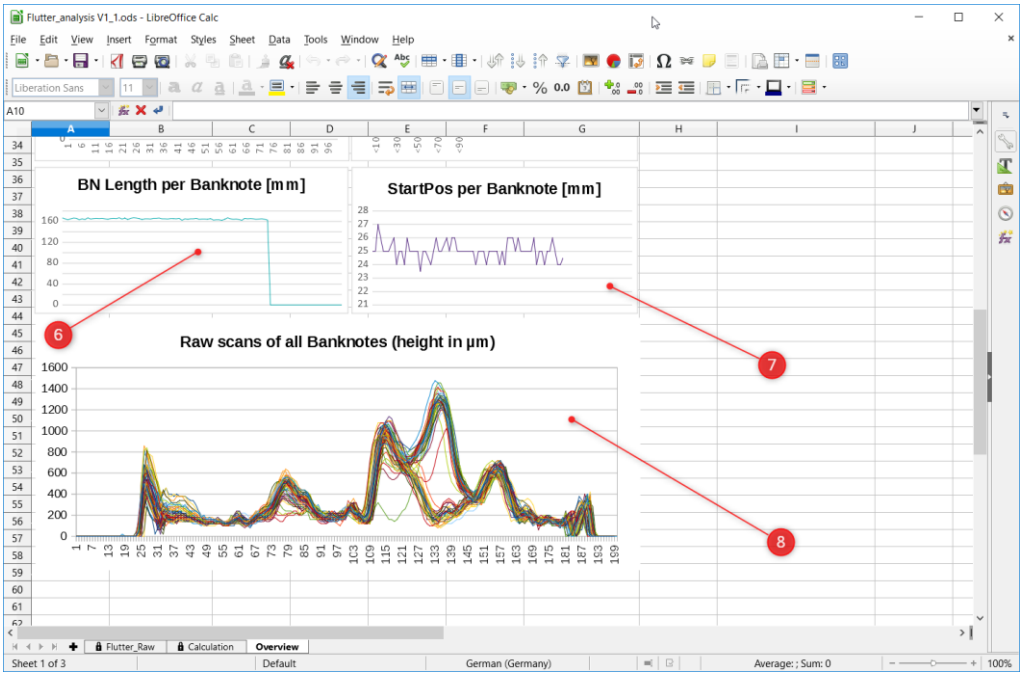


Figure 96: Flutter Analyses Tool (cont.)

(6) Banknote Length per Banknote: Shows the measured length for each banknote
(7) StartPos per Banknote: Shows the starting position for each banknote
(8) Raw scans of all Banknotes: Shows an overlay of all scanned banknotes

Note that the analyses information is valid when data has been captured by a Flutter Detector in CS casing. In case of data coming from a Flutter Detector in Detector casing, some analyses information is invalid or is not shown properly. Specifically, displays (6) and (7) shall be ignored.

13. Technical Specification

These specifications apply to the BSM simulator and the Device simulator.

Operating Conditions

AC Input Voltage range	100-240V, 50/60Hz, 500W max.
Operating Temperature	10 to 40°C
Conducted & Radiated EMI	EN 61326, CISPR 11, Class B EN 61000-3-2, Class A EN 61000-3-3
Immunity	EN 61326 IEC 61000-4-2, IEC 61000-4-3, IEC 61000-4-4, IEC 61000-4-5, IEC 61000-4-6, IEC 61000-4-8, IEC 61000-4-11

14. Annex: Description of BSMS Test Sets for BSM Acceptance

14.1. BSMS_CS-007 Sorting to stackers

This test case is used together with the *TS* which is filled with two banknotes one *FIT* and one *UNFIT*. It verifies that the *CS* reports *FIT* and *UNFIT* alternately.

14.1.1. Functionality

1. Bring the system to FEED_OFF
2. Wait until **Continue** is pressed
3. Transition to SORTING
4. Check that the *CS* reports *FIT* and *UNFIT* alternately

14.1.2. Inheritance

Test case inherits from *BSMS_DEV-G01_CS Internal Trigger*. See there for parameter descriptions.

14.2. BSMS_DET-003 Sorting to stackers

This test case is used together with the *TS* which is filled with two banknotes one *FIT* and one *UNFIT*. It verifies that the *Detector* reports *FIT* and *UNFIT* alternately.

14.2.1. Functionality

1. Bring the system to FEED_OFF
2. Wait until **Continue** is pressed
3. Transition to SORTING
4. Check that the *Detector* reports *FIT* and *UNFIT* alternately

14.2.2. Inheritance

Test case inherits from *BSMS_CS-007 Sorting to stackers*. See there for parameter descriptions.

14.3. BSMS_DEV-003 HTTP and additional services

This test set is used to check that a device supports an HTTP web server as well as mandatory and, optionally, device specific Additional Services in certain states.

14.3.1. Functionality

The test case runs the startup sequence until the device reaches the DS_FEED_OFF state. It displays a link to the HTTP port of the device, allowing the user to open the Service and Maintenance page of the device. Further, the user may access the FTP file services of the device using an FTP client (e.g. WinSCP). On **Continue** the BSMS enters BS_ERROR allowing to test if HTTP server of the device is still offered in the DS_ERROR state.

14.3.2. Inheritance

Test case inherits from *BSMS_BSM-020 HTTP and additional services*. See there for parameter descriptions.

14.4. BSMS_DEV-005 State Transitions

This test case performs various state transitions and checks if the DUT reacts correctly. When it reaches the BS_SORTING state the second time, it waits for one of the connected devices to go to the DS_ERROR state.

14.4.1. Functionality

This test expects the device to follow the BSMS and make following devices state transitions. DS_START_UP, DS_INITIALISATION, DS_INITIALISED, DS_FEED_OFF, DS_READY_TO_SORT, DS_SORTING, DS_FEED_OFF, DS_READY_TO_SHUT_DOWN, DS_SHUT_DOWN, DS_START_UP, DS_INITIALISATION, DS_INITIALISED, DS_FEED_OFF, DS_READY_TO_SORT, DS_SORTING, DS_ERROR, DS_READY_TO_SHUT_DOWN and DS_SHUTDOWN.

14.4.2. Inheritance

Test case inherits from *BSMS_DEV-005 State Transitions*. See there for parameter descriptions.

14.5. BSMS_DEV-006 Powerlink Errors

Checks the correct reaction of a device to specific Powerlink errors.

14.5.1. Functionality

1. Transitions the BSMS to BS_FEED_OFF
2. Waits until **Continue** is pressed
3. Injects a CRC error in the PReq frame for the DUT. The DUT is expected to detect a “Loss of PReq” error.
4. Checks that the device transitions to DS_ERROR
5. Waits until **Continue** is pressed
6. Transitions the BSMS to BS_FEED_OFF
7. Injects a CRC error in a SoC frame for the DUT. The DUT is expected to detect a “Loss of SoC” error.
8. Checks that the device transitions to DS_ERROR

14.5.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

14.6. BSMS_DEV-007 Software Update

This test case checks that the DUT can be updated with a software update as described by CDI2.

14.6.1. Functionality

The test performs the following steps:

1. Go to BS_FEED_OFF and waits for the user to **Continue**.

2. The user uploads a “firmware.bin” to the device using a standard FTP client, e.g. WinSCP
3. Upon **Continue** the test performs the software update sequence and resets the device.
4. The BSMS waits until the device has done the software update and it is ready again.
5. The BSMS enters BS_FEED_OFF

14.6.2. Inheritance

Test case inherits from *BSMS_BSM-023 Software Updates*. See there for parameter descriptions.

14.7. BSMS_DEV-010 TTS Error

This test set checks the response of a device to different TTS errors.

14.7.1. Functionality

Runs the following tests:
1. “Loss of TC” (E)
2. “TC out of range” (W)
3. “BP without BNID” (W)
4. “BNID without BP” (W)

For each test the following steps are performed:
1. Transitions the system up to FEED_OFF and wait until **Continue** is pressed
2. Inject error
3. Retrieve and display error state and `current_error.xml`
4. Wait until **Continue** is pressed
5. Transitions the system up to FEED_OFF

14.7.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

14.8. BSMS_DEV-014 TTS Reset

This test case checks that the DUT reacts correctly to TTS reset signals with different lengths.

14.8.1. Functionality

The test performs the following steps:
1. Go to FEED_OFF
2. Perform a reset with a short reset pulse on **Continue**
3. Check that the device ignores this reset pulse
4. Perform a proper reset pulse on **Continue**
5. Check that the device resets properly

14.8.2. Inheritance

Test case inherits from *BSMS_BSM-032 Reset Devices*. See there for parameter descriptions.

14.9. BSMS_DEV-G01_CS Internal Trigger

This test supports the testing of a single camera system.

14.9.1. Functionality

First, the system starts up, initialises and transitions into the state BS_FEED_OFF. Then it waits for the user to prepare the various network analysis tools (e.g. ProfiTap, Wireshark, …). Upon **Continue** the test sorts 1000 banknotes and shuts down. After that the user can start analysing the network logs.

Device Settings

The camera system is expected to use node id 1.

14.9.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

14.10. BSMS_DEV-G01_DET Internal Trigger

This test supports the testing of a single detector.

14.10.1. Functionality

First, the system starts up, initialises and transitions into the state BS_FEED_OFF. Then it waits for the user to prepare the various network analysis tools (e.g. ProfiTap, Wireshark, …). Upon **Continue** the test sorts 1000 banknotes and shuts down. After that the user can start analysing the network logs.

Device Settings

The camera system is expected to use node id 1.

14.10.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

14.11. BSMS_DEV-G01_IEU Internal Trigger

This test supports the testing of a single IEU.

14.11.1. Functionality

First, the system starts up, initialises and transitions into the state BS_FEED_OFF. Then it waits for the user to prepare the various network analysis tools (e.g. ProfiTap, Wireshark, …). Upon **Continue** the test sorts 1000 banknotes and shuts down. After that the user can start analysing the network logs.

Device Settings

The IEU is expected to use node id 1.

14.11.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

14.12. BSMS_IEU-002 IDB Error

Checks the correct reaction to specific IDB errors.

During sorting, the *BSMS* injects a minor error for which the *IEU* is expected to continue operation. After that sorting is continued to allow for the IDB connection to be manually disconnected.

14.12.1. Functionality

1. The system is brought up to SORTING
2. 100 banknotes are sorted, containing a single IDB error
3. The system goes to FEED_OFF
4. Waits until **Continue** is pressed
5. Sorts another 1000 banknotes to allow for the IDB cable to be unplugged manually.

14.12.2. Inheritance

Test case inherits from *BSMS_DEV-G01_IEU Internal Trigger*. See there for parameter descriptions.

15. Annex: Description of DS Simulator-only Test Sets for BSMS Acceptance

15.1. DS_CS-007 Sorting to Stackers

This test simulates a single camera system that always reports FIT, UNFIT as fitness results.

15.1.1. Functionality

The camera system is compliant with the CDI2 spec and reacts to all state transitions normally. It reports FIT, UNFIT as fitness result alternately.

Device Settings

The camera system is configured with node id 1.

15.1.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

15.2. DS_DET-003 Sorting to Stackers

This test simulates a single detector that always reports FIT, UNFIT as fitness results.

15.2.1. Functionality

The detector is compliant with the CDI2 spec and reacts to all state transitions normally. It reports FIT, UNFIT as fitness result alternately.

Device Settings

The camera system is configured with node id 1.

15.2.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

15.3. DS_DEV-003 HTTP and additional services

This test set supports *BSMS* in testing the HTTP and additional services of a device.

15.3.1. Functionality

The device follows the BSMS states and enters DS_ERROR when the BSMS transitions to BS_ERROR. It provides the HTTP and additional services in DS_ERROR as well.

15.3.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

15.4. DS_DEV-006 Powerlink Errors

This test case provides a device which follows the BSM to all states while it is checking for correct behaviour on Powerlink errors. It has to be stopped by pressing **Stop**.

15.4.1. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

15.5. DS_DEV-007 Software Update

This test case verifies that the *BSMS* can perform a software updates of a device.

15.5.1. Functionality

The test performs the following steps:

6. Go to DS_FEED_OFF
7. Follow software update
8. Checks that the received “firmware.bin” is correct
9. Follows the BSM through the software update procedure and enters DS_FEED_OFF finally.

15.5.2. Inheritance

Test case inherits from *DS_BSM-023*. See there for parameter descriptions.

15.6. DS_DEV-G01_CS Sorting

This test set simulates a single camera system as a standalone device. It is a general test set to be used in different test cases.

15.6.1. Functionality

The camera system is compliant with the CDI2 spec and reacts to all state transitions normally. The test succeeds if the CS receives 1000 banknote triggers and is shut down cleanly.

Device Settings

The camera system is configured with node id 1.

15.6.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

15.7. DS_DEV-G01_DET Sorting

This test simulates a single detector. It is a general test set to be used in different test cases.

15.7.1. Functionality

The detector is compliant with the CDI2 spec and reacts to all state transitions normally. The test succeeds if it receives 1000 banknote triggers and is shut down cleanly.

Device Settings

The detector is configured with node id 1.

15.7.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

15.8. DS_DEV-G01_IEU Sorting

This test simulates a single IEU. It is a general test set to be used in different test cases.

15.8.1. Functionality

The IEU is compliant with the CDI2 spec and reacts to all state transitions normally. The test succeeds if it receives 1000 triggers and is shut down cleanly.

Device Settings

The IEU is configured with node id 1.

15.8.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16. Annex: Description of DS Test Sets for Device Acceptance

16.1. DS_BSM-002 Distance

This test case tests the worst case distances and timeouts that are allowed by the CDI2 spec. To test all device types it provides 4 devices, namely 1 CS, 1 IEU, and 2 detectors.

16.1.1. Functionality

All *BNRESULTs* have the Judgement code `FIT`, so that it can easily be checked that no specimen was sorted to sorting gate `REJECT`. All devices are set up to provide their results as late as possible:

10. The CS provides its images as late as possible: `tIMAGE_BFA = tBP_BFA + tMaxBN + 5 ms`
11. The IEU provides its BNRESULT as late as possible: `tRES_IEU = tIMAGE + 45 ms`
12. The detectors provide their BNRESULT as late as possible: `tRES_DET = tBP + tMaxBN + 20 ms`

16.1.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.2. DS_BSM-006 Flutter

This test provides a way to deploy and trigger the flutter detector. The flutter detector is connected to TTS1 of the BSM and thus receiving clock and trigger via TTS. The device simulator supports the startup protocol on the DMB, instead of the flutter detector. The device actually operates in the non-TTS mode.

16.2.1. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.3. DS_BSM-014 Sort test charts

This test follows the BSM when sorting test charts. It checks that the BSM overrules the *BNRECOGINTION* received from the CS and sends a fixed *BNRECOGNITION* with Series=0 (Test/Calibration), Denomination=1 (Blank Sheet) and Orientation=1 (Front) instead.

16.3.1. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.4. DS_BSM-017 Powerlink and Nettime

This test supports the testing of the Powerlink protocol and its timing, and that the real-time master clock is transmitted by the BSM. It uses the *DS-BSM Standard Setup* and configures it for the *BSM-017 Powerlink and Nettime* test.

16.4.1. Functionality

The behaviour of the devices is compliant with the CDI2 specification. They react to all state transitions and expect to receive 1000 banknotes.

Additionally, the devices display and update the received Nettime in their status message.

16.4.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.5. DS_BSM-020 HTTP and additional services

Checks that the BSM offers a HTTP browser, supports Additional Services in certain states and that it supports device specific Additional Services.

16.5.1. Functionality

1. Follows the BSM state changes
2. Offers HTTP and additional service files
3. Uploads to writable files are printed to the message field
4. Pressing **Continue** lets one device transition to DS_ERROR
5. To end the test press **Stop**

16.5.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.6. DS_BSM-021 State Transitions

This test case uses the *DS-BSM Standard Setup* and configures it for the *BSM-021 State Transitions* test.

16.6.1. Functionality

The devices react normally to the state transitions of the BSM except for nodes 3 and 8.

Device **#3** takes the maximum allowed time to transition between states. The times are configurable using the test case parameters. To guarantee that the times are in the allowed range they are compensated for the transmission time (10 ms).

When the BSM reaches BS_SORTING for the second time, the user is prompted to click **Continue**. When the **Continue** is clicked in this state, the device **#8** will transition to DS_ERROR, which should trigger the BSM and consequently all other devices to transition into the ERROR state.

This test records the BSM state transitions. The BSM is expected to run through following states. BS_FEED_OFF, BS_REQUEST_TO_SORT, BS_SORTING, BS_FEED_OFF, BS_REQUEST_TO_SHUT_DOWN, BS_SHUTDOWN, BS_INITIALISATION, BS_FEED_OFF, BS_REQUEST_TO_SORT, BS_SORTING, BS_ERROR, BS_REQUEST_TO_SHUT_DOWN and BS_SHUTDOWN.

16.6.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.7. DS_BSM-022 Powerlink Errors

This test injects errors on the Powerlink and checks the reaction by the BSM.

16.7.1. Functionality

If this node has the node id specified in the parameters it will perform the following steps:

6. Waits for SYS_FEED_OFF
7. Waits until **Continue** is pressed
8. Disable Powerlink transmitter for four cycles (misses one PRes)
9. Checks if the BSM transitioned to BS_ERROR
10. Waits until SYS_FEED_OFF is reached again
11. Waits until **Continue** is pressed
12. Disable Powerlink transmitter for 1000 cycles (simulating loss of device)
13. Checks if the BSM transitioned to BS_ERROR
14. Done

All other nodes will just follow the state transitions of the system.

16.7.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.8. DS_BSM-023 Software Update

This DS test set is used to check, that the BSM can perform software updates of a device.

16.8.1. Functionality

The test performs the following steps:

15. Go to DS_FEED_OFF
16. Follow software update
17. Check that the received “firmware.bin” is correct
18. Follow BSM back to DS_FEED_OFF

16.8.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.9. DS_BSM-024 Non-TTS Operation

This test case uses the *DS-BSM-Non-TTS* setup and checks that the BSM supports devices without a physical TTS connection.

16.9.1. Functionality

This test case sorts 1000 banknotes in non-TTS mode. In fact, it behaves exactly the same as the *DS_BSM-G01* test case, except that all devices operate in the non-TTS mode.

16.9.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.10. DS_BSM-030 IDB Error

Checks the BSM for correct reaction to specific IDB errors.

During sorting, the DS injects a minor error for which the BSM is expected to continue operation. After that sorting is continued to allow for the IDB connection to be manually disconnected.

16.10.1. Functionality

13. The simulator is set up to report `FIT` for all banknotes.
14. The IDB sender is set up to introduce a single error into the IDB stream.

16.10.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.11. DS_BSM-031 Sorting to Stackers

This test case uses the *DS-BSM Standard Setup* and configures it for the “BSM-031 Sorting to Stackers” test.

16.11.1. Functionality

The devices expect to receive 400 banknotes. They are prepared to set the Judgement Code to a certain value. All devices set the first 100 banknotes to `FIT`, the next 100 to `REJECT`, the next 100 to `UNFIT`, and the last 100 to `SPECIAL1`.

16.11.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.12. DS_BSM-033 XML Error

During this test case the CS sends an invalid XML string, which should cause an error on the BSM.

16.12.1. Parameters:

Device Info Content

The content of the `device_info.xml` that is send by the device to the BSM.

16.12.2. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.13. DS_BSM-G01-HR Sorting

This test case provides the *DS-BSM Standard Setup* with 16 devices and sorts 1000 banknotes.

In contrast to DS_BSM-G01, in DS_BSM-G01-HR the IDB sender is configured for the transmission of optional streams, with different image sizes and higher resolution.

Stream 0 : 488x920
Stream 1 : 488x920
Stream 2 : 488x920

Stream 3 : 488x920
Stream 4 : 488x920
Stream 5 : 488x920
Stream 6 : 488x920
Stream 7 : 488x920
Stream 8 : 488x920
Stream 9 : 488x920
Stream 10 : 488x920
Stream 11 : 488x920
Stream 12 : 488x920
Stream 13 : 488x920
Stream 14 : 488x920
Stream 15 : 976x1840
Stream 16 : 976x1840
Stream 17 : 976x1840
Stream 18 : 976x16
Stream 19 : 976x16

16.13.1. Functionality

The devices behave correctly according to the CDI2 standard and react to the state changes of the BSM. The devices expect to be initialised and brought into the DS_FEED_OFF state. After DS_FEED_OFF they expect to go to the DS_SORTING state, via the DS_READY_TO_SORT state, and to receive exactly 1000 banknote triggers. The test finishes when the devices are shut down and stopped cleanly.

16.13.2. Triggering

The devices in TTS mode use the TTS-BP signal, while devices in non-TTS mode use the trigger announcements received with the *BSMINFO* packet.

16.13.3. Parameters

This test case supports input parameters, allowing the user to adapt the testcase to different use cases.

Number of Banknotes

The expected number of banknotes to sort.

Run Forever

19. False: The device stops the testcase after the expected number of banknotes.
20. True: The device does not abort when a BSM testcase stops and/or enters BS_ERROR. Instead, it accepts a new startup sequence of the BSM and continues normal operation. The Number of Banknotes parameter is ignored.

Device Command Line

An extra command line argument to the simulated component.

Fitness Results

A comma separated list of values (FIT, UNFIT, REJECT, SPECIAL1, SPECIAL2, SPECIAL3 SPECIAL4 or SPECIAL5) that is returned as result code for each banknote. When the end of the list is reached the next result will start again with the first entry.

DS Command Line

An extra command line argument to the simulated component.

Static recognition

If not empty, a set of three comma separated integers specifying orientation, series and denomination

Poweroff Stop Count

The stop count decrements with every NMTStopCmd the device receives. When it reaches 0 the device will power down and check if the test has passed. An initial value of 0 disables this feature.

Short Self-Test Duration (ms)

Time in milliseconds the devices will need to perform a short self-test, including system state transition times (10ms). If specified as *min:max*, a random value between min and max is chosen.

Intensive Self-Test Duration (ms)

Time in milliseconds the devices will need to perform an intensive self-test, including system state transition times (10ms). If specified as *min:max*, a random value between min and max is chosen.

Stall Short Self-Test (duration:cycle)

Let a Short Self-Test take a certain duration (ms) at a specific self-test cycle (cycle counting starts at 0). This is done once and overrides the Short Self-Test Duration parameter.

Stall Intensive Self-Test (duration:cycle)

Let an Intensive Self-Test take a certain duration (ms) at a specific self-test cycle (cycle counting starts at 0). This is done once and overrides the Intensive Self-Test Duration parameter.

Set Maintenance State at Short Self-Test (state@cycle)

Set maintenance status flags at a specific Short Self-Test cycle (cycle counting starts at 0). The state code may range from 0 to 15. The maintenance status flags are reset to 0 at the next, either short or intensive self-test.

Set Maintenance State at Intensive Self-Test (state@cycle)

Set maintenance status flags at a specific Intensive Self-Test cycle (cycle counting starts at 0). The state code may range from 0 to 15. The maintenance status flags are reset to 0 at the next, either short or intensive self-test.

Test Tag

Operator-specified textual metadata that is stored along with the BSM results package.

IDB Sender

The IDB sender module sends prepared images on the IDB whenever it is triggered by the CS. This module generates a message that indicates the number of images it has sent.

16.13.4. Debugging

Logfiles are provided in the result package. See *idb_x.log* in the *IDB_DEV* folder for the IDB receiver and *everything.log* in the *DMB_DEVICE_results/log* folders for CN logs.

16.14. DS_BSM-G01-HRMAX Sorting

This test case provides the *DS-BSM Standard Setup* with 16 devices and sorts 1000 banknotes.

In contrast to DS_BSM-G01, in DS_BSM-G01-HRMAX the IDB sender is configured for the transmission of optional streams with different image sizes and higher resolution. The testcase transmits 33 streams and utilizes maximum IDB bandwidth.

Stream 0 : 488x920
Stream 1 : 488x920
Stream 2 : 488x920
Stream 3 : 488x920
Stream 4 : 488x920
Stream 5 : 488x920
Stream 6 : 488x920
Stream 7 : 488x920
Stream 8 : 488x920
Stream 9 : 488x920
Stream 10 : 488x920
Stream 11 : 488x920
Stream 12 : 488x920
Stream 13 : 488x920
Stream 14 : 488x920
Stream 15 : 976x1840
Stream 16 : 976x1840
Stream 17 : 976x1840
Stream 18 : 976x1840
Stream 19 : 488x920
Stream 20 : 488x920
Stream 21 : 488x920
Stream 22 : 488x920
Stream 23 : 488x920
Stream 24 : 488x920
Stream 25 : 488x920
Stream 26 : 488x920
Stream 27 : 488x920
Stream 28 : 488x920
Stream 29 : 488x920
Stream 30 : 488x920

Stream 31 : 488x920
Stream 32 : 488x920

16.14.1. Functionality

The devices behave correctly according to the CDI2 standard and react to the state changes of the BSM. The devices expect to be initialised and brought into the DS_FEED_OFF state. After DS_FEED_OFF they expect to go to the DS_SORTING state, via the DS_READY_TO_SORT state, and to receive exactly 1000 banknote triggers. The test finishes when the devices are shut down and stopped cleanly.

16.14.2. Triggering

The devices in TTS mode use the TTS-BP signal, while devices in non-TTS mode use the trigger announcements received with the *BSMINFO* packet.

16.14.3. Parameters

This test case supports input parameters, allowing the user to adapt the testcase to different use cases.

Number of Banknotes

The expected number of banknotes to sort.

Run Forever

21. False: The device stops the testcase after the expected number of banknotes.
22. True: The device does not abort when a BSM testcase stops and/or enters BS_ERROR. Instead, it accepts a new startup sequence of the BSM and continues normal operation. The Number of Banknotes parameter is ignored.

Device Command Line

An extra command line argument to the simulated component.

Fitness Results

A comma separated list of values (FIT, UNFIT, REJECT, SPECIAL1, SPECIAL2, SPECIAL3 SPECIAL4 or SPECIAL5) that is returned as result code for each banknote. When the end of the list is reached the next result will start again with the first entry.

DS Command Line

An extra command line argument to the simulated component.

Static recognition

If not empty, a set of three comma separated integers specifying orientation, series and denomination

Poweroff Stop Count

The stop count decrements with every NMTStopCmd the device receives. When it reaches 0 the device will power down and check if the test has passed. An initial value of 0 disables this feature.

Short Self-Test Duration (ms)

Time in milliseconds the devices will need to perform a short self-test, including system state transition times (10ms). If specified as *min:max*, a random value between min and max is chosen.

Intensive Self-Test Duration (ms)

Time in milliseconds the devices will need to perform an intensive self-test, including system state transition times (10ms). If specified as *min:max*, a random value between min and max is chosen.

Stall Short Self-Test (duration:cycle)

Let a Short Self-Test take a certain duration (ms) at a specific self-test cycle (cycle counting starts at 0). This is done once and overrides the Short Self-Test Duration parameter.

Stall Intensive Self-Test (duration:cycle)

Let an Intensive Self-Test take a certain duration (ms) at a specific self-test cycle (cycle counting starts at 0). This is done once and overrides the Intensive Self-Test Duration parameter.

Set Maintenance State at Short Self-Test (state@cycle)

Set maintenance status flags at a specific Short Self-Test cycle (cycle counting starts at 0). The state code may range from 0 to 15. The maintenance status flags are reset to 0 at the next, either short or intensive self-test.

Set Maintenance State at Intensive Self-Test (state@cycle)

Set maintenance status flags at a specific Intensive Self-Test cycle (cycle counting starts at 0). The state code may range from 0 to 15. The maintenance status flags are reset to 0 at the next, either short or intensive self-test.

Test Tag

Operator-specified textual metadata that is stored along with the BSM results package.

IDB Sender

The IDB sender module sends prepared images on the IDB whenever it is triggered by the CS. This module generates a message that indicates the number of images it has sent.

16.14.4. Debugging

Logfiles are provided in the result package. See *idb_x.log* in the *IDB_DEV* folder for the IDB receiver and *everything.log* in the *DMB_DEVICE_results/log* folders for CN logs.

16.15. DS_BSM-G01-RF-HR Sorting (Run Forever)

This test case inherits from *DS_BSM-G01-HR Sorting* and uses *Run Forever mode = true* default parameters. It can be used for continuous operation, as it does not terminate when the BSM stops operation and will accept re-initialization.

16.15.1. Inheritance

Test case inherits from *DS_BSM-G01-HR Sorting*. See there for parameter descriptions.

16.16. DS_BSM-G01-RF Sorting (Run Forever)

This test case inherits from *DS_BSM-G01 Sorting* and uses *Run Forever mode = true* default parameters. It can be used for continuous operation, as it does not terminate when the BSM stops operation and will accept re-initialization.

16.16.1. Inheritance

Test case inherits from *DS_BSM-G01 Sorting*. See there for parameter descriptions.

16.17. DS_BSM-G01 Sorting

This test case provides the *DS-BSM Standard Setup* with 16 devices and sorts 1000 banknotes.

16.17.1. Functionality

The devices behave correctly according to the CDI2 standard and react to the state changes of the BSM. The devices expect to be initialised and brought into the DS_FEED_OFF state. After DS_FEED_OFF they expect to go to the DS_SORTING state, via the DS_READY_TO_SORT state, and to receive exactly 1000 banknote triggers. The test finishes when the devices are shut down and stopped cleanly.

16.17.2. Triggering

The devices in TTS mode use the TTS-BP signal, while devices in non-TTS mode use the trigger announcements received with the *BSMINFO* packet.

16.17.3. Parameters

This test case supports input parameters, allowing the user to adapt the testcase to different use cases.

Number of Banknotes

The expected number of banknotes to sort.

Run Forever

1. False: The device stops the testcase after the expected number of banknotes.
2. True: The device does not abort when a BSM testcase stops and/or enters BS_ERROR. Instead, it accepts a new startup sequence of the BSM and continues normal operation. The Number of Banknotes parameter is ignored.

Device Command Line

An extra command line argument to the simulated component.

Fitness Results

A comma separated list of values (FIT, UNFIT, REJECT, SPECIAL1, SPECIAL2, SPECIAL3 SPECIAL4 or SPECIAL5) that is returned as result code for each banknote. When the end of the list is reached the next result will start again with the first entry.

DS Command Line

An extra command line argument to the simulated component.

Static recognition

If not empty, a set of three comma separated integers specifying orientation, series and denomination

Poweroff Stop Count

The stop count decrements with every NMTStopCmd the device receives. When it reaches 0 the device will power down and check if the test has passed. An initial value of 0 disables this feature.

Short Self-Test Duration (ms)

Time in milliseconds the devices will need to perform a short self-test, including system state transition times (10ms). If specified as *min:max*, a random value between min and max is chosen.

Intensive Self-Test Duration (ms)

Time in milliseconds the devices will need to perform an intensive self-test, including system state transition times (10ms). If specified as *min:max*, a random value between min and max is chosen.

Stall Short Self-Test (duration:cycle)

Let a Short Self-Test take a certain duration (ms) at a specific self-test cycle (cycle counting starts at 0). This is done once and overrides the Short Self-Test Duration parameter.

Stall Intensive Self-Test (duration:cycle)

Let an Intensive Self-Test take a certain duration (ms) at a specific self-test cycle (cycle counting starts at 0). This is done once and overrides the Intensive Self-Test Duration parameter.

Set Maintenance State at Short Self-Test (state@cycle)

Set maintenance status flags at a specific Short Self-Test cycle (cycle counting starts at 0). The state code may range from 0 to 15. The maintenance status flags are reset to 0 at the next, either short or intensive self-test.

Set Maintenance State at Intensive Self-Test (state@cycle)

Set maintenance status flags at a specific Intensive Self-Test cycle (cycle counting starts at 0). The state code may range from 0 to 15. The maintenance status flags are reset to 0 at the next, either short or intensive self-test.

Test Tag

Operator-specified textual metadata that is stored along with the BSM results package.

IDB Sender

The IDB sender module sends prepared images on the IDB whenever it is triggered by the CS. This module generates a message that indicates the number of images it has sent.

16.17.4. Debugging

Logfiles are provided in the result package. See *idb_x.log* in the *IDB_DEV* folder for the IDB receiver and *everything.log* in the *DMB_DEVICE_results/log* folders for CN logs.

17. Annex: Description of BSMS Simulator-only Test Set for DS Acceptance

17.1. BSMS_BSM-002 Distance

This test case tests the worst case distances and timeouts that are allowed by the CDI2 spec. The BSMS sorts 1000 banknotes and measures the timeouts.

17.1.1. Functionality

The test case is limited to 4 devices, namely 1 CS, 1 IEU, and 2 detectors. Every device should be set up to deliver its *BNRESULT* at the last possible moment allowed by the CDI2 spec. The test case will measure the time between the BP signal and the arrival of the *BNRESULT*. If a *BNRESULT* does not arrive in time, the test fails immediately.

Furthermore, the Wireshark network analyser is used to record the Powerlink network packets for further analyses. The CDI2 statistics plugin provides means to measure the trigger to *BNRESULT* as well.

17.1.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.2. BSMS_BSM-006 Flutter

This test sorts 1000 banknotes using the TS to test the flutter detector.

17.2.1. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.3. BSMS_BSM-014 Sort test charts

This test runs the simulator to sort the test charts and an additional 1000 blanks. It overrules the series information from the *BNRECOGNITION* to be Series=0 (Test/Calibration), Denomination=1 (Blank Sheet) and Orientation=1 (Front). During the test it checks if all devices gave a judgment of `FIT`.

17.3.1. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.4. BSMS_BSM-017 Powerlink and Nettime

This test set supports the testing of the Powerlink protocol and its timing, and testing of the real-time master clock.

17.4.1. Functionality

First, the system starts up, initialises and transitions into `BS_FEED_OFF`. Then it stops and the user may prepare the various network analysis tools (e.g. ProfiTap, Wireshark, …). Then the user may start sorting by pressing **Continue**. The test will sort 1000 banknotes and shutdown when finished. After that the user can analyse the recorded network logs.

17.4.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.5. BSMS_BSM-020 HTTP and additional services

This test case provides access to the HTTP and additional services on the DS. It provides links to the HTTP interface and to offered files.

17.5.1. Functionality

15. Transitions up to BS_FEED_OFF
16. Follows devices going to DS_ERROR
17. Offers links to all devices in the instructions section

17.5.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.6. BSMS_BSM-021 State Transitions

This test case performs various state transitions and checks if the devices react correctly. When it reaches the BS_SORTING state the second time, it waits for one of the connected devices to go to the DS_ERROR state.

17.6.1. Functionality

The BSMS runs through following states. BS_FEED_OFF, BS_REQUEST_TO_SORT, BS_SORTING, BS_FEED_OFF, BS_REQUEST_TO_SHUT_DOWN, BS_SHUTDOWN, BS_FEED_OFF, BS_REQUEST_TO_SORT, BS_SORTING, BS_ERROR, BS_REQUEST_TO_SHUT_DOWN and BS_SHUTDOWN.

17.6.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.7. BSMS_BSM-022 Powerlink Errors

This test reacts to Powerlink errors in a decent way.

17.7.1. Functionality

18. Transitions up to BS_FEED_OFF
19. Waits for a Powerlink error
20. Waits until **Continue** is pressed
21. Transitions back to BS_FEED_OFF
22. Waits for another Powerlink error

17.7.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.8. BSMS_BSM-023 Software Updates

This test performs a software update procedure for a device and measures the timing of the process.

17.8.1. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.9. BSMS_BSM-024 Non-TTS Operation

This test case configures the *BSMS* for the *DS-BSM-Non-TTS* setup and checks that the devices work properly in non-TTS operation.

17.9.1. Functionality

This test case sorts 1000 banknotes in non-TTS mode. In fact, it behaves exactly the same as the *BSMS_BSM-G01* test case, except that all devices operate in the non-TTS mode.

The *BSMS* is setup to work with devices in non-TTS operation. Hence, all checks regarding the TTS-Ready signal are omitted.

17.9.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.10. BSMS_BSM-030 IDB Error

This test allows the DS to test for IDB error reaction. First it sorts 100 BNs, then it waits for user input and sends another 900 banknotes.

17.10.1. Functionality

23. Transition up to `SYS_SORTING`
24. Sort 100 banknotes
25. Transition to `SYS_FEED_OFF`
26. Wait until **Continue** is pressed
27. Sort another 900 banknotes

17.10.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.11. BSMS_BSM-031 Sorting to Stackers

This test sorts 400 banknotes to different stackers. It expects the devices to set the Judgement code to specific values. The first 100 banknotes must be set to `FIT`, the next 100 to `REJECT`, the next 100 to `UNFIT`, and the last 100 to `SPECIAL1`.

17.11.1. Functionality

The BSMS automatically checks the responses of the devices and reports a result message.

17.11.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.12. BSMS_BSM-032 Reset Devices

This test checks if the DS supports device resets.

17.12.1. Functionality

28. Transitions the system up to FEED_OFF
29. Waits until **Continue** is pressed
30. Performs a reset
31. Transitions the system back up to FEED_OFF

17.12.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.13. BSMS_BSM-033 XML Error

During this test case the CS sends an invalid XML string, which should cause an error on the BSM.

17.13.1. Parameters:

XML Error Target

Specifies the node id of the device the manipulated XMLs should be send to.

Device Config Content

Contains the content that is send as `device_config.xml` to the node specified by *XML Error Target*.

Machine Info Content

Contains the content that is send as `machine_info.xml` to the node specified by *XML Error Target*.

17.13.2. Inheritance

Test case inherits from *BSMS_BSM-G01 Sorting*. See there for parameter descriptions.

17.14. BSMS_BSM-G01-HR Sorting

This test case provides the BSM configuration for a *DS-BSM Standard Setup* with 16 devices and sorts 1000 banknotes. It is set up to broadcast the Powerlink NetTime and RelativeTime in each SoC frame, and it fulfils all TTS requirements.

In contrast to BSMS_BSM-G01, in BSMS_BSM-G01-HR the IDB receiver is configured to receive 20 streams (mandatory + optional streams).

17.14.1. Functionality

Initially the test brings all devices to the FEED_OFF state. After all devices reached the FEED_OFF state it goes to the SORTING state, via the READY_TO_SORT state, and sorts 1000 banknotes. Then it shuts itself

and the devices down and checks if it has received all of the 1000 *BNRESULTs* for all of the banknotes from all devices.

17.14.2. Parameters

This test case supports input parameter, allowing the user to adapt the testcase to different use cases.

Number of Banknotes

The number of banknotes to sort.

Manual Mode

32. False: The BSMS starts sorting immediately after BS_FEED_OFF is reached.
33. True: The BSMS stops at BS_FEED_OFF and waits for the user to press Continue. This option is used when the test requires further preparations before sorting starts. For example a measurement instrument has to be connected or the transport simulator needs time to reach nominal speed.

Trigger source

34. Internal: The BSMS generates an internal banknote trigger as well as an internal transport clock.
35. External: Configures the BSMS to use an external transport clock and trigger signal, usually provided by the transport simulator. The external clock is expected to have a resolution of 0.1 mm.

Trigger distance (ms)

When trigger source is set Internal: Specifies the time between two simulated banknotes are triggered. Use 20 ms to set a sorting speed of 50 banknotes per second.

Internal TC Clock Rate (Hz)

Clock rate of the transport clock when in Internal trigger mode. The TC is counting tenths of millimeters so a value of 125000 results in a transport speed of 12.5 m/s.

BSM Command Line

An extra command line argument to the simulated component.

No Camera System

36. False: Normal operation. BNINFO packets are generated upon reception of BNRECOGNITION from a Camera System.
37. True: BSMS provides simulated BNINFO packets. Use this option only in case no Camera System is installed.

First BNID

Starting point of banknote identifications allocated during this test run.

BN Series

BN series sent out in BNINFO (Only relevant in “No Camera System” mode).

BN Denomination

BN denomination sent out in BNINFO (Only relevant in “No Camera System” mode).

BN Orientation

BN orientation sent out in BNINFO (Only relevant in “No Camera System” mode).

BNINFO Delay (ms)

Delay in milliseconds before the BSM sends out the BNINFO (Only relevant in “No Camera System” mode).

SORTING Timeout (ms)

Wait this many milliseconds for expected BNRESULT messages before aborting the running test. Set to 0 to wait forever.

Short Self-Test Cycle (BN)

Sort this many BNs between short self-tests. Set to 0 for no short self-tests. The BSMS transitions from `BS_SORTING` to `BS_REQUEST_TO_SORT` to allow detectors to run a short self-test.

Intensive Self-Test Cycle (BN)

Sort this many BNs between intensive self-tests. Set to 0 for no intensive self-tests. The BSMS transitions from `BS_SORTING` to `BS_FEED_OFF` to allow detectors to run an intensive self-test. The Intensive Self-Test Cycle interval shall be longer than the Short Self-Test Cycle interval.

Test Tag

Operator-specified textual metadata that is stored along with the BSM results package.

17.14.3. IDB Receiver

The IDB receiver module stores images received on the IDB and checks the IDB network quality. This module generates a message that indicates the number of images it has received as well as the number of errors they contained. The IDB receiver stores up to 100 banknote images, depending on image size and main memory size.

17.14.4. Debugging

Logfiles are provided in the result package. See *idb_x.log* in the *IDB_BSM* folder for the IDB receiver and *everything.log* in the *BSM_SIM_results/log* folder for MN logs.

17.15. BSMS_BSM-G01-HRMAX Sorting

This test case provides the BSM configuration for a *DS-BSM Standard Setup* with 16 devices and sorts 1000 banknotes. It is set up to broadcast the Powerlink NetTime and RelativeTime in each SoC frame, and it fulfils all TTS requirements.

In contrast to BSMS_BSM-G01, in BSMS_BSM-G01-HRMAX the IDB receiver is configured to receive 33 streams at maximum IDB bandwidth.

17.15.1. Functionality

Initially the test brings all devices to the `FEED_OFF` state. After all devices reached the `FEED_OFF` state it goes to the `SORTING` state, via the `READY_TO_SORT` state, and sorts 1000 banknotes. Then it shuts itself

and the devices down and checks if it has received all of the 1000 *BNRESULTs* for all of the banknotes from all devices.

17.15.2. Parameters

This test case supports input parameter, allowing the user to adapt the testcase to different use cases.

Number of Banknotes

The number of banknotes to sort.

Manual Mode

38. False: The BSMS starts sorting immediately after BS_FEED_OFF is reached.
39. True: The BSMS stops at BS_FEED_OFF and waits for the user to press Continue. This option is used when the test requires further preparations before sorting starts. For example a measurement instrument has to be connected or the transport simulator needs time to reach nominal speed.

Trigger source

40. Internal: The BSMS generates an internal banknote trigger as well as an internal transport clock.
41. External: Configures the BSMS to use an external transport clock and trigger signal, usually provided by the transport simulator. The external clock is expected to have a resolution of 0.1 mm.

Trigger distance (ms)

When trigger source is set Internal: Specifies the time between two simulated banknotes are triggered. Use 20 ms to set a sorting speed of 50 banknotes per second.

Internal TC Clock Rate (Hz)

Clock rate of the transport clock when in Internal trigger mode. The TC is counting tenths of millimeters so a value of 125000 results in a transport speed of 12.5 m/s.

BSM Command Line

An extra command line argument to the simulated component.

No Camera System

42. False: Normal operation. BNINFO packets are generated upon reception of BNRECOGNITION from a Camera System.
43. True: BSMS provides simulated BNINFO packets. Use this option only in case no Camera System is installed.

First BNID

Starting point of banknote identifications allocated during this test run.

BN Series

BN series sent out in BNINFO (Only relevant in “No Camera System” mode).

BN Denomination

BN denomination sent out in BNINFO (Only relevant in “No Camera System” mode).

BN Orientation

BN orientation sent out in BNINFO (Only relevant in “No Camera System” mode).

BNINFO Delay (ms)

Delay in milliseconds before the BSM sends out the BNINFO (Only relevant in “No Camera System” mode).

SORTING Timeout (ms)

Wait this many milliseconds for expected BNRESULT messages before aborting the running test. Set to 0 to wait forever.

Short Self-Test Cycle (BN)

Sort this many BNs between short self-tests. Set to 0 for no short self-tests. The BSMS transitions from BS_SORTING to BS_REQUEST_TO_SORT to allow detectors to run a short self-test.

Intensive Self-Test Cycle (BN)

Sort this many BNs between intensive self-tests. Set to 0 for no intensive self-tests. The BSMS transitions from BS_SORTING to BS_FEED_OFF to allow detectors to run an intensive self-test. The Intensive Self-Test Cycle interval shall be longer than the Short Self-Test Cycle interval.

Test Tag

Operator-specified textual metadata that is stored along with the BSM results package.

17.15.3. IDB Receiver

The IDB receiver module stores images received on the IDB and checks the IDB network quality. This module generates a message that indicates the number of images it has received as well as the number of errors they contained. The IDB receiver stores up to 100 banknote images, depending on image size and main memory size.

17.15.4. Debugging

Logfiles are provided in the result package. See *idb_x.log* in the *IDB_BSM* folder for the IDB receiver and *everything.log* in the *BSM_SIM_results/log* folder for MN logs.

17.16. BSMS_BSM-G01 Sorting

This test case provides the BSM configuration for a *DS-BSM Standard Setup* with 16 devices and sorts 1000 banknotes. It is set up to broadcast the Powerlink NetTime and RelativeTime in each SoC frame, and it fulfils all TTS requirements.

17.16.1. Functionality

Initially the test brings all devices to the FEED_OFF state. After all devices reached the FEED_OFF state it goes to the SORTING state, via the READY_TO_SORT state, and sorts 1000 banknotes. Then it shuts itself and the devices down and checks if it has received all of the 1000 *BNRESULTs* for all of the banknotes from all devices.

17.16.2. Parameters

This test case supports input parameter, allowing the user to adapt the testcase to different use cases.

Number of Banknotes

The number of banknotes to sort.

Manual Mode

1. False: The BSMS starts sorting immediately after BS_FEED_OFF is reached.
2. True: The BSMS stops at BS_FEED_OFF and waits for the user to press Continue. This option is used when the test requires further preparations before sorting starts. For example a measurement instrument has to be connected or the transport simulator needs time to reach nominal speed.

Trigger source

1. Internal: The BSMS generates an internal banknote trigger as well as an internal transport clock.
2. External: Configures the BSMS to use an external transport clock and trigger signal, usually provided by the transport simulator. The external clock is expected to have a resolution of 0.1 mm.

Trigger distance (ms)

When trigger source is set Internal: Specifies the time between two simulated banknotes are triggered. Use 20 ms to set a sorting speed of 50 banknotes per second.

Internal TC Clock Rate (Hz)

Clock rate of the transport clock when in Internal trigger mode. The TC is counting tenths of millimeters so a value of 125000 results in a transport speed of 12.5 m/s.

BSM Command Line

An extra command line argument to the simulated component.

No Camera System

1. False: Normal operation. BNINFO packets are generated upon reception of BNRECOGNITION from a Camera System.
2. True: BSMS provides simulated BNINFO packets. Use this option only in case no Camera System is installed.

First BNID

Starting point of banknote identifications allocated during this test run.

BN Series

BN series sent out in BNINFO (Only relevant in “No Camera System” mode).

BN Denomination

BN denomination sent out in BNINFO (Only relevant in “No Camera System” mode).

BN Orientation

BN orientation sent out in BNINFO (Only relevant in “No Camera System” mode).

BNINFO Delay (ms)

Delay in milliseconds before the BSM sends out the BNINFO (Only relevant in “No Camera System” mode).

SORTING Timeout (ms)

Wait this many milliseconds for expected BNRESULT messages before aborting the running test. Set to 0 to wait forever.

Short Self-Test Cycle (BN)

Sort this many BNs between short self-tests. Set to 0 for no short self-tests. The BSMS transitions from `BS_SORTING` to `BS_REQUEST_TO_SORT` to allow detectors to run a short self-test.

Intensive Self-Test Cycle (BN)

Sort this many BNs between intensive self-tests. Set to 0 for no intensive self-tests. The BSMS transitions from `BS_SORTING` to `BS_FEED_OFF` to allow detectors to run an intensive self-test. The Intensive Self-Test Cycle interval shall be longer than the Short Self-Test Cycle interval.

Test Tag

Operator-specified textual metadata that is stored along with the BSM results package.

17.16.3. IDB Receiver

The IDB receiver module stores images received on the IDB and checks the IDB network quality. This module generates a message that indicates the number of images it has received as well as the number of errors they contained. The IDB receiver stores up to 1000 banknote images, depending on image size and main memory size.

17.16.4. Debugging

Logfiles are provided in the result package. See *idb_x.log* in the *IDB_BSM* folder for the IDB receiver and *everything.log* in the *BSM_SIM_results/log* folder for MN logs.

18. Annex: BSMS utility testcases

18.1. BSM-003 Get Version Infos

This test case retrieves the software versions of the installed packages.

18.1.1. Functionality

This test case calls a shell script that collects the information from the controller and all DMB nodes. The results are logged to `versions.log` and `versions-summary.txt` (see download package folder `IDB_BSM/log`).

18.2. BSMS-999 Shutdown

This test case performs a shutdown of all CDI2 components.

18.2.1. Functionality

This test case calls a shell script that shuts down all CDI2 components one by one.

19. Annex: DS utility test cases

19.1. DS-003 Get Version Infos

This test case retrieves the software versions of the installed packages.

19.1.1. Functionality

This test case calls a shell script that collects the information from the controller and all DMB nodes. The results are logged to `versions.log` and `versions-summary.txt` (see download package folder `IDB_DEV/log`).

19.2. DS-999 Shutdown

This test case performs a shutdown of all CDI2 components.

19.2.1. Functionality

This test case calls a shell script that shuts down all CDI2 components one by one.

20. Disclaimer

This document describes the usage of the CDI2 BSM and CDI2 Device Simulator for testing and developing CDI2 banknote sorting machines and CDI2 detectors. It was prepared with care and all information contained herein was reviewed. However, AIT accepts no responsibility or liability whatsoever with regard to the content of the document, and assumes no liability for any loss or damage, either direct or indirect, arising from the use or implementation of information contained in this document, including mistakes, incompleteness or discrepancies between this document and the product described.
This disclaimer is not intended to limit the AIT's liability in contravention of any requirements laid down in applicable national law or to exclude its liability for matters which cannot be excluded under such law.

21. License Note

Any intellectual property contained in this document is presumably protected. This document does not grant a license for the use of intellectual property of AIT or third parties.