
STM32CubeMonitor-RF software tool for wireless performance measurements

Introduction

STM32CubeMonitor-RF (STM32CubeMonRF) is a software tool, which helps designers to test their products based on STMicroelectronics STM32 wireless microcontrollers.

The tool performs the following operations:

- It sends and receives test packets to check the efficiency of radio frequency boards and compute the packet error rate (PER) on Bluetooth® LE and IEEE 802.15.4 technologies.
- It sends commands to the Bluetooth® LE controller for standardized tests.
- It sends and receives Bluetooth® LE commands for fast application prototyping.
- It configures a variety of beacons via Bluetooth® LE commands.
- It transfers data over-the-air (OTA) from one device to another, to configure or program a wireless remote device.
- It sends commands to an OpenThread device for application prototyping.
- It explores a Thread network and displays it with all the relevant information.
- It provides a sniffer tool to analyze IEEE 802.15.4 frames with Wireshark®.

This user manual applies to STM32CubeMonitor-RF version 2.18.0 and later.



Contents

1	Getting started	9
1.1	Download and setup	9
1.2	Welcome screen	10
1.3	Main screen	11
1.3.1	Menu bar	12
1.3.2	Connection bar	12
1.3.3	Panels	13
1.3.4	Log area	14
2	Connection to target	15
2.1	Use case descriptions and definitions	15
2.2	VCOM/UART connection	16
2.2.1	VCOM connection	16
2.2.2	UART connection	18
2.2.3	VCP device	19
2.3	Opening COM	20
3	Bluetooth® LE mode	22
3.1	Presentation	22
3.1.1	Panels	22
3.2	Bluetooth® LE stack	23
3.3	ACI Commands panel	23
3.3.1	How to send an ACI command	24
3.3.2	Search function	24
3.3.3	Filter usage	25
3.3.4	How to fill parameters. Fixed field/editable field	25
3.3.5	Log functionalities	26
3.4	RF test panel	30
3.4.1	Transmitter test mode (TX)	31
3.4.2	Receiver test mode (RX)	33
3.4.3	PER	36
3.5	Scripts	42
3.5.1	Launching scripts	42

3.5.2	Script recording	44
3.5.3	Scripts modification	44
3.5.4	Script report	45
3.5.5	List of script commands	46
3.6	Command-line interface (CLI)	49
3.6.1	Installation	49
3.6.2	Getting started	49
3.6.3	Arguments for CLI commands	50
3.6.4	Basic commands	50
3.6.5	Informative commands	51
3.6.6	Execution commands	51
3.6.7	Examples	52
3.7	OTA transfer	58
3.7.1	OTA presentation	58
3.7.2	OTA procedure	59
3.7.3	Use the tool to perform an OTA update	60
3.8	Beacon	64
3.8.1	Beacon presentation	64
3.8.2	Beacon configuration methods:	65
3.8.3	Configuration of the beacon with STM32CubeMonitor-RF	66
3.9	ACI Utilities	70
3.9.1	Remote services discovering	71
3.9.2	Advertising	77
4	OpenThread mode	79
4.1	Presentation	79
4.1.1	Panel	79
4.2	Commands tab	81
4.3	OpenThread scripts tab	83
4.3.1	OpenThread script example	84
4.3.2	List of script commands	84
4.4	Network Explorer tab	85
4.4.1	Controls	85
4.4.2	Display area	87
4.4.3	Infobox	89
4.4.4	Log area	89

5	802.15.4 RF test mode	90
5.1	Presentation	90
5.2	Transmitter test mode (TX)	91
5.2.1	Frame mode	92
5.2.2	Continuous modulated mode	93
5.2.3	Continuous wave mode	93
5.3	Receiver test (RX) mode	94
5.3.1	Packet error rate (PER) test	95
5.3.2	Link quality assessment (LQI) test	96
5.3.3	Energy detection (ED) test	97
5.3.4	Channel clear assessment (CCA) test	97
5.4	Packet error rate (PER) mode	98
5.4.1	Connecting the additional device to play the role of a packet generator (tester)	99
5.4.2	Configure the parameters of the tester	100
5.4.3	Configure the parameters of the device under test (DUT)	101
5.4.4	Configure the measurement	102
6	802.15.4 sniffer	105
6.1	Presentation	105
6.2	Prerequisite	105
6.2.1	Sniffer device	105
6.2.2	Wireshark	105
6.2.3	Python™	105
6.3	Setup verification	106
6.3.1	Sniffer launch	107
6.3.2	Select interface	107
6.3.3	Channel configuration	108
6.3.4	Sniffing start	108
Appendix A Beacon configuration format		109
Revision history		110

List of tables

Table 1. OTA loader address table 59

Table 2. Search filtering 60

Table 3. Measurement setting 103

Table 4. Beacon configuration format 109

Table 5. Document revision history 110

List of figures

Figure 1.	Welcome screen	10
Figure 2.	Main screen	11
Figure 3.	Menu bar.	12
Figure 4.	Connection bar	12
Figure 5.	ACI Commands panel.	13
Figure 6.	Log area	14
Figure 7.	Typical connection with a Nucleo board	15
Figure 8.	Connection with a remote device	15
Figure 9.	Connection with a second device	16
Figure 10.	VCOM connection Bluetooth® LE	16
Figure 11.	VCOM connection Thread	17
Figure 12.	UART connection	18
Figure 13.	VCP connection	19
Figure 14.	Opening COM.	20
Figure 15.	Successful COM.	20
Figure 16.	ACI Commands panel.	22
Figure 17.	How to send an ACI command	24
Figure 18.	Search button	25
Figure 19.	Fixed parameter	25
Figure 20.	Editable parameter	25
Figure 21.	Predefined values	25
Figure 22.	Help details	26
Figure 23.	Log functionalities	26
Figure 24.	More button.	27
Figure 25.	Message details	27
Figure 26.	Purple error messages	28
Figure 27.	Gray second board messages	28
Figure 28.	Save ACI log	29
Figure 29.	LOG_ACI_CubeMonitor-RF.csv	29
Figure 30.	Test mode selection	30
Figure 31.	Change the test mode	30
Figure 32.	Select the test mode	30
Figure 33.	Transmitter test mode.	31
Figure 34.	Transmitting message	32
Figure 35.	Transmitted packets count	33
Figure 36.	Receiver test mode.	33
Figure 37.	RSSI measurement	34
Figure 38.	RSSI measurement graph	35
Figure 39.	RF RSSI measurement large display	36
Figure 40.	PER definition	36
Figure 41.	PER tester connection	37
Figure 42.	PER tester connected.	38
Figure 43.	PER tester configuration.	38
Figure 44.	DUT configuration.	39
Figure 45.	PER test parameters	40
Figure 46.	PER and RSSI measurement graph.	41
Figure 47.	Launching scripts	42
Figure 48.	Script execution	43

Figure 49.	Script recording buttons	44
Figure 50.	Sample script	44
Figure 51.	Script report	45
Figure 52.	Script verdict	46
Figure 53.	Script pause	47
Figure 54.	Example	47
Figure 55.	Loop simple example	47
Figure 56.	Loop second simple example	48
Figure 57.	Loop decrement	48
Figure 58.	Loop specific increment	48
Figure 59.	Nested loop	48
Figure 60.	Example of loop script verdict	49
Figure 61.	Display of loop script verdict	49
Figure 62.	Script loop error	49
Figure 63.	Contents of <i>myAciLog.txt</i> displayed in a spreadsheet tool (sendCommand)	55
Figure 64.	Contents of <i>myAciLog.txt</i> displayed in a spreadsheet tool (script)	57
Figure 65.	Contents of <i>myAciLog.txt</i> displayed in a spreadsheet tool (error investigation 1 of 2)	58
Figure 66.	Contents of <i>myAciLog.txt</i> displayed in a spreadsheet tool (error investigation 2 of 2)	58
Figure 67.	Search procedure	60
Figure 68.	Scanning	61
Figure 69.	No device found	61
Figure 70.	Device found	61
Figure 71.	Select the device and parameters	62
Figure 72.	Configuring in OTA	63
Figure 73.	Progress bar	63
Figure 74.	Beacon presentation	64
Figure 75.	Beacon usage	64
Figure 76.	Online beacon	65
Figure 77.	Offline beacon	65
Figure 78.	Selecting the Beacon mode	66
Figure 79.	Common parameters	66
Figure 80.	iBeacon parameters	67
Figure 81.	Eddystone UID	68
Figure 82.	Eddystone URL	68
Figure 83.	Online mode transfer configuration	69
Figure 84.	Offline mode transfer configuration	69
Figure 85.	ACI Utilities panel	70
Figure 86.	Select checkbox	70
Figure 87.	Scan parameters	71
Figure 88.	Scanning	72
Figure 89.	Select device	72
Figure 90.	Back	72
Figure 91.	Init	72
Figure 92.	Connecting	73
Figure 93.	Connection error	73
Figure 94.	Connected icon	73
Figure 95.	Services list	73
Figure 96.	Characteristics list	74
Figure 97.	Read value	74
Figure 98.	Write value	75
Figure 99.	Indicate value changed	75
Figure 100.	Notify value changed	76

Figure 101. Notifying	76
Figure 102. Advertising parameters	77
Figure 103. Advertising	78
Figure 104. Connected	78
Figure 105. OpenThread - Command tab	79
Figure 106. OpenThread - Script tab	80
Figure 107. OpenThread - Network Explorer tab	80
Figure 108. OpenThread common bottom area	81
Figure 109. OpenThread command tab top area	81
Figure 110. Script buttons	82
Figure 111. Command list	82
Figure 112. Command details	82
Figure 113. Scripts tab	83
Figure 114. Sample script	84
Figure 115. OpenThread network explorer tab	85
Figure 116. Auto connection functions	85
Figure 117. Project and background management	86
Figure 118. Explore and size choice controls	86
Figure 119. Display area	87
Figure 120. Zoom and motion controls	88
Figure 121. Infobox	89
Figure 122. Log area	89
Figure 123. Test mode selection	90
Figure 124. Transmitter test mode	91
Figure 125. Field and picklist defining the frame	92
Figure 126. Help frame information	92
Figure 127. Continuous modulated test mode	93
Figure 128. Receiver test mode	94
Figure 129. PER frames received	95
Figure 130. PER frame reception completed	95
Figure 131. LQI measurement	96
Figure 132. ED measurement	97
Figure 133. CCA measurement	98
Figure 134. Packet tester connection	99
Figure 135. PER tester configuration	100
Figure 136. DUT configuration	101
Figure 137. PER test parameters	102
Figure 138. Standard display	103
Figure 139. Chart display	103
Figure 140. Large PER display	104
Figure 141. Large RSSI display	104
Figure 142. Large LQI display	104
Figure 143. Prerequisite check	106
Figure 144. Wireshark interfaces	107
Figure 145. Wheel	107
Figure 146. Channel choice	108
Figure 147. Sniffing	108

1 Getting started

STM32CubeMonitor-RF supports the STM32WB, STM32WBA, and STM32WB0 series microcontrollers based on the Arm^{®(a)} Cortex[®]-M processor.



1.1 Download and setup

STM32CubeMonitor-RF is compatible with Windows^{®(b)}, Linux^{®(c)}, and macOS^{®(d)} operating systems.

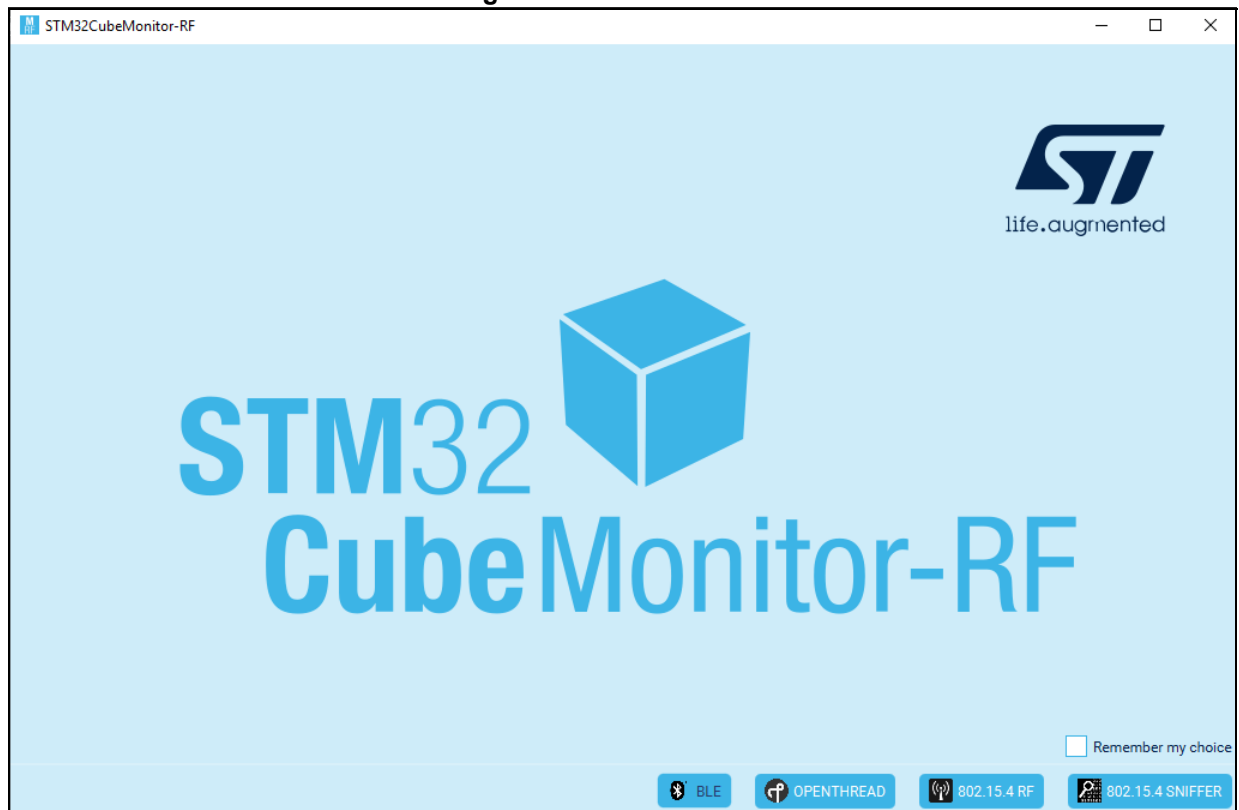
The information on how to install the application is described in the release note, which explains the compatibilities and new features available in the tool.

Refer to the STM32CubeMonRF release note (RN0104) to install and configure the application.

-
- a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
 - b. Windows is a trademark of the Microsoft group of companies.
 - c. Linux[®] is a registered trademark of Linus Torvalds.
 - d. macOS[®] is a trademark of Apple Inc., registered in the U.S. and other countries and regions.

1.2 Welcome screen

Figure 1. Welcome screen



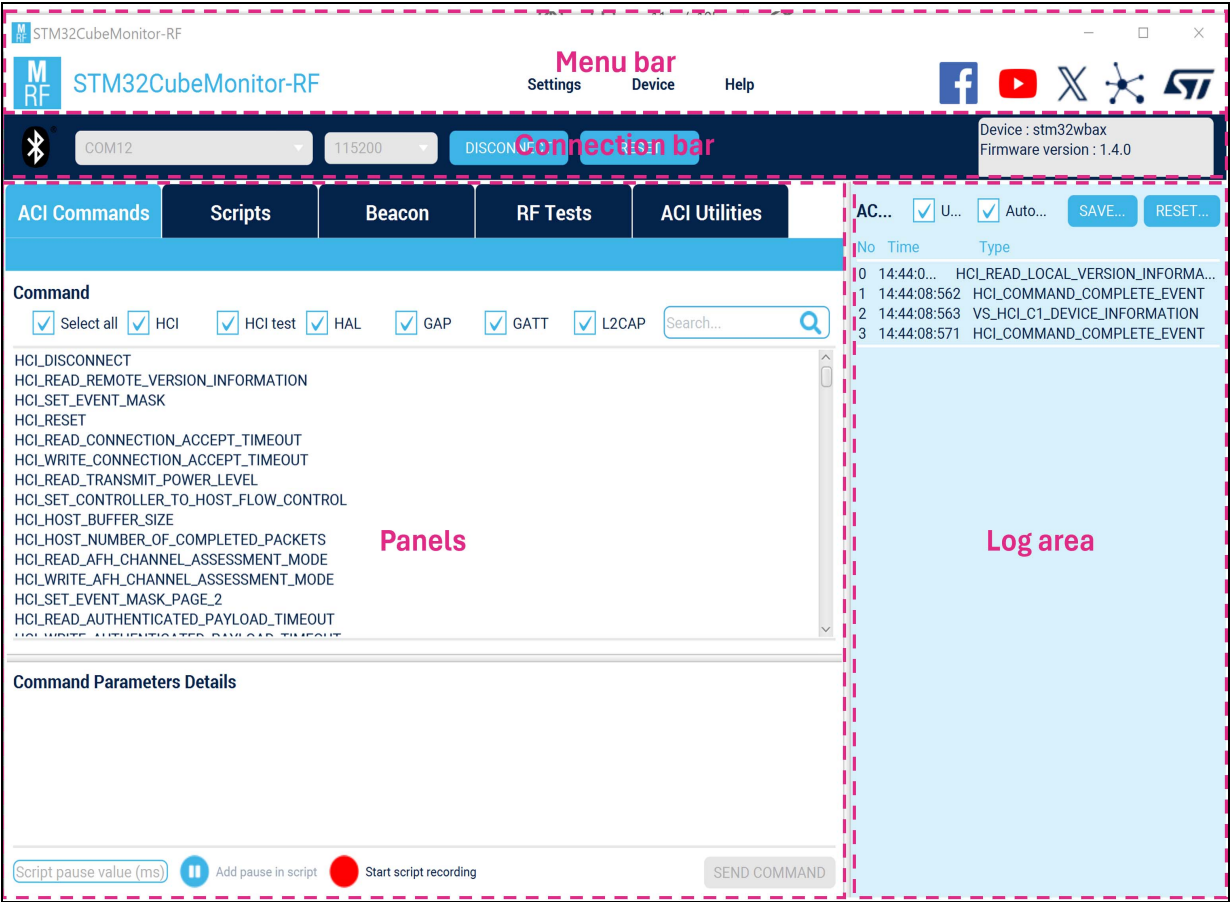
Launching the application opens the welcome screen, where the users select the mode they want to use: Bluetooth® LE, OpenThread, 802.15.4 RF, or 802.15.4 sniffer.

The checkbox *Remember my choice* memorizes the selection so that the next application launch directly opens it, without the welcome screen, except for the 802.15.4 sniffer.

1.3 Main screen

The main screen of the tool is divided into four sections: the menu bar, the connection bar, the panels, and the log area.

Figure 2. Main screen



1.3.1 Menu bar

Figure 3. Menu bar



The application header includes a menu bar for specific tools and displays useful information.

The *Settings* menu allows users to change the mode and reset the default mode selection. Resetting the selection causes the welcome screen to reappear.

The *Device* menu provides information and actions for the connected board.

The *Help* menu provides details about the tool version in use.

The social network links are available in the upper-right corner. This area includes five shortcuts to access social networks:

- The Facebook™ icon leads to the official STMicroelectronics Facebook page.
- The YouTube™ icon leads to the official STMicroelectronics YouTube page.
- The X icon leads to the official STMicroelectronics X page.
- The Share icon leads to the ST community website.
- The STMicroelectronics icon leads to the STMicroelectronics website.

1.3.2 Connection bar

Figure 4. Connection bar



The connection bar shows information about the device connected to the application.

The icon on the left side shows the selected mode.

The first picklist is used to select the COM port.

The second picklist is used to select the serial baud rate.

Buttons enable connecting to, disconnecting from, or resetting the target.

Information about the connected component is displayed on the right.

The *RESET* button is used to reinitialize the Bluetooth® LE wireless stack. When many tests are performed, the button must be used to reset the stack at the start of each test.

CM0 and CM4 information for the Arm® Cortex®-M0+ and Cortex®-M4 cores only applies to STM32WB microcontrollers. For STM32WBA microcontrollers, the hardware and firmware versions are displayed, while only the firmware version is displayed for STM32WB0 microcontrollers.

1.3.3 Panels

The panels are used to perform specific operations. Each panel groups different functions. The *ACI Commands* panel example is illustrated in [Figure 5](#).

Figure 5. ACI Commands panel

ACI Commands | **Scripts** | **Beacon** | **RF Tests** | **ACI Utilities**

Command

☒ Select all
 ☒ HCI
 ☒ HCI test
 ☒ HAL
 ☒ GAP
 ☒ GATT
 ☒ L2CAP

HCI_DISCONNECT
 HCI_READ_REMOTE_VERSION_INFORMATION
 HCI_SET_EVENT_MASK
HCI_RESET
 HCI_READ_TRANSMIT_POWER_LEVEL
 HCI_READ_LOCAL_VERSION_INFORMATION
 HCI_READ_LOCAL_SUPPORTED_COMMANDS
 HCI_READ_LOCAL_SUPPORTED_FEATURES
 HCI_READ_BD_ADDR
 HCI_READ_RSSI
 HCI_LE_SET_EVENT_MASK
 HCI_LE_READ_BUFFER_SIZE
 HCI_LE_READ_LOCAL_SUPPORTED_FEATURE
 HCI_LE_SET_RANDOM_ADDRESS
 HCI_LE_SET_ADVERTISING_PARAMETERS
 HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER
 HCI_LE_SET_ADVERTISING_DATA

Command Parameters Details

Parameter	Value	Literal	Info
HCI packet indicator	0x01	HCI Command Packet	
Op_Code	0x0C03	HCI_RESET	
Parameter_Total_Length	0x00		

The main panels are *ACI Commands*, *Scripts*, *Beacon*, *RF Tests*, and *ACI Utilities*. Each panel is detailed in a specific section of the document:

- [Section 3.3: ACI Commands panel](#)
- [Section 3.5: Scripts](#)
- [Section 3.8: Beacon](#)
- [Section 3.4: RF test panel](#)
- [Section 3.9: ACI Utilities](#)

1.3.4 Log area

Figure 6. Log area

The screenshot shows a log interface with a header bar containing 'ACI log', 'Update' (checked), 'Autoscroll' (checked), and 'SAVE LOG' and 'RESET LOG' buttons. Below the header is a table of log entries. Entry 44, 'ACI_GATT_INIT' at time 14:57:35.712, is highlighted in yellow. To the right of this entry, an arrow points to the word 'Message'. Below the highlighted entry, a detailed view is shown with a table of parameters. An arrow points from the text 'Message detail' to this detailed view. The detailed view table has columns 'Parameter', 'Value', and 'Literal'. The parameters listed are 'HCI packet indicator' (0x01, HCI Command Packet), 'Op_Code' (0xFD01, ACI_GATT_INIT), and 'Parameter_Total_Length' (0x00). A '+ More' button is at the bottom right of the detailed view. The main log table continues with entries 45 through 49.

No	Time	Type
36	14:57:15.809	HCI_READ_LOCAL_VERSION_INFORMATION
37	14:57:15.812	Command Complete
38	14:57:15.814	VS_HCI_C1_DEVICE_INFORMATION
39	14:57:15.822	Command Complete
40	14:57:18.590	HCI_RESET
41	14:57:18.594	Command Complete
42	14:57:27.561	ACI_HAL_SET_TX_POWER_LEVEL
43	14:57:27.563	Command Complete
44	14:57:35.712	ACI_GATT_INIT
45	14:57:35.715	Command Complete
46	14:57:44.128	ACI_GAP_INIT
47	14:57:44.133	Command Complete
48	14:57:57.478	ACI_GATT_UPDATE_CHAR_VALUE
49	14:57:57.481	Command Complete

Parameter	Value	Literal
HCI packet indicator	0x01	HCI Command Packet
Op_Code	0xFD01	ACI_GATT_INIT
Parameter_Total_Length	0x00	

The log area shows the messages exchanged between the application and the connected devices. The list shows all message names and details. The log area is described in [Section 3.3.5](#).

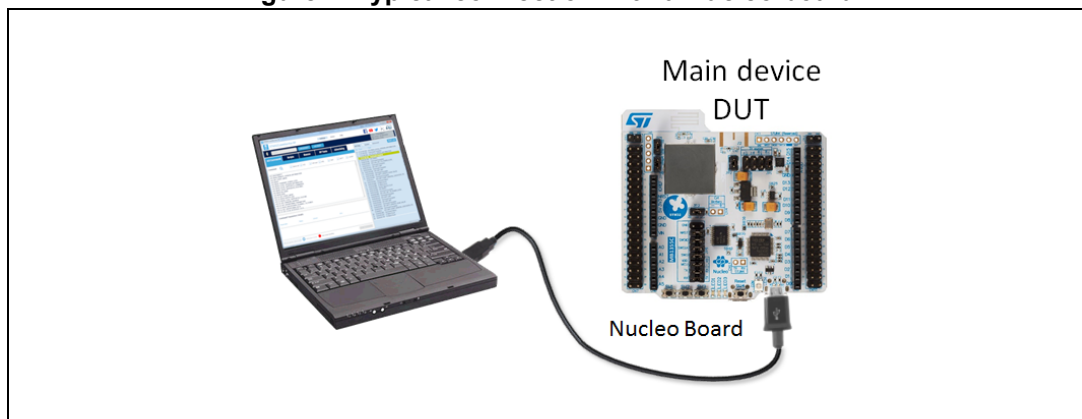
2 Connection to target

2.1 Use case descriptions and definitions

STM32CubeMonitor-RF usually connects to one STM32WB, STM32WBA, or STM32WB0 device. The connection is performed through a UART, either by a physical port or a Virtual COM port (VCP).

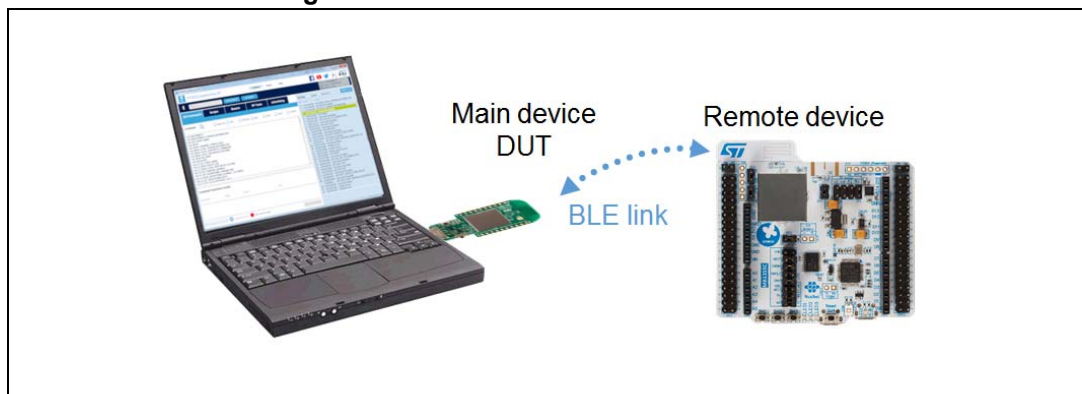
The device connected is usually named the *main device*. This is the board that the user wants to exercise with the tool. It is also called the device under test (DUT).

Figure 7. Typical connection with a Nucleo board



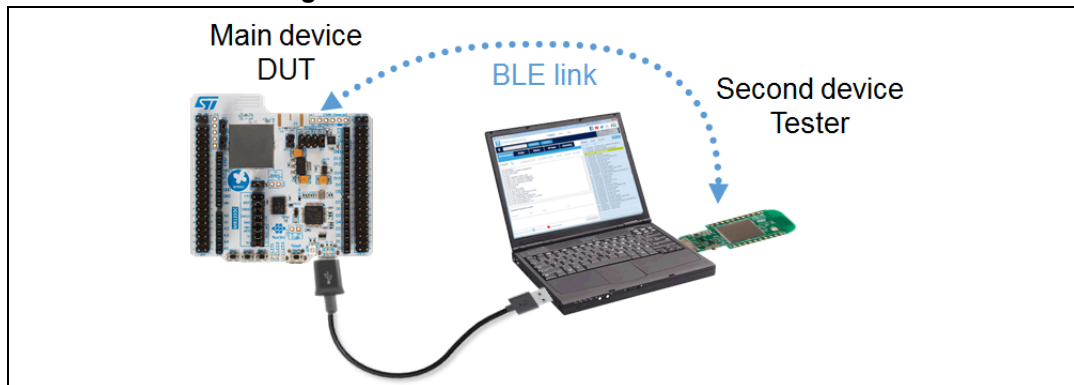
Some operations, like downloading over-the-air, involve communication with another device. This other device is referred to as the *remote device* in this document.

Figure 8. Connection with a remote device



One RF test uses two boards to measure the error rate of a packet transfer. For such a test, a second device is connected; it is named the *second device*. This latter device is the tester, the main device being the device to evaluate (DUT).

Figure 9. Connection with a second device



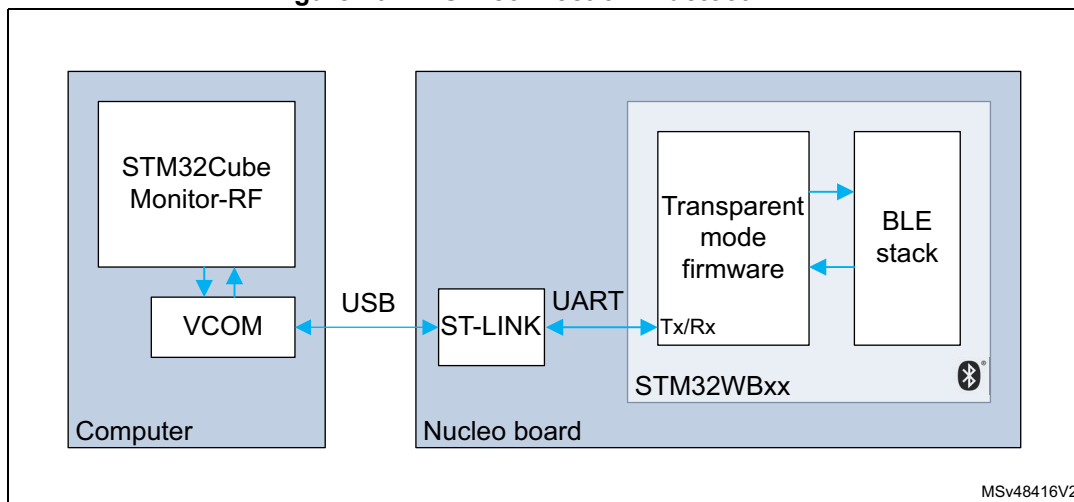
2.2 VCOM/UART connection

The connection must transfer the host controller interface (HCI) or command-line interface (CLI) commands between STM32CubeMonitor-RF and the wireless stack in the STM32WB, STM32WBA, or STM32WB0 device. HCI commands are used for Bluetooth® LE applications. CLI commands are used for Thread and 802.15.4 RF tests. The application opens a serial port (virtual or physical) and communicates with the target through this link. Many configurations are possible. The most common ones are described in this section.

2.2.1 VCOM connection

The connection with a Nucleo board uses a Virtual COM port and goes through ST-LINK. It is highly recommended to regularly update ST-LINK firmware to the latest available version.

Figure 10. VCOM connection Bluetooth® LE



The application opens the Virtual COM port and sends the data to the VCOM driver.

When a byte is sent, the VCOM transfers the data over USB to the ST-LINK embedded in the Nucleo board. The ST-LINK transfers the data on UART lines to the STM32WB, STM32WBA, or STM32WB0 microcontroller.

For Bluetooth® LE, a special firmware in the STM32WB, STM32WBA, or STM32WB0 microcontroller, called Transparent mode, copies the data received on the Rx pin to the Bluetooth® LE stack. Data sent back by the Bluetooth® LE stack follows the reverse path.

The Transparent mode firmware is available in the STM32CubeWB firmware package (refer to folder `\Projects\xxx\Applications\BLE\BLE_TransparentMode.`).

For the STM32WB devices, the wireless *stack* firmware *stm32wb5x_BLE_Stack_full_extended_fw.bin* is available in `\Projects\STM32WB_Copro_Wireless_Binaries`.

For information on Transparent Mode firmware loading for the STM32WB05xN network coprocessor, refer to the UM3406 user manual.

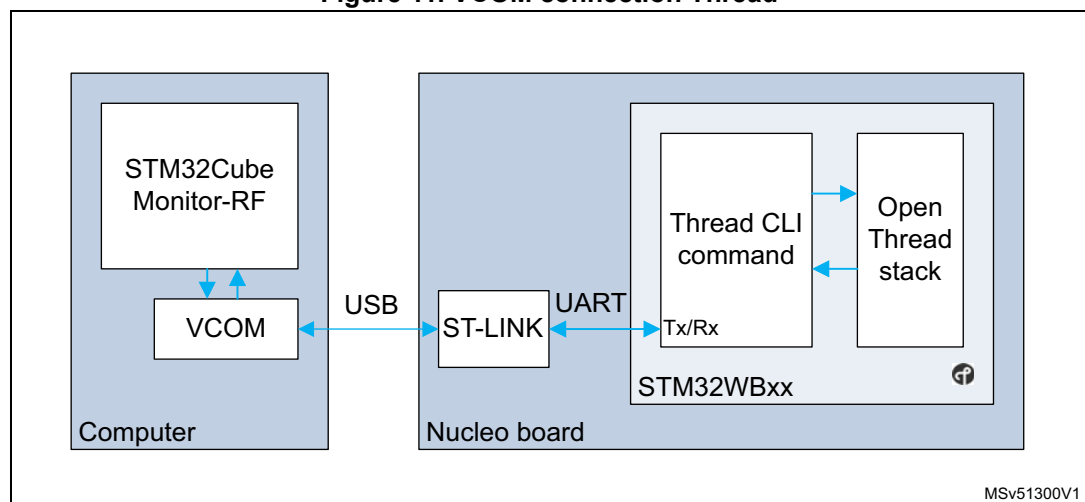
For more information, refer to [Section 3.2](#).

For Thread, the *Thread_Cli_cmd* firmware copies the data from the UART to the OpenThread command-line interpreter. Data sent back by the interpreter are forwarded to the UART.

For STM32WB devices, the CLI firmware source code is available in the STM32CubeWB firmware package (refer to the `|Projects|xxx|Applications|Thread|Thread_Cli_Cmd` folder). The wireless stack firmware *stm32wbxx_Thread_FTD_fw.bin* is available in `|Projects|STM32WB_Copro_Wireless_Binaries`.

For STM32WBA devices, the source is available in the STM32CubeWBA firmware package (refer to `|Projects|NUCLEO-WBAxxxx|Applications|Thread|Thread_Cli_Cmd` folder).

Figure 11. VCOM connection Thread



For 802.15.4 RF tests, the *Cli_Phy_802_15_4* firmware transfers the data from the UART to the 802.15.4 wireless stack. Data sent back by the stack follows the reverse path.

For STM32WB devices, the `Phy_802_15_4_Cli` source is available in the STM32CubeWB firmware package (refer to the `\Projects\P-NUCLEO-WB55.Nucleo\Applications\Phy_802_15_4\Phy_802_15_4_Cli` folder). The wireless stack firmware `stm32wb5x_Phy_802_15_4_fw.bin` is available in the `\Projects\STM32WB_Copro_Wireless_Binaries` folder.

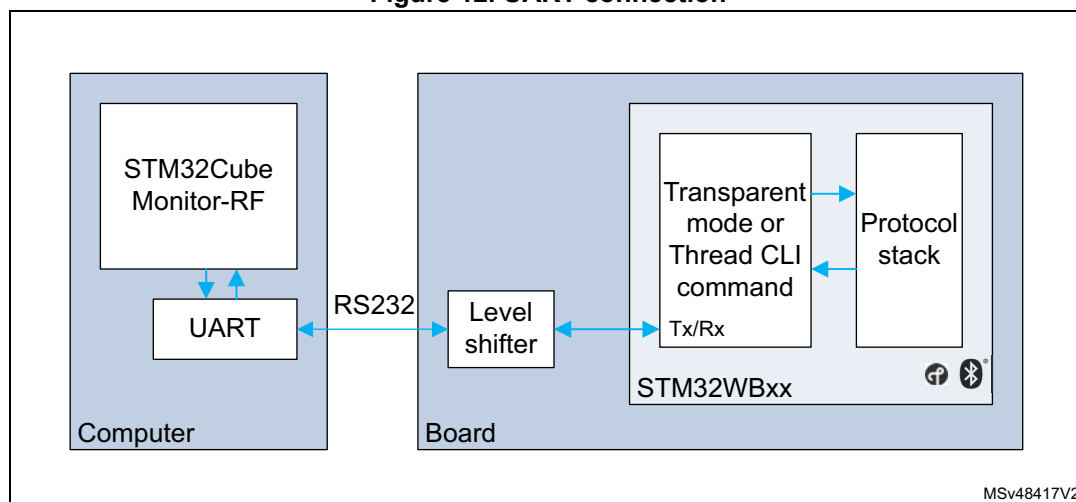
For STM32WBA devices, the source is available in the STM32CubeWBA firmware package (`\Projects\NUCLEO-WBAxxx\Applications\Phy_802_15_4\Phy_Cli` folder).

When a USB-to-serial converter replaces the ST-LINK part, the VCOM driver might be installed automatically on the computer. For the converter without an automatic driver setup, the user must install the VCOM driver manually.

2.2.2 UART connection

It is possible to use a physical UART link to connect directly to any board.

Figure 12. UART connection



In this case, data are sent directly in serial mode through the level shifter. Refer to the Transparent mode or CLI command release note for UART configuration.

The UART connection can be used to connect an STM32WB55 USB dongle for 802.15.4 RF tests.

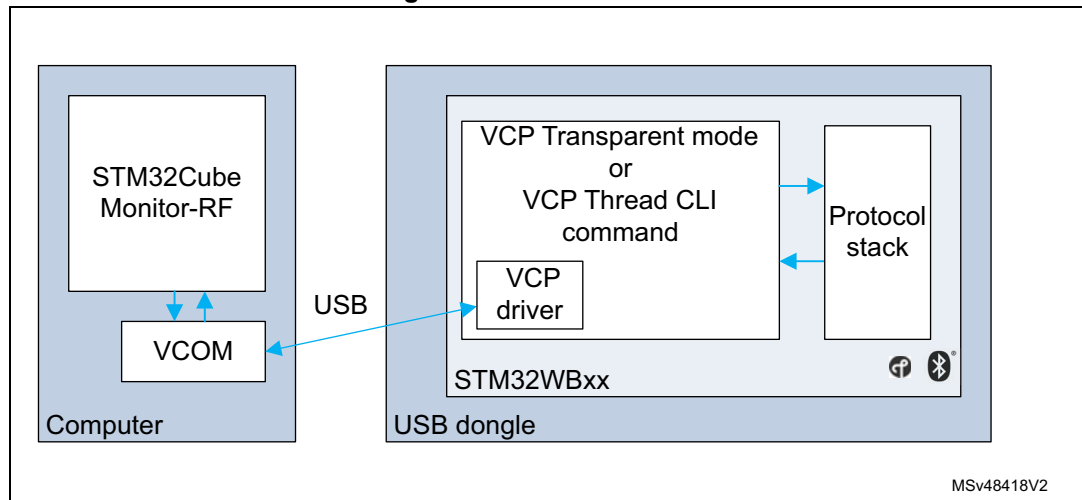
To configure the USB dongle for the 802.15.4 test:

1. Build the Nucleo firmware `Phy_802_15_4_Cli` and flash it.
(`STM32Cube_FW_WB_Vx.x.x\Projects\P-NUCLEO-WB55.Nucleo\Applications\Phy_802_15_4\Phy_802_15_4_Cli`)
2. Flash with DFU the wireless stack `stm32wb5x_Phy_802_15_4_fw.bin` in the dongle (binary in `Projects\STM32WB_Copro_Wireless_Binaries`).
3. Move solder bridge SB2 to SB6 (connection of PB7 to CN2.7).
4. Connect the serial cable to PB7 (PC Tx) and PB6 (PC Rx), PB7 is on CN2.7 and PB6 on CN2.6.

2.2.3 VCP device

In this case, no UART is involved. The data goes directly from the computer to the microcontroller through the USB.

Figure 13. VCP connection



A special VCP firmware is used. It implements a VCP driver to copy the data from the USB port to the protocol stack. The VCOM driver might be installed automatically on the computer or the user needs to install it manually. This configuration is used with the STM32WB55 USB dongle reference board and the Nucleo sniffer configuration.

1. For Bluetooth® LE:
The firmware is in
`\Projects\NUCLEO WB55.USB Dongle\Applications\BLE\BLE_TransparentModeVCP`.
The wireless stack is in:
`\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x\stm32wb5x_BLE_Stack_full_extended_fw.bin`
For more information, refer to [Section 3.2](#).
2. For Thread:
The firmware source code is in
`\Projects\NUCLEO WB55.USB Dongle\Applications\Thread\Thread_Cli_Cmd`.
The wireless stack is in the
`\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x\stm32wb5x_Thread_FT_D_fw.bin` folder.
3. For the 802.15.4 sniffer:
The wireless stack is in the
`\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x\stm32wb5x_Mac_802_15_4_fw.bin` folder. The firmware binaries are `Mac_802_15_4_Sniffer_Dongle.bin` and `Mac_802_15_4_Sniffer_Nucleo.bin`. The location changes with the operating system.
 - a) For Windows®, the firmware is in the
`<Public documents>\STMicroelectronics\STM32CubeMonitor-RF\sniffer` folder,

which means `C:\Users\Public\Documents\STMicroelectronics\STM32CubeMonitor-RF\firmwares`.

- b) For Linux®, the firmware is in the `<userhome>/STMicroelectronics/STM32CubeMonitor-RF/sniffer folder`.
- c) For macOS®, the firmware is inside the document folder provided in the setup package: `\Users\Public\Documents\STMicroelectronics\STM32CubeMonitor-RF\firmwares\Mac_802_15_4_Sniffer.bin`

2.3 Opening COM

To start using the application, connect it to the device under test in the connection bar.

Figure 14. Opening COM



The procedure is:

- Connect the board to the computer. If VCOM or VCP is used, a driver must be installed. It might take a few seconds at the first connection. For some devices, drivers need to be installed manually.
- Select the serial port to use in the picklist (*Comx* on Windows® and *ttyACMx* on Linux® and Mac®). On a Mac® computer, the port *cu.Bluetooth-Incoming-Port* is the Mac Bluetooth® adapter. This port is not connected to the STM32 device and must not be used).
- Select the serial baud rate in the picklist. This picklist is only accessible in the Bluetooth® mode. 921600 is recommended for STM32WB0 devices while 115200 must be used for other STM32WB or STM32WBA devices.
- Click *CONNECT*

The board is connected, and the version is displayed on the right side of the bar.

Figure 15. Successful COM



When the *CONNECT* button is pressed, the software attempts to communicate with the device to read the firmware and hardware versions. If the connection is not working, the tool displays an error and disconnects the COM port.

Caution: In the case of a connection error, the user must check these points:

- At first board connection, driver loading might take some time, or the driver might not install automatically. If the tool is not showing the COM port in the list, check that the drivers are properly installed.
- When a user tries to open a device with the wrong mode or an unsupported baud rate, for example, open a Bluetooth® LE device in Thread mode, the device cannot decode the command sent and might freeze or crash the target software. In this case, it is necessary to unplug/replug the board to reset it.
- Delay on Ubuntu®(a):
 - On Ubuntu, the *modemmanager* process checks the COM port when the board is plugged in. Due to this activity, the COM port is busy for a few seconds, and STM32CubeMonitor-RF cannot connect.
 - The user must wait for the end of the *modemmanager* activity before opening the COM port.
 - If the user does not need *modemmanager*, it can be uninstalled with *sudo apt-get purge modemmanager*.
- Port not visible or connectable on Linux®:
 - The user might not have the proper access rights for ttyACM. Ubuntu requires adding the user to the dial-out group with the *sudo adduser <username> dialout* command (replace username with user name).
- If another application opens the port, the tool is unable to connect.
- When a USB device is removed, the Virtual COM port is not closed automatically, and the software might not be informed of the disconnection. If a USB device is inserted when the Virtual COM port is already opened, the board is not mounted in the system. To solve this, close the COM port on STM32CubeMonitor-RF, disconnect the USB cable, and reinsert it. In some rare cases, it is mandatory to enable or disable the COM port in the OS device manager.

a. Ubuntu is a registered trademark of Canonical Ltd.

3 Bluetooth® LE mode

3.1 Presentation

3.1.1 Panels

The panels are used to perform a specific operation. Each panel regroupes different functions, as [Figure 16](#) shows when the *ACI Commands* panel is selected.

Figure 16. ACI Commands panel

ACI CommandsScriptsBeaconRF TestsACI Utilities

Command

☒ Select all☒ HCI☒ HCI test☒ HAL☒ GAP☒ GATT☒ L2CAP

Search...

HCI_DISCONNECT

HCI_READ_REMOTE_VERSION_INFORMATION

HCI_SET_EVENT_MASK

HCI_RESET

HCI_READ_TRANSMIT_POWER_LEVEL

HCI_READ_LOCAL_VERSION_INFORMATION

HCI_READ_LOCAL_SUPPORTED_COMMANDS

HCI_READ_LOCAL_SUPPORTED_FEATURES

HCI_READ_BD_ADDR

HCI_READ_RSSI

HCI_LE_SET_EVENT_MASK

HCI_LE_READ_BUFFER_SIZE

HCI_LE_READ_LOCAL_SUPPORTED_FEATURE

HCI_LE_SET_RANDOM_ADDRESS

HCI_LE_SET_ADVERTISING_PARAMETERS

HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER

HCI_LE_SET_ADVERTISING_DATA

HCI_LE_SET_SCAN_RESPONSE_DATA

Command Parameters Details

Parameter	Value	Literal	Info
HCI packet indicator	0x01	HCI Command Packet	
Op_Code	0x0C2D	HCI_READ_TRANSMIT_POWER_LE...	
Parameter_Total_Length	0x03		
Connection_Handle	0x002A		Specifies which Connection_Handle's Tr...
Type	0x00		0x00: Read Current Transmit Power Lev...

Script pause value (ms)

Add pause in script

Start script recording

SEND COMMAND

The main panels are *ACI Commands*, *Scripts*, *Beacon*, *RF Tests*, and *ACI Utilities*. They are detailed in the next section.

3.2 Bluetooth® LE stack

From STM32Cube_FW_WB_V1.14.0, there are some STM32WB_Copro_Wireless_Binaries variants. STM32CubeMonitor-RF needs to use *stm32wb5x_BLE_Stack_full_extended_fw.bin*.

For more information, refer to

https://github.com/STMicroelectronics/STM32CubeWB/tree/master/Projects/STM32WB_Copro_Wireless_Binaries.

For the STM32WBA devices, there exist different variants for the Bluetooth® LE stack, such as *Full*, *Basic*, *Link Layer Only*, and *Link Layer Only Basic*. Note that the default *BLE_TransparentMode.bin* firmware is based on the Basic feature variant and does not support all commands. A new build is required to support extra commands such as extended advertising. Refer to *Middlewares\ST\STM32_WPAN\ble\stack\doc\STM32WBA_BLE_Stack_User_Manual.html* for more details.

3.3 ACI Commands panel

The application command interface (ACI) panel is used to send commands to the main device Bluetooth® LE stack. Categories group commands. These commands allow the user to configure the Bluetooth® LE stack and activate communication with remote devices.

3.3.1 How to send an ACI command

Figure 17. How to send an ACI command

The screenshot shows a software interface for sending Bluetooth LE commands. At the top, there's a 'Command' section with a list of commands. A search bar is on the right. Below the list, the 'Command Parameters Details' section is visible, showing parameters like 'Hci packet indicator', 'Op_Code', 'Parameter_Total_Length', 'Connection_Handle', and 'Type'. A 'SEND COMMAND' button is at the bottom right.

Command

☒ Select all ☒ HCI ☒ HCI test ☒ HAL ☒ GAP ☒ GATT ☒ L2CAP Search...

HCI_DISCONNECT
 HCI_READ_REMOTE_VERSION_INFORMATION
 HCI_SET_EVENT_MASK
 HCI_RESET
HCI_READ_TRANSMIT_POWER_LEVEL
 HCI_READ_LOCAL_VERSION_INFORMATION
 HCI_READ_LOCAL_SUPPORTED_COMMANDS
 HCI_READ_LOCAL_SUPPORTED_FEATURES
 HCI_READ_BD_ADDR
 HCI_READ_RSSI
 HCI_LE_SET_EVENT_MASK
 HCI_LE_READ_BUFFER_SIZE
 HCI_LE_READ_LOCAL_SUPPORTED_FEATURE
 HCI_LE_SET_RANDOM_ADDRESS
 HCI_LE_SET_ADVERTISING_PARAMETERS
 HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER
 HCI_LE_SET_ADVERTISING_DATA
 HCI_LE_SET_SCAN_RESPONSE_DATA

Command Parameters Details

Parameter	Value	Literal	Info
Hci packet indicator	0x01	HCI Command Packet	
Op_Code	0x0C2D	HCI_READ_TRANSMIT_POWER_LE...	
Parameter_Total_Length	0x03		
Connection_Handle	0x002A		Specifies which Connection_Handle's Tr...
Type	0x00		0x00: Read Current Transmit Power Lev...

Script pause value (ms) 0x01 Start script recording **SEND COMMAND**

Before sending any command to the main device, the device must be connected.

To send an ACI command:

1. Select a command name in the command list (for example HCI_READ_TRANSMIT_POWER_LEVEL).
The command parameters are displayed in the *Command Parameters Details* area.
2. Fill in the parameters of the command. Default values are used otherwise.
3. Click on **SEND COMMAND**. The command is sent to the main device.

3.3.2 Search function

The search icon is used quickly to select a command in the list:

- Click on the magnifier icon. A text box is created
- Type the name to search. As soon as a character is entered, matching commands are filtered in the list. The match might be any part of the command name. It is not necessary to start from the beginning.
- Click once on the command to select it (do not use double-click).

Figure 18. Search button

ACI Commands

Scripts

Beacon

RF Tests

ACI Utilities

Command

☒ Select all

☒ HCI

☒ HCI test

☒ HAL

☒ GAP

☒ GATT

☒ L2CAP

local

X

HCI_READ_LOCAL_VERSION_INFORMATION
HCI_READ_LOCAL_SUPPORTED_COMMANDS
HCI_READ_LOCAL_SUPPORTED_FEATURES
HCI_LE_READ_LOCAL_SUPPORTED_FEATURE
HCI_LE_READ_LOCAL_P256_PUBLIC_KEY
HCI_LE_READ_LOCAL_RESOLVABLE_ADDRESS

3.3.3 Filter usage

This features group and name the commands. The groups are:

- HCI
- HCI test
- HAL
- GATT
- GAP
- L2CAP

The picklist at the top of the area allows seeing only some groups to find the commands more easily. Click on *Select all* to see all the commands in the list.

3.3.4 How to fill parameters. Fixed field/editable field

Some parameters have fixed values and are not editable, while others are free or take only some values. The tool guides the user to fill in the parameters:

- Fixed parameter: this parameter is not editable. The specification or logic defines the value. This applies to the *length* value automatically computed by the tool.

Figure 19. Fixed parameter

Parameter_Total_Len... 0x03

- Editable parameter: the editable parameter is surrounded by a blue rounded box. The value is editable inside the field. Edit is blocked if the value is too long for the field.

Figure 20. Editable parameter

Connection_Handle 0x002A

- Predefined values: when the choice is limited, a picklist is displayed to help the user select the values.

Figure 21. Predefined values

Type 0x01

For some parameters, some help is available in the column *Info*. To see the help details, put the pointer on the wanted parameter info, and a bubble displays the details.

Figure 22. Help details



3.3.5 Log functionalities

The log area is on the right part of the screen. It displays the messages exchanged with the boards.

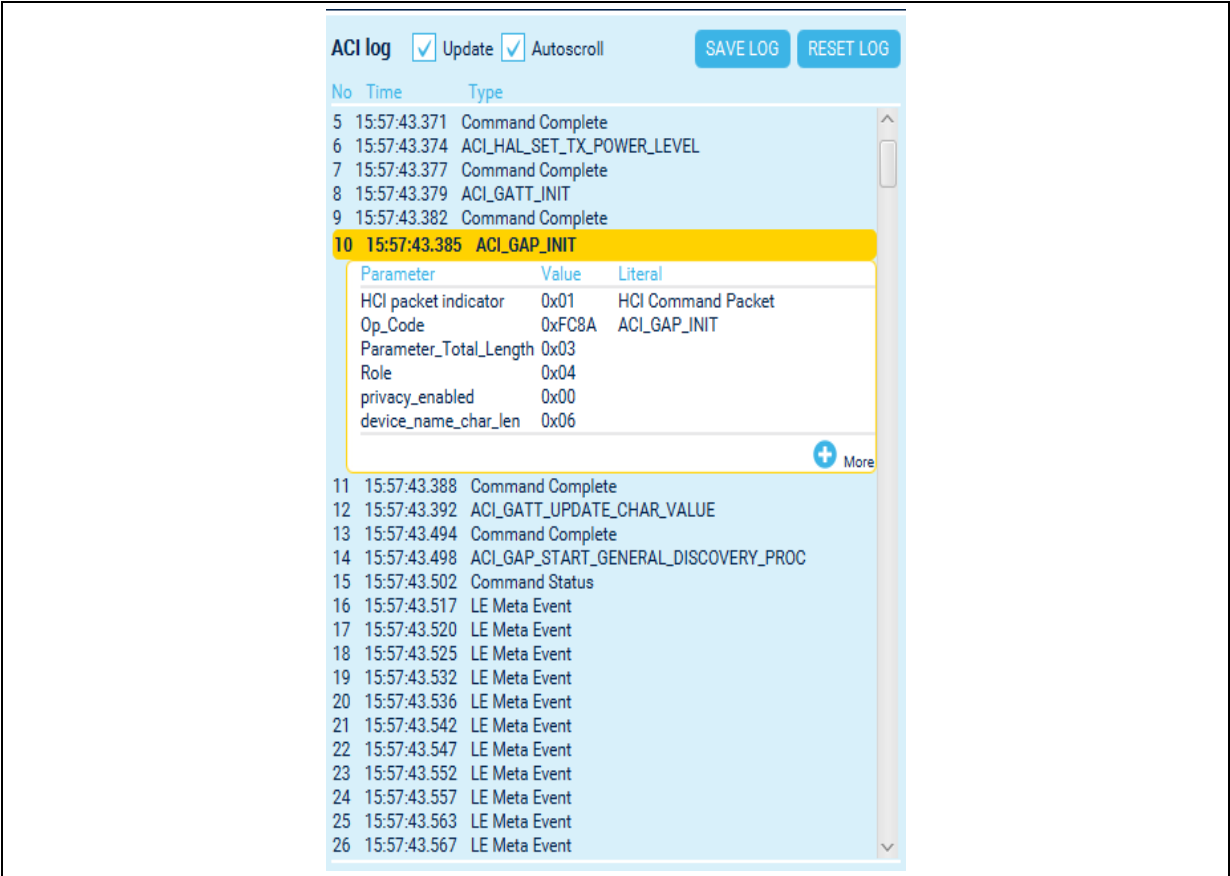
When a command is sent, most of the time an immediate answer comes from the board. It is a *Command Status* or a *Command Complete*.

The commands with *Command Status* usually have other events coming later. These events are also displayed in the log area.

Some asynchronous events might come from the device and be displayed in this area.

The tool keeps the last 1000 lines. When the limit is reached, the oldest lines are automatically discarded.

Figure 23. Log functionalities



The user can scroll through the list using the scroll bar on the right-hand side.

When a line is selected, the content of the selected message is displayed in the green area, with one line for each parameter.

The text ends with ... when it is not possible to display the complete text. It is possible to change the log area width to display longer texts.

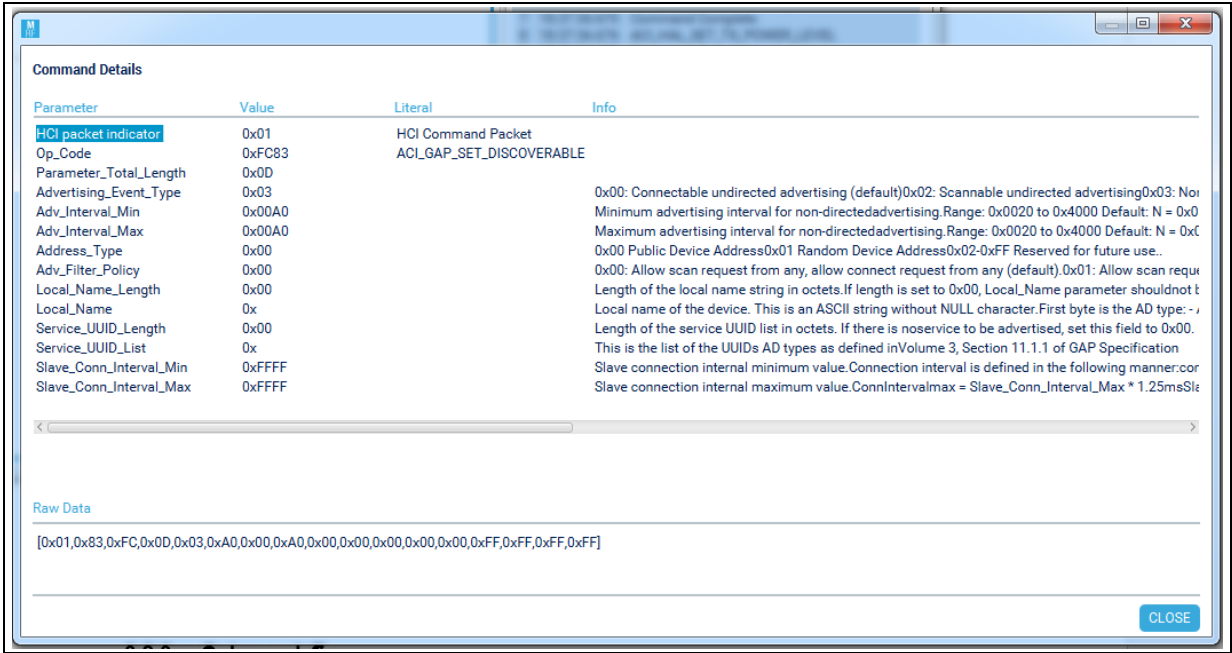
Details

Figure 24. More button



Sometimes, all the information in a message does not fit in the area used for the log. The button shown in [Figure 24](#) opens a new window showing the message details:

Figure 25. Message details



The details show all decoded message parameters. The *Literal* column shows a predefined text for the parameter values (opcode and others). The *Info* column provides some description of the parameter content.

The raw data in the bottom part is the data sent/received over UART, without decoding.

In this window, it is possible to copy information to paste it into other windows.

An efficient solution to compare two messages is to open multiple detail windows at the same time.

Color code

The logs use a color code to identify the device used and highlight errors.

A line with purple text shows that the status in the message is different from zero, which indicates an error.

Figure 26. Purple error messages

No	Time	Type
8	09:25:32.205	HCI_LE_CREATE_CONNECTION
9	09:25:32.212	Command Status

A log on a dark gray background comes from a second board. When the two boards are connected, the main device (DUT) has a normal color log while the second device tester has a darker background. This is helpful to understand the sequences involving the two devices.

Figure 27. Gray second board messages

No	Time	Type
14	09:27:34.029	HCI_READ_LOCAL_VERSION_INFORMATION
15	09:27:34.036	Command Complete
16	09:27:37.634	HCI_READ_LOCAL_VERSION_INFORMATION
17	09:27:37.672	Command Complete

Update button

When the *Update* tick box is not selected, the messages are not added to the log area. The line number continues to increase anyway but is not displayed until the *Update* tick box is enabled.

Auto-scroll

When the *Auto-scroll* box is ticked, the log area always displays the last log received. To check the log history, untick the box that disables the auto-scroll.

Reset log

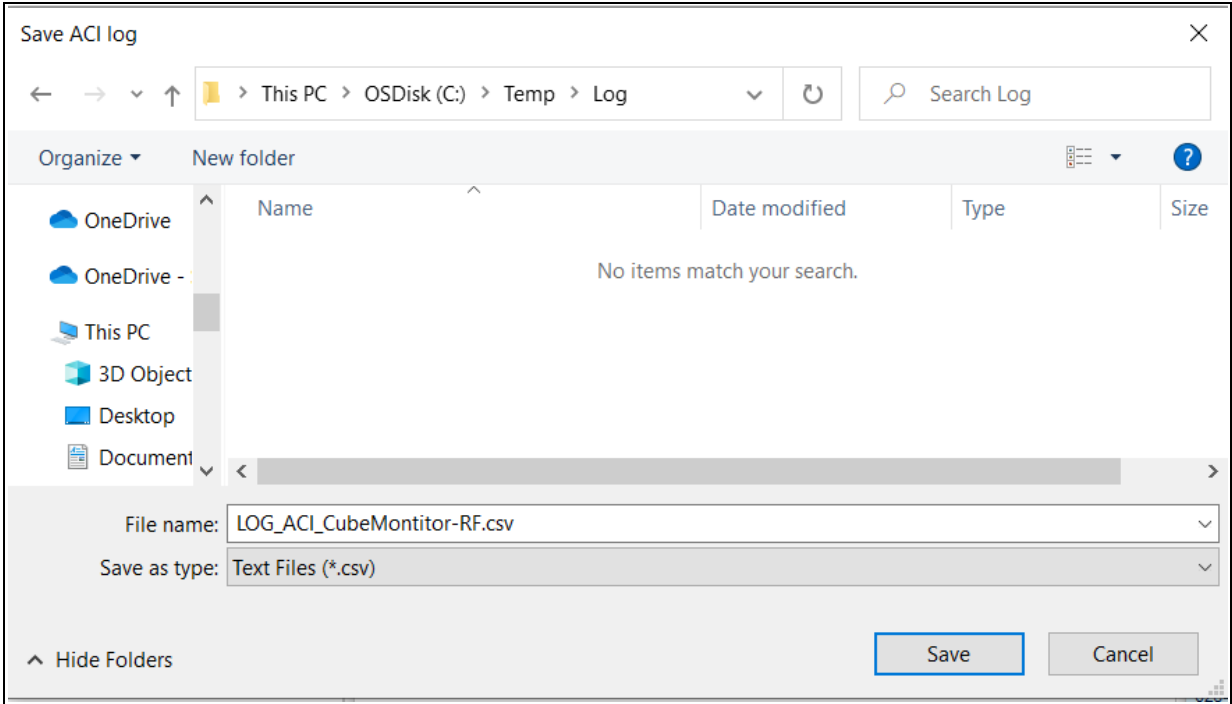
The *RESET LOG* button allows wiping the log displayed in the log area. The line number is not affected, but the memory used by older logs is made free.

The *RESET LOG* button resets the ACI log buffer used for log saving executed by the *SAVE LOG* action described just after

Save log

The *SAVE LOG* button allows saving the ACI log text file in csv format.

Figure 28. Save ACI log



In the csv file, there is one line per parameter. Then there are several lines per type of command. The number, the time, as well as the type fields are repeated for each parameter. Refer to [Figure 29](#) as an example.

Figure 29. LOG_ACI_CubeMonitor-RF.csv

A		B	C	D	F		G	
Nb	Time	PortCom	Type	Parameter	Value	Literal	Information	
1	39:11.6	COM9	Command Complete	HCI packet indicator	0x04	HCI Event Packet	<div>Aci Log detail in text file</div>	
6	1	39:11.6	COM9	Command Complete	Event_Code	0x0C		Command Complete
7	1	39:11.6	COM9	Command Complete	Parameter_Total_Length	0x0C		
8	1	39:11.6	COM9	Command Complete	Num_HCI_Command_Packets	0x01		The number of HCI command packets which are allowed to be sent to the controller from the host. Range for N:0-255.
9	1	39:11.6	COM9	Command Complete	Command_Opcode	0x1001		Opcode of this command which caused this event.
10	1	39:11.6	COM9	Command Complete	Status	0x00		SUCCESS
11	1	39:11.6	COM9	Command Complete	HCI_Version	0x0C		https://www.bluetooth.com/specifications/assigned-numbers/host-controller-interface
12	1	39:11.6	COM9	Command Complete	HCI_Revision	0x0070		Revision of the Current HCI in the BR/EDR Controller
13	1	39:11.6	COM9	Command Complete	LMP/PAL_Version	0x0C		Version of the Current LMP or PAL in the Controller - see: https://www.bluetooth.com/specifications/assigned-numbers/link-manager
14	1	39:11.6	COM9	Command Complete	Manufacturer_Name	0x0030		Manufacturer Name of the BR/EDR Controller - see: https://www.bluetooth.com/specifications/assigned-numbers/Company-identifiers
15	1	39:11.6	COM9	Command Complete	LMP/PAL_Subversion	0x2170		Subversion of the Current LMP or PAL in the Controller
16	1	39:11.6	COM9	Command Complete	Raw	[0x04:0x0E:0x0C:0x01:0x10:0x00:0x0C:0x70:0x00:0x0C:0x30:0x00:0x70:0x21]		
17	2	39:11.6	COM9	V5_HCI_C1_DEVICE_INFORMATION HCI packet indicator	0x20	HCI M4 Command Packet		
18	2	39:11.6	COM9	V5_HCI_C1_DEVICE_INFORMATION Op Code	0xF062	V5_HCI_C1_DEVICE_INFORMATION		
19	2	39:11.6	COM9	V5_HCI_C1_DEVICE_INFORMATION Parameter_Total_Length	0x00			
20	2	39:11.6	COM9	V5_HCI_C1_DEVICE_INFORMATION Raw	[0x20:0xF062:0xF0:0x00]			
21	3	39:11.6	COM9	Command Complete	HCI packet indicator	0x21	HCI M4 Event Packet	

The **SAVE LOG** button does not reset the buffer. A further **SAVE LOG** action saves not only the ACI log generated from the last **SAVE LOG** action but also the anterior-generated ACI log.


The buffer is erased only on the **Reset log** action.

3.4 RF test panel

The RF test panel is used to perform the radio frequency tests on the main device. The RF tests are grouped into three test modes: Transmitter (TX), Receiver (RX), and Packet error rate (PER):

- The TX test is dedicated to radio frequency emission, for tones and packets.
- The RX test is for packet reception.
- The PER test is a quality transmission test between two devices.

Figure 30. Test mode selection



The screenshot shows a software interface with a top navigation bar containing five tabs: 'ACI Commands', 'Scripts', 'Beacon', 'RF Tests' (which is highlighted), and 'ACI Utilities'. Below the tabs is a section titled 'Test mode' with a blue header. Underneath, there is a 'Select test mode' section with three radio button options: 'Transmitter test (TX)' (which is selected), 'Packet error test (PER)', and 'Receiver test (RX)'. At the bottom right of the panel is a blue button labeled 'SELECT TEST MODE'.

The first action after connecting a device is to select the mode to test and then click on the *SELECT TEST MODE* button.

When the user selects a test mode, it is mandatory to go back to the selection page to change the test mode:

- Click on the *Change test mode*


Figure 31. Change the test mode



The screenshot shows a single button with a blue circular icon containing a white left-pointing arrow and the text 'Back' to its right.

- Click on *test mode* in the top bar.

Figure 32. Select the test mode



The screenshot shows a blue button with the text 'Test mode > Transmitter (TX)'. A mouse cursor is pointing at the button.

Note: To avoid incorrect configuration of the device, the test mode is unchangeable when transmission or reception is ongoing. The user must first stop the transmission and then change the test mode.

3.4.1 Transmitter test mode (TX)

The TX mode is used to set the Bluetooth® LE transmitter in emission. Two transmission modes are defined: *transmission of data*, or *emission of tone*.

Figure 33. Transmitter test mode

Test mode > Transmitter (TX)

Transmitter

PA Level: 31 (+6dBm)

TX Frequency: 2402 MHz (Channel 37)

Length of Data: 0x25

Packet Payload: 0x00 - Pseudo-Random bit sequence 9

PHY: 0x01 - Transmitter set to use the LE 1M PHY

Back START TONE START TX

Test measurement

Transmitted packets count:

Received packets count:

Packet Error Rate (PER):

RSSI:

Tone generation

The tone generation performs the emission of a continuous sinus wave on the RF. The parameters for the tone are tone power level and tone frequency. The power level is the power at the chip output.

To start tone generation:

1. Enter the Transmitter panel test mode.
2. Select the power level with the picklist.
3. Select the frequency with the TX frequency picklist. The list is sorted by frequency; the data/advertising channel index is indicated in parentheses. The advertising channel index does not follow the frequency order. Channels 37, 38, and 39 are the advertising

channels. Refer to the BLUETOOTH SPECIFICATION version 4.2 [Vol 6 part B] chapter 1.4.1 for details.

4. Select the PHY modulation to use (the unsupported modulations by the device are not listed).
5. Click on the *START TONE* button.

The emission starts, the *START TONE* button is changed to *STOP TONE*, and *Transmitting information* is displayed.

Figure 34. Transmitting message



6. To stop the tone generation, click on *STOP TONE*, and the emission stops.

It is mandatory to stop transmission to change to another test mode.

Packet transmission

It is possible to send some data packets in test mode. The parameters are power level, transmission frequency, length, and content of the data to send.

Power and level parameters are the same as tone parameters.

The packet data is selected in the *Packet payload* picklist. Eight types of payloads are available:

- A pseudo-random bit sequence 9 (PRBS9)
- A pattern of alternating bits *0b11110000*
- A pattern of alternating bits *0b10101010*
- A pseudo-random bit sequence 15 (PRBS15)
- A pattern of fixed bits *0b11111111*
- A pattern of fixed bits *0b00000000*
- A pattern of alternating bits *0b00001111*
- A pattern of alternating bits *0b01010101*

The *Length of data* picklist defines the sequence length. This is the length of the data payload in bytes. The PHY box is used to select the modulation.

To start packet emission:

- Select the power level with the picklist.
 - Select the frequency with the TX frequency picklist.
 - Select the length of the packet to send
 - Select the content of the packet payload
 - Click on *START TX*
- The emission starts, the start button is changed to *STOP TX*, and *Transmitting* is displayed. The sequence is repeated until the test is stopped.

To stop the transmission, click on *STOP TX*. The number of packets transmitted during the test is displayed in the test measurement area.

Figure 35. Transmitted packets count

Transmitted packets count	1724
---------------------------	------

If the number of packets received by the reception device is known, it is manually entered in the *Received Packet Number* box. The *Packet Error Rate* is automatically computed (refer to [Section 3.4.3: PER](#) for details).

3.4.2 Receiver test mode (RX)

The Receiver mode is used to put the main device in Reception mode and count packets received.

Figure 36. Receiver test mode

Test mode > Receiver (RX)

Receiver

RX Frequency
2402 MHz (Channel 37)

PHY
0x01 - Receiver set to use the LE 1M PHY

Index modulation
0x00 - Assume transmitter will have a standard modulation index

☐ Get RSSI

Back
START RX

Packets reception

- Select the frequency to use.
- Select the PHY and the modulation index to be used.
- Click on *START RX*. The reception starts, *Receiving* is displayed with an animation, and the button changes to *STOP RX*.

To stop reception, click on *STOP RX*. The count of received packets is retrieved from the main device and displayed in the *Received packet number*.

If the number of transmitted packets is known, it might be entered manually in the *Transmitted packet number*. The *Packet Error Rate (PER)* is automatically computed (refer to [Section 3.4.3](#) for details).

If the *Get RSSI* checkbox is selected, the tool performs an RSSI measurement.

RSSI measurement

The RSSI indicates the signal level received by the RF. The value reported by the RF is not an absolute value because the reception level is dependent on the board layout and antenna design. Note that only the STM32WB devices support RSSI measurement.

When the RSSI option is selected, the user must define the measurement interval. The default value is 3 seconds. The RSSI value is displayed at the end of each measurement period.

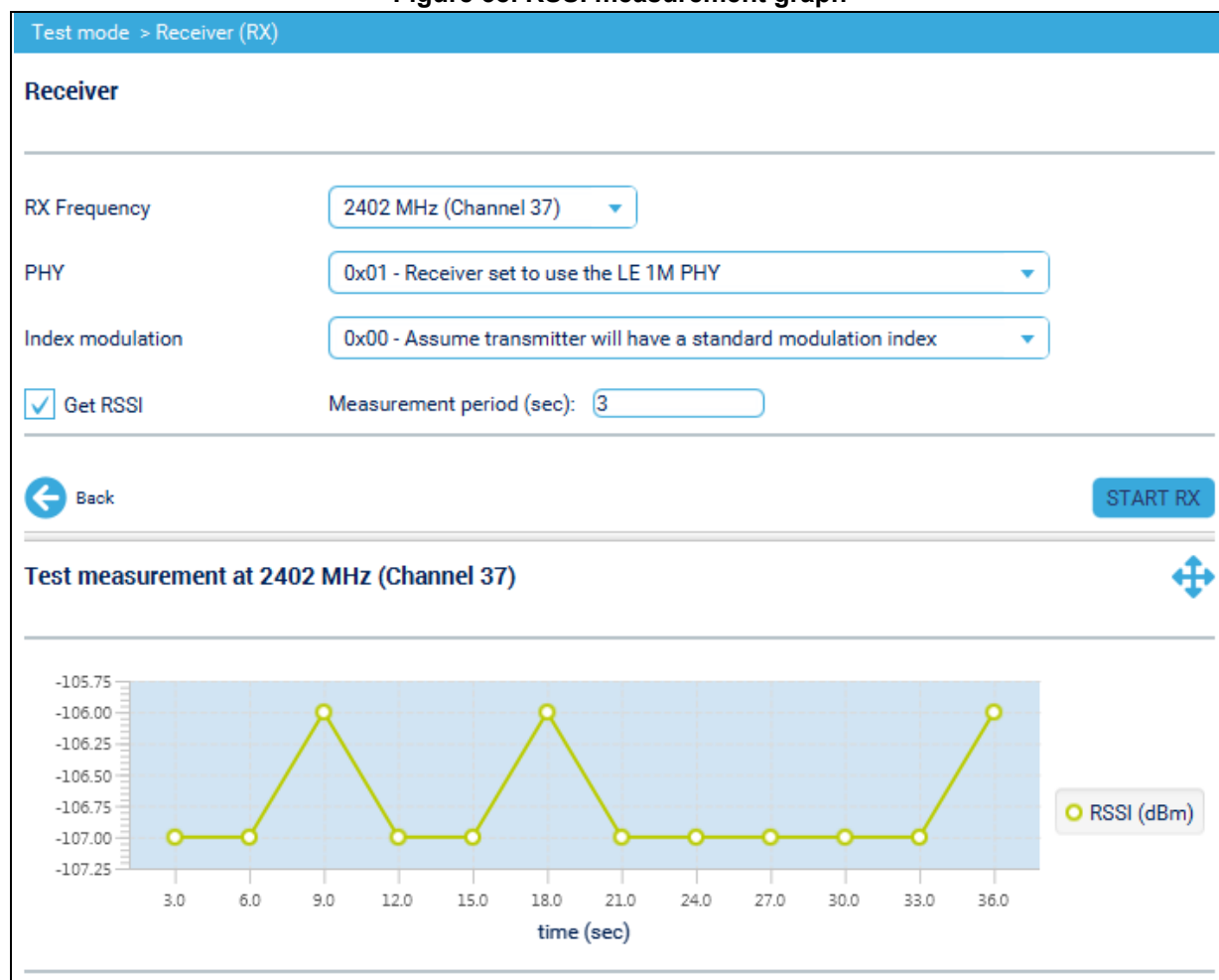
It is possible to switch between detailed value, plot view, and display, with the blue button on the right (bar chart, arrows, or blue lines).

Figure 37. RSSI measurement

The screenshot shows the 'Test mode > Receiver (RX)' interface. Under the 'Receiver' section, there are settings for 'RX Frequency' (2402 MHz (Channel 37)), 'PHY' (0x01 - Receiver set to use the LE 1M PHY), and 'Index modulation' (0x00 - Assume transmitter will have a standard modulation index). A checkbox 'Get RSSI' is checked, and the 'Measurement period (sec)' is set to 3. A 'STOP RX' button is visible. Below this, the 'Test measurement at 2402 MHz (Channel 37)' section shows 'Transmitted packets count' and 'Received packets count' fields, both empty. A 'Receiving...' status indicator with a circular progress bar is shown. The 'Packet Error Rate (PER):' field is empty, and the 'RSSI' field displays '-107.00 dBm'. A small bar chart icon is located to the right of the measurement title.

Note: When the RSSI measurement is performed, the number of received packets is not available in the tool. When the measurement is stopped, the Received packet number field is cleared, and an information message is displayed.

Figure 38. RSSI measurement graph



Note: The graph length is limited to 250 points. When the limit is reached, the oldest points are discarded.

Figure 39. RF RSSI measurement large display

Test mode > Receiver (RX)

Receiver

RX Frequency: 2402 MHz (Channel 37) ▼

PHY: 0x01 - Receiver set to use the LE 1M PHY ▼

Index modulation: 0x00 - Assume transmitter will have a standard modulation index ▼

☒ Get RSSI Measurement period (sec): 3

← Back START RX

Test measurement at 2402 MHz (Channel 37) ☰

RSSI

-62.00 dBm

3.4.3 PER

PER definition

The packet error rate (PER) is an indicator of the quality of transmission between the two devices. The measurement proposed in the tool covers the whole transmission chain from the transmitter to the receiver.

The packet error rate is computed with the number of packets sent and the number of packets received. A good transmission gives a low PER. A high PER means that the transmission is not good.

Figure 40. PER definition

$$\text{PER} = 100 \times \frac{N_{\text{tx}} - N_{\text{rx}}}{N_{\text{tx}}} \%$$

Ntx: Number of packets sent. Nrx: Number of packets received. PER: The result is in percent.

A bad PER might be an issue from the transmitter or the receiver and depends on parameters like the distance between devices, antennas, PCB design, and interferences. To limit the parameters influencing the measurements, it is advised to use one reference board with well-known performances in the setup.

PER test mode

The tool provides a special test mode dedicated to the PER test. In this mode, two devices need to be connected to the computer:

- The first device under test (DUT)
- The second device acts as a packet generator (tester)

After the connection of the DUT (main device, connected in the application top bar), the PER test mode is selectable on the RF test page.

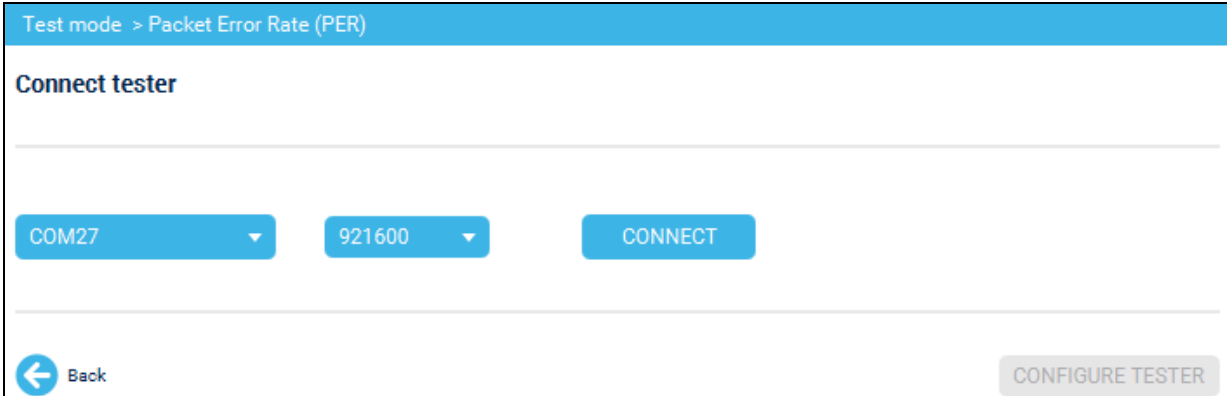
The configuration of the PER test is done with a sequence of panels:

- Tester connection
- Tester configuration
- DUT configuration
- Test parameters

The first step is to connect the tester:

PER tester connection

Figure 41. PER tester connection



Test mode > Packet Error Rate (PER)

Connect tester

COM27 921600 CONNECT

Back CONFIGURE TESTER

- Plug the device into the computer with the same requirements as the first device. Refer to [Section 2.2](#).
- Select the serial port to use in the picklist.
- Click on the *CONNECT* button.

Figure 42. PER tester connected

Test mode > Packet Error Rate (PER)

Connect tester

COM27 921600 DISCONNECT

Device : stm32wb05x
Firmware version : 1.0.0

Disconnect 2nd device to change test mode CONFIGURE TESTER

- The board information is displayed on the right.

When the second device is connected, it is not possible to change the mode. Disconnect the device first, and then use the *back* button.

Click on *CONFIGURE TESTER* to set the tester parameters:

Figure 43. PER tester configuration

Test mode > Packet Error Rate (PER) > COM33

Configure tester (COM33)

PA Level 31 (+6dBm)

TX Frequency 2402 MHz (Channel 37)

Length of Data 0x25

Packet Payload 0x00 - Pseudo-Random bit sequence 9

PHY 0x01 - Transmitter set to use the LE 1M PHY

Back CONFIGURE DUT

- Select the TX power level with the picklist.
- Select the transmission frequency with the *TX Frequency* picklist.
- Select the length of the packet to send (the same as the TX test).
- Select the content of the packet payload. For the PER test, it is recommended to use the *Pseudo-Random bit sequence 9* reference pattern. Patterns containing only 0 or 1 must not be used for PER. Other patterns can be used.
- Select the PHY to use.

Click on *CONFIGURE DUT* to set the *Device Under Test* configuration:

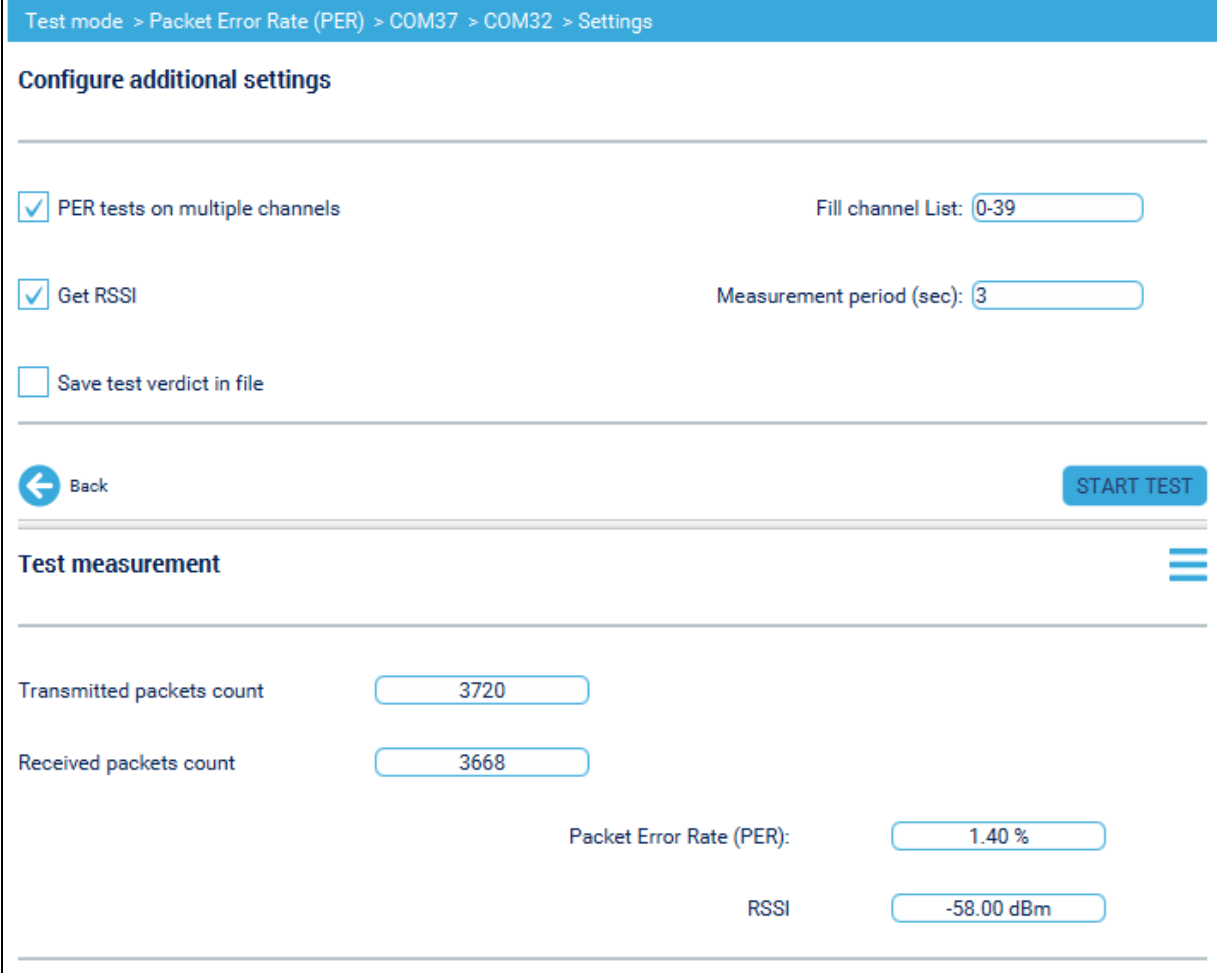
Figure 44. DUT configuration

The screenshot shows a web-based configuration interface for a Device Under Test (DUT). At the top, a blue breadcrumb trail reads "Test mode > Packet Error Rate (PER) > COM33 > COM26". Below this, the title "Configure Device under test (DUT) (COM26)" is displayed. The main configuration area contains three dropdown menus: "RX Frequency" set to "2402 MHz (Channel 37)", "PHY" set to "0x01 - Receiver set to use the LE 1M PHY", and "Index modulation" set to "0x00 - Assume transmitter will have a standard modulation index". At the bottom left is a "Back" button with a left arrow icon, and at the bottom right is a blue "CONFIGURE PARAM" button.

Select the reception frequency, the PHY, and the modulation index for the receiver board. The tool uses by default the same frequency as the tester, but the user might modify it.

Click on *CONFIGURE PARAM* to set the test configuration:

Figure 45. PER test parameters



Test mode > Packet Error Rate (PER) > COM37 > COM32 > Settings

Configure additional settings

☒ PER tests on multiple channels Fill channel List:

☒ Get RSSI Measurement period (sec):

☐ Save test verdict in file

← Back START TEST

Test measurement

Transmitted packets count

Received packets count

Packet Error Rate (PER):

RSSI

- **PER tests on multiple channels:** when this option is selected, the PER test is performed on a list of predefined channels. When the box is ticked, the *Channel list* is displayed. The *Value 0-39* indicates all channels from 0 to 39. It is also possible to put values separated by a comma: *0,1,5* or to mix: *0,1,10-15*. The measurement period is the time of each PER test to be performed.
- **Get RSSI:** this option adds some RSSI measurement between each PER measurement. When activated, the tool performs a PER test for the measurement period, computes PER, and then makes an RSSI check. Note that RSSI measurement is not supported for STM32WB0x devices.
- **Save test verdict in file:** this option generates a test report of the measurements. When the option is selected, a *SELECT FILE* button is displayed. The user must select the report file before starting the tests. The report is saved at the end of the tests.

When the option has been configured, click on the *START TEST* button:

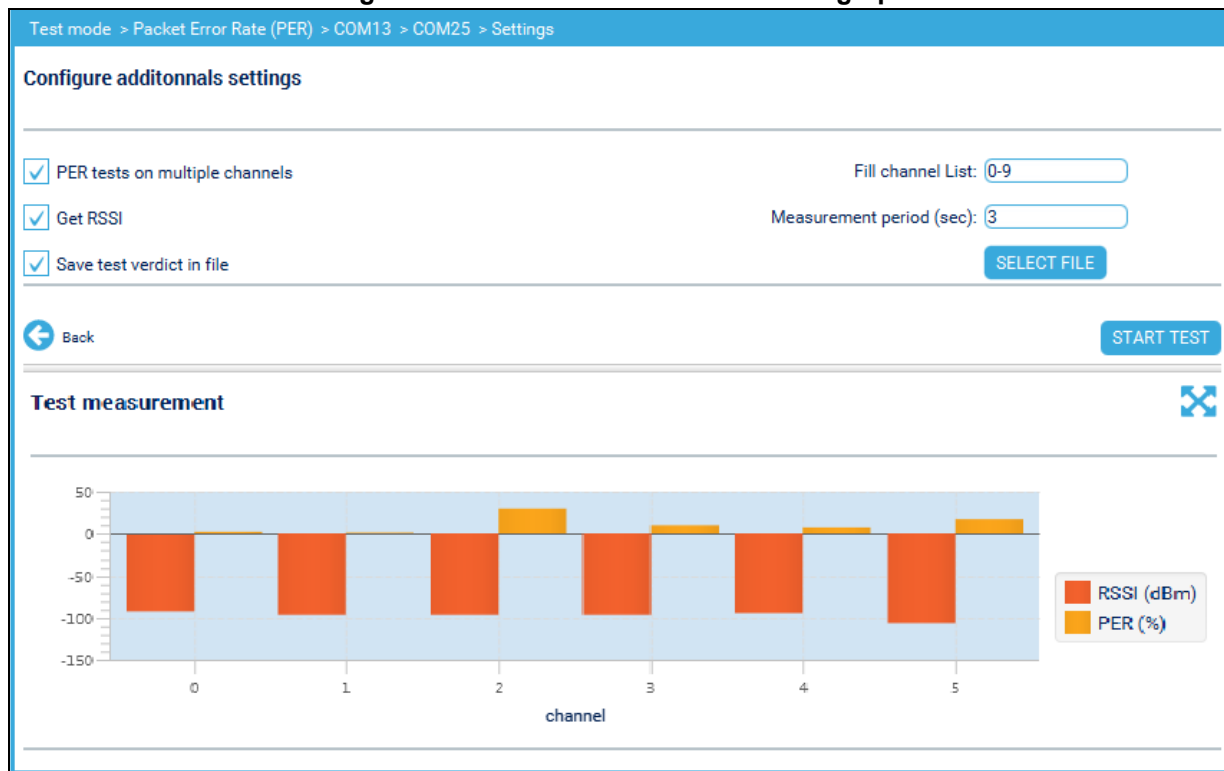
- The DUT is set in Reception mode,
- Then the tester starts.

The button is changed to *STOP* and *Testing...* is displayed.

The test continues until the user presses the *STOP* button, or after all channels are measured for multichannel tests.

The result is displayed in the bottom part. It is possible to switch between the numerical values and the chart with the blue bar icon.

Figure 46. PER and RSSI measurement graph



3.5 Scripts

Scripts are used to execute in sequence some commands stored in a text file. Scripts help avoid entering each command manually for repetitive tasks.

3.5.1 Launching scripts

Figure 47. Launching scripts

The screenshot displays the 'Scripts' tab of the ACI (Application Configuration Interface) software. The interface includes a top navigation bar with tabs for 'ACI Commands', 'Scripts' (which is active), 'Beacon', 'RF Tests', and 'ACI Utilities'. Below the navigation bar, the 'Script' section is visible. It contains a checkbox labeled 'Generate report' which is checked. Underneath the checkbox is a text input field containing the file path 'C:\Users\Public\Documents\STMicroelectronics\STM32CubeMonitor-RF\scripts\BLE_Sample'. To the right of this text field is a blue 'BROWSE' button. At the bottom right of the main content area, there is a blue 'START SCRIPT' button.

Scripts are stored in text files and are editable with any text editor.

To execute a script:

- Select the script file with the browse button or directly enter the file name.
- Click on the *Start script* button.
- The script is displayed and executed. The line in execution is highlighted in green. The ACI results are updated in the log area.
- The script is manually stopped with the *Stop script* button.

Figure 48. Script execution

Script

☒ Generate report

```

# STM32CubeMonitor-RF sample script
# Line starting with # are comments

# Empty line will be skipped

# Syntax to send ACL command : Send(ACL_CMD_NAME;Parameter1Value;Parameter2Value; ...)
# Parameter value are in hexadecimal, with format 0x12...
# Send reset command :
Send(HCI_RESET)

# Wait few milliseconds
Wait(500)

# Send another command : Set power level
Send(ACL_HAL_SET_TX_POWER_LEVEL;0x01;0x07)

# Pause command
Pause("This is a pause")

# Start Tone
Send(ACL_HAL_TONE_START;0x04;0x00)

# Wait 3 seconds
Wait(3000)

# Send stop tone
Send(ACL_HAL_TONE_STOP)

```

Script examples are provided with the tool, such as sample script, loop, and beacon creation.

For Windows, scripts are in the

C:\Users\Public\Documents\STMicroelectronics\STM32CubeMonitor-RF\scripts (public documents) folder.

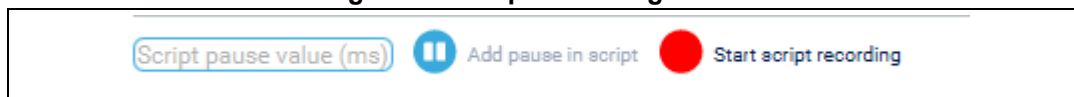
For Linux, they are in *<userhome>/STMicroelectronics/STM32CubeMonitor-RF/scripts*.

For macOS, it is inside the document folder provided in the setup package.

3.5.2 Script recording

The ACI commands used in the ACI panel are directly recorded in a script. Some script buttons are located at the bottom of the ACI panel:

Figure 49. Script recording buttons



Use the red button to start recording. Pause is inserted with the *Add pause in script* button.

At the end of the recording, click the *Stop* button. The tool asks for the script name before saving.

3.5.3 Scripts modification

The script is created or modified with a text editor. It uses a simple syntax to list the ACI command to send and the action to perform.

Figure 50. Sample script

```
# Send reset command:
Send(HCI_RESET)

# Wait a few milliseconds
Wait(500)

# Send another command: Set power level
Send(ACI_HAL_SET_TX_POWER_LEVEL;0x01;0x07)

# Start tone
Send(ACI_HAL_TONE_START; 0x00)

# Wait 3 seconds
Wait(3000)

# Send stop tone
Send (ACI_HAL_TONE_STOP)

# Pause command
Pause("End of script")
```

The lines starting with # are comments ignored by the tool. Empty lines are skipped.

Other lines are commands. The line starts with the command name, followed by parameters in brackets separated by a semicolon.

3.5.4 Script report

It is possible to have a script report generated at the end of script execution. The script report stores the status of each ACI command executed by the script.

Figure 51. Script report

SCRIPT REPORT			
Script name: SampleScript.txt			
Test date: 18/12/2017 17:27:52			
Verdict: SUCCESS No error detected			
Command	Sent	ACI status	ACI raw result
HCI_RESET	OK	0x00	
[0x04, 0x0E, 0x04, 0x01, 0x03, 0x0C, 0x00]			
ACI_HAL_SET_TX_POWER_LEVEL	OK	0x00	
[0x04, 0x0E, 0x04, 0x01, 0x0F, 0xFC, 0x00]			
ACI_HAL_TONE_START	OK	0x00	
[0x04, 0x0E, 0x04, 0x01, 0x15, 0xFC, 0x00]			
ACI_HAL_TONE_STOP	OK	0x00	
[0x04, 0x0E, 0x04, 0x01, 0x16, 0xFC, 0x00]			
END of report			

The result is stored in a new file, in the same path as the script, with a name in the form: *verdict_SampleScript_18-12-2017_17-27-52*. The name is built with the concatenation of:

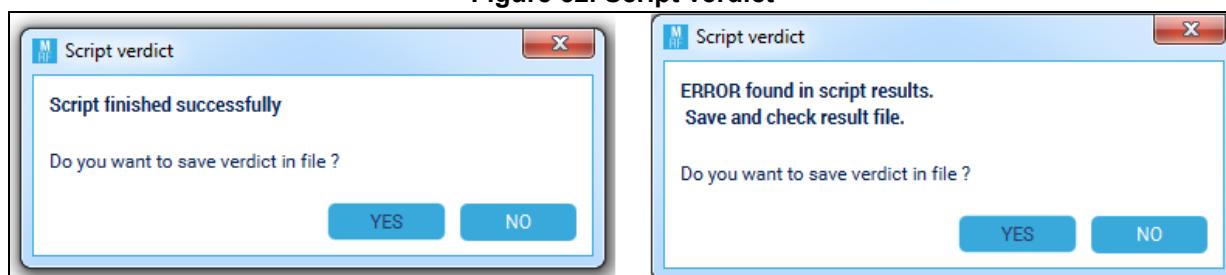
- *Verdict_*
- Script name
- Current date
- Current time
- .txt extension

In the report, the *Sent* column holds the status of the command transfer to the board. If parameters are missing, the command is not sent.

The *ACI status* column has the status of the ACI response. *0x00* is a success status while other values are errors.

At the end of script execution, a popup with the verdict (error found or finish successfully) is displayed and asks if the report must be saved:

Figure 52. Script verdict



If the user presses *yes*, the report is generated in the folder of the current script. If the user presses *no*, the report is not saved.

If the *Generate report* tick box is not checked, no report is generated at the end of the script. The script successfully means that there is no error in the script syntax, and the status of operations is *OK* (error code = 0). The value measured and the performance are not verified, there is no *PASS/FAIL* criterion on the results.

3.5.5 List of script commands

Send an ACI command

The ACI commands are sent with the instruction *Send*: `Send (ACI_CMD_NAME; Parameter1Value; Parameter2Value...)`

The elements inside the parenthesis are separated by semicolons.

The first element is the command name. It is the name as it is displayed in the tool.

The next elements are the parameters. The value must be entered in hexadecimal format and start with *0x*. The optional parameters can be left empty. The length is dependent on the size parameter in the ACI command.

Note: The *Command Packet Type*, *Opcode*, and *Parameter Total Length* are filled in by the application. They must not be added to the parameters.

Wait for a specific time

It is possible to add a delay with the instruction *Wait*:

```
Wait (3000)
```

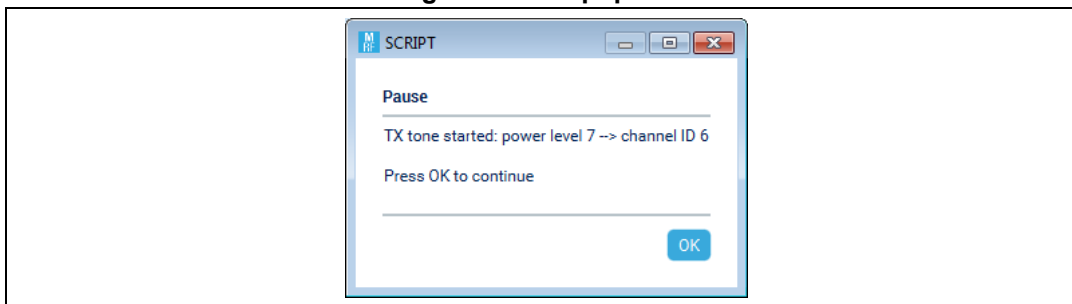
This instruction delays the script execution for three seconds. Time is given in milliseconds.

In the ACI panel screen, a pause is inserted in the script with the *Add pause in script* button.

Pause command in the script

The *Pause* command adds a pause during the proceeding of the script. This command opens a pop-up window customized with the user's comment.

Figure 53. Script pause



The *OK* button allows the user to continue the script.

Command: *Pause (user comment)*

The user text must be enclosed between quote marks (").

Figure 54. Example

```
# Pause demo script

# Start tone
Send (ACI_HAL_TONE_START; 0x04)

# Pause command
Pause ("TX tone started")

# Send stop tone
Send (ACI_HAL_TONE_STOP)
```

Loop command in the script

A loop can be used in the script to repeat some actions automatically.

Loop usage

To repeat a part of a script, the commands must be enclosed between two instructions:

- *Loop (count, 0, 5)*: This instruction indicates the beginning of the repeated section. The *count* variable is the name given to the counter. The first value is the start value and the second one is the end value. In this example, the counter *count* is increased from 0 to 5. There are six iterations.
- *EndLoop* indicates the end of the loop. If the counter reaches the end value, execution continues to the next line. If the counter does not reach the end value, it is updated, and execution goes back to the *Loop* instruction.

Figure 55. Loop simple example

```
Loop (count; 1; 3)
Pause ("test the loop")
EndLoop
```


This script, given as an example in [Figure 55](#), displays three times: *test the loop*.

Using the counter value

It is possible to use the counter value in other lines of the script to change the parameter values during script execution. When the counter name is embedded inside square brackets, the tool inserts the counter value.

Figure 56. Loop second simple example

```
Loop (count; 1; 3)
Pause ("The loop counter is [count]")
EndLoop
```

The script in [Figure 56](#) displays *The loop counter is 1*, then *The loop counter is 2*, and finally, *The loop counter is 3*.

Some parameters require hexadecimal values. In this case, add an ampersand (&) after the first bracket. The tool replaces the counter name with the hexadecimal value.

If count = 10, 0xA replaces [&count].

Special count option

The counter value can increase or decrease. If the start value is bigger than the end value, the counter is decremented.

Figure 57. Loop decrement

```
Loop (mycount; 3; 1)
```

In the countdown example ([Figure 57](#)), *mycount* takes the values 3, 2, and 1.

The counter can have a specific increment value when a third value is added to the loop instruction, as shown in [Figure 58](#):

Figure 58. Loop specific increment

```
Loop (mycount; 1; 6; 2)
```

This example counts with a step of 2. Successive values are 1, 3, and 5. The loop stops at 5 because 7 is higher than 6.

The loop can include another loop. It is mandatory to use a different counter name.

Figure 59. Nested loop

```
Loop (row;4;5)
  Loop (column;3;2)
    Pause ("coord: [row] [column]")
  EndLoop
EndLoop
```

The script provided as a nested loop example in [Figure 59](#) displays: *coord: 4 3*, *coord: 4 2*, *coord: 5 3*, and *coord 5 2*.

Loop script verdict

The loop generates some special lines in the verdict file. The added lines help the user follow the execution.

The script shown in [Figure 60](#) generates the verdict shown in [Figure 61](#).

Figure 60. Example of loop script verdict

```
Loop (FREQ, 13, 15)
EndLoop
```

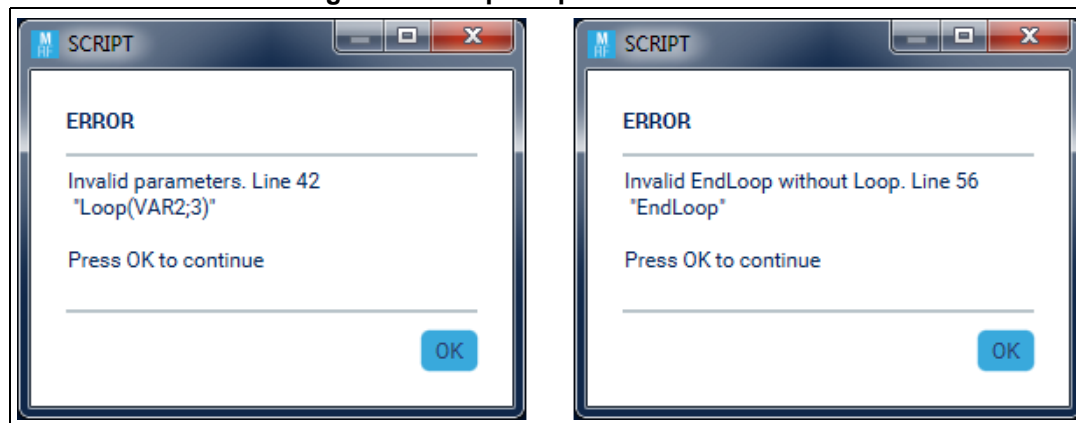
Figure 61. Display of loop script verdict

```
-- Loop Start (FREQ=13) --
-- Loop (FREQ=14) --
-- Loop (FREQ=15) --
-- Loop End (FREQ) --
```

The beginning and end of the loop are indicated, and the counter value is also inserted in decimal for each iteration.

In the case of an error in an instruction, a pop-up warns the user when the script is executed and the line is skipped. If a Loop instruction is missing or invalid, the EndLoop generates the *Invalid EndLoop without Loop* warning message.

Figure 62. Script loop error



3.6 Command-line interface (CLI)

3.6.1 Installation

No specific installation is required because the command-line interface is included with the STM32CubeMonitor-RF graphical user interface tool.

3.6.2 Getting started

To access the CLI, open a terminal and launch the tool with an argument from the installation directory. When no argument is used, the GUI launches.

By default, the installation directory is:

- macOS®: `/Applications/STM32CubeMonitor-RF.app/Contents/MacOS`
- Linux®: `/local/home/<user>/STMicroelectronics/STM32CubeMonitor-RF`
- Windows®: `C:\Program Files\STMicroelectronics\STM32CubeMonitor-RF`

To list all available CLI commands, execute:

```
./STM32CubeMonitor-RF --help
```

3.6.3 Arguments for CLI commands

Optional arguments

- | | |
|---|---|
| <p>-a, --aciLog=<aciLogFile></p> | <p>Name of the file to save the ACI log.</p> <p>To get the ACI log as described in Section 3.3.5: Log functionalities, this argument must be defined with a filename to allow the recording in this file. The file is in csv format. It can be opened using a spreadsheet tool to improve data viewing.</p> |
| <p>-v, --verbose</p> | <p>Verbose mode that displays the raw details concerning the board identification on connection.</p> |
| <p>-b, --baudrate=<baudRate></p> | <p>Define the UART baud rate (the default value is 115200 bauds). Refer to Section 2.3: Opening COM if the default baud rate is not relevant.</p> |

Required arguments

- | | |
|--------------------------------------|---|
| <p>-p, --port=<COMxx></p> | <p>Define the COM port to use. If the COM ports are unknown, first use the list command to identify them.</p> |
|--------------------------------------|---|

3.6.4 Basic commands

list

```
./STM32CubeMonitor-RF list                      [-a=<aciLogFile>]
```

This command lists all available COM ports, including those not connected to an STM32 Bluetooth® LE board.

getInfo

```
./STM32CubeMonitor-RF getInfo                      -p=<COMxx>
                                                    [-b=<baudRate>]
                                                    [-v]
                                                    [-a=<aciLogFile>]
```

This command gets the version information of the STM32 Bluetooth® LE board connected to the COM port set as a required argument (refer to [Section 3.6.3: Arguments for CLI commands](#) for details).

3.6.5 Informative commands

getCommands

```
./STM32CubeMonitor-RF getCommands -p=<COMxx>
                               [-v]
                               [-a=<aciLogFile>]
                               [-b=<baudRate>]
```

This command provides all the available ACI commands for the STM32 Bluetooth® LE board connected to the COM port set as a required argument (refer to [Section 3.6.3: Arguments for CLI commands](#) for details).

getParameters

```
./STM32CubeMonitor-RF getParameters -p=<COMxx>
                                      -n=<ACICommandName>
                                      [-v]
                                      [-a=<aciLogFile>]
                                      [-b=<baudRate>]
```

This command provides all the available ACI commands for the STM32 Bluetooth® LE board connected to the COM port set as a mandatory argument (refer to [Section 3.6.3: Arguments for CLI commands](#) for details).

Specific required argument:

```
-n,      --name=<ACICommandName> Name of the ACI command (use the
                                getCommands command to retrieve this name).
```

Refer to [Section 3.6.3: Arguments for CLI commands](#) for details about the other possible arguments for this command.

3.6.6 Execution commands

SendCommand

```
./STM32CubeMonitor-RF sendCommand -p=<COMxx>
                                      -c=<ACICommand>
                                      [-v]
                                      [-a=<aciLogFile>]
                                      [-b=<baudRate>]
```

Use this command to execute a unitary ACI command on the target connected to the COM port set as a mandatory argument (refer to [Section 3.6.3: Arguments for CLI commands](#) for details).

Specific required argument:

```
-c,      --command=<ACICommand> ACI command to execute. The command format is
                                the same as the one used in the script format (refer
                                to Section 3.5.5: List of script commands).
```


Format example:

```
"Send(ACI_HAL_WRITE_CONFIG_DATA;0x00;0x06;0x222233445570)"
```

The first parameter is the ACI command name. The next parameters are the values of the ACI command arguments.

script

```
./STM32CubeMonitor-RF script          -p=<COMxx>
                                         -s=<scriptFile>
                                         [-r=<verdictFile>]
                                         [-v]
                                         [-a=<aciLogFile>]
                                         [-b=<baudRate>]
```

Use this command to execute a script file on the target connected to the COM port set as a mandatory argument (refer to [Section 3.6.3: Arguments for CLI commands](#) for details).

Specific required argument:

```
-s,      --script=<scriptFile>  Script file to execute. Refer to Section 3.5: Scripts
                                for guidelines and examples.
```

Specific optional argument:

```
-r,      --result=<verdictFile>  Name of the file to register the verdict result.
```

3.6.7 Examples

list

Command:

```
$ ./STM32CubeMonitor-RF list
```

Output:

```
COM5 ; COM4 ; COM44 ; COM3 ; COM7
```

Comment:

This command lists all available COM ports, including those not connected to an STM32 Bluetooth® LE board.

getInfo

Command:

```
$ ./STM32CubeMonitor-RF getInfo -p COM7
```

Output:

```
CM0 FUS version           :1.2.0
CM0 Safeboot version      :0.0.0
CM0 version               :1.18.0.3
```



```
CM4 version           :0.0.1
Device                :stm32wbxx
Device Id             :0x495
Device Revision       :8193
Device package type   :a
Hardware version      :2.1.0
Manufacturer          :STMicroelectronics
Stack info            :Full stack
Tag Number            :119
```

getCommands

Command:

```
$ ./STM32CubeMonitor-RF getCommands -p COM7
```

Output:

```
HCI_DISCONNECT
HCI_READ_REMOTE_VERSION_INFORMATION
HCI_SET_EVENT_MASK
HCI_RESET
HCI_READ_CONNECTION_ACCEPT_TIMEOUT
HCI_WRITE_CONNECTION_ACCEPT_TIMEOUT
HCI_READ_TRANSMIT_POWER_LEVEL
HCI_SET_CONTROLLER_TO_HOST_FLOW_CONTROL
HCI_HOST_BUFFER_SIZE
HCI_HOST_NUMBER_OF_COMPLETED_PACKETS
HCI_READ_AFH_CHANNEL_ASSESSMENT_MODE
HCI_WRITE_AFH_CHANNEL_ASSESSMENT_MODE
HCI_SET_EVENT_MASK_PAGE_2
HCI_READ_AUTHENTICATED_PAYLOAD_TIMEOUT
HCI_WRITE_AUTHENTICATED_PAYLOAD_TIMEOUT
HCI_SET_ECOSYSTEM_BASE_INTERVAL
HCI_CONFIGURE_DATA_PATH
[...]
ACI_L2CAP_COC_DISCONNECT
ACI_L2CAP_COC_FLOW_CONTROL
ACI_L2CAP_COC_TX_DATA
VS_HCI_C1_WRITE_REGISTER
VS_HCI_C1_READ_REGISTER
```

Comment:

This command lists all available Bluetooth® LE ACI commands on the board connected.

getParameters

Command:

```
$ ./STM32CubeMonitor-RF getParameters -p COM7 -n
ACI_GATT_WRITE_CHAR_VALUE
```


Output:

```
[PARAMETER] = (hci_packet_indicator) [VALUE] = (0x01) [LITERAL] = (HCI
Command Packet) [IS_ENUMERABLE] = (false) [IS_EDITABLE] = (false)
[NEED_CHECK] = (false)

[PARAMETER] = (op_code) [VALUE] = (0xFD1C)
[LITERAL] = (ACI_GATT_WRITE_CHAR_VALUE) [IS_ENUMERABLE] = (false)
[IS_EDITABLE] = (false) [NEED_CHECK] = (false)

[PARAMETER] = (parameter_total_length) [VALUE] = (0x06)
[IS_ENUMERABLE] = (false) [IS_EDITABLE] = (false) [NEED_CHECK] = (false)

[PARAMETER] = (connection_handle) [VALUE] = (0x0000)
[MIN_VALUE] = (0x0000) [MAX_VALUE] = (0xEA3F) [DEFAULT_VALUE] = (0x0000)
[IS_ENUMERABLE] = (false) [IS_EDITABLE] = (true) [NEED_CHECK] = (false)

[PARAMETER] = (attr_handle) [VALUE] = (0x0000) [IS_ENUMERABLE] = (false)
[IS_EDITABLE] = (true) [NEED_CHECK] = (false)

[PARAMETER] = (attribute_val_length) [VALUE] = (0x01)
[DEFAULT_VALUE] = (0x01) [IS_ENUMERABLE] = (false) [IS_EDITABLE] = (true)
[NEED_CHECK] = (true)

[PARAMETER] = (attribute_val) [VALUE] = (0x00) [IS_ENUMERABLE] = (false)
[IS_EDITABLE] = (true) [NEED_CHECK] = (false)
```

Comment:

This command retrieves the characteristics of each parameter to be used with the ACI command set as one of its mandatory arguments.

getParameters is used to obtain the name of the parameters, their formats through their default values, and their orders to help the creation of the execution commands.

sendCommandCommand:

```
$ ./STM32CubeMonitor-RF senCommand -p COM7 -c "Send(HCI_RESET)" -a
~/myAciLog.txt
```

The *myAciLog.txt* file lists the ACI command sent together with its result as displayed in the AciLog screen of the GUI. The commands used to connect the board before the launch of the ACI command requested are also listed.

~/myAciLog.txt:

```
Nb, Time, Port Com, Type, Parameter, Value, Literal, Information
0,18:04:49:580,COM7,HCI_READ_LOCAL_VERSION_INFORMATION,hci_packet_i
ndicator,0x01,HCI Command Packet,
0,18:04:49:580,COM7,HCI_READ_LOCAL_VERSION_INFORMATION,op_code,0x10
01,HCI_READ_LOCAL_VERSION_INFORMATION,
0,18:04:49:580,COM7,HCI_READ_LOCAL_VERSION_INFORMATION,parameter_to
tal_length,0x00,
[...]
4,18:04:49:706,COM7,HCI_RESET,hci_packet_indicator,0x01,HCI Command
Packet,
4,18:04:49:706,COM7,HCI_RESET,op_code,0x0C03,HCI_RESET,
4,18:04:49:706,COM7,HCI_RESET,parameter_total_length,0x00,
```


4,18:04:49:706,COM7,HCI_RESET,Raw,[0x01:0x03:0x0C:0x00]
 5,18:04:49:708,COM7,HCI_COMMAND_COMPLETE_EVENT,hci_packet_indicator,0x04,HCI Event packet,
 5,18:04:49:708,COM7,HCI_COMMAND_COMPLETE_EVENT,event_code,0x0E,HCI_COMMAND_COMPLETE_EVENT,
 5,18:04:49:708,COM7,HCI_COMMAND_COMPLETE_EVENT,parameter_total_length,0x04,
 5,18:04:49:708,COM7,HCI_COMMAND_COMPLETE_EVENT,num_hci_command_packets,0x01,The Number of HCI command packets which are allowed to be sent to the Controller from the Host.
 5,18:04:49:708,COM7,HCI_COMMAND_COMPLETE_EVENT,command_opcode,0x0C03,HCI_RESET,Opcode of the command which caused this event.
 5,18:04:49:708,COM7,HCI_COMMAND_COMPLETE_EVENT,status,0x00,Success,Status error code.
 5,18:04:49:708,COM7,HCI_COMMAND_COMPLETE_EVENT,Raw,[0x04:0x0E:0x04:0x01:0x03:0x0C:0x00]

For improved data viewing, open the csv-formatted *myAciLog.txt* file in a spreadsheet tool as shown in [Figure 63](#).

Figure 63. Contents of *myAciLog.txt* displayed in a spreadsheet tool (sendCommand)

	A	B	C	D	E	F	G	H	I	J
1	Nb	Time	Port Com	Type	Parameter	Value	Literal	Information		
2	0	18:04:49:580	COM7	HCI_READ_LOCAL_VERSION_INFORMATION	hci_packet_indicator	0x01		HCI Command Packet		
3	0	18:04:49:580	COM7	HCI_READ_LOCAL_VERSION_INFORMATION	op_code	0x1001		HCI_READ_LOCAL_VERSION_INFORMATION		
4	0	18:04:49:580	COM7	HCI_READ_LOCAL_VERSION_INFORMATION	parameter_total_length	0x00				
5	0	18:04:49:580	COM7	HCI_READ_LOCAL_VERSION_INFORMATION	Raw	[0x01:0x01:0x10:0x00]				
6	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	hci_packet_indicator	0x04		HCI Event packet		
7	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	event_code	0x0E		HCI_COMMAND_COMPLETE_EVENT		
8	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	parameter_total_length	0x0C				
9	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	num_hci_command_packets	0x01			The number of HCI command packets	
10	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	command_opcode	0x1001		HCI_READ_LOCAL_VERSION_INFORMATION	Opcode of this command which caused this event	
11	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00		Success	Status error code	
12	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	hci_version	0x0D			Version of the HCI Specification	
13	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	hci_subversion	0x0077			Revision of the HCI implementation	
14	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	lmp_version	0x0D			Version of the Current LMP specification	
15	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	company_identifier	0x0030			Company identifier for the manufacturer	
16	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	lmp_subversion	0x2177			Subversion of the Current LMP implementation	
17	1	18:04:49:586	COM7	HCI_COMMAND_COMPLETE_EVENT	Raw	[0x04:0x0E:0x04:0x01:0x03:0x0C:0x00:0x0D:0x77:0x00:0x30:0x0D:0x77:0x21]				
18	2	18:04:49:595	COM7	VS_HCI_C1_DEVICE_INFORMATION	hci_packet_indicator	0x20		Local Command Packet		
19	2	18:04:49:595	COM7	VS_HCI_C1_DEVICE_INFORMATION	op_code	0xFD62		VS_HCI_C1_DEVICE_INFORMATION		
20	2	18:04:49:595	COM7	VS_HCI_C1_DEVICE_INFORMATION	parameter_total_length	0x00				
21	2	18:04:49:595	COM7	VS_HCI_C1_DEVICE_INFORMATION	Raw	[0x20:0x62:0xFD:0x00]				
22	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	hci_packet_indicator	0x21		Local Event Packet		
23	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	event_code	0x0E		HCI_COMMAND_COMPLETE_EVENT		
24	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	parameter_total_length	0x42				
25	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	num_hci_command_packets	0x01			The number of HCI command packets	
26	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	command_opcode	0xFD62		VS_HCI_C1_DEVICE_INFORMATION	Opcode of this command which caused this event	
27	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00		Success	Status error code	
28	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	revision_id	0x2001			Revision Id (from DBGMCU_IDCODE)	
29	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	device_code_id	0x0495			Device Code Id (from DBGMCU_IDCODE)	
30	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	package_type	0x0A			Package Type (from package data)	
31	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	device_type_id	0x26			Device Type Id (from FLASH UID64)	
32	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	st_company_id	0x00080E1			ST Company Id (from FLASH UID64)	
33	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	uid64	0x00129B82			UID64 (from FLASH UID64)	
34	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	uid96	0x20343743594D50130047003A			UID96 from Unique Device ID register	
35	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	safe_boot_information	0x00000000			Safe Boot Information (Not Applicable for this device)	
36	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	rss_information	0x000000001000000601020000			Rss Information (Not Applicable for this device)	
37	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	cm0 and wireless fw version	0x01120003			CM0+ Wireless FW Information (Not Applicable for this device)	
38	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	cm0 and wireless fw memory size	0x161E002F			CM0+ Wireless FW Information (Not Applicable for this device)	
39	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	cm0 and wireless fw stackinfo	0x00000006			CM0+ Wireless FW Information (Not Applicable for this device)	
40	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	cm0 and wireless fw reserved	0x00000000			CM0+ Wireless FW Information (Not Applicable for this device)	
41	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	local_fw_information	0x00000100			Local FW Information (Hard coded)	
42	3	18:04:49:597	COM7	HCI_COMMAND_COMPLETE_EVENT	Raw	[0x21:0x0E:0x42:0x01:0x62:0xFD:0x00:0x01:0x20:0x95:0x04:0x0A:0x26:0xE1:0x80:0x0A]				
43	4	18:04:49:706	COM7	HCI_RESET	hci_packet_indicator	0x01		HCI Command Packet		

scriptCommand:

```
$ ./STM32CubeMonitor-RF script -p COM7 -a ~/myAciLog.txt -r
~/myResult.txt -s ~/myScript_AdvertisingScan_GAP_Extended.txt
```

myScript_AdvertisingScan_GAP_Extended.txt is the filename of the script file in this example. The file content is as follows:

```
# Address Initialisation
Send(ACI_HAL_WRITE_CONFIG_DATA;0x00;0x06;0x222233445570)
#Send(ACI_HAL_SET_TX_POWER_LEVEL;0x00;0x11)
# GATT Initialisation
Send(ACI_GATT_INIT)
# GAP Initialisation with all roles with 7 bytes reserved for the
name
Send(ACI_GAP_INIT;0x0F;0x00;0x07)
# Set the name
Send(ACI_GATT_UPDATE_CHAR_VALUE;0x0005;0x0006;0x00;0x07;0x434150595
24F44)
Send(ACI_GAP_SET_IO_CAPABILITY;0x01)
Send(ACI_GAP_SET_AUTHENTICATION_REQUIREMENT;0x01;0x01;0x02;0x00;0x0
7;0x10;0x01;0x00013A57;0x00)
# ----- Extended advertising -----
Send(ACI_GAP_ADV_SET_CONFIGURATION;0x00;0x00;0x0013;0x000007D0;0x00
0007D0;0x07;0x00;0x00;0x112233445566;0x00;0x01;0x00;0x01;0x00;0x00)
Send(ACI_GAP_ADV_SET_ADV_DATA;0x00;0x03;0x00;0x08;0x320102434150090
4)
Send(ACI_GAP_ADV_SET_ENABLE;0x01;0x01;0x00;0x0000;0x00)
#----- Scan -----
Send(ACI_GAP_START_GENERAL_DISCOVERY_PROC;0x07D0;0x07D0;0x00;0x01)
Wait(4000)
Send(ACI_GAP_ADV_SET_ENABLE;0x00;0x01;0x00;0x0000;0x00)
```

Output:

Execution status : ok

In this example, the verdict result is set in the file *myResult.txt* as follows:

SCRIPT REPORT

Script name : myScript_AdvertisingScan_GAP_Extended.txt

Test date : 18/03/2025 09:58:41

Verdict : SUCCESS No error detected

Command	Sent	ACI status	
ACI raw result			
ACI_HAL_WRITE_CONFIG_DATA	OK	0x00	
[0x04, 0x0E, 0x04, 0x01, 0x0C, 0xFC, 0x00]			
ACI_GATT_INIT	OK	0x00	
[0x04, 0x0E, 0x04, 0x01, 0x01, 0xFD, 0x00]			
ACI_GAP_INIT	OK	0x00	
[0x04, 0x0E, 0x0A, 0x01, 0x8A, 0xFC, 0x00, 0x05, 0x00, 0x06, 0x00, 0x08, 0x00]			
ACI_GATT_UPDATE_CHAR_VALUE	OK	0x00	

[0x04, 0x0E, 0x04, 0x01, 0x06, 0xFD, 0x00]		
ACI_GAP_SET_IO_CAPABILITY	OK	0x00
[0x04, 0x0E, 0x04, 0x01, 0x85, 0xFC, 0x00]		
ACI_GAP_SET_AUTHENTICATION_REQUIREMENT	OK	0x00
[0x04, 0x0E, 0x04, 0x01, 0x86, 0xFC, 0x00]		
ACI_GAP_ADV_SET_CONFIGURATION	OK	0x00
[0x04, 0x0E, 0x04, 0x01, 0xC0, 0xFC, 0x00]		
ACI_GAP_ADV_SET_ADV_DATA	OK	0x00
[0x04, 0x0E, 0x04, 0x01, 0xC2, 0xFC, 0x00]		
ACI_GAP_ADV_SET_ENABLE	OK	0x00
[0x04, 0x0E, 0x04, 0x01, 0xC1, 0xFC, 0x00]		
ACI_GAP_START_GENERAL_DISCOVERY_PROC	OK	0x00
[0x04, 0x0F, 0x04, 0x00, 0x01, 0x97, 0xFC]		
ACI_GAP_ADV_SET_ENABLE	OK	0x00
[0x04, 0x0E, 0x04, 0x01, 0xC1, 0xFC, 0x00]		
END of report		

In this example, the ACI log is registered in the *myAciLog.txt* file. For improved data viewing, open it in a spreadsheet tool as shown in [Figure 64](#).

Figure 64. Contents of *myAciLog.txt* displayed in a spreadsheet tool (script)

A	B	C	D	E	F	G	H
1	Nb	Time	Port	Com Type	Parameter	Value	Literal
2	0	15:55:38:748	COM7	HCI_READ_LOCAL_VERSION_INFORMATION	hci_packet_indicator	0x01	Information
3	0	15:55:38:748	COM7	HCI_READ_LOCAL_VERSION_INFORMATION	op_code	0x1001	HCI_READ_LOCAL_VERSION_INFORMATION
4	0	15:55:38:748	COM7	HCI_READ_LOCAL_VERSION_INFORMATION	parameter_total_length	0x00	
5	0	15:55:38:748	COM7	HCI_READ_LOCAL_VERSION_INFORMATION	Raw	[0x01:0x01:0x10:0x00]	
6	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	hci_packet_indicator	0x01	HCI Event Packet
7	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	event_code	0x0E	HCI_COMMAND_COMPLETE_EVENT
8	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	parameter_total_length	0x0C	
9	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	num_hci_command_packets	0x01	The number of HCI command packets which are allowed to be sent to the controller from the host.
10	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	command_opcode	0x1001	HCI_READ_LOCAL_VERSION_INFORMATION
11	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success
12	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	hci_version	0x00	Version of the HCI Specification supported by the Controller. See Bluetooth Assigned Numbers.
13	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	hci_subversion	0x0077	Revision of the HCI implementation in the Controller. This value is vendor-specific.
14	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	lmp_version	0x00	Version of the Current LMP supported by the Controller. See Bluetooth Assigned Numbers.
15	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	company_identifier	0x0030	Company identifier for the manufacturer of the Controller. See Bluetooth Assigned Numbers.
16	1	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	lmp_subversion	0x1777	Subversion of the Current LMP in the Controller. This value is vendor-specific.
17	2	15:55:38:751	COM7	HCI_COMMAND_COMPLETE_EVENT	Raw	[0x01:0x0E:0x0C:0x01:0x01:0x10:0x00:0x00:0x00:0x77:0x00:0x00:0x00:0x77:0x00:0x77:0x021]	
18	2	15:55:38:758	COM7	VS_HCI_C1_DEVICE_INFORMATION	hci_packet_indicator	0x20	Local Command Packet
19	2	15:55:38:758	COM7	VS_HCI_C1_DEVICE_INFORMATION	op_code	0xF062	VS_HCI_C1_DEVICE_INFORMATION
20	2	15:55:38:758	COM7	VS_HCI_C1_DEVICE_INFORMATION	parameter_total_length	0x00	
21	2	15:55:38:758	COM7	VS_HCI_C1_DEVICE_INFORMATION	Raw	[0x20:0x62:0xF0:0x00]	
22	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	hci_packet_indicator	0x21	Local Event Packet
23	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	event_code	0x0E	HCI_COMMAND_COMPLETE_EVENT
24	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	parameter_total_length	0x01	
25	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	num_hci_command_packets	0x02	The number of HCI command packets which are allowed to be sent to the controller from the host.
26	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	command_opcode	0xF062	VS_HCI_C1_DEVICE_INFORMATION
27	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success
28	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	revision_id	0x2001	Revision Id (from DBGMCI_JDCCODE register).
29	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	device_code_id	0x0495	Device Code Id (from DBGMCI_JDCCODE register)
30	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	package_type	0x01	Package Type (from package data register)
31	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	device_type_id	0x26	Device Type Id (from FLASH UID04)
32	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	st_package_id	0x00009E1	ST Package Id (from FLASH UID04)
33	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	uid04	0x0129980	UID04 (from FLASH UID04)
34	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	uid06	0x2034374394050130046003B	UID06 from Unique Device ID register
35	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	safe_boot_information	0x00000000	Safe Boot Information (Not Applicable for STM32WB)
36	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	rs_information	0x00000000100000001020000	RMS Information (Not Applicable for STM32WB)
37	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	cm0_and_wireless_fw_version	0x01129003	CM0-Wireless FW Information (Not Applicable for STM32WB)
38	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	cm0_and_wireless_fw_memory_size	0x0116002F	CM0-Wireless FW Information (Not Applicable for STM32WB)
39	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	cm0_and_wireless_fw_stackinfo	0x00000006	CM0-Wireless FW Information (Not Applicable for STM32WB)
40	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	cm0_and_wireless_fw_reserved	0x00000000	CM0-Wireless FW Information (Not Applicable for STM32WB)
41	3	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	local_fw_information	0x00000100	Local FW Information (Hard coded in user flash)
42	4	15:55:38:759	COM7	HCI_COMMAND_COMPLETE_EVENT	Raw	[0x21:0x0E:0x42:0x01:0x62:0xF0:0x00:0x01:0x20:0x05:0x04:0x0A:0x26:0xE1:0x80:0x00:0x88:0x0B:0x02:0x00:0x46:0x05:0x43:0x30:0x43:0x20:0x00:0x00]	
43	4	15:55:38:876	COM7	ACI_HAL_WRITE_CONFIG_DATA	hci_packet_indicator	0x01	ACI Command Packet
44	4	15:55:38:876	COM7	ACI_HAL_WRITE_CONFIG_DATA	op_code	0xC0C	ACI_HAL_WRITE_CONFIG_DATA
45	4	15:55:38:876	COM7	ACI_HAL_WRITE_CONFIG_DATA	parameter_total_length	0x08	

How to investigate in the case of a script error

Command:

```
$ ./STM32CubeMonitor-RF script -p COM7 -a ~/myAciLog.txt -r
~/myResult.txt -s ~/myScript AdvertisingScan GAP Extended.txt
```

Output:

```
Execution status : failed, see verdict result for details if option
activated
```

Use a spreadsheet tool to check the *myAciLog.txt* file and search the error in the status as shown [Figure 65](#) in and [Figure 66](#).

Figure 65. Contents of *myAciLog.txt* displayed in a spreadsheet tool (error investigation 1 of 2)

	A	B	C	D	E	F	G
1	Time	Port Co	Type	Parameter	Value	Literal	Information
11	1 17:15:58:899	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.
27	3 17:15:58:909	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.
55	5 17:15:59:035	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.
66	7 17:15:59:039	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x62	Invalid operation	Status error code.
80	9 17:15:59:043	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x0C	Command Disallowed	Status error code.
99	11 17:15:59:047	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.
111	13 17:15:59:050	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.
131	15 17:15:59:156	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.
157	17 17:15:59:161	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.
173	19 17:15:59:265	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.
189	21 17:15:59:269	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.
202	23 17:15:59:272	COM7	HCI_COMMAND_STATUS_EVENT	status	0x00	Success	Status error code.
619	46 17:16:03:275	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x00	Success	Status error code.

Figure 66. Contents of *myAciLog.txt* displayed in a spreadsheet tool (error investigation 2 of 2)

6	17:15:59:038	COM7	ACI_GATT_INIT	hci_packet_indicator	0x01	HCI Command Packet	
6	17:15:59:038	COM7	ACI_GATT_INIT	op_code	0xFD01	ACI_GATT_INIT	
6	17:15:59:038	COM7	ACI_GATT_INIT	parameter_total_length	0x00		
6	17:15:59:038	COM7	ACI_GATT_INIT	Raw	[0x010x010xFD0x00]		
7	17:15:59:039	COM7	HCI_COMMAND_COMPLETE_EVENT	hci_packet_indicator	0x04	HCI Event packet	
7	17:15:59:039	COM7	HCI_COMMAND_COMPLETE_EVENT	event_code	0x0E	HCI_COMMAND_COMPLETE_EVENT	
7	17:15:59:039	COM7	HCI_COMMAND_COMPLETE_EVENT	parameter_total_length	0x04		
7	17:15:59:039	COM7	HCI_COMMAND_COMPLETE_EVENT	num_hci_command_packets	0x01		
7	17:15:59:039	COM7	HCI_COMMAND_COMPLETE_EVENT	command_opcode	0xFD01	ACI_GATT_INIT	The Number of HCI command packets which are allowed to be sent to the Controller from the Host.
7	17:15:59:039	COM7	HCI_COMMAND_COMPLETE_EVENT	status	0x62	Invalid operation	Opcode of the command which caused this event.
7	17:15:59:039	COM7	HCI_COMMAND_COMPLETE_EVENT	Raw	[0x040x0E0x040x010x010xFD0x62]		Status error code.

In this example, the ACI_GATT_INIT has been already done, resulting in an error.

3.7 OTA transfer

3.7.1 OTA presentation

Over-the-air (OTA) transfer executes the transfer of data from a device to a remote device without a cable. Data are applicative data, like user configuration, pictures, music, or firmware. STM32CubeMonitor-RF provides a transfer function from the computer to the remote device over Bluetooth® LE.

In this section, the computer or device sending the data is named the *Source device*.

The source device transfers the data and the OTA loader to the address requested by the user.

The implementation example does not include security in the transfer process. The users are expected to change their loader or application to perform the security verification based on their requirements.

The OTA process is described in the application note *Over-the-air application and wireless firmware update for STM32WB series microcontrollers* (AN5247), available on www.st.com. Read these documents for the details of device configuration and OTA procedure. In this user manual, there is only a summary of the procedure, to explain how to use the tool. Read the application note to get detailed information about the target software and the Bluetooth® LE services used.

OTA loader

The OTA loader is the first application that starts at boot or reboot. The OTA loader checks the boot conditions, and if the flash memory is empty.

When the bootloader starts in OTA mode, the loader creates an OTA service and some characteristics required to perform the OTA transfer. These attributes are used to perform the transfer.

The loader fits in the first six sectors of the flash memory. So, the block at the address *0x6000* is free and used to upload user data.

Table 1. OTA loader address table

Flash memory address	Flash memory content
+0x0000	OTA bootloader
+0x1000	OTA bootloader
+0x...	OTA bootloader...
+0x6000	Free for user data
+0x7000	User application
+0x8000	User application
+0x....	User application...

- In the STM32WB sample code, the binary is stored at the *0x7000* address.
- In the STM32WBA sample code, the binary is stored at the *0x7C00* address. The bootloader starts at this address after upload.

3.7.2 OTA procedure

The OTA procedure occurs between one source device and the target device. The process is based on operations:

1. Activate the OTA mode on the target device.
2. Connect in OTA mode and transfer data.

Activation of the OTA mode

The computer sends an indication to the target device to reboot in OTA mode, with the download information. The target restarts in OTA mode and erases the flash memory area required for the transfer.

Connection in OTA mode and data transfer

The source device first connects to the OTA loader and discovers the details of the service and characteristics to be able to transfer the data. Then the sequence is:

1. Configure the target device to send an indication to the source device.
2. Write in the target device the command to initiate the procedure, with the exact storage address.
3. Write each block of data. Depending on the optimized or not MTU size, the blocks are 20- or 248-byte long, and the binary must be transferred in many blocks.
4. At the end of the last block, write the confirmation that all blocks have been sent.
5. The source device waits for the reboot confirmation from the target.

3.7.3 Use the tool to perform an OTA update

The OTA function is available in the device menu in the menu bar. Click on the device and then click on OTA updater.

Search procedure

The first operation is to find the target device. The tool needs to perform a scan of Bluetooth® LE devices and list all the devices with OTA capabilities.

Figure 67. Search procedure

The screenshot shows the 'OTA Updater' window. It features a 'SEARCH FOR DEVICES' button, an 'Advertising filter' checkbox (checked), a 'Select device' dropdown, 'Device type' (STM32WB5x/WB3x), 'Target CPU' (CPU1 : M4), an 'Optimize MTU Size' checkbox (checked), 'Image base address (hex)' (0x7000, Default: 0x7000), 'Image file path' with a 'BROWSE' button, and an 'UPDATE' button at the bottom right.

The tool provides an advertising filter to refine the search procedure with an advertising message.

Table 2. Search filtering

Filter	Search method	Comment
No filter	Scans all Bluetooth® LE devices and provides the list.	Some devices listed are not compatible with OTA.
Advertising filter	Scans all Bluetooth® LE devices and provides a list of devices with ST OTA information.	Gives only the list of compatible devices.

To start the search, click on the *SEARCH FOR DEVICES* button.

Figure 68. Scanning

The search procedure starts.

If no target device is found, the tool indicates *No device found*.

Figure 69. No device found

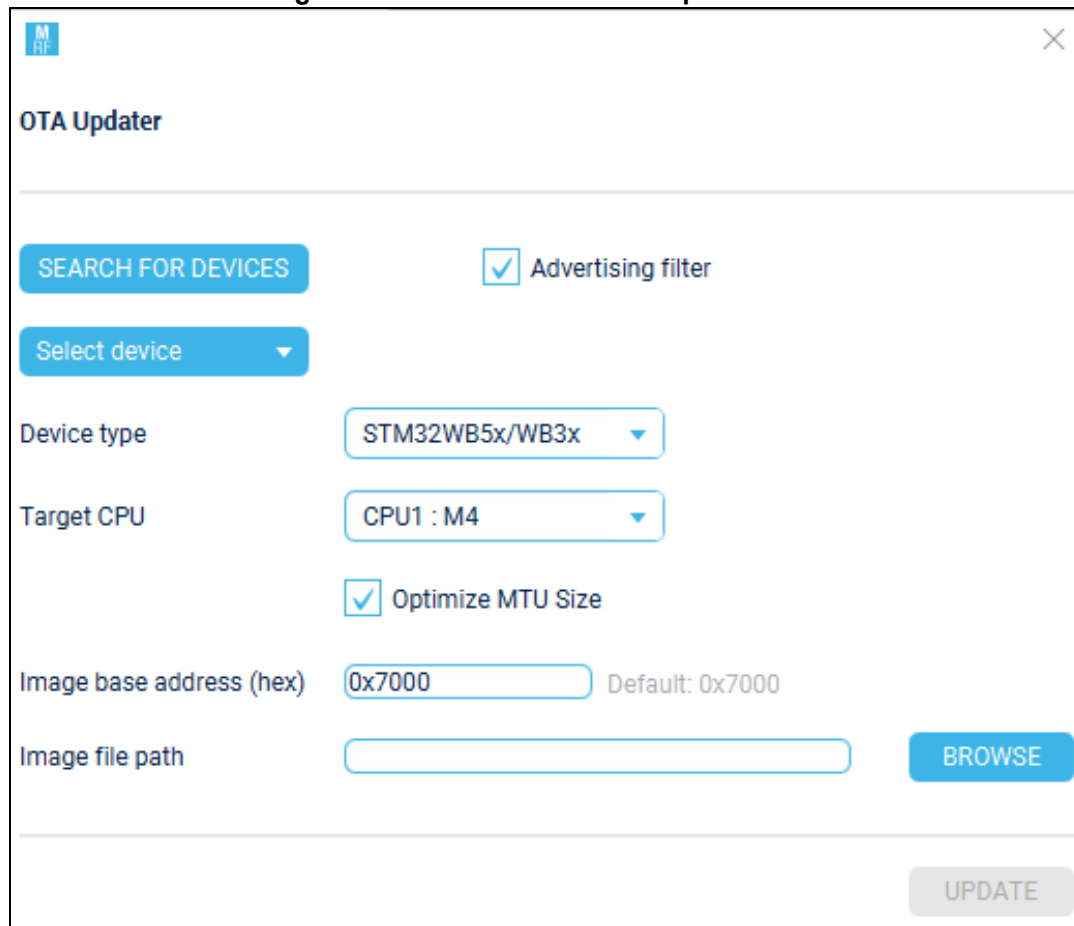
If a candidate device is found, the select device box changes to blue.

Figure 70. Device found

Device and parameters selection

After the search procedure, if one or more devices are found, the user selects the device with the picklist *Select device*.

Figure 71. Select the device and parameters



The screenshot shows a window titled "OTA Updater" with a close button in the top right corner. Inside the window, there is a "SEARCH FOR DEVICES" button and a checked checkbox for "Advertising filter". Below these is a "Select device" dropdown menu. Further down, there are two dropdown menus: "Device type" set to "STM32WB5x/WB3x" and "Target CPU" set to "CPU1 : M4". There is also a checked checkbox for "Optimize MTU Size". Below these are two input fields: "Image base address (hex)" with the value "0x7000" and a note "Default: 0x7000", and "Image file path" with an empty text box and a "BROWSE" button to its right. At the bottom right of the window is an "UPDATE" button.

The picklist displays the list of boards found:

- For a device with Bluetooth® LE characteristics:
Bluetooth® LE address - Device name - OTA enabled
- For a device already in OTA mode:
Bluetooth® LE address - Device name - OTA loader

Select the firmware target:

- For user data or user application firmware, select the *CPU1: M4*
- For the wireless stack, select the *CPU2: M0+*

Select the device type:

- STM32WB5x/WB3x product lines
- STM32WB1x product line. Note that for this device type, only *Target CPU1: M4* can be updated.
- STM32WBAxx product line. Note that for this device type, only *Target CPU1* can be updated

The *Optimize MTU size* option allows the user to increase the ATT maximum transmission unit (MTU) from 20 to 248 bytes.

The image base address is the place where the binary file must be stored on the target device. It is a hexadecimal value and must be a multiple of *0x800* for STM32WB5x/WB3x/WB0x, *0x1000* for STM32WB1x, or *0x2000* for STM32WBx to match with the flash memory sector. For the wireless stack, the address is the temporary location in the CPU1 user part area. The default address value is given aside.

The image file path is the binary file to load. Enter the path in the box, or use the BROWSE button to select the file to download.

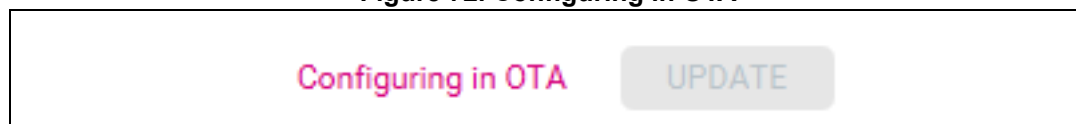
The configuration is finished and the software is ready to start the update procedure.

Flashing the remote device

Press the *UPDATE* button to start flashing the target device.

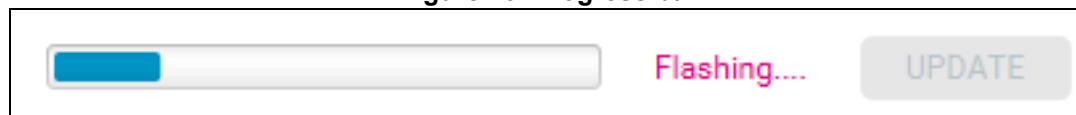
1. First step: if the selected device has an OTA characteristic, the tool first restarts the device in OTA bootloader mode. The indication *Configuring in OTA* is displayed.

Figure 72. Configuring in OTA



2. Second step: the transfer process to the OTA bootloader is performed. The data are transferred in blocks of 20 bytes. To avoid the overload of log windows, the log information related to block transfer is not displayed. Only the flow control event and the errors are shown.
3. A progress bar monitors the memory load.

Figure 73. Progress bar



At the end of the update process,

4. The target device reboots.
5. The user closes the OTA panel or starts a new search to flash another device.

3.8 Beacon

3.8.1 Beacon presentation

A beacon is an active device discoverable by other devices.

The beacon device only sends information by advertisement and does not receive any data.

The data shared by the beacon are very small. A connected device receives them and the application on the device is notified of a beacon presence. The application uses the cloud to get more information and act accordingly.

Figure 74. Beacon presentation



When an application is informed of beacon proximity, it uses the beacon identification to request the web server more information about the beacon. The application gets information related to the geographical position of the beacon or the action to perform, like displaying commercial ads or starting an interactive application.

Figure 75. Beacon usage



Many organizations have created beacons. The specifications from Apple® and Google® are frequently used:

- iBeacon: This is the Apple format. The beacon broadcasts fixed content, to identify the beacon easily.
- Eddystone UID: Google defines it. The beacon transmits fixed content (UID), the box that is a unique ID, referenced in the Google database to interact with applications.
- Eddystone URL: Another Google format. It provides a short URL to use for the *Physical web*.
- Eddystone TLM: An additional beacon advertising information providing beacon information (battery status, temperature).
- Eddystone EID: Like UID, but broadcasts encrypted data to provide better security.

3.8.2 Beacon configuration methods:

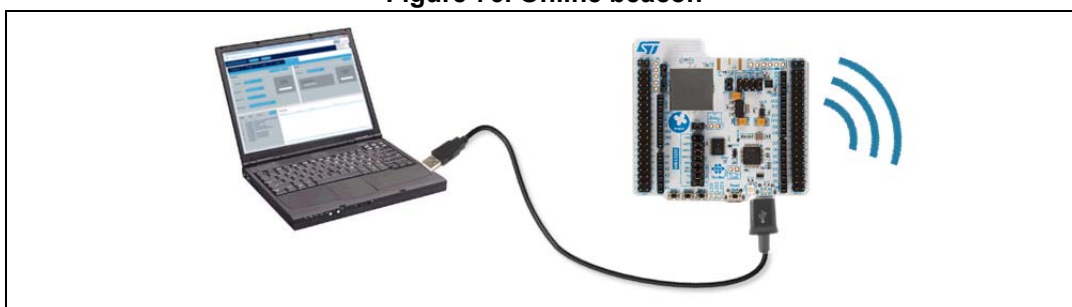
STM32CubeMonitor-RF is used to generate and configure beacons. Different methods have been defined to accommodate the user's needs. This chapter describes the different methods supported.

Online beacon

In online mode, the tool is directly configuring the main device in a beacon. The tool sends ACI commands to configure the boards in the Advertising mode and configure the content of the advertising packet. The main device acts as a beacon until turned off.

The main advantage of this method is to configure a beacon with a board in Transparent mode quickly. The drawback is that the configuration is lost when the board is reset or powered off.

Figure 76. Online beacon

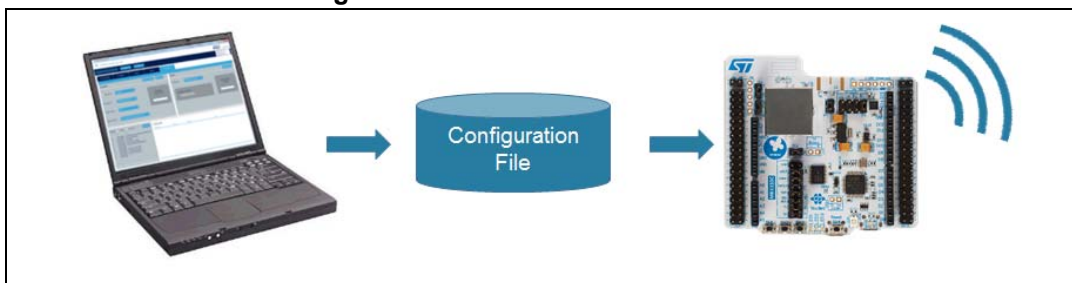


Offline beacon

The Offline beacon mode is used to prepare the configuration of a board not directly connected to the STM32CubeRF-Monitor. The parameters to configure the beacon are stored in a data file. The file is used to configure a target board running the beacon example firmware. The file must be stored in the target flash memory at the 0x6000 address. The beacon firmware reads the data and configures the advertising block accordingly. Details of the configuration file are described in [Table 4](#).

The interest of the method is to have an independent beacon, which is useful if the user needs many beacon boards at the same time. It is possible to keep many configuration files to change the configuration quickly. The drawback is that the configuration file must be transferred manually to the target device so it is less flexible than the Online mode.

Figure 77. Offline beacon



Selecting the Beacon mode

The selection of the configuration mode is the first action to prepare the beacon. The user must choose the mode when selecting the beacon tab.

Figure 78. Selecting the Beacon mode

ACI Commands

Scripts

Beacon

RF Tests

ACI Utilities

Configuration mode

Select configuration mode

☒ Directly configure device through wired connection (On line)

☐ Generate configuration and update device through wired connection (Off line)

SELECT CONFIGURATION MODE

Select one of the two bullets and click on *SELECT CONFIGURATION MODE*.

3.8.3 Configuration of the beacon with STM32CubeMonitor-RF

To configure the beacon,

- 1. Select the configuration method.
- 2. Fill in the beacon parameters, some are common for all beacons, and others are specific for the beacon type.
- 3. Generate/transfer the configuration. Additional information might be required according to the configuration method.

Common parameters

Some beacon parameters are common for all kinds of beacons. The common parameters are at the top of the beacon panel:

Figure 79. Common parameters

Parameters

Reference TX power level (dBm)

-56

PA Level

31 (+6dBm)

Beacon Address

123456789AAA

☒ Public Address

☐ Random Address

The first parameter is the *Reference TX power level*, and the second parameter is the real *TX power level*.

To save batteries, the power level of the beacon might be lowered, reducing consumption and visibility. Using high power extends the range of visibility but drains more power. The user needs to define the power level based on the power source and beacon purpose.



The device detecting the beacon needs to estimate if the beacon is close or far. Unfortunately, the received power level is not enough to estimate the real distance:

- Some beacons might transmit with high power, while others are using low power.
- The design of the beacon antenna might be efficient.

The reference power information is added to help determine the distance. This is the power level received at one meter from the beacon. The application uses this value and the received strength to estimate the distance, independently of the real TX power used and the beacon characteristics.

The easiest solution to fill this parameter is to configure a beacon with the required Tx level, and then measure the received level at one meter. Then the beacon is reconfigured with the value measured at one meter in the *Reference TX power level* field.

The second set of parameters is the beacon address. There are three possibilities:

- Set the address in the box and tick the *public address* to use the entered address.
- Tick the *random address* checkbox. A random address is used.
- If nothing is selected, the default public address of the board is used.

iBeacon parameters

Figure 80. iBeacon parameters

First, select the type: iBeacon (default choice)

The users must check the Apple website for information about the iBeacon structure and the conditions to use iBeacon for their project: <https://developer.apple.com/ibeacon/>

More information is also available at

<https://en.wikipedia.org/wiki/iBeacon>. <https://en.wikipedia.org/wiki/iBeacon>.

The company code is a value based on the Bluetooth® LE SIG group-assigned values. For iBeacon, the Apple value is used: *0x004C*. The assigned values are available on the SIG website: <https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers>

The beacon UUID is the unique identifier for a group of beacons. Apple explains how to define the identifier in the document *Getting started with iBeacon* available at <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>.

The user defines the major and minor codes to identify logically different beacons sharing the same UUID.

When all parameters are updated, click on *CONFIGURE*. The data are ready for transfer (refer to [Configuration transfer](#)).

Eddystone UID parameters

Figure 81. Eddystone UID

The Eddystone UID parameters are the beacon UID, a 16-byte identifier, formed by:

- The *Name Space*, 10 bytes used to group some beacons in a logical pool. Google describes the way to generate the value. Refer to <https://github.com/google/eddystone/tree/master/eddystone-uid>.
- The *Beacon Instance*, 6 bytes gives a unique ID inside the pool.

When a beacon is discovered on a smartphone, the UID value is not directly usable by the phone application. Google offers a cloud service to associate one or more data with a beacon. The smartphone application retrieves this information to perform the required actions.

The last option is the *Enable TLM* tick box. When TLM is used, the beacon interleaves some status information inside the normal beacon advertisement. The TLM frame has information about battery level, temperature, the time the beacon is on, and the number of frames transmitted. The TLM information is not known by the tool. So, the firmware must manage directly by itself. Consequently:

- The TLM option is not used for the Online configuration mode.
- For Offline and OTA modes, a bit is set in the configuration file (refer to [Appendix A](#)).

Eddystone URL parameters

Figure 82. Eddystone URL

The Eddystone URL format is just sending a URL in the advertising message. To optimize space, the start and end of the URL might be compressed.

1. Select the URL prefix: the prefix is encoded in 1 byte in the advertising.
2. Fill in the rest of the URL in the URL box, without a prefix. The URL is analyzed, and if the end of the URL is encodable, the tool encodes it. A long URL does not work, so ST recommends using the URL short service to obtain a short URL.

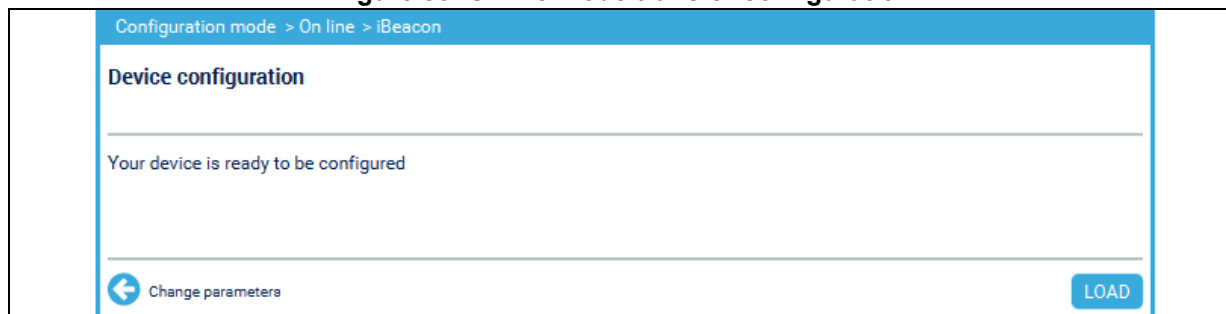
The TLM option is the same as the UID beacon.

Configuration transfer

The transfer depends on the selected configuration mode.

1. Online mode transfer configuration

Figure 83. Online mode transfer configuration

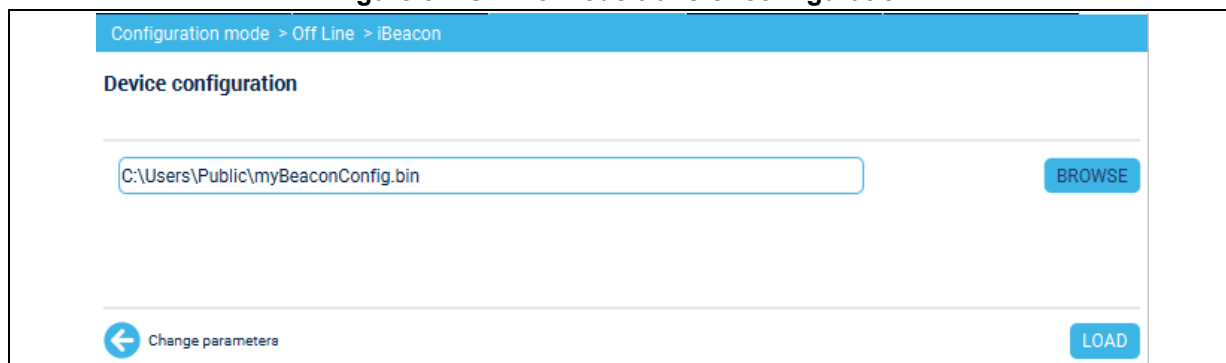


The screenshot shows a web interface for online mode transfer configuration. At the top, a blue header bar contains the text "Configuration mode > On line > iBeacon". Below this, the section is titled "Device configuration". A message states "Your device is ready to be configured". At the bottom, there is a "Change parameters" button with a left arrow icon and a "LOAD" button.

No extra parameters are required: just click on *LOAD* and the main device is initialized and configured in the beacon.

2. Offline mode transfer configuration

Figure 84. Offline mode transfer configuration



The screenshot shows a web interface for offline mode transfer configuration. At the top, a blue header bar contains the text "Configuration mode > Off Line > iBeacon". Below this, the section is titled "Device configuration". There is a text input field containing the path "C:\Users\Public\myBeaconConfig.bin" and a "BROWSE" button to its right. At the bottom, there is a "Change parameters" button with a left arrow icon and a "LOAD" button.

Use the *BROWSE* button to select the path and file to create. Copy the file to the target device using a flash memory programmer or another tool.

3.9 ACI Utilities

The *ACI Utilities* panel is used to configure the device to perform either the advertising signal or to discover remote devices and explore their services and characteristics.

Figure 85. ACI Utilities panel

ACI CommandsScriptsBeaconRF TestsACI Utilities

Init

Initialization parameters

☐ Discover remote services

☐ Advertising

Address

0x112233445566

Power

31 (+6dBm)

Name

STM32WB

Discoverability mode

General discoverable

Adv type

0x00 - ADV_IND (Connectable undirected advertising)

Advertising channel map

☒ CH37☐ CH38☐ CH39

Own address type

0x00 - Public Device Address

Advertising interval (20 to 10240 ms)

1280

Min

1280

Max

Slave connection interval (7.5 to 4000 ms)

MinMax

Use empty value for non specific Min/Max

SCAN

START ADVERTISING

The first action is to select to discover remote services, manage to advertise, or both, by clicking the appropriate checkbox.

Figure 86. Select checkbox

☐ Discover remote services

☐ Advertising

3.9.1 Remote services discovering

The remote service discovery performs a scan of the remote devices in the area.

Figure 87. Scan parameters

The screenshot shows the 'Init' tab under 'ACI Utilities'. The 'Initialization parameters' section is active. It contains the following fields and controls:

- ☒ Discover remote services
- ☐ Advertising
- Address: 0x112233445566
- Power: 31 (+6dBm)
- Name: STM32WB
- Discoverability mode: General discoverable
- Adv type: 0x00 - ADV_IND (Connectable undirected advertising)
- Advertising channel map: ☒ CH37, ☐ CH38, ☐ CH39
- Own address type: 0x00 - Public Device Address
- Advertising interval (20 to 10240 ms): 1280 Min 1280 Max
- Slave connection interval (7.5 to 4000 ms): [] Min [] Max Use empty value for non specific Min/Max

At the bottom right, there is a blue 'SCAN' button and a greyed-out 'START ADVERTISING' button.

To perform a scan of the available devices:

1. Enter the device address.
2. Select the power level with the picklist.
3. Enter the device name.
4. Click on the **SCAN** button to start the discovery.

The search procedure starts, and it is possible to stop it using the **STOP** button.

Figure 88. Scanning



If no remote device is found, the tool indicates *No device found*. Otherwise, the user chooses one of the devices found in the *Select Device* box.

Figure 89. Select device



At this stage, the user performs another scan procedure upon request.

- Click on the back button

Figure 90. Back



- Click on the *Init* in the top bar

Figure 91. Init



Or connect to the selected remote device, by clicking on the *CONNECT* button.

Figure 92. Connecting



If the connection fails, an error is displayed.

Figure 93. Connection error

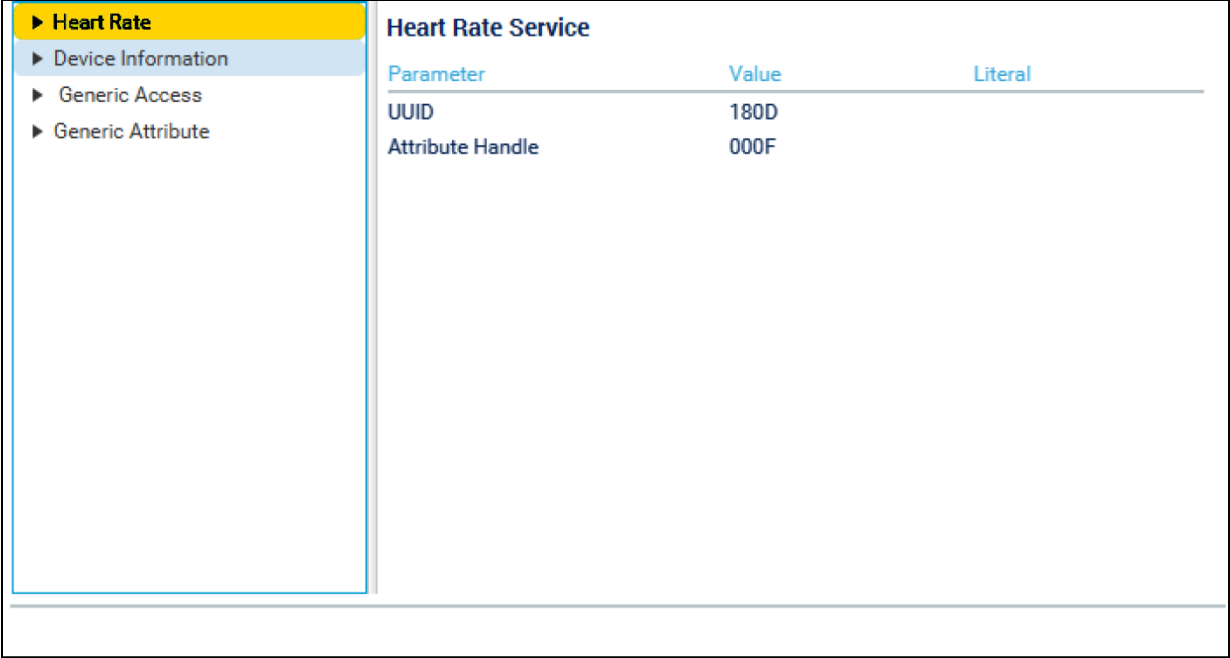


Once connected, the connect icon appears in blue and the list of available services is proposed.

Figure 94. Connected icon



Figure 95. Services list



When the user selects a service, its details are displayed. Clicking on the arrow displays the characteristics linked to the above service.

Figure 96. Characteristics list

<div>▼ Heart Rate</div> <div>Heart Rate Control Point</div> <div>Body Sensor Location</div> <div>Heart Rate Measurement</div> <div>▶ Device Information</div> <div>▶ Generic Access</div> <div>▶ Generic Attribute</div>	Heart Rate Service		
	Parameter	Value	Literal
	UUID	180D	
	Attribute Handle	000F	

The user can select a parameter and, depending on it, read, or write a value and be notified of the value change. Note that read and write long characteristics are not supported, or authenticated signed write.

To read a value, the user clicks on the *READ* button.

Figure 97. Read value

<div>▼ Heart Rate</div> <div>Heart Rate Control Point</div> <div>Body Sensor Location</div> <div>Heart Rate Measurement</div> <div>▶ Device Information</div> <div>▶ Generic Access</div> <div>▶ Generic Attribute</div>	Body Sensor Location Characteristic		
	Parameter	Value	Literal
	UUID	2A38	
	Handle	0013	
	Properties	02	Read
	Value handle	0014	
	Value length	8bit	
	Value	0x04	
			READ

To write a value, the user enters the new value and clicks on the *WRITE* button.

Figure 98. Write value

Parameter	Value	Literal
UUID	2A39	
Handle	0015	
Properties	08	Write
Value handle	0016	
Value length	8bit	
Value	123456	

WRITE

There are two ways to be informed of a value change: either via the indicated method or a notification, depending on the method property supported by the remote device.

To receive an indication of value change, the user can click on the *INDICATE* button.

Figure 99. Indicate value changed

Parameter	Value	Literal
UUID	2A05	
Handle	0002	
Properties	20	Indicate
Value handle	0003	
Value length	uint16	
Value		
Client Characteristic Configuration UUID	2902	
Client Characteristic Configuration handle	0004	

INDICATE

To receive a notification upon value change, the user can click on the *NOTIFY* button.

Figure 100. Notify value changed

Parameter	Value	Literal
UUID	2A37	
Handle	0010	
Properties	10	Notify
Value handle	0011	
Value length	8bit	
Value		
Client Characteristic Configuration UUID	2902	
Client Characteristic Configuration handle	0012	

NOTIFY

Upon each change, a notification (resp. indication) is received and the new value is displayed. The user can be informed of multiple characteristic value changes at the same time. To stop the notification (resp. indication), the user can click on the *UN-NOTIFY* button (resp. *UN-INDICATE*).

On disconnection, all registered notifications are removed.

Figure 101. Notifying

ACI Commands | Scripts | Beacon | RF Tests | **ACI Utilities**

Init > scanner

Found devices

0x112233445566 - STM32WB

DISCONNECT

Heart Rate Measurement Characteristic

Parameter	Value	Literal
UUID	2A37	
Handle	0010	
Properties	10	Notify
Value handle	0011	
Value length	8bit	
Value	0x0400FC9000451F	
Client Characteristic Configuration UUID	2902	
Client Characteristic Configuration handle	0012	

UN-NOTIFY

ACI log ☒ Update ☒ Autoscroll SAVE LOG RESET LOG

No	Time	Type
4270	16:01:52.555	LE Meta Event
4271	16:01:52.556	LE Meta Event
4272	16:01:52.558	LE Meta Event
4273	16:01:52.559	LE Meta Event
4274	16:01:52.563	LE Meta Event
4275	16:01:52.575	LE Meta Event
4276	16:01:52.576	LE Meta Event
4277	16:01:52.579	LE Meta Event
4278	16:01:52.584	LE Meta Event
4279	16:01:52.589	LE Meta Event
4280	16:01:52.594	LE Meta Event
4281	16:01:52.595	LE Meta Event
4282	16:01:52.596	Vendor Specific Event
4283	16:02:07.983	ACL_GAP_CREATE_CONNECTION
4284	16:02:07.988	Command Status
4285	16:02:14.369	Vendor Specific Event
4286	16:02:14.371	LE Meta Event
4287	16:02:14.396	ACL_GATT_DISC_ALL_PRIMARY_SERVICES
4288	16:02:14.399	Command Status
4289	16:02:14.563	Vendor Specific Event
4290	16:02:14.722	Vendor Specific Event
4291	16:02:14.723	Vendor Specific Event
4292	16:02:14.725	ACL_GATT_DISC_ALL_CHAR_OF_SERVICE
4293	16:02:14.728	Command Status
4294	16:02:14.964	Vendor Specific Event
4295	16:02:15.123	Vendor Specific Event
4296	16:02:15.283	Vendor Specific Event
4297	16:02:15.284	Vendor Specific Event
4298	16:02:15.286	ACL_GATT_DISC_ALL_CHAR_DESC
4299	16:02:15.290	Command Status
4300	16:02:15.445	Vendor Specific Event
4301	16:02:15.605	Vendor Specific Event
4302	16:02:15.763	Vendor Specific Event
4303	16:02:15.763	Vendor Specific Event
4304	16:02:57.304	ACL_GAP_TERMINATE
4305	16:02:57.307	Command Status
4306	16:02:57.364	Disconnection Complete
4307	16:03:00.420	ACL_GAP_CREATE_CONNECTION
4308	16:03:00.425	Command Status

3.9.2 Advertising

Figure 102. Advertising parameters



ACI Commands	Scripts	Beacon	RF Tests	ACI Utilities
Init				
Initialization parameters				
<div> <input type="checkbox"/> Discover remote services <input checked="" type="checkbox"/> Advertising </div>				
Address		0x112233445566		
Power		31 (+6dBm)		
Name		STM32WB		
Discoverability mode		General discoverable		
Adv type		0x00 - ADV_IND (Connectable undirected advertising)		
Advertising channel map		<input checked="" type="checkbox"/> CH37 <input type="checkbox"/> CH38 <input type="checkbox"/> CH39		
Own address type		0x00 - Public Device Address		
Advertising interval (20 to 10240 ms)		1280 Min 1280 Max		
Slave connection interval (7.5 to 4000 ms)		Min Max Use empty value for non specific Min/Max		
				<div> <div>SCAN</div> <div>START ADVERTISING</div> </div>

To activate the Advertising mode:

1. Enter the device address.
2. Select the power level with the picklist.
3. Enter the device name.
4. Select the advertising type with the picklist.
5. Select at least one channel from 37, 38, and 39.
6. Enter the advertising interval.
7. Enter an optional target connection interval.
8. Click on the *START ADVERTISING* button to start the procedure.



The search procedure starts, and the advertising icon appears in blue. It is possible to stop it using the *STOP ADVERTISING* button.

Figure 103. Advertising

ACI Commands	Scripts	Beacon	RF Tests	ACI Utilities
Init				
Initialization parameters  				
<div> <input type="checkbox"/> Discover remote services <input checked="" type="checkbox"/> Advertising </div>				
Address <input type="text" value="0x112233445566"/>				
Power <input type="text" value="31 (+6dBm)"/>				
Name <input type="text" value="STM32WB"/>				
Discoverability mode <input type="text" value="General discoverable"/>				
Adv type <input type="text" value="0x00 - ADV_IND (Connectable undirected advertising)"/>				
Advertising channel map <input checked="" type="checkbox"/> CH37 <input type="checkbox"/> CH38 <input type="checkbox"/> CH39				
Own address type <input type="text" value="0x00 - Public Device Address"/>				
Advertising interval (20 to 10240 ms) <input type="text" value="1280"/> Min <input type="text" value="1280"/> Max				
Slave connection interval (7.5 to 4000 ms) <input type="text"/> Min <input type="text"/> Max Use empty value for non specific Min/Max				
<div> <input type="button" value="SCAN"/> <input type="button" value="STOP ADVERTISING"/> </div>				

The connect icon might appear in blue if a remote device connects. In that case, advertising stops.

Figure 104. Connected

Initialization parameters  

4 OpenThread mode

4.1 Presentation

4.1.1 Panel

The OpenThread main panel is organized with three tabs, *Commands*, *Scripts*, and *Network Explorer*.

Figure 105. OpenThread - Command tab

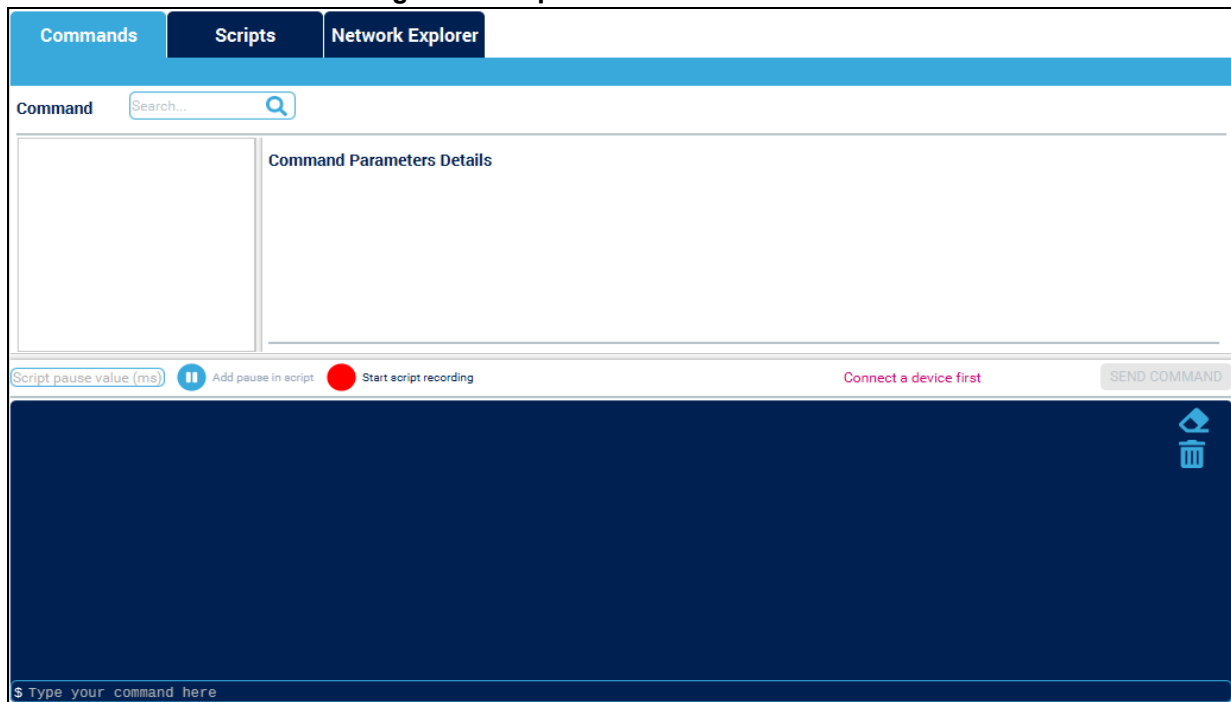
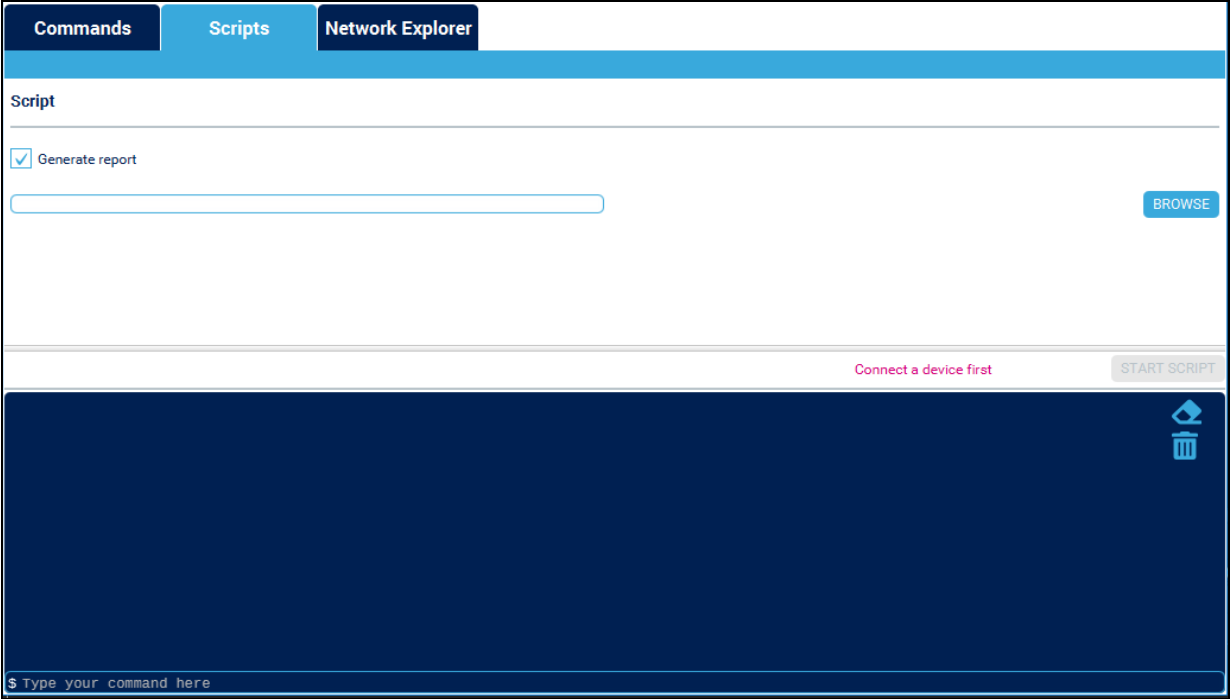


Figure 106. OpenThread - Script tab



The first two tabs have one common bottom area, the terminal area.

Figure 107. OpenThread - Network Explorer tab

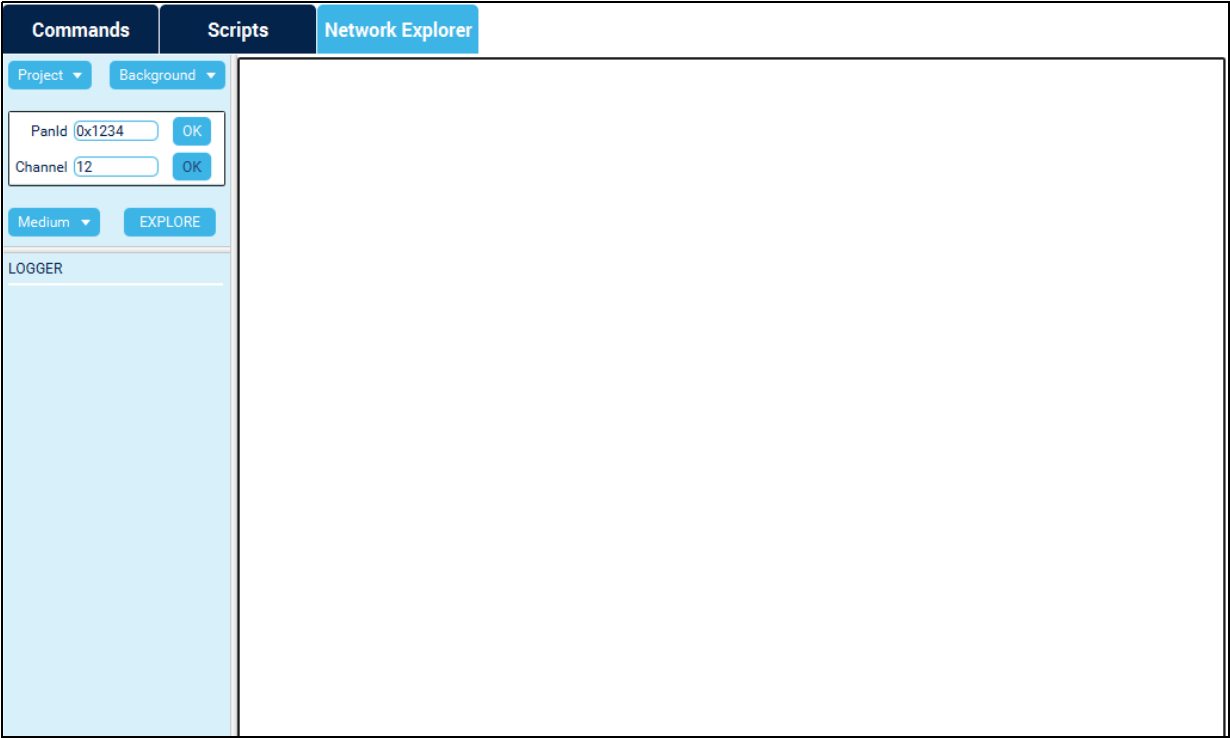
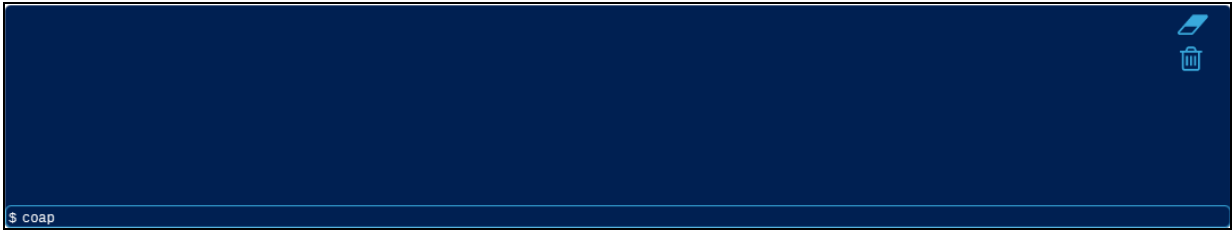


Figure 108. OpenThread common bottom area



The terminal area is used to show the messages exchanged between the application and the target. The commands sent to the target can be seen and the responses received from the target. Those messages can be cleared with the rubber icon.

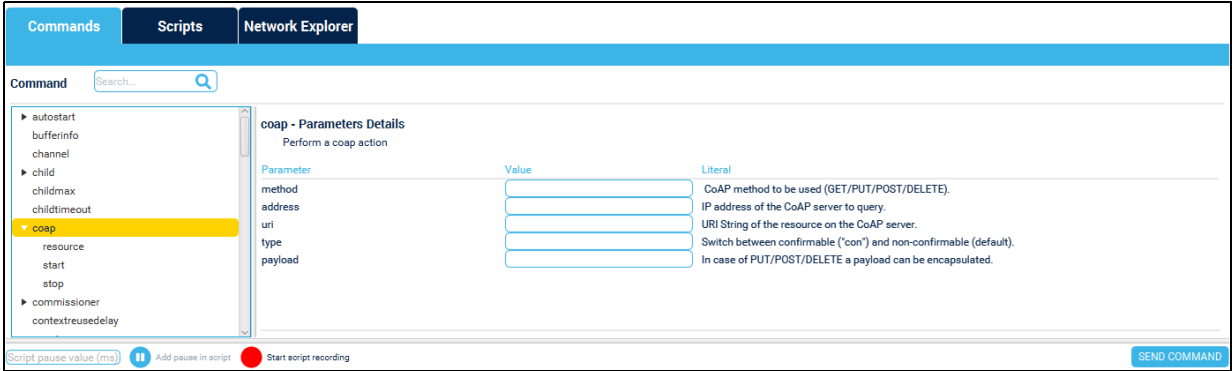
The bottom line with a \$ character is a command line. The user types the command with the parameters and presses <Enter> to send the command. The commands sent with this line are recorded in the history file and can be recalled with up and down arrows. This history can be deleted with the trash icon.

One other way is using the commands list and parameter area to fill the line, then the user can modify the line and send a command with the entering key. The commands list and parameters area are described in the chapter Commands tab.

4.2 Commands tab

This tab is dedicated to the OpenThread commands and parameters. The top area gives access to the commands list and parameters. Some commands can be used to read and send values, others are only commands sent to the OpenThread stack.

Figure 109. OpenThread command tab top area



For commands used to send data, the *SEND COMMAND* button sends the command with parameters to the target.

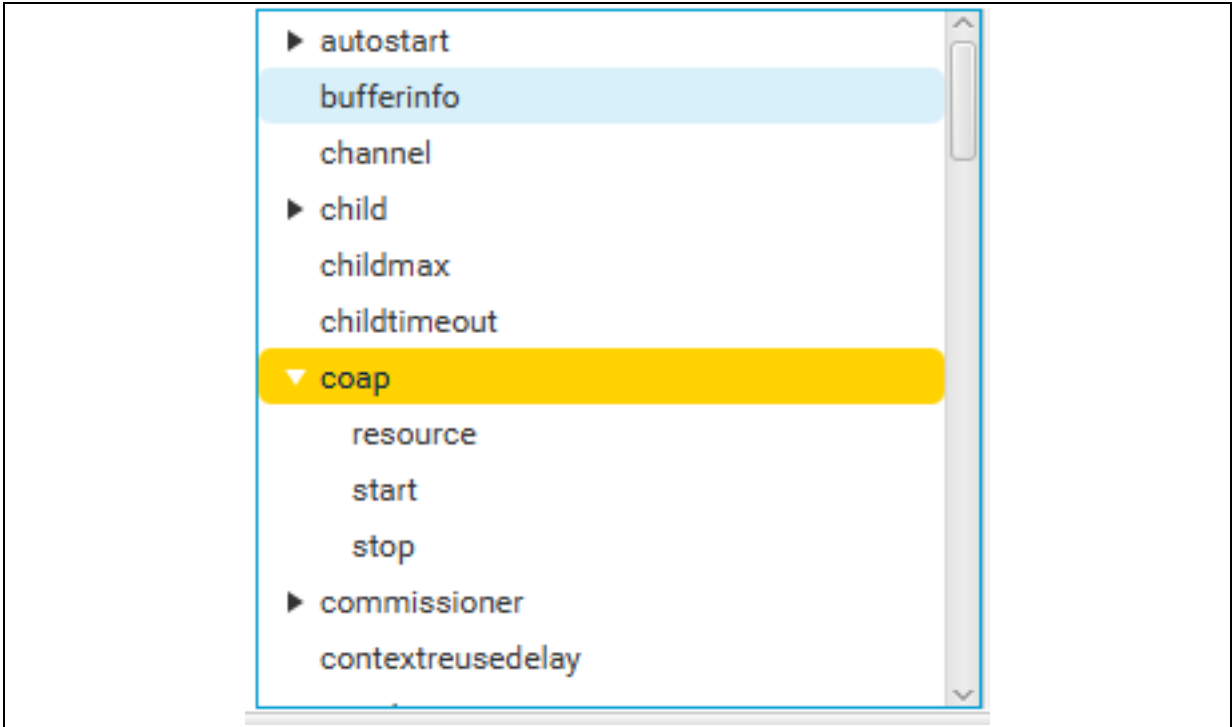
For commands able to read information, two buttons are available: *READ* and *SEND COMMAND*. The *READ* button sends the command without parameters to read the value. The *SEND COMMAND* button sends the command with parameters to the target.

Figure 110. Script buttons



The *Start script recording* and *Add pause in script* buttons allow saving a script. This part is described in [Section 4.3: OpenThread scripts tab](#).

Figure 111. Command list



The command list is arranged in alphabetical order, and accessible from the tree, for example below the *coap* command, there are *coap resource*, *coap start*, and *coap stop* commands.

Figure 112. Command details

coap - Parameters Details

Perform a coap action

Parameter	Value	Info
method	<input type="text"/>	Coap method
address	<input type="text"/>	Coap target address
uri	<input type="text"/>	Coap uri
payload	<input type="text"/>	Coap payload
type	<input type="text"/>	Connection type

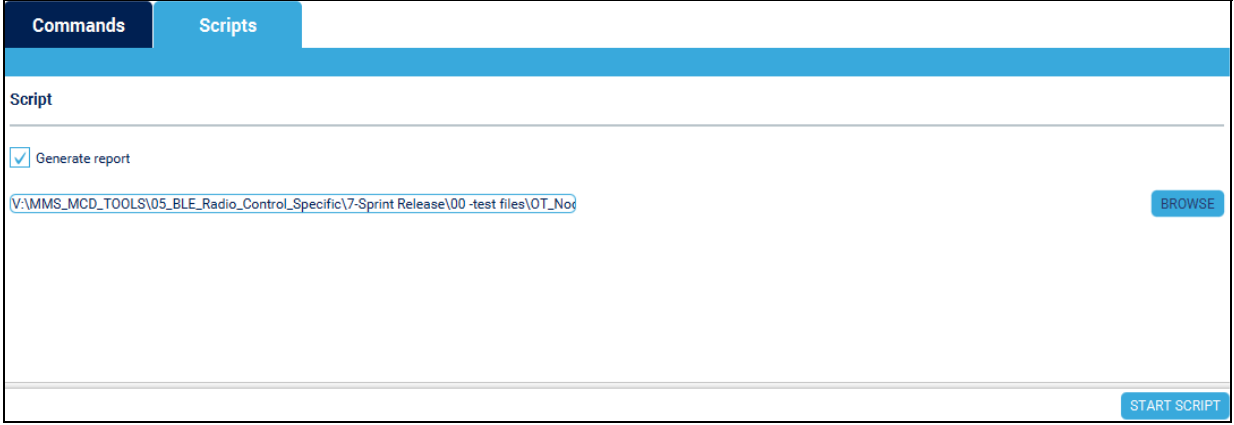
The command name and the definition are in the upper part of the command details area. Below is one table of parameters with the parameter name, there is one writable field to define the value and information concerning this parameter.

Note that when sending a reset command on an STM32WB USB dongle, the VCP device is reset and disconnects. A new connection procedure is needed to perform further tests.

4.3 OpenThread scripts tab

The OpenThread scripts tab is used to launch the script stored in a text file.

Figure 113. Scripts tab



The screenshot shows the 'Scripts' tab in a software interface. At the top, there are two tabs: 'Commands' and 'Scripts'. The 'Scripts' tab is selected and highlighted in blue. Below the tabs, there is a section labeled 'Script'. Inside this section, there is a text area containing the file path: `V:\MMS_MCD_TOOLS\05_BLE_Radio_Control_Specific\7-Sprint Release\00 -test files\OT_No`. To the right of the text area is a blue button labeled 'BROWSE'. Below the text area, there is a large empty space. At the bottom right of the 'Script' section, there is a blue button labeled 'START SCRIPT'.

The scripts use the same syntax as the Bluetooth® LE scripts. The OpenThread specificities are described in this chapter. Consult the Bluetooth® LE script description in [Section 3.5](#) for general information.

4.3.1 OpenThread script example

Figure 114. Sample script

```
#STM32CubeMonitor-RF sample script
# OpenThread ping node script

#Pause command
Pause ("Ready to start the test")

#Send reset command:
Send (reset)

#Set channel
Send (channel 11)

#Set the PAN ID:
Send (panid 0x1234)

#Bring up the IPv6 interface:
Send (ifconfig up)

#Start Thread protocol operation:
Send (thread start)

#Wait for a few seconds and verify that the device has become a
Thread leader:
wait (5000)

#Check state
Send (state)

#Ipaddr
Send (ipaddr)
```

4.3.2 List of script commands

The OpenThread scripts use the same commands as Bluetooth® LE, but the *Send* command is modified to send Thread commands.

The OpenThread commands are sent with the *Send* instruction:

```
Send (OPENTHREAD_CMD_NAME Parameter1Value Parameter2Value).
```

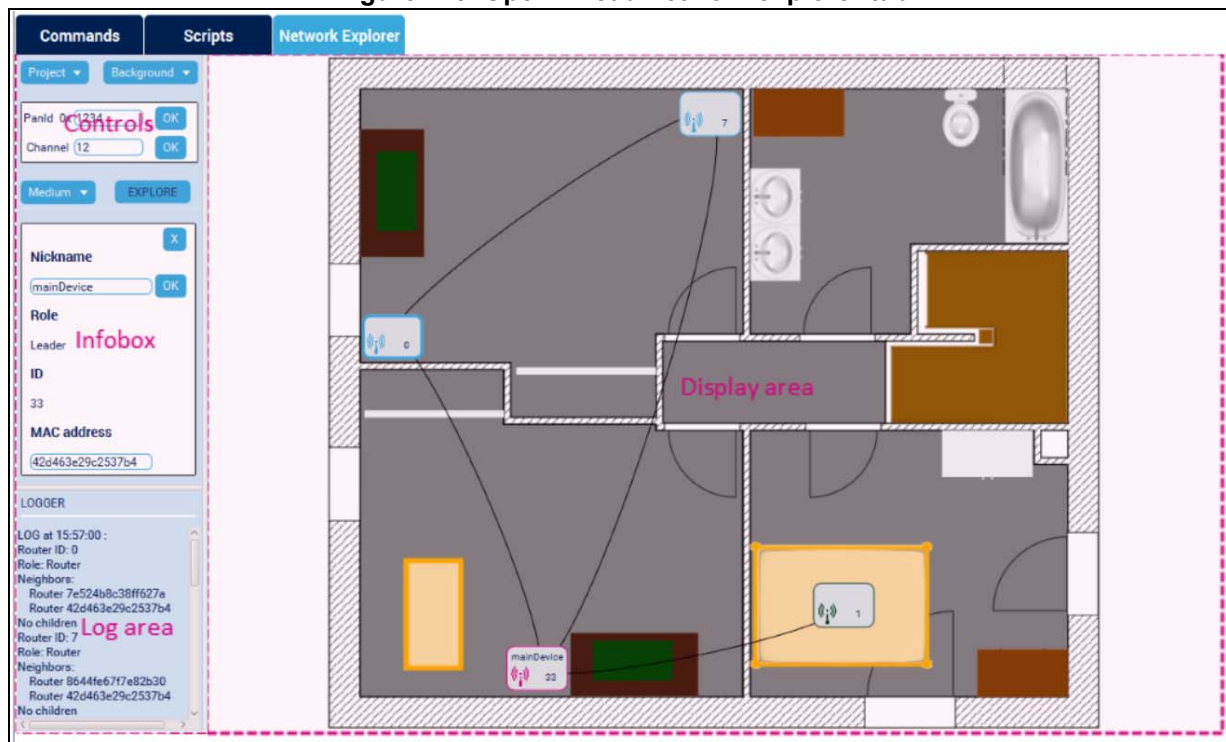
The part inside the brackets is the command line to send.

4.4 Network Explorer tab

This feature can only be used if the DUT has the *Thread_Cli_cmd* firmware to be able to copy data from the UART to the OpenThread command-line interpreter. Refer to [Section 2.2.1](#) for further information about firmware.

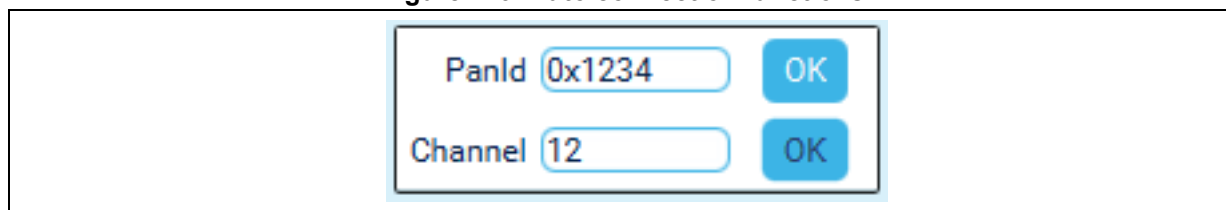
This tab is dedicated to the exploration and display of the network to which the DUT device is attached. The representation of the network is displayed in the central area. There are some basic control functions in the up-left corner of the pane. Just below, there is information on the selected node in an infobox plus logs of the exploration.

Figure 115. OpenThread network explorer tab



4.4.1 Controls

Figure 116. Auto connection functions



The network explorer tab easily configures the panId and channel of the DUT device.

The panId is entered in hexadecimal format, with no need to specify *0x*. The value must be contained between *0x0000* and *0xffff*. The value *0xffff* corresponds to an unconfigured panId.

The channel is defined in decimal format and must be contained in [11;26]. The panId must be configured before configuring the channel. As from the STM32WB V1.14 version, the network key is initialized to a random value, so this parameter is set to *00112233445566778899aabbccddeeff*. It is the default controller key used by the previous firmware.

For both parameters, if the filled value is in the wrong format, nothing is changed, and the actual value of the device remains displayed. Moreover, if a network exploration is ongoing neither parameter can be changed.

At the first DUT connection or when switching to the network explorer tab, the tool checks the current values of both parameters and displays them in the fields as information.

Figure 117. Project and background management



The two menu buttons on the top of the control area give control to the project itself and the background image.

The *Project* menu proposes three choices:

1. *New choice* cleans the current session by resetting the display area and stopping the ongoing exploration if there is one.
2. *Open choice* opens a file explorer to choose a backup of a project to use in the session. When a project is loaded from the *Open choice* box, there is a two-step process:
 - The saved image is first restored as the background of the right area.
 - When a scan is started, if a saved project device is detected, it is instantly displayed in its last place with its former nickname. This association is based on the unique MAC addresses of the devices.
3. *Save choices* saves the current project. In this backup, there is the background image, the location of the icons on that image, and the nicknames of the devices.

The *Background* menu allows either to:

- Remove the background image.
- Open a file explorer to put an image in the display area as a background.

Figure 118. Explore and size choice controls

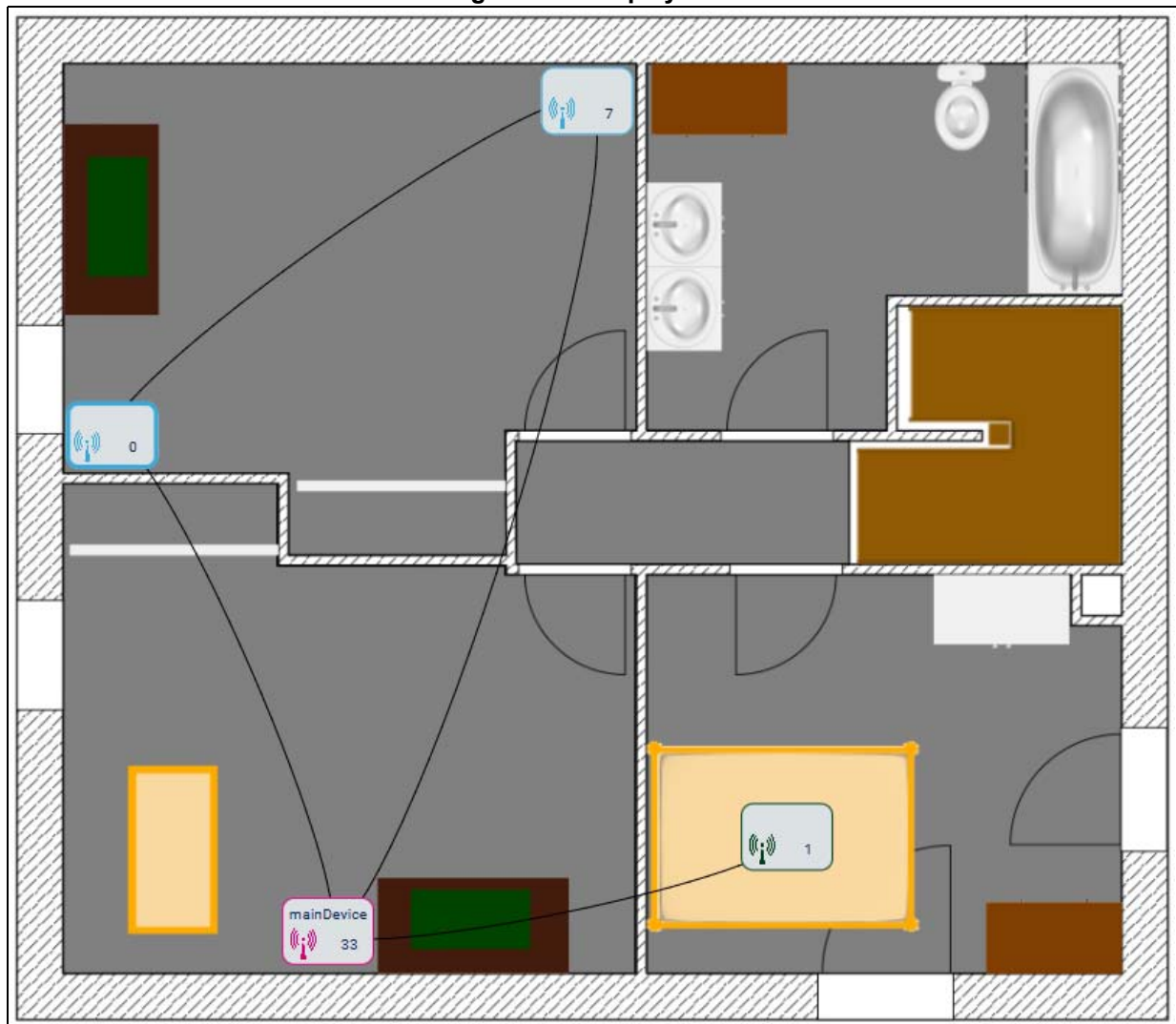


Once the DUT device is connected to a Thread network, the EXPLORE button starts the network exploration sequences. It turns to *STOP* when the exploration is ongoing.

The *Choice box* at the left of the EXPLORE button chooses the size of the icon between three standard sizes: *Small*, *Medium*, and *Large*. It can be changed at any time. The size of the icons is adapted according to the dimensions of the background image.

4.4.2 Display area

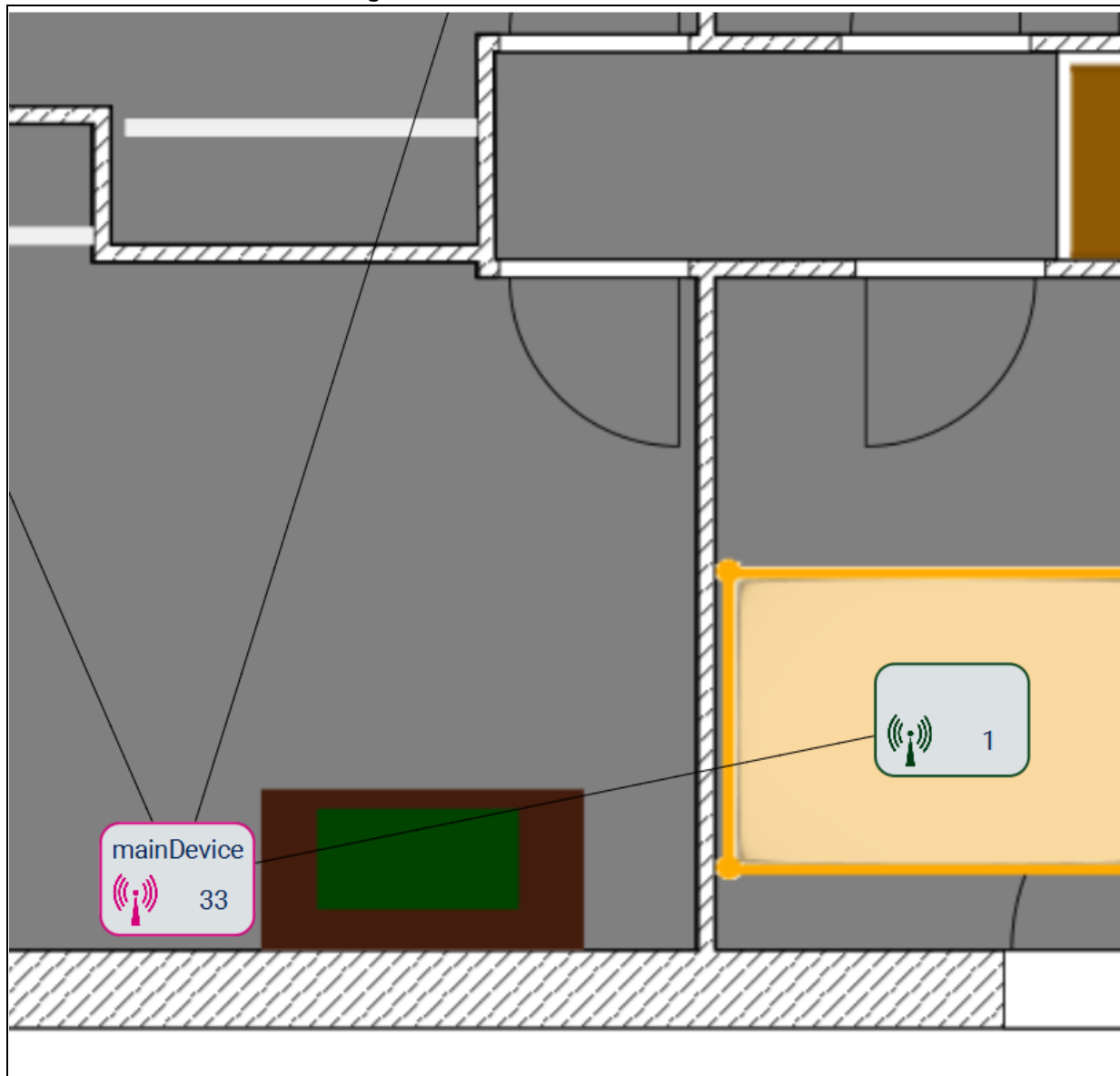
Figure 119. Display area



The result of the network exploration is displayed with icons representing the devices of the network and their links. Each icon gives three types of information:

1. The color of the borders and the logo (pink for a *Leader*, cyan for a *Router*, and green for a *Child*) gives the role.
2. The number on the right side of the logo gives the ID.
3. The eventual nickname is written above the logo.

Figure 120. Zoom and motion controls

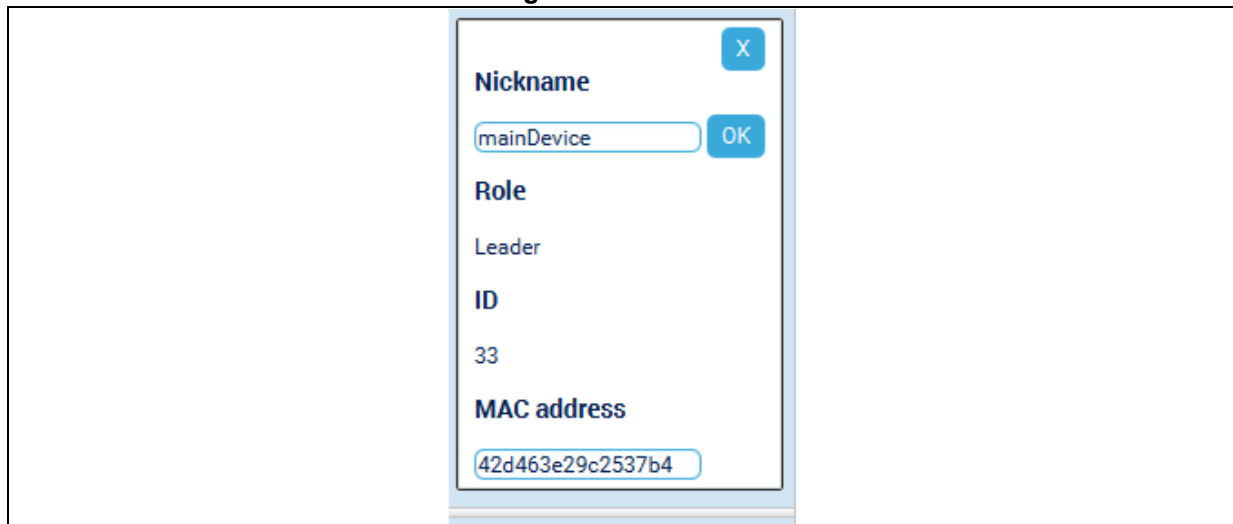


In the display area, it is possible to make several kinds of movement:

- Holding the left click of the mouse can move an icon everywhere inside the right area. If dragged on another icon, it turns gray and is automatically replaced if dropped on another icon to avoid overlays.
- Zoom in or out is done with the mouse wheel. The motion is centered on the mouse pointer.
- The whole content of the right area can be moved by holding the right click of the mouse. However, this movement is subject to constraints, as the zone background (imported image or default blank background) cannot leave the zone completely.
- A double-click (left) anywhere on the area centers the background and restores the zoom x1.

4.4.3 Infobox

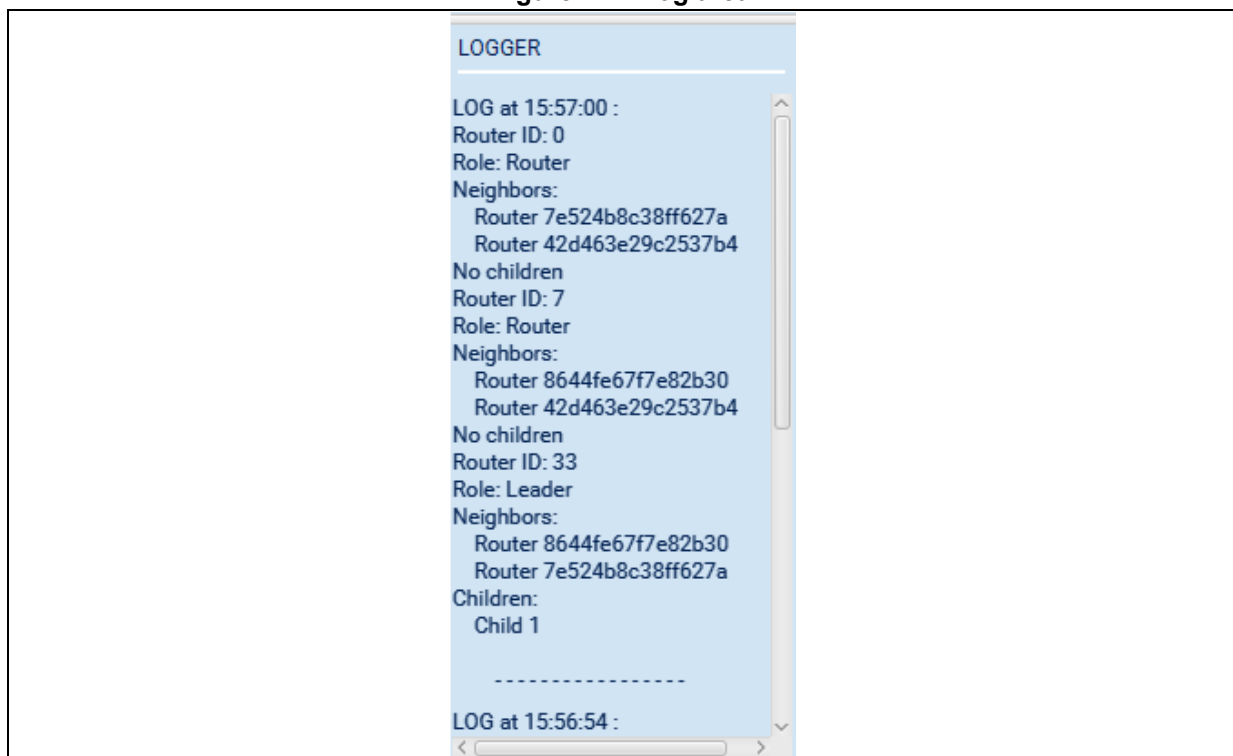
Figure 121. Infobox



An infobox can be instantiated just below the control area by clicking on the concerned icon. It allows modification of the node nickname and indicates its role, ID, and MAC address.

4.4.4 Log area

Figure 122. Log area



The log area is in the bottom-left part of the tab. It prints the last two exploration results of the network in written form. This area is updated after each new exploration.

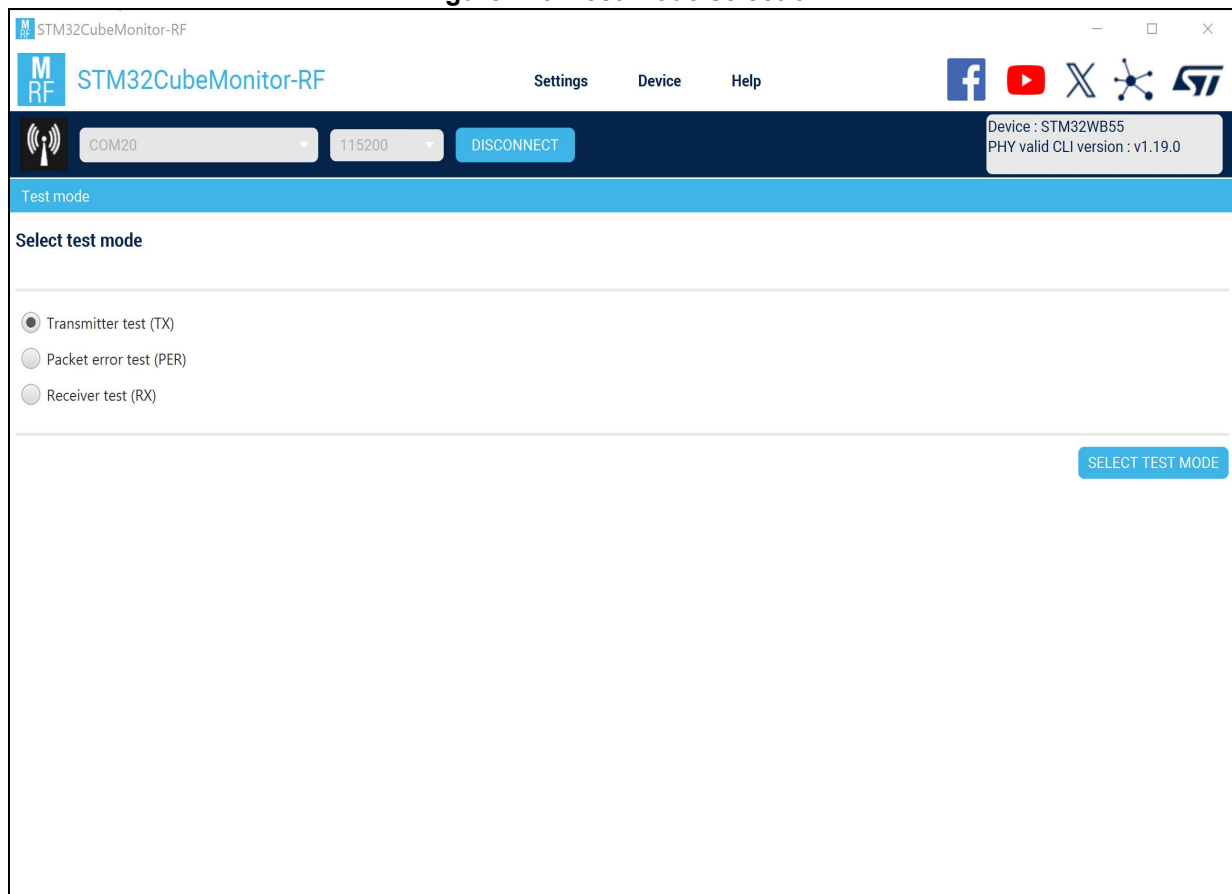
5 802.15.4 RF test mode

5.1 Presentation

The RF test panel performs the 802.15.4 radio frequency tests on the main device. Three test modes are available: Transmitter (TX), Receiver (RX), and Packet error rate (PER):

- The TX test sets the device in emission (TX continuous).
- The RX test sets the device in reception.
- The PER test sets the device in reception and one additional device is used as a packet generator.

Figure 123. Test mode selection



The user selects the mode by checking the radio button and pressing the *SELECT TEST MODE* key to switch on the new panel.

To change the mode, it is necessary to come back to this panel. There is a *Back* key and a breadcrumb link in each test panel to come back to this *Test mode selection* panel.

5.2 Transmitter test mode (TX)

This test mode configures the 802.15.4 device in emission. Two TX modes are available, Frame and Continuous modulated modes.

Figure 124. Transmitter test mode

The screenshot shows the 'Transmitter (TX)' configuration screen. The 'Power level' is set to 'High Power' (checked) with a value of '+6 dBm'. The 'TX Frequency' is set to '2405 MHz (Channel 11)'. The 'TX Mode' dropdown menu is open, showing options: 'Continuous wave' (highlighted), 'Frame', 'Continuous modulated', and 'Continuous wave'. A 'No modulated signal' label is next to the dropdown. The 'Back' button is on the left and the 'START TX' button is on the right. The 'Test measurement' section is empty.

The user must:

- For the STM32WBA devices, check or uncheck the *High Power* box according to high-power support.
- Select the power level:
 - For the STM32WB MCUs, select the range from +6 to -21 dBm.
 - For the STM32WBA MCUs, the supported range depends on high-power support:
 - Select from +3 to -20 dBm in Low-power mode
 - Select from +10 to -20 dBm in High-power mode
- Select the TX frequency (Channel 11 - 2405 MHz to Channel 26 - 2480 MHz).
- Select the TX mode, Frame, Continuous modulated, or Continuous wave.

5.2.1 Frame mode

This mode allows the user to send a MAC frame. Either the user selects one frame available in the picklist or it fills the field.

Figure 125. Field and picklist defining the frame

TX Mode: Frame

0x0C,0x01,0x00,0x01,0x11,0x12,0x13,0x14,0x15,0x16,0x17 100

0x0C,0x01,0x00,0x01,0x11,0x12,0x13,0x14,0x15,0x16,0x17 100

(Tx to Rx turnaround time) 0x0C,0x21,0x00,0xA1,0x11,0x12,0x13,0x14,0x15,0x16,0x17 100

(Ack required - a second board in RX_start is mandatory) 0x0C,0x61,0x98,0x00,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7 100 10

(Data Frame type and Ack required - a second board in RX_start is mandatory) 0x0D,0x61,0x98,0x00,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8 100 10

(MAC command Frame type and Ack required - a second board in RX_start is mandatory) 0x0E,0x63,0x98,0x00,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9 100 10

(Used for validation) 0x25,0x01,0x00,0x01,0xC0,0x73,0xBD,0x69,0x37,0x15,0x72,0x1E,0x85,0x29,0x46,0xA2,0xBF,0x27,0x91,0x8B,0x4C,0x2A,0x05,0x37,0x58,0xB2,0xFF,0x5C,0x6E,0x24,0xBE,0x3

(Big frame size) 0x3F,0x41,0x98,0x00,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B,0x0C,0x0D,0x0E,0x0F,0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1A,0x1B,0x1C,0x1D,0x1E,0x1F,0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2A,0x2B,0x2C,0x2D,0x2E,0x2F,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3A,0x3B,0x3C,0x3D,0x3E,0x3F,0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4A,0x4B,0x4C,0x4D,0x4E,0x4F,0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x5A,0x5B,0x5C,0x5D,0x5E,0x5F,0x60,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6A,0x6B,0x6C,0x6D,0x6E,0x6F,0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x7A,0x7B,0x7C,0x7D,0x7E,0x7F,0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0x9D,0x9E,0x9F,0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF,0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0xDA,0xDB,0xDC,0xDD,0xDE,0xDF,0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,0xEA,0xEB,0xEC,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0xFA,0xFB,0xFC,0xFD,0xFE,0xFF

Note: In the picklist, there is the frame required for the certification test of TX to RX turnaround time.

The help information is visible with a mouse over the question mark, on the right side of the field.

Figure 126. Help frame information

0x0C,0x01,0x00,0x01,0x11,0x12,0x13,0x14,0x15,0x16,0x17 100

Send the specified MAC frame:
 Example: 0x0B,0x41,0x98,0x00,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6 100 0 0 0x4A
 Format: Frame | Option 1 | Option 2 | Option 3 | Option 4
 Frame: 1st byte is length.
 2nd byte is MAC FCF byte 1.
 bits 0-2 : Frame type : 000 beacon.
 001 data.
 010 Acknowledgment.
 011 MAC command.
 bits 3 : Security enabled.
 bits 4 : Frame pending.
 bits 5 : Ack request.
 bits 6 : Intra PAN.
 bits 7 : Reserved.
 3rd byte is MAC FCF byte 2.
 bits 0-1 : Reserved.
 bits 2-3 : Dest addr. mode.
 bits 4-5 : Frame version.
 bits 6-7 : Source addr. mode.
 4th byte is sequence number.
 5th byte to last one minus 2 for MAC data.
 Last 2 bytes for MAC FCS (CRC) must not be entered as they will be added by HW.
 Option 1: Number of frames to be sent (default = 1).
 Option 2: Delay between 2 consecutive frames in ms (default = 0).
 Option 3: If not 0 then stop transmission when Tx result is not 0x00 (default = 0 = continue).
 Option 4: RSSI threshold in HEXA to apply during transmission (default = 0x20).

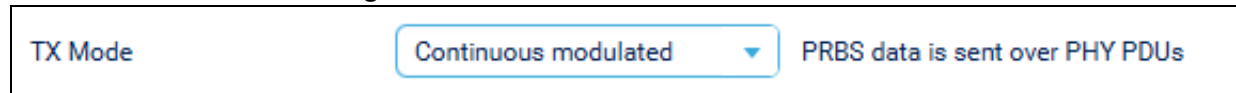
The **START TX** button is enabled when the frame in the field is valid. Press the start button to launch the transmission, the button is disabled until the frame is transmitted.

5.2.2 Continuous modulated mode

This mode transmits a continuous signal where pseudo-random binary sequence (PRBS) data is sent over PHY PDU.

Press the *START TX* button to launch the transmission. The label of the button is switched to *STOP TX* and allows the user to stop the transmission.

Figure 127. Continuous modulated test mode



5.2.3 Continuous wave mode

This mode transmits a continuous signal with no modulation.

Press the *START TX* button to launch the transmission. The label of the button is switched to *STOP TX* and allows the user to stop the transmission.

5.3 Receiver test (RX) mode

This test mode configures the device in reception and requires an external generator.
Four tests are available:

- 1. PER (packet error rate) requires an external frame generator.
- 2. LQI (link quality indicator) requires an external frame generator.
- 3. ED (energy detection) requires an external continuous wave generator.
- 4. CCA (channel clear assessment) requires an external frame generator.

Note: LQI, ED, and CCA tests are available with PHY valid CLI version v1.8.1 and upper.

Figure 128. Receiver test mode

Test mode > Receiver (RX)

Receiver

RX Frequency

2405 MHz (Channel 11)

Test mode

PER (Packet Error Rate)

PER (Packet Error Rate)

LQI (Link Quality Indicator)

ED (Energy Detection)

CCA (Channel Clear Assessment)

Back

START RX

Test measurement

NB frames received

RSSI (last received packet)

LQI (last received packet)

5.3.1 Packet error rate (PER) test

This test requests to use an external frame generator and follow the procedure below:

- Select the channel to be tested.
- Press the *START RX* button; the device enters *Receiver mode* and the button switches to *STOP RX*.
- With one external generator, send the frames to test in the frequency selected above.
- On the application side, the frames received appear in a gray part. This part is available from PHY valid CLI version v1.8.0 and upper.

Figure 129. PER frames received

Test measurement

NB frames received: ...

RSSI (last received packet): ...

LQI (last received packet): ...

STOP RX

Testing

```

Info[RX length:[0x0C], RX FCF:[0x9861], RX FCS (CRC):[0xDC8B], RSSI:[-49], LQI:[168]]
Fram[0x0C,0x61,0x98,0x00,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xDC]
Multiple buffer[Index:[0], usage:[1]]
Fram[0x0C,0x61,0x98,0x00,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xDC]
Multiple buffer[Index:[0], usage:[1]]
Fram[0x0C,0x61,0x98,0x00,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xDC]
Multiple buffer[Index:[0], usage:[1]]
Fram[0x0C,0x61,0x98,0x00,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xDC]
Info[RX length:[0x0C], RX FCF:[0x9861], RX FCS (CRC):[0xDC8B], RSSI:[-49], LQI:[168]]
Fram[0x0C,0x61,0x98,0x00,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xDC]
Multiple buffer[Index:[0], usage:[1]]
Fram[0x0C,0x61,0x98,0x00,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xDC]
Multiple buffer[Index:[0], usage:[1]]
  
```

- Once the frames are completely sent, press the *STOP RX* button. The three fields NB frames, RSSI, and LQI are filled. The button switches to *START RX*.

Figure 130. PER frame reception completed

Test measurement

NB frames received: 100

RSSI (last received packet): -49

LQI (last received packet): 168

START RX

Back

```

nb packet:[100]
nb rejected packet:[0]
nb filtered packet:[0]
nb RF lock failed:[0]

Last good Frame received:
[0x0C,0x61,0x98,0x00,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xDC]
Context after last good Frame:
RX length:[0x0C], RX FCF:[0x9861], RX FCS (CRC):[0xDC8B], RSSI:[-49], LQI:[168]]

In last good Frame received:
RSSI:[-49]
LQI:[168]
  
```

- Depending on the number of frames sent, the PER can be calculated using the value in the *NB frames received* field.

5.3.2 Link quality assessment (LQI) test

This test requests to use an external frame generator and follow the procedure below:

- Select the channel to be tested.
- Either the measurement is done in continuous (default mode) or step-by-step checking the *Single measurement* item.
- With one external generator, send the RF signal to test in the frequency selected above.
- Press the *START RX* button to launch the LQI measurement.
- The instantaneous measurement appears on the right side and is also reported in the chart.

Figure 131. LQI measurement



5.3.3 Energy detection (ED) test

This test requests to use an external frame generator and follow the procedure below:

- Select the channel to be tested.
- Either the measurement is done in continuous (default mode) or step-by-step checking the *Single measurement* item.
- With one external generator, send the RF CW signal in the frequency selected above.
- Press the *START RX* button to launch the ED measurement.

The instantaneous measurement appears on the right side and is also reported in the chart.

Figure 132. ED measurement



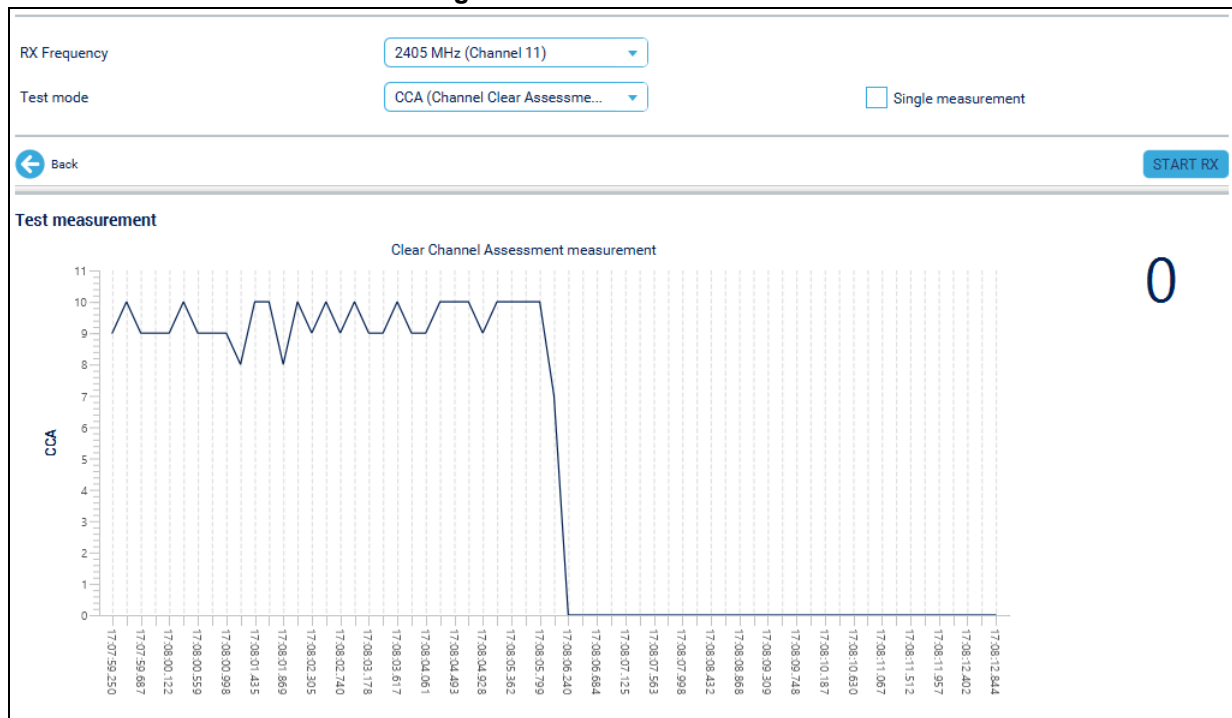
5.3.4 Channel clear assessment (CCA) test

This test requests to use an external frame generator and follow the procedure below:

- Select the channel to be tested.
- Either the measurement is done continuously (default mode) or step by step when checking the *Single measurement* item.
- With one external generator, send the RF signal to test in the frequency selected above.
- Press the *START RX* button to launch the CCA measurement.

The instantaneous measurement appears on the right side and is also reported in the chart.

Figure 133. CCA measurement



5.4 Packet error rate (PER) mode

This mode configures the device in reception and one other device to play the role of the generator.

The tool makes three measurements:

- RSSI: Received signal strength indication
- LQI: Link quality indicator
- PER: Packet error rate - computed with the number of frames received and the number of frames sent

$$100 \times (\text{Number of frames sent} - \text{number of frames received}) / \text{Number of frames sent}$$

Four steps are necessary:

- Connect the additional device to play the role of a packet generator (tester).
- Configure the parameters of the tester.
- Configure the parameters of the device under test (DUT).
- Configure the measurement.

5.4.1 Connecting the additional device to play the role of a packet generator (tester)

Figure 134. Packet tester connection

Test mode > Packet Error Rate (PER)

Connect tester

COM17 DISCONNECT PHY valid CLI version : v1.3.0

Disconnect 2nd device to change test mode CONFIGURE TESTER

Test measurement

Channel

PER (%)	<input type="text"/>
NB frames received	<input type="text"/>
RSSI (last received packet)	<input type="text"/>
LQI (last received packet)	<input type="text"/>

- Plug one additional device into the computer (the same requirements as the first device, refer to [Section 2.2](#)).
- Select the serial port to use in the picklist.
- Click on the **CONNECT** key, the device information must appear on the right side of the **connect** key.

When the second device is connected, it is not possible to change the mode. First, the user needs to disconnect the device and then press the *back* button.

Click on **CONFIGURE TESTER** to set the tester parameters.

5.4.2 Configure the parameters of the tester

Figure 135. PER tester configuration

Test mode > Packet Error Rate (PER) > Configure tester

Configure tester (COM17)

Power level ☒ Hight Power +6 dBm

TX Frequency 2405 MHz (Channel 11)

[← Back](#) [CONFIGURE DUT](#)

Test measurement

	Channel
PER (%)	<input type="text"/>
NB frames received	<input type="text"/>
RSSI (last received packet)	<input type="text"/>
LQI (last received packet)	<input type="text"/>

The user must:

- For the STM32WBA devices, check or uncheck the High Power box according to high-power support.
- Select the power level in the *Power Level* picklist.
- Select the frequency in the *TX frequency* picklist. This parameter is used only for the Single measurement mode. It is not used for Continuous or Multiple-channel modes. It is applied to the tester device.

Click on *CONFIGURE DUT* to set the Device Under Test configuration.



5.4.3 Configure the parameters of the device under test (DUT)

Figure 136. DUT configuration

Test mode > Packet Error Rate (PER) > Configure tester > Configure DUT

Configure Device Under Test (DUT) (COM20)

RX Frequency2405 MHz (Channel 11) ▾

 Back 

Test measurement

	Channel
PER (%)	<input type="text"/>
NB frames received	<input type="text"/>
RSSI (last received packet)	<input type="text"/>
LQI (last received packet)	<input type="text"/>

The user must:

- Select the frequency in the *RX frequency* picklist. It is the frequency of the DUT.

Click on *CONFIGURE PARAM* to set the test configuration:

5.4.4 Configure the measurement

Figure 137. PER test parameters

Test mode > Packet Error Rate (PER) > Configure tester > Configure DUT > Configure param

Configure Additional Settings

☒ Continuous measurement

☒ Multiple channels

☒ Save test verdict in file
 No file choosen SELECT FILE

← Back
START TEST

Test measurement

	Channel
PER (%)	<input type="text"/>
NB frames received	<input type="text"/>
RSSI (last received packet)	<input type="text"/>
LQI (last received packet)	<input type="text"/>

Three measurement modes are available:

- **Single measurement** measures once the frame number is defined. The frequency of the tester is the one defined in the *PER tester configuration* panel (TX frequency). The frequencies of DUT are as defined in the *DUT configuration* panel.
- **Continuous measurement** repeats the measurement on frame number until the user presses the *Stop Test* key. The DUT and tester frequencies are identical. They are defined once in the *DUT configuration* panel.
- **Multiple channels** measure the frequency defined in the *Fill channel list* field. The default values are 11-26, which means that all channels are in the range of 11 to 26. It is possible to use a comma to define channel by channel: 12,15,24 or mix both: 11,14-20,25,26. The user can interrupt the test with the *Stop Test* key.

The results of continuous and multiple-channel measurements can be saved in a csv file. The user must check the *Save test verdict in file* checkbox and define the file name by the *SELECT FILE* key before starting the test.

Table 3. Measurement setting

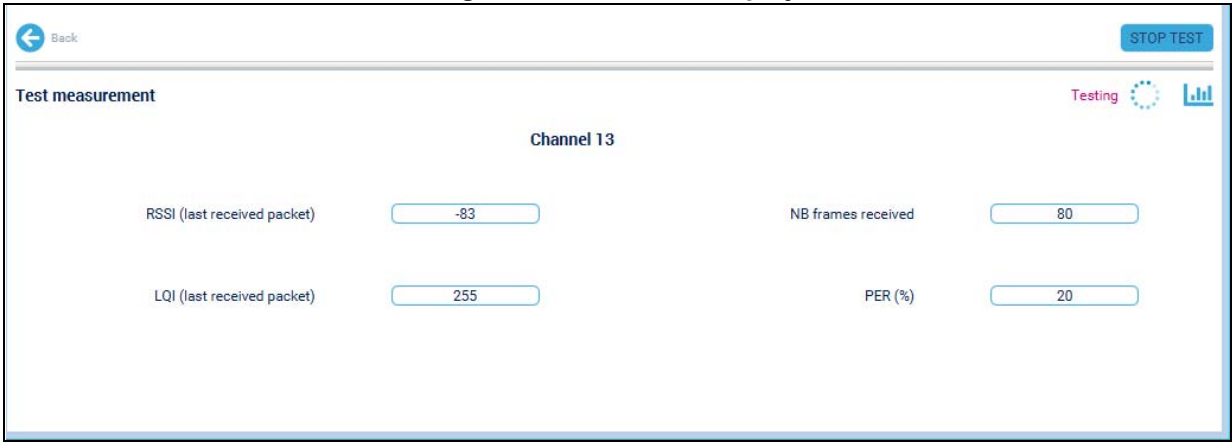
Measurements	Single	Continuous	Multiple channels
Continuous measurement checkbox	Unchecked	Checked	Checked
Multiple-channel checkbox	Unchecked	Unchecked	Checked
Save test verdict in file checkbox	Not available	Available	Available

Three display modes are available:

1. Standard display

There are the PER and RSSI values and LQI for one channel.

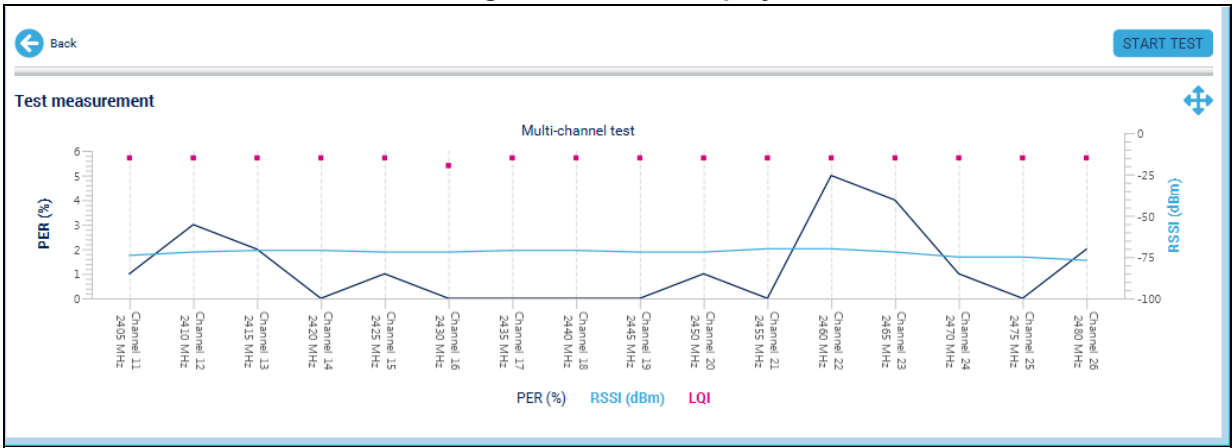
Figure 138. Standard display



2. Chart display

In the same chart, there are the PER and RSSI values and LQI for channels that the user defines.

Figure 139. Chart display



3. Large display

The user can switch between a PER display and RSSI and LQI using the arrow icons on the left and right.

Figure 140. Large PER display

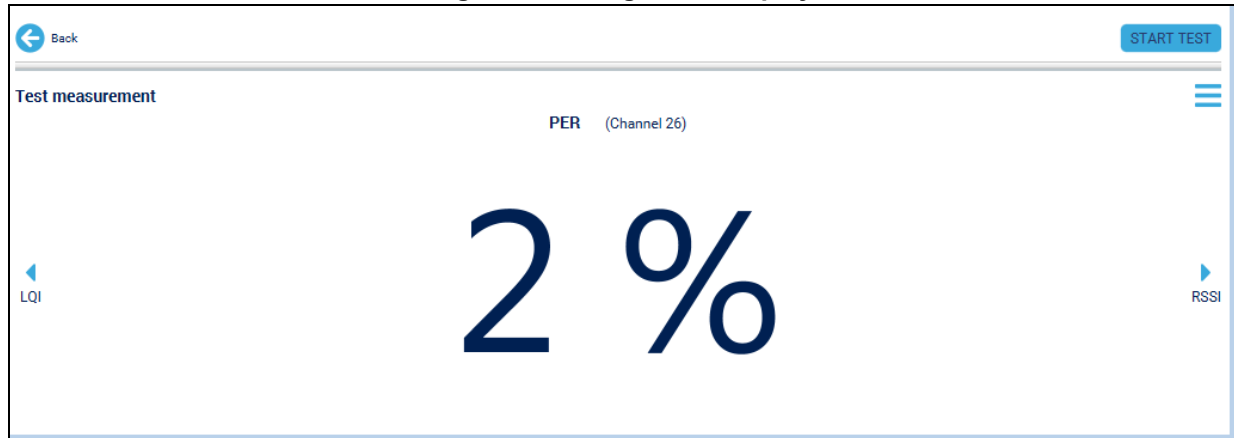


Figure 141. Large RSSI display

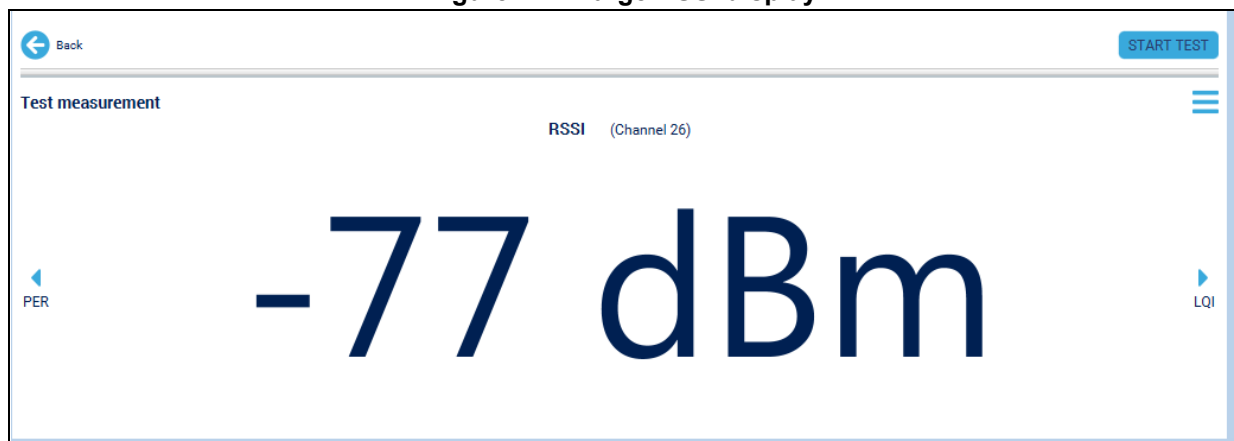


Figure 142. Large LQI display



6 802.15.4 sniffer

6.1 Presentation

The 802.15.4 sniffer allows the user to detect and log 802.15.4 packets between the devices communicating in the neighborhood of the sniffer device. Packets captured by the device are logged and formatted in a readable format, thanks to Wireshark, an external free software tool.

6.2 Prerequisite

6.2.1 Sniffer device

To configure the device as a sniffer, refer to [Section 2.2.3](#). Once done, connect the STM32WBx5 Nucleo board to the host computer using the USB_USER connector.

Make sure that the 5 V source jumper connector is plugged into the USB MCU.

6.2.2 Wireshark

Install Wireshark v2.4.6 or later, available from <http://www.wireshark.org>, and add the installation path to the path environment variable if it is not already done.

Once done, the user must copy the Python™ sniffer script `stm32cubeMonRf_sniffer.py` and the associated `stm32cubeMonRf_sniffer.bat` file in the Wireshark *extcap* directory. Files are available in the sniffer directory where the tool is installed, by default for Windows: *\Program Files (x86)\STMicroelectronics\STM32CubeMonitor-RF\sniffer*.

The Wireshark *extcap* path is available in the Help/About Wireshark menu under the *Folders* tab.

Under macOS and Linux, the *stm32cubeMonRf_sniffer.py* file must have execute permission.

6.2.3 Python™

Install Python™ v2.7.x or later available from <https://www.python.org/downloads>. Then add the installation path to the path environment variable if it is not already done.

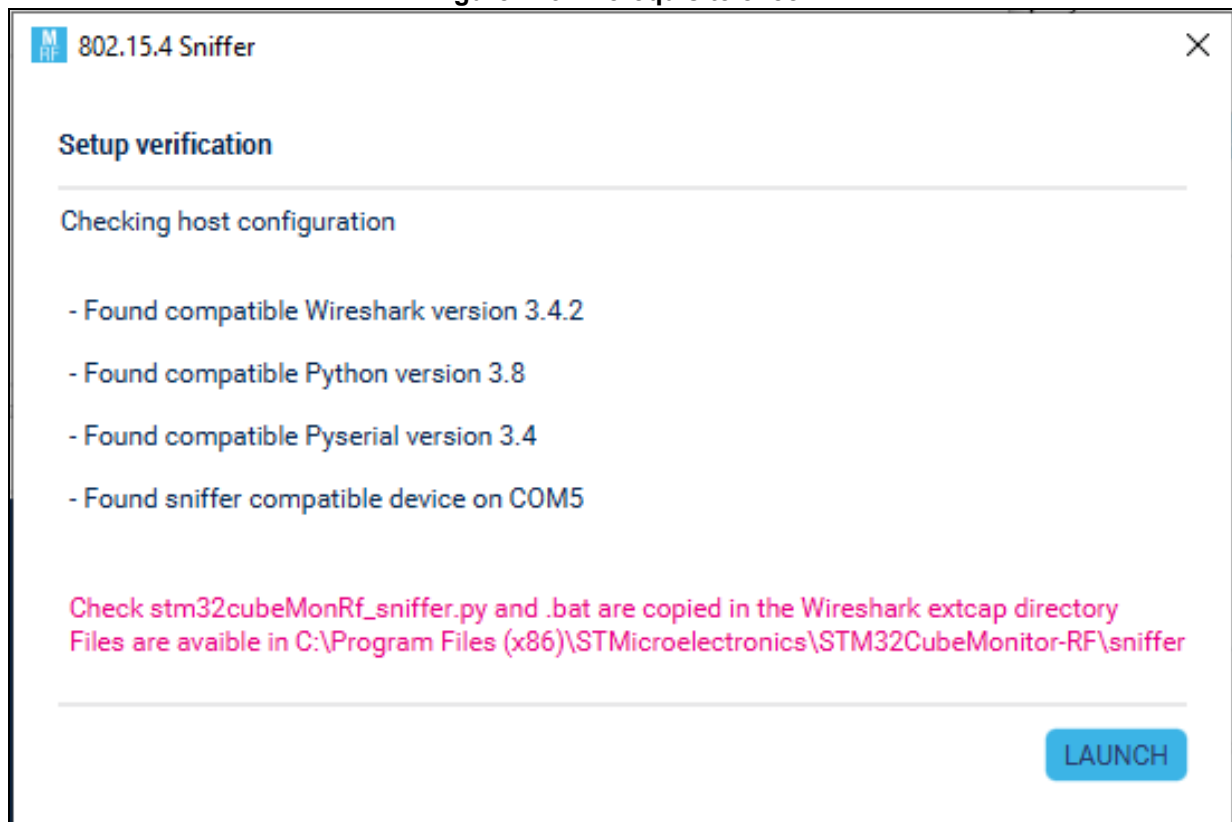
The user also needs to install the Python™ serial port extension, *pyserial*, available from <https://pypi.org/project/pyserial>.

6.3 Setup verification

The 802.15.4 sniffer, available as a button on the home screen or in the Settings/mode menu on the menu bar, can invoke the sniffer. In both cases, the tool checks that the prerequisites are fulfilled.

If this is not the case, the user is asked to correct it. Otherwise, the following pop-up window is displayed. To launch the sniffer, click on the LAUNCH button.

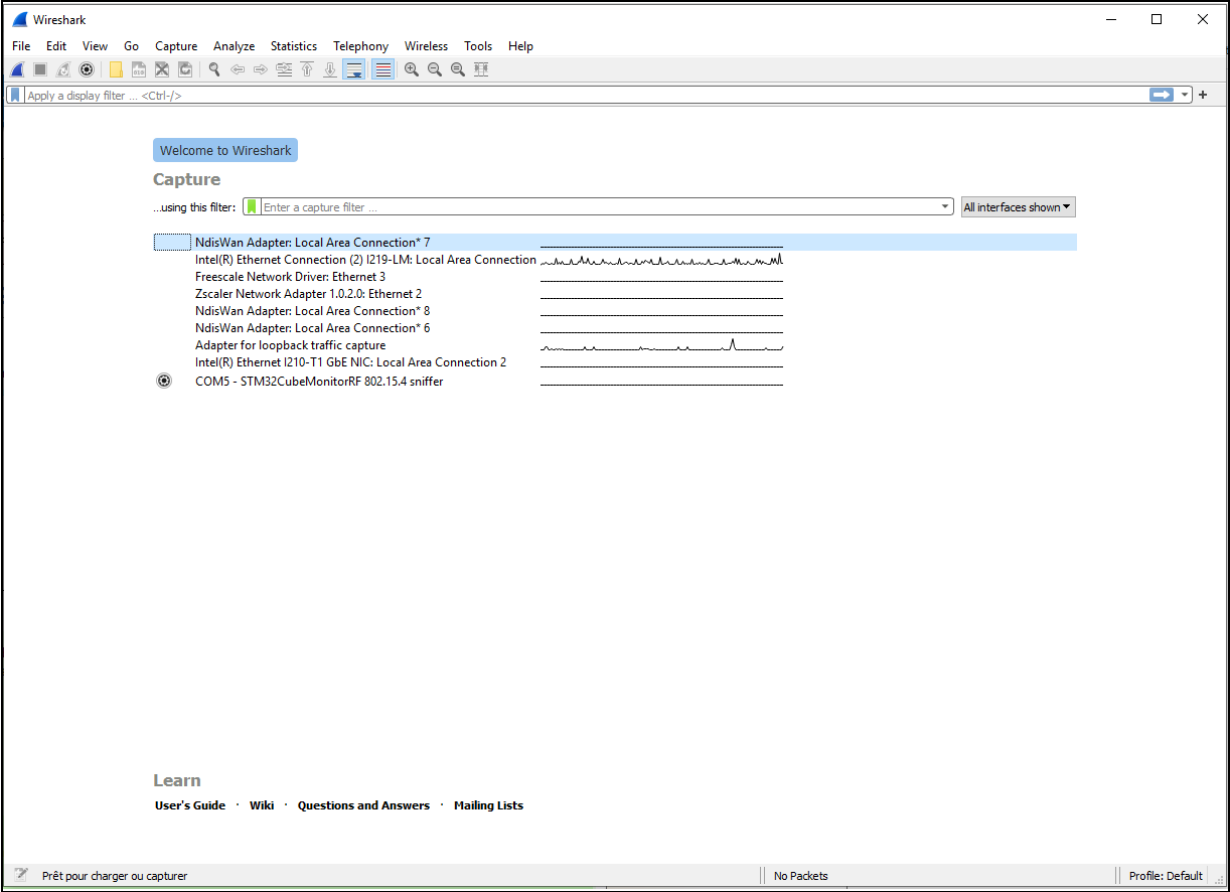
Figure 143. Prerequisite check



6.3.1 Sniffer launch

Once Wireshark is launched, the user is proposed to choose the interface to sniff.

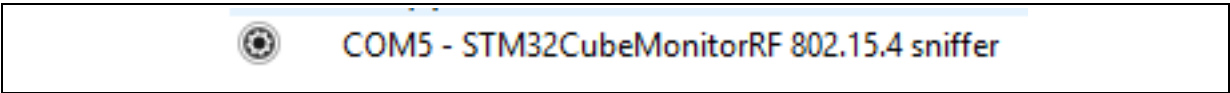
Figure 144. Wireshark interfaces



6.3.2 Select interface

Choose the interface corresponding to the device configured for sniffing by clicking on the wheel.

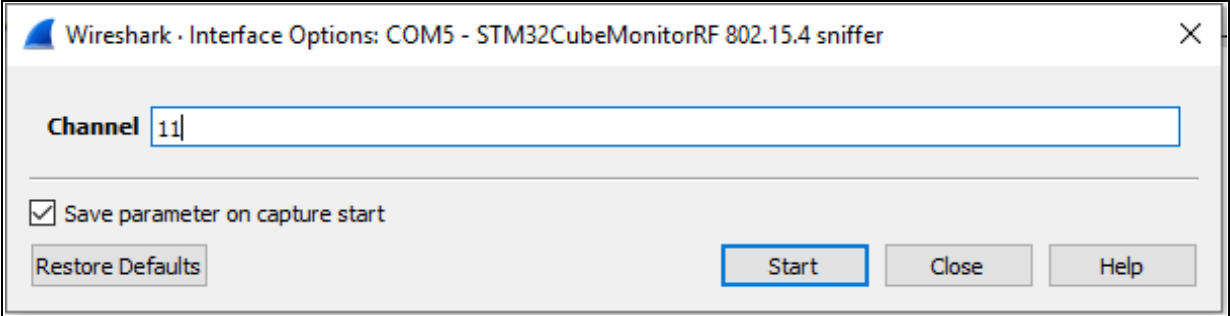
Figure 145. Wheel



6.3.3 Channel configuration

The user is asked to choose the channel to be sniffed.

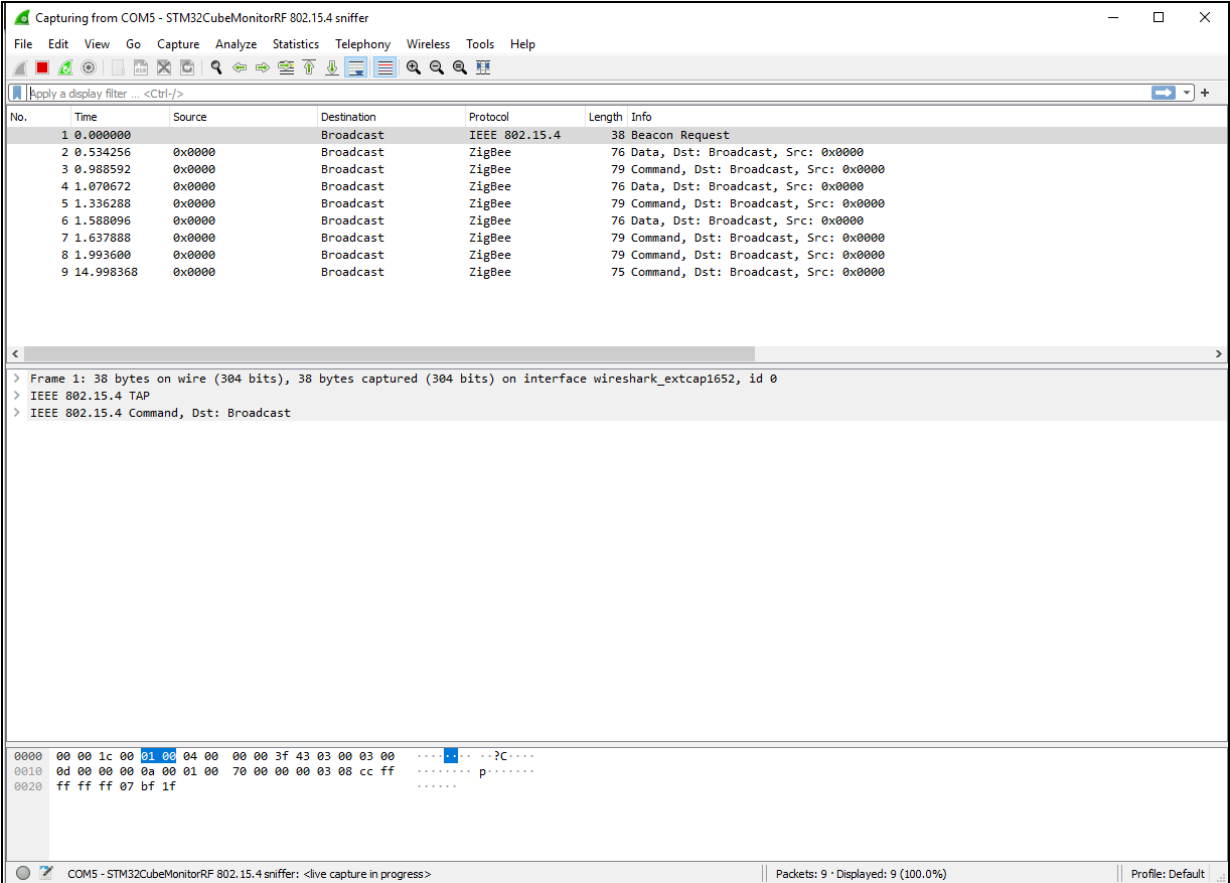
Figure 146. Channel choice



6.3.4 Sniffing start

Once the channel is selected, click on start. The list of sniffed packets appears at the top with details of the selected packet in the middle and the packet byte at the bottom.

Figure 147. Sniffing



Appendix A Beacon configuration format

The beacon configuration file is binary. Its content is explained in [Table 4](#).

Table 4. Beacon configuration format

Byte #	Name	Value	Description
0	Address type	0,1,2	Address type: 0 = board default address 1 = random address 2 = static address provided in the block
1 - 6	Address	address	Static address for the beacon. Valid only if the address type is 2.
7	Tx power	0x00-0x1F	Tx power to be used for the beacon. Value <i>PA_Level</i> of command ACI_HAL_SET_TX_POWER_LEVEL: 0 to 31
8	Beacon additional feature	0 or 1	0: No additional feature 1: TLM activated Other values reserved
9	Advertising payload length	13-32	Length of payload data
10 - 41	Advertising payload	-	Beacon advertisement payload, to be inserted in the advertisement

Revision history

Table 5. Document revision history

Date	Revision	Changes
27-Nov-2017	1	Initial version
25-Jan-2018	2	Updated: – Introduction – Section 3.3.2: Test mode receiver (RX) Added: – Two tables: Table 2: Specific AD encoding for code example and Table 3: Search filtering – Twelve new figures – Section 3.2.1: How to send an ACI command – Section 3.2.2: Search function – Section 3.4.4: Script report – Section: Pause command in the script – Section 3.5.3: Advertising change for OTA in ST example
23-Aug-2018	3	Complete content reorganized to explain tool support to the original Bluetooth® Low Energy mode in Section 3 and the new OpenThread mode in Section 4 .
13-Feb-2019	4	Updated: – Section 4: OpenThread mode and most of the figures with the new version tool Added: – Section 5: 802.15.4 RF test mode
12-Jul-2019	5	Updated: – Tool version 2.4.0 – Section 3.5: OTA transfer simplified. Details are reported in the application note.
30-Mar-2020	6	Added: – STM32WB35 support with updated paths – Section 4.4 on Thread® network exploration feature
12-Nov-2020	7	Updated: – Tool version 2.6.0 Added: Six new sections: – Section 5.2.1: Frame mode – Section 5.2.2: Continuous modulated mode – Section 5.3.1: Packet error rate (PER) test – Section 5.3.2: Link quality assessment (LQI) test – Section 5.3.3: Energy detection (ED) test – Section 5.3.4: Channel clear assessment (CCA) test dealing with all the applicable tests in the Transmitter test mode (TX) and Receiver test mode (RX)

Table 5. Document revision history (continued)

Date	Revision	Changes
08-Feb-2021	8	<p>All modifications linked to the new feature 802.15.4 sniffer</p> <p>Updated:</p> <ul style="list-style-type: none"> – Introduction – Section 1.2: Welcome screen with Figure 1 – Section 2.2.3: VCP device – Figure 61, Figure 62, Figure 65, Figure 105, and Figure 114 <p>Added:</p> <ul style="list-style-type: none"> – Section 6: 802.15.4 sniffer with Figure 141 to Figure 145
22-Jul-2021	9	<p>All modifications linked to the new tool version 2.8.0</p> <p>Updated:</p> <ul style="list-style-type: none"> – Section 2.2: VCOM/UART connection – Section 3.3.3: PER – Section 3.4.3: Scripts modification – Section 3.5.3: Use the tool to perform an OTA update – Section 3.6.3: Configuration of the beacon with STM32CubeMonitor-RF – Figure 61, Figure 62, Figure 65, and Figure 80 to Figure 82
07-Jul-2022	10	<p>Updated:</p> <ul style="list-style-type: none"> – Section 4.4: Network Explorer tab with the default master key of the former firmware – Section 2.2.1: VCOM connection to Section 2.2.3: VCP device linked to the added section below <p>Added:</p> <ul style="list-style-type: none"> – Section 3.2: Bluetooth® Low Energy stack
29-Nov-2022	11	<p>All modifications linked to the new tool version 2.10.0</p> <p>Added:</p> <ul style="list-style-type: none"> – Section 2.3: Opening COM regarding connection error in the wrong mode <p>Updated:</p> <ul style="list-style-type: none"> – Section 3.2: Bluetooth® Low Energy stack with Table 1 – Section 3.5.1: Launching scripts path – Figure 72: Selecting the Beacon mode <p>Removed:</p> <ul style="list-style-type: none"> – Beacon OTA configuration including former Figure 72 – Configuration transfer OTA mode scan transfer including former Figure 80 to Figure 82
06-Mar-2023	12	<p>All modifications linked to the new tool version 2.11.0</p> <p>Added:</p> <ul style="list-style-type: none"> – Support to STM32XWBxx microcontrollers – Save log in Section 3.3.5: Log functionalities with new Figure 28 and Figure 29 – Section 5.2.3: Continuous wave mode <p>Updated:</p> <ul style="list-style-type: none"> – Figure 2, Figure 6, Figure 23, Figure 97, and Figure 120

Table 5. Document revision history (continued)

Date	Revision	Changes
04-Jul-2023	13	All modifications linked to the new tool version 2.12.0 Removed <i>Table 1 Bluetooth® Low Energy stack and STM32CubeMonitor-RF feature compatibility</i>
23-Nov-2023	14	All modifications linked to the new tool version 2.13.0 Updated: – Applicable tool version quoted in Introduction – Figure 2 , Figure 3 , Figure 63 , Figure 64 , and Figure 67 – Source code paths in Section 2.2.1: VCOM connection
14-Mar-2024	15	All modifications linked to the new tool version 2.14.0 Updated: – Applicable tool version quoted in Introduction – Figure 119 , Figure 120 , Figure 124 , and Figure 130 to Figure 133
02-Jul-2024	16	All modifications linked to the new tool version 2.15.0 Updated: – Applicable tool version quoted in Introduction – Support of STM32WB0xx microcontrollers as mentioned in Getting started
22-Nov-2024	17	Modifications linked to the new tool version 2.16.0 Updated: – Applicable tool version quoted in Introduction – Support of STM32WB06x microcontrollers as mentioned in Getting started – Figure 63 , Figure 64 , and Figure 67
18-Feb-2025	18	Modifications linked to the new tool version 2.17.0. Updated Getting started to add support of STM32WBA6x devices. Updated Section 2.2.1: VCOM connection . Minor text edits across the whole document.
30-Jun-2025	19	Modifications linked to the new tool version 2.18.0: – Updated Section 2.2.1: VCOM connection with Transparent Mode firmware loading for the STM32WB05xN network coprocessor – Updated the connection issue to Linux® in Section 2.3: Opening COM – Updated Figure 84: Offline mode transfer configuration in Section 3.8.3: Configuration of the beacon with STM32CubeMonitor-RF – Added Section 3.6: Command-line interface (CLI) Changed Bluetooth® Low Energy to Bluetooth® LE and applied minor text edits throughout the document.

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2025 STMicroelectronics – All rights reserved