UM3534

User manual

Getting started with STM32CubeWiSE – Bluetooth® LE Explorer

Introduction

This document describes the STM32CubeWiSE - Bluetooth® LE Explorer software package (STM32CubeWiSEbe), which consists of the following components:

- STM32CubeWiSE - Bluetooth® LE Explorer PC GUI application
- STM32CubeWiSE - Bluetooth® LE Explorer script launcher standalone utility

The STM32CubeWiSE - Bluetooth® LE Explorer PC GUI is a graphical user application interface, which allows standard and vendor-specific Bluetooth® LE stack commands to be sent to the device controller and events to be received from it. It also provides a script engine, which can be used to load and run user scripts based on Bluetooth® LE stack commands and related events.

These scripts can also be run via a standalone script launcher utility through a PC command prompt window, outside the PC GUI application context. A set of sample scripts is provided in the software package.

1 General information

STM32CubeWiSEbe devices embed the Arm® Cortex®-M7.

For information on the Arm® Cortex®-M0+, refer to the technical reference manuals, available from the www.arm.com website.

1.1 Acronyms

Acronyms

Table 1. **List of acronyms**

Acronym	Definition
ACI	Application command interface
GUI	Graphical user interface
HCI	Host controller interface
RF	Radio frequency
USB	Universal serial port

2 Getting started

This section describes all system requirements to run the STM32CubeWiSE - Bluetooth® LE Explorer PC GUI application, as well as the relative SW package installation procedure.

2.1 System requirements

The STM32CubeWiSE - Bluetooth® LE Explorer PC GUI application has the following minimum requirements:

- PC with Intel® processor running one of the following Microsoft® operating systems: Windows® 11
- At least 2 Gbytes of RAM
- USB ports

2.2 STM32CubeWiSE - Bluetooth® LE Explorer SW package setup

1. Extract the content of the `stm32wise-bewin-vx-y-z.zip` file into a temporary directory.
2. Launch the `STM32CubeWiSE-Bluetooth LE Explorer_Vx.y.z.exe` file and follow the on-screen instructions.

2.3 STM32CubeWiSE - Bluetooth® LE Explorer SW package folders

The STM32CubeWiSE - Bluetooth® LE Explorer SW package files are organized into the following folders:

- App: contains `STM32CubeWiSE-Bluetooth_LE_Explorer.exe` and `STM32CubeWiSE-Bluetooth_LE_Explorer_Script_Launcher.exe`
- Docs: contains the supported Bluetooth ®LE stack commands and events descriptions, scripts descriptions and script launcher utility description.
- Firmware: contains the BLE_TransparentMode applications for the supported STM32 Bluetooth® LE microcontrollers to be loaded as prerequisites for running the PC GUI application and script launcher utility. User is requested to open the STM32CubeProgrammer tool and follow these steps:
  1. Plug the selected STM32WBA, STM32WB0 NUCLEO kit on PC USB port;
  2. On STM32CubeProgrammer tool, select and connect to the plugged NUCLEO kit;
  3. Select the STM32WBA, STM32WB0 BLE_TransparentMode application related to the specific device variant available on Firmware folder;
  4. Download the selected binary images;
  5. Disconnect the NUCLEO kit from STM32CubeProgrammer and unplug and plug again the NUCLEO kit to PC USB port.

Note: For STM32WB0 devices, the BLE_TransparentMode source is available in the STM32CubeWB0 firmware package (refer to \Projects\NUCLEO-WB0xxxx\Applications\BLE\BLE_TransparentMode folder).

Note: For STM32WBA devices, the BLE_TransparentMode source is available in the STM32CubeWBA firmware package (refer to \Projects\NUCLEO-WBAxxxx\Applications\BLE\BLE_TransparentMode folder).

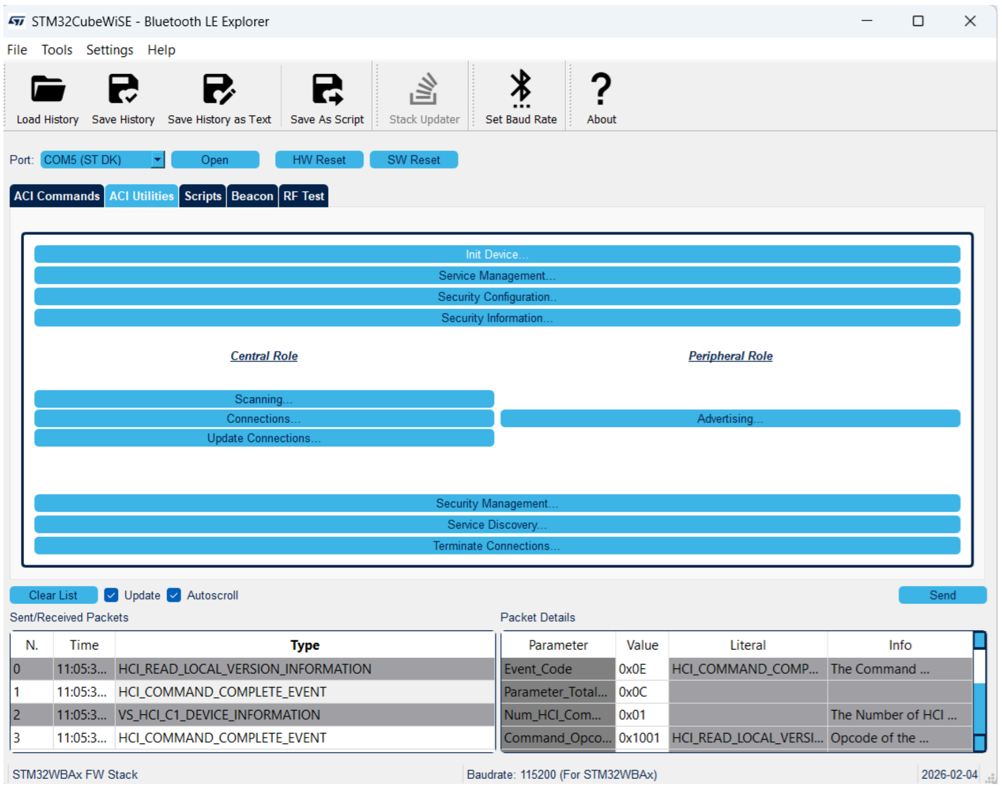
Release notes and license files are located in the root folder.

3 STM32CubeWiSE - Bluetooth® LE Explorer PC GUI application description

This section describes the main functions of the PC GUI application. You can run this utility by clicking on the STM32CubeWiSE - Bluetooth® LE Explorer GUI icon.

3.1 Main window

Figure 1. Main window of STM32CubeWiSE - Bluetooth® LE Explorer PC GUI



N.	Time	Type
0	11:05:3...	HCI_READ_LOCAL_VERSION_INFORMATION
1	11:05:3...	HCI_COMMAND_COMPLETE_EVENT
2	11:05:3...	VS_HCI_C1_DEVICE_INFORMATION
3	11:05:3...	HCI_COMMAND_COMPLETE_EVENT

Parameter	Value	Literal	Info
Event_Code	0x0E	HCI_COMMAND_COMP...	The Command ...
Parameter_Total...	0x0C		
Num_HCI_Com...	0x01		The Number of HCI ...
Command_Opco...	0x1001	HCI_READ_LOCAL_VERSI...	Opcode of the ...

STM32WBAx FW Stack   Baudrate: 115200 (For STM32WBAx)   2026-02-04

DT79081v1

The STM32CubeWiSE - Bluetooth® LE Explorer PC GUI application main window has different zones, some of which can be resized.

Port and interface selection

The uppermost zone allows the user to open the COM port associated with the STM32 Bluetooth® LE device.

When a COM port is opened, the following information is displayed:

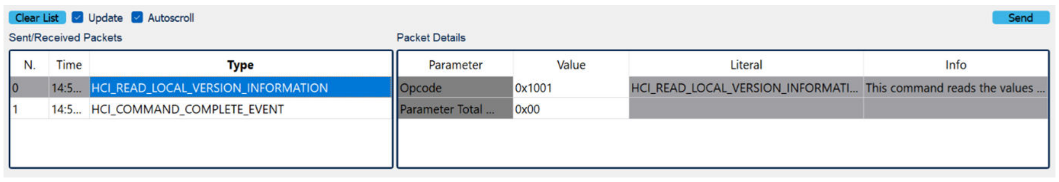
- Device HW version
- Bluetooth LE FW stack version

Packet history and details

Two frames at the bottom of the main window show packets sent to and received from the Bluetooth® LE controller, as well as other events:

- The frame on the left (sent/received packets) holds a history of all packets.
- The frame on the right (packet details) shows all the details of the selected packet as per the command packet table.

Figure 2. Packet history and details



Sent/Received Packets

N.	Time	Type
0	14:5...	HCI_READ_LOCAL_VERSION_INFORMATION
1	14:5...	HCI_COMMAND_COMPLETE_EVENT

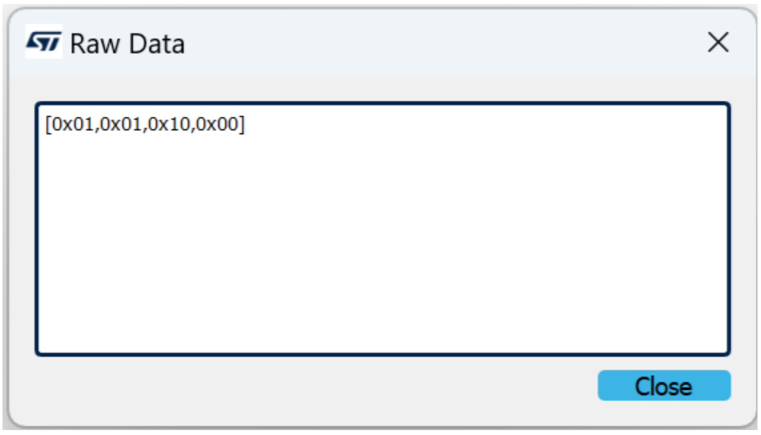
Packet Details

Parameter	Value	Literal	Info
Opcode	0x1001	HCI_READ_LOCAL_VERSION_INFORMATI...	This command reads the values ...
Parameter Total ...	0x00		

DT79083V1

Double-clicking on a row of the sent/received packet table shows the raw packet.

Figure 3. Raw packet dump



```
[0x01,0x01,0x10,0x00]
```

DT79084V1

Some events (displayed in yellow cells) can provide other information. HCI packets sent towards the Bluetooth® LE device are displayed in gray cells while received packets are shown inside white cells.

The sent/received packets table can be cleared by clicking on clear list button. Update and auto-scrolling checkboxes enable or disable updating and auto-scrolling of the sent/received packets table while new packets are sent or received (however, information is still printed).

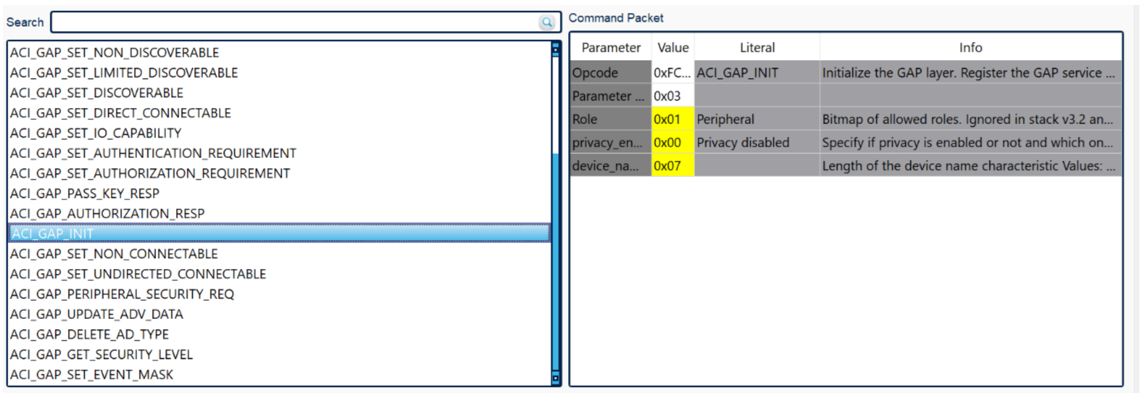
The sent/received packets can be stored and later reloaded on the PC GUI, by using the utilities provided on file menu:

- **Save history and save history as text**: saves the current list of sent commands and received events to a CSV file or to a simple text file.
- **Load history**: loads a list of sent commands and received events from a CSV file.
- **Save as Python script**: stores the current list of sent commands and received events as a script file (python format), which can be (by adding specific code for handling events, parameters) and then used in the PC GUI script window as part of a user application scenario (refer to Section 3.4: Scripts window).

3.2 ACI commands window

The "ACI Commands" window tab contains a list of all the available commands. Commands can be filtered by checking/ unchecking boxes under the filter section. After clicking on one of the commands, all packet fields are displayed on the command packet table in the upper-right section of the tab.

Figure 4. Command packet table



ACI_GAP_SET_NON_DISCOVERABLE
ACI_GAP_SET_LIMITED_DISCOVERABLE
ACI_GAP_SET_DISCOVERABLE
ACI_GAP_SET_DIRECT_CONNECTABLE
ACI_GAP_SET_IO_CAPABILITY
ACI_GAP_SET_AUTHENTICATION_REQUIREMENT
ACI_GAP_SET_AUTHORIZATION_REQUIREMENT
ACI_GAP_PASS_KEY_RESP
ACI_GAP_AUTHORIZATION_RESP
ACI_GAP_INIT
ACI_GAP_SET_NON_CONNECTABLE
ACI_GAP_SET_UNDIRECTED_CONNECTABLE
ACI_GAP_PERIPHERAL_SECURITY_REQ
ACI_GAP_UPDATE_ADV_DATA
ACI_GAP_DELETE_AD_TYPE
ACI_GAP_GET_SECURITY_LEVEL
ACI_GAP_SET_EVENT_MASK

Parameter	Value	Literal	Info
Opcode	0xFC...	ACI_GAP_INIT	Initialize the GAP layer. Register the GAP service ...
Parameter ...	0x03		
Role	0x01	Peripheral	Bitmap of allowed roles. Ignored in stack v3.2 an...
privacy_en...	0x00	Privacy disabled	Specify if privacy is enabled or not and which on...
device_na...	0x07		Length of the device name characteristic Values: ...

The command packet table contains four columns:

- **Parameter**: name of the packet field as per volume 2, part E of Bluetooth specification
- **Value**: field value represented in hexadecimal format (right-click on a cell to change the format)
- **Literal**: meaning of the current field value
- **Info**: description of the corresponding field

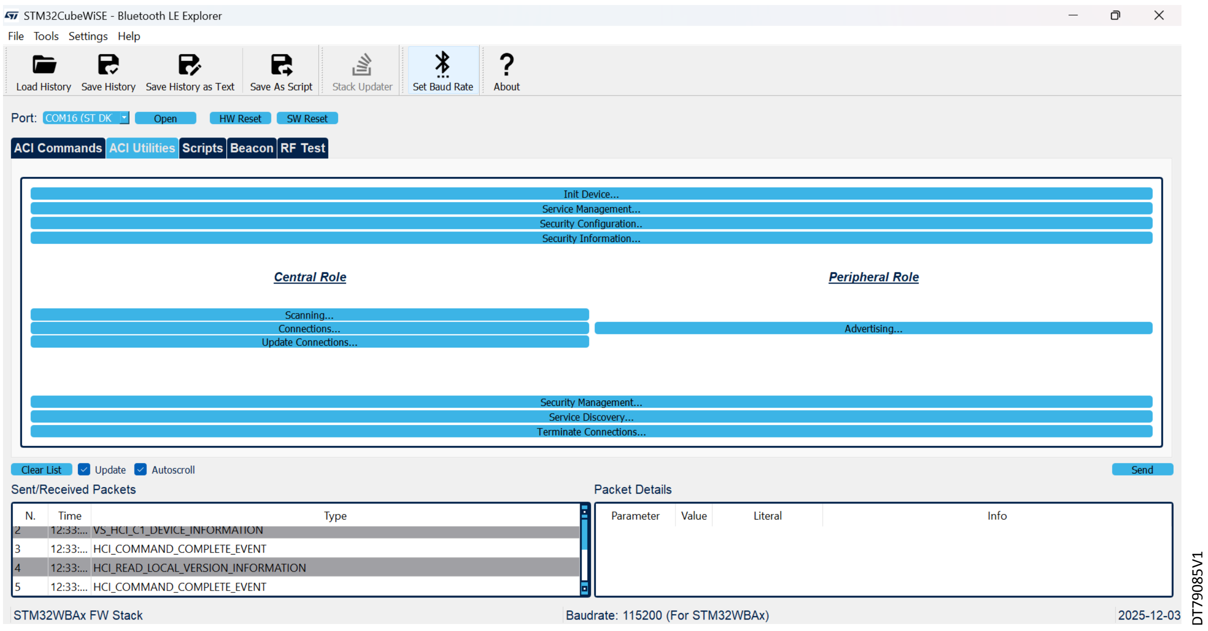
Only the yellow cells of this table can be modified by the user. Parameter total length is fixed or automatically calculated after modifying cell content.

After fields have been modified, the command can be sent using the send button.

3.3 ACI Utilities window

The "ACI Utilities" window provides several tabs for testing some application scenarios.

Figure 5. ACI Utilities window



N.	Time	Type
2	12:33:...	VS_HCI_C1_DEVICE_INFORMATION
3	12:33:...	HCI_COMMAND_COMPLETE_EVENT
4	12:33:...	HCI_READ_LOCAL_VERSION_INFORMATION
5	12:33:...	HCI_COMMAND_COMPLETE_EVENT

Central and peripheral roles are supported with the Bluetooth$^{®}$ LE operations described in the following tables.

Table 2. ACI utilities: available general operations

Operation	Associated actions	Notes
Init Device...	Allows a device to be initialized by selecting: • Role • Address type (public, random) and value • Tx power level • Power mode • Device name	-
Security Configuration...	Allows the device security database to be erased. Allows security settings to be configured: • IO capabilities • Authentication requirements	-
Service Management...	Allows adding a service by selecting: • UUID type (16 or 128 bits) • Service type (primary or secondary) • Set max. number of records For each service, it allows a characteristic and related descriptors to be added by selecting: • UUID type (16 or 128 bits) • Properties • Security permissions • Variable length or not • Length	After a service, characteristic or descriptor are defined, the user can edit its parameters and/or delete it. Once a service and its characteristics, descriptors have been defined, click OK to add them to the GATT database. The defined GATT database is shown on a specific view. The "Save View" button allows the defined services and associated characteristics to be stored in order to be reloaded later with the "Load View" button.

Operation	Associated actions	Notes
	• GATT event mask • Encryption key size	
Security Management...	Allows security operations (send security request, send pairing request,...) to be performed.	-
Security Information...	Allows the list of the bonded devices to be got.	-
Service Discovery...	Allows discovery of all services and related characteristics and descriptors of available connections.	Service start handle, end handle and UUID are showed. For each selected service the related characteristics information are showed (attribute handle, property, value handle and UUID). For the available characteristic with notify or indication property, it is possible to enable the notification/indication. It is also possible to store as view the discovered services and associated characteristics for a later usage through the "Load View" button on "Service Management …" window. This allows user to build his own "Services/ Characteristics View" database from discovered peer device services and characteristics, and store them for possible future application scenarios.
Terminate Connection...	Allows terminating the available connections	-

The "Service Management..." window provides the load and save view buttons. A set of predefined views ,related to standard services and profiles are available on the STM32CubeWiSE - Bluetooth® LE Explorer SW package SW package, app/Services_View folder ("alert notification server, blood pressure sensor, glucose sensor, heart rate sensor, health thermometer sensor, find me target, phone alert server, HID, time, …")

User can load them through the "Load View" button and add to his application database.

Table 3. **ACI utilities: available central operations**

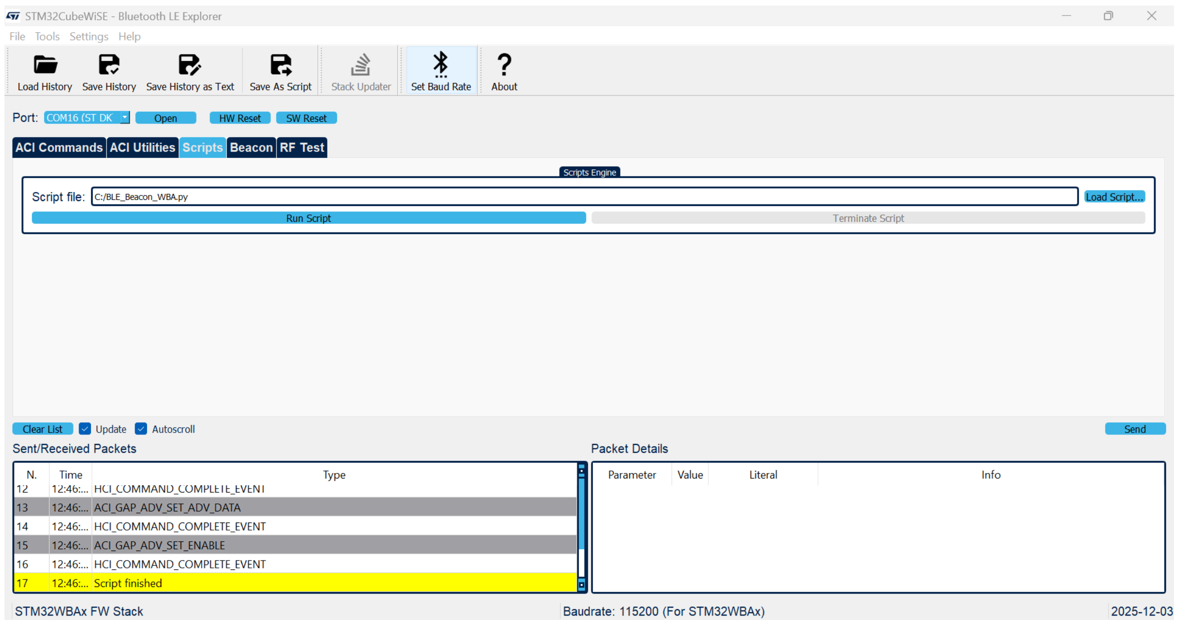
Operation	Associated actions	Notes
Scanning...	Allows placing the device in scanning mode by selecting: • GAP procedure • Enable or disable filters • Set scanning PHY • Set passive or active scan • Set scanning interval and window	-
Connections...	Allows connecting to a peer device by: • Searching for the devices in advertising • Selecting the device to which to connect • Selecting the connection parameters: • Peer address and type • Scan interval and window • Connection interval • Latency • Supervision timeout • Connection event length	The addresses of the detected advertising devices are displayed
Update Connections...	Allows updating the connection parameters of available connections by: • Selecting the specific connection to be updated	-

Operation	Associated actions	Notes
	• Setting the new connection parameters: • Connection interval • Latency • Supervision timeout • Connection event length	
Advertising...	Allows placing a peripheral device in advertising mode by setting: • Advertising parameters • Advertising data • Scan response data	-

3.4 Scripts window

The "Scripts" window allows the user to load and run a python script built using the available set of ACI commands and related events. For a list of supported HCI and ACI script commands and parameters, refer to those commands available in the "ACI Commands" window.

Figure 6. Scripts window section



Port: COM16 (ST DK

Script file: C:/BLE_Beacon_WBA.py

Update ✓ Autoscroll ✓

Sent/Received Packets

N.	Time	Type
12	12:46:...	HCI_COMMAND_COMPLETE_EVENT
13	12:46:...	ACI_GAP_ADV_SET_ADV_DATA
14	12:46:...	HCI_COMMAND_COMPLETE_EVENT
15	12:46:...	ACI_GAP_ADV_SET_ENABLE
16	12:46:...	HCI_COMMAND_COMPLETE_EVENT
17	12:46:...	Script finished

Packet Details

Parameter	Value	Literal	Info

STM32WBAx FW Stack   Baudrate: 115200 (For STM32WBAx)   2025-12-03

DT79086V1

The script engine supports other utility commands listed in the following table.

Table 4. Script window: utility commands

Command name	Parameters	Description
ERROR	User message	Raises an exception with a user-defined debug message
CLEAR_QUEUE	None	Removes all the pending events collected within the internal event queue
GET_CHAR	None	Allows a specific char to be entered as input (as the C get_char() API)
GET_ALL_COM_PORT	None	Returns the list of all COM ports (ST DK kits and not)
GET_ALL_ST_DK_COM_PORT	None	Returns the list of all ST DK kits COM ports sorted by COM port number
GET_FILE	None	Allows a specific file to be selected as input
GET_NAME	None	Returns the device name within an advertising packet
GET_VALUE	Array of bytes	Converts the array of bytes to an integer value; for example: X= [0x33,0x22] GET_VALUE(X) = 0x2233
GET_LIST	Integer, number of bytes	Converts the integer value to an array of number of bytes; for example: X= 0x2233 GET_LIST(X, 2)= [0x33,0x22]
GET_STACK_VERSION	None	Returns the device information (HW version and FW version) as (hw, fw)
GET_RAND_KEY	None	Returns a random number between 0 and 999999
HW_RESET	None	HW reset
INFO	String to be displayed	Opens a message window and show the input parameter. Script is blocked until user presses OK button

Command name	Parameters	Description
INSERT_FLOAT_NUMBER	None	Allows a float value
INSERT_INT_NUMBER	None	Allows an integer value to be entered
INSERT_PASS_KEY	None	Allows entering a pass key value for the security pass key method
IS_STM32WB0X	None	Returns TRUE if the device is a STM32WB0 series device, FALSE otherwise
IS_STM32WBAX	None	Returns TRUE if the device is a STM32WBA series device, FALSE otherwise
PRINT	String	Print utility: displays information on GUI sent/received packets (only integer decimal format is supported)
RESET	None	SW reset
SLEEP	time	Sleeps for the period 'time' in seconds
SET_PUBLIC_ADDRESS	Public address	Set public address (optional)
SENSORDEMO_GET_TEMPERATURE	None	Allows retrieval of the temperature value from the read response event (only for the SensorDemo_Central script)
SENSORDEMO_GET_ACCELERATION	None	Allows retrieval of the acceleration values (x,y,z) from the notification event (only for the SensorDemo_Central script)
TIME	None	Returns the time as a floating point number expressed in second since the epoch, in UTC

The following pseudo code describes how to initialize an STM32WBA device as a peripheral using a simple python™ script:

```
#Reset device
HCI_RESET()
#Init GATT
ACI_GATT_INIT()
#Init GAP as peripheral device
ACI_GAP_INIT(Role=PERIPHERAL)
```

The following pseudo code describes how to initialize a STM32WB0 device as a peripheral using a simple python™ script:

```
#Reset device
HCI_RESET()
status, _ = ACI_GATT_SRV_PROFILE_INIT()
if status !=0x00:
    ERROR('ACI_GATT_SRV_PROFILE_INIT CALL FAILED')
status = ACI_GAP_INIT()
if status !=0x00:
    ERROR('ACI_GAP_INIT CALL FAILED')
status, _, _, _ = ACI_GAP_PROFILE_INIT(Role=PERIPHERAL)
if status !=0x00:
    ERROR('ACI_GAP_PROFILE_INIT CALL FAILED')
```

When a script calls a command that generates specific events, the script can detect them by using the: `WAIT_EVENT (event_code=None, timeout=None, continueOnEvtMiss=False, **param_checks)` command.

Table 5. **WAIT_EVENT macro-command**

Command name	Description	Parameters	Return
WAIT_EVENT	It waits for an event with 'Event Code' parameter equal to event_code. If no event_code is indicated, the macro-command waits for any event.	• event_code = none (default) • timeout = none (default) • continueOnEvtMiss = false (default)	• An event with its parameters None, if a timeout occurs and the input parameter "continueOnEvtMiss" is set to true. • An event with its parameters

Command name	Description	Parameters	Return
	Optional filtering parameters allow additional filters to be defined on event fields.	• param_checks = optional filtering parameters	• None, if a timeout occurs and the input parameter "continueOnEvtMiss" is set to true

The WAIT_EVENT macro-command waits for an event with 'Event Code' parameter equal to event_code. If no event_code is indicated, the macro-command waits for any event.

The timeout parameter allows the event timeout to be set. If no timeout is set, the macro-command awaits until an event occurs. If a timeout (greater than zero) is set and continueOnEvtMiss is false and no event occurs before the timeout, an HCITimeoutError error happens. Otherwise, if the input parameter continueOnEvtMiss is true and a timeout (greater than zero) is set, the macro-command returns the value none even when no event occurs before the timeout.

If one or more optional filtering parameters are specified, the macro-command performs a check on them and it returns only the first detected event that satisfies these parameters. The events received before the one returned are discarded.

The WAIT_EVENT() command return value can be:

- An event
- None, if a timeout occurs and the input parameter "continueOnEvtMiss" is set to true

An HCI timeout error exception is raised when a timeout occurs. Each coming event is stored in an internal queue. User is requested to call a specific WAIT_EVENT utility command in order to handle each received event and remove it from the internal queue. The CLEAR_QUEUE utility command allows the internal queue to be cleared in order to remove all the collected events not handled through the WAIT_EVENT utility command. The event_code parameter can be one of the following values:

Table 6. **Event codes with relative event parameter types**

event_code	Event parameter type	Event parameter type value
HCI_LE_META_EVENT	Subevent_Code	Refer to specific device Bluetooth® LE Stack ACI APIs and events documentation available on SW package, docs\gui_aci_html folders (ACI events, HCI LE meta events section)
HCI_VENDOR_EVENT	Ecode	Refer to specific device Bluetooth® LE stack ACI APIs and events doxygen documentation available on SW package, Docs Docs\gui_aci_html folder (ACI events, ACI GAP events, ACI GATT/ATT events, ACI L2CAP events, ACI HAL event sections)
NA	NA	Refer to specific device Bluetooth® LE stack ACI APIs and events doxygen documentation available on SW package, Docs Docs\gui_aci_html folder (ACI events, HCI event section)

Below are some code examples using the WAIT_EVENT() macro-command:

Example 1:

```
#Wait any events
evt = WAIT_EVENT()
if (evt.event_code == HCI_LE_META_EVENT):
   #User specific code ……
elif (evt.event_code==HCI_VENDOR_EVENT):
   #User specific code ……
```

Example 2:

```
# Wait an HCI_LE_META_EVENT
evt = WAIT_EVENT(HCI_LE_META_EVENT)
# Using evt.get_param('Subevent_Code').val it's possible to identify the specific HCI_LE_META
_EVENT
# parameter type value
evtCode = evt.get_param('Subevent_Code').val
# Check if received event is HCI_LE_ENHANCED_CONNECTION_COMPLETE_EVENT
if (evtCode == HCI_LE_ENHANCED_CONNECTION_COMPLETE_EVENT):
  # If connection complete status is success, get connection handle
  if evt.get_param('Status').val==0x00:
     conn_handle= evt.get_param('Connection_Handle').val
```

Example 3:

```
#Wait HCI_VENDOR_EVENT event_code
evt = WAIT_EVENT(HCI_VENDOR_EVENT)
#Using evt.get_param('Ecode').val it's possible to identify the specific HCI_VENDOR_EVENT par
ameter type value
evtCode= evt.get_param('Ecode').val
if IS_STM32WB0X:
  if(evtCode == ACI_GATT_CLT_NOTIFICATION_EVENT):
     conn_handle=evt.get_param('Connection_Handle').val
elif IS_STM32WBAX:
  if(evtCode == ACI_GATT_NOTIFICATION_EVENT):
     conn_handle=evt.get_param('Connection_Handle').val
```

Example 4:

```
#Wait the Ecode
ACI_GATT_PROC_COMPLETE_EVENT (HCI_VENDOR_EVENT #event_code).
if IS_STM32WB0X
:
   WAIT_EVENT(HCI_VENDOR_EVENT, timeout=30, Ecode=ACI_GATT_CLT_PROC_COMPLETE_EVENT)
elif IS_STM32WBAX:
   WAIT_EVENT(HCI_VENDOR_EVENT, timeout=30, Ecode=ACI_GATT_PROC_COMPLETE_EVENT)
```

Note: If no timeout parameter is specified, it awaits until, respectively, the `ACI_GATT_CLT_PROC_COMPLETE_EVENT`*,* `ACI_GATT_PROC_COMPLETE_EVENT` *.*

Example 5:

```
#Wait an event for 10 seconds with continueOnEvtMiss set to True
#If no event occurs, the script continues (no exception is raised).
WAIT_EVENT(timeout=10, continueOnEvtMiss =True)
```

Note: If continueOnEvtMiss parameter is set to false and if no event within the selected timeout occurs, an exception is raised.

Example 6:

```
#Wait the HCI_DISCONNECTION_COMPLETE_EVENT event_code
WAIT_EVENT(HCI_DISCONNECTION_COMPLETE_EVENT)
```

Example 7:

```
#Create a connection on STM32WBA and wait for the HCI_LE_ENHANCED_CONNECTION_COMPLETE_EVENT
Status = ACI_GAP_EXT_CREATE_CONNECTION(Peer_Address=[0x12, 0x34, 0x00, 0xE1, 0x80, 0x02])
event= WAIT_EVENT(HCI_LE_META_EVENT,timeout=30,Subevent_Code= HCI_LE_ENHANCED_CONNECTION_COMP
LETE_EVENT)
if event.get_param('Status').val==0x00:
    # Store the connection handle
    conn_handle= event.get_param('Connection_Handle').val
    #User defined code…
```

Example 8:

```
#Create a connection on STM32WB0 and wait for the HCI_LE_ENHANCED_CONNECTION_COMPLETE_EVENT
Status = ACI_GAP_CREATE_CONNECTION(Peer_Address=[0x12, 0x34, 0x00, 0xE1, 0x80, 0x02])
event= WAIT_EVENT(HCI_LE_META_EVENT, timeout=30, Subevent_Code=
                  HCI_LE_ENHANCED_CONNECTION_COMPLETE_EVENT)
if event.get_param('Status').val==0x00:
    #Store the connection handle
    conn_handle= event.get_param('Connection_Handle').val
    #User defined code…
```

3.4.1 Scripts engine loading and running steps

To load and run a python™ script using the script engine, the following steps must be observed:

1. In the STM32CubeWiSE - Bluetooth® LE Explorer, "Scripts" window, script engine section, click on tab “…”, browse to the script location and select the script.
2. Click on the “Run Script” tab to run the script. The execution flow (commands and events) is displayed in the “Sent/Received Packets” section.

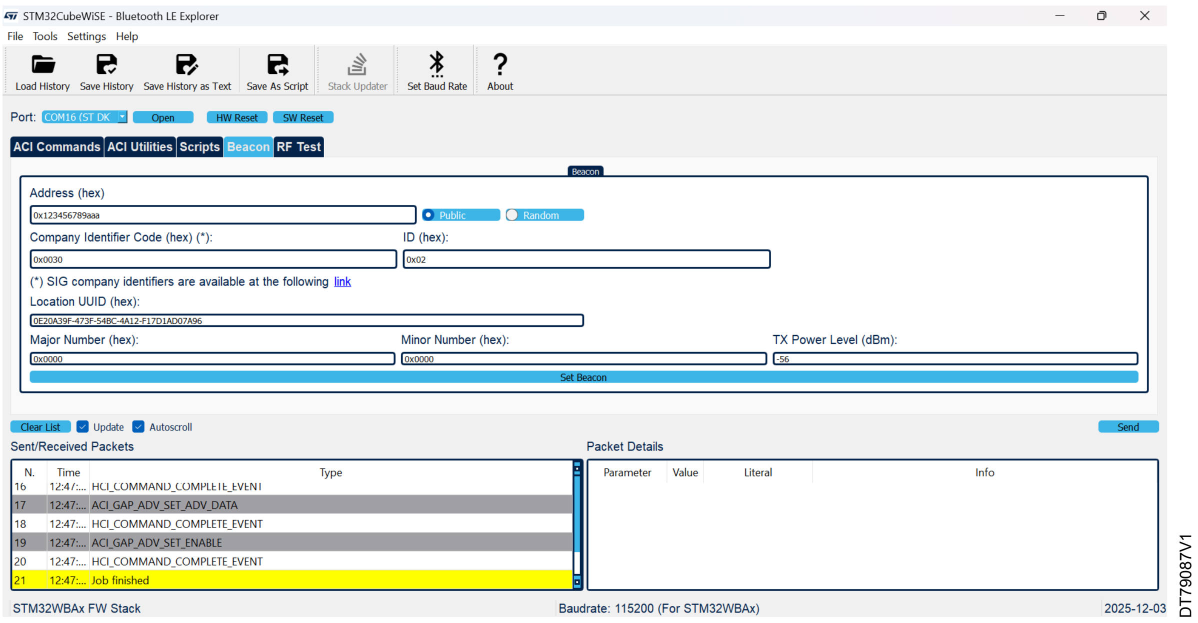
In the STM32CubeWiSE - Bluetooth® LE Explorer SW package some reference scripts are available in the "app/scripts" folder.

Note: 1. *To write and use the scripts, the user should have some knowledge of the Python language (Python 3.7.6), and a good understanding of the Bluetooth® LE stack ACI commands and related events.*

3.5 Beacon window

The STM32CubeWiSE - Bluetooth® LE Explorer PC GUI "Beacon" window provides some tabs to configure a supported STM32 Bluetooth® LE devices as a Bluetooth® LE Beacon device that transmits advertising packets with specific manufacturer data.

Figure 7. Beacon window



Port: COM16 (ST DK

Address (hex): 0x123456789aaa [✓ Public] Random

Company Identifier Code (hex) (*): 0x0030

ID (hex): 0x02

(*) SIG company identifiers are available at the following link

Location UUID (hex): 0E20A39F-473F-54BC-4A12-F17D1AD07A96

Major Number (hex): 0x0000

Minor Number (hex): 0x0000

TX Power Level (dBm): -56

Sent/Received Packets

N.	Time	Type
16	12:47:...	HCI_COMMAND_COMPLETE_EVENT
17	12:47:...	ACI_GAP_ADV_SET_ADV_DATA
18	12:47:...	HCI_COMMAND_COMPLETE_EVENT
19	12:47:...	ACI_GAP_ADV_SET_ENABLE
20	12:47:...	HCI_COMMAND_COMPLETE_EVENT
21	12:47:...	Job finished

Packet Details

Parameter	Value	Literal	Info

STM32WBAx FW Stack   Baudrate: 115200 (For STM32WBAx)   2025-12-03

The user can configure the advertising data fields in the following table for the Bluetooth® LE Beacon device through the Beacon window configuration parameters.

Table 7. Beacon window configuration parameters

Data field	Description	Notes
Address	Device address	-
Device address type	Public or random	-
Company identifier code	SIG company identifier	Default is 0x0030 (STMicroelectronics)
ID	Beacon ID	Fixed value
Location UUID	Beacons UUID	Used to distinguish specific Beacons from others
Major number	Identifier for a group of beacons	Used to group a related set of Beacons
Minor number	Identifier for a single beacon	Used to identify a single Beacon
TxPower	Level 2's complement of the Tx power	Used to establish how far you are from the device

To configure the selected platform as a Bluetooth® LE Beacon device, click on the “Set Beacon” tab.

3.6 RF Test window

The STM32CubeWiSE - Bluetooth® LE Explorer GUI provides the "RF Test" window that permits to perform the following tests:

1. Start/stop a tone on a specific Bluetooth® LE RF channel
2. Perform a Bluetooth® LE packer error rate (PER) tests using Bluetooth® LE direct test mode (DTM) commands.

3.6.1 Start/stop a tone

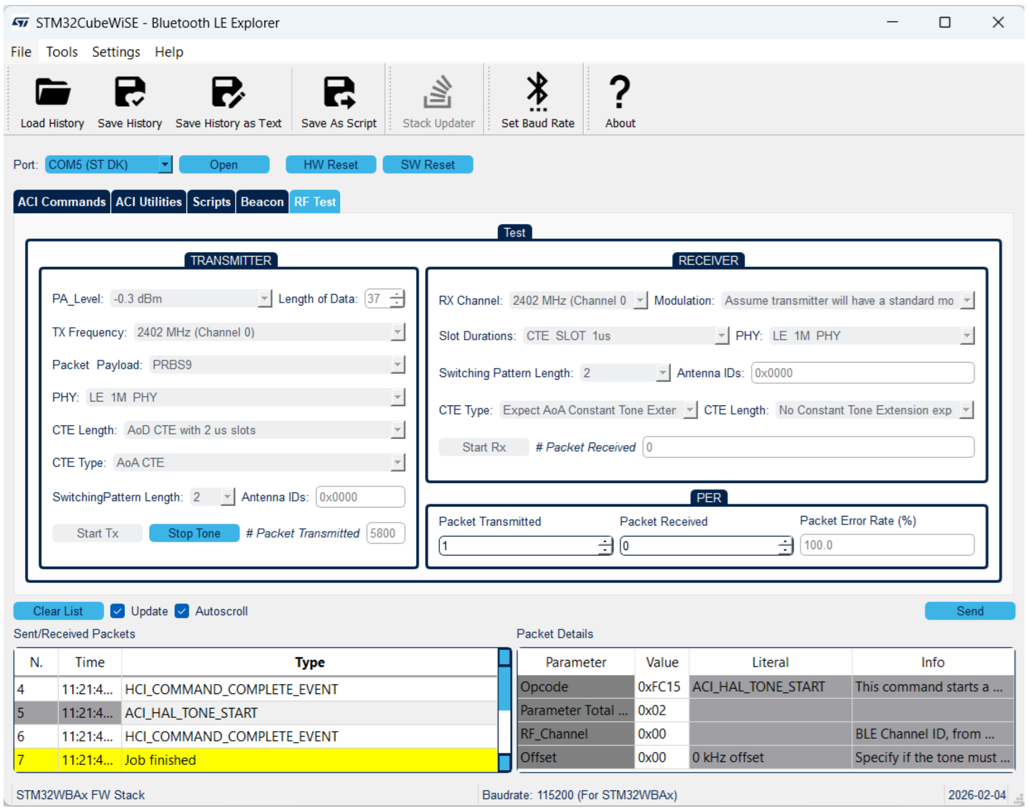
To start a tone on a specific RF Bluetooth® LE channel:

1. Connect a supported device platforms to PC USB port
2. Launch an instance of STM32CubeWiSE - Bluetooth® LE Explorer PC GUI
3. Open the relative COM port
4. Go to RF Test window and in the TRANSMITTER section:
   a. Set Bluetooth® LE channel by TX frequency combo box
   b. Set TX power in the related combo box
   c. Click on the "Start Tone" button

To stop a tone on a specific RF Bluetooth® LE channel:

1. Go to RF Test window and in the TRANSMITTER section:
   a. Click on stop tone button (stop button is available only when a tone is started)

Figure 8. Start a tone



Port: COM5 (ST DK)

TRANSMITTER

PA_Level: -0.3 dBm Length of Data: 37
TX Frequency: 2402 MHz (Channel 0)
Packet Payload: PRBS9
PHY: LE 1M PHY
CTE Length: AoD CTE with 2 us slots
CTE Type: AoA CTE
SwitchingPattern Length: 2 Antenna IDs: 0x0000
Packet Transmitted 5800

RECEIVER

RX Channel: 2402 MHz (Channel 0 Modulation: Assume transmitter will have a standard mo
Slot Durations: CTE SLOT 1us PHY: LE 1M PHY
Switching Pattern Length: 2 Antenna IDs: 0x0000
CTE Type: Expect AoA Constant Tone Exter CTE Length: No Constant Tone Extension exp
Packet Received 0

PER

Packet Transmitted	Packet Received	Packet Error Rate (%)
1	0	100.0

☑ Update ☑ Autoscroll

Sent/Received Packets

N.	Time	Type
4	11:21:4...	HCI_COMMAND_COMPLETE_EVENT
5	11:21:4...	ACI_HAL_TONE_START
6	11:21:4...	HCI_COMMAND_COMPLETE_EVENT
7	11:21:4...	Job finished

Packet Details

Parameter	Value	Literal	Info
Opcode	0xFC15	ACI_HAL_TONE_START	This command starts a ...
Parameter Total ...	0x02		
RF_Channel	0x00		BLE Channel ID, from ...
Offset	0x00	0 kHz offset	Specify if the tone must ...

STM32WBAx FW Stack | Baudrate: 115200 (For STM32WBAx) | 2026-02-04

DT79088v1

3.6.2 Direct test mode (DTM) tests

The STM32CubeWiSE - Bluetooth® LE Explorer PC GUI "RF Test" window let you target a packet error rate test scenario using the Bluetooth® LE direct test mode commands.

Two sections are available:

1. TRANSMITTER section to transmit reference packets to a fixed interval
2. RECEIVER section to receive reference packets to a fixed interval

TRANSMITTER section

This section let you set the following items:

- The power level of transmitter
- The frequency of transmitter
- Length of data to transmit in each packet
- Packet payload format as defined on Bluetooth® LE specification, direct test mode section
- PHY used by transmitter

Clicking on "Start Tx" button, test reference packets are sent to a fixed interval.

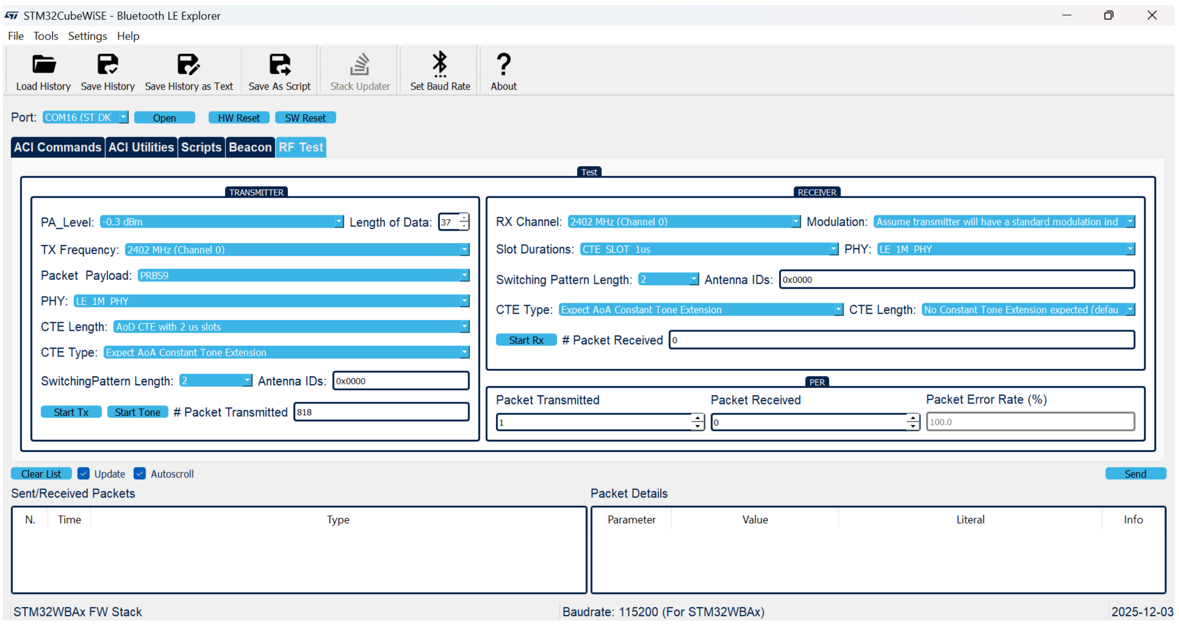
RECEIVER section

This section let you set the following items:

- The frequency or channel of receiver
- PHY used by receiver
- Modulation

Clicking on "Start Rx" button, test reference packets are received to a fixed interval.

Figure 9. Transmitter and receiver sections



Port: COM16 (ST DK

TRANSMITTER

PA_Level: -0.3 dBm Length of Data: 37
TX Frequency: 2402 MHz (Channel 0)
Packet Payload: PRBS9
PHY: LE 1M PHY
CTE Length: AoD CTE with 2 us slots
CTE Type: Expect AoA Constant Tone Extension
SwitchingPattern Length: 2 Antenna IDs: 0x0000
Packet Transmitted 818

RECEIVER

RX Channel: 2402 MHz (Channel 0) Modulation: Assume transmitter will have a standard modulation ind
Slot Durations: CTE SLOT 1us PHY: LE 1M PHY
Switching Pattern Length: 2 Antenna IDs: 0x0000
CTE Type: Expect AoA Constant Tone Extension CTE Length: No Constant Tone Extension expected (defau
Packet Received 0

PER

Packet Transmitted	Packet Received	Packet Error Rate (%)
1	0	100.0

Sent/Received Packets

N.	Time	Type

Packet Details

Parameter	Value	Literal	Info

STM32WBAx FW Stack Baudrate: 115200 (For STM32WBAx) 2025-12-03

3.6.3 Packet error rate (PER) test procedure

To perform a packet error rate test using the standard Bluetooth® LE direct test mode commands the following steps are needed.

Start PER test

1. Connect two supported STM32 Bluetooth® LE platforms to PC USB ports
2. Open two instances of STM32CubeWiSE - Bluetooth® LE Explorer PC GUI on both devices (called TX and RX devices, respectively)
3. In each instance of STM32CubeWiSE - Bluetooth® LE Explorer PC GUI, open COM port relative to the TX/RX device
4. Ensure the antennas are plugged into the devices where applicable
5. In the GUI related to RX device:
   – Go to RF Test window, RECEIVER section
   – Set RX frequency or channel
   – Set PHY
   – Set modulation
   – Click on "Start Rx" button to start receiver test
6. In the GUI related to TX device:
   – Go to RF Test window, TRANSMITTER section
   – Set TX power
   – Set TX frequency
   – Set length of data
   – Set packet payload format
   – Set PHY
   – Click on "Start Tx" button, to start transmitter test

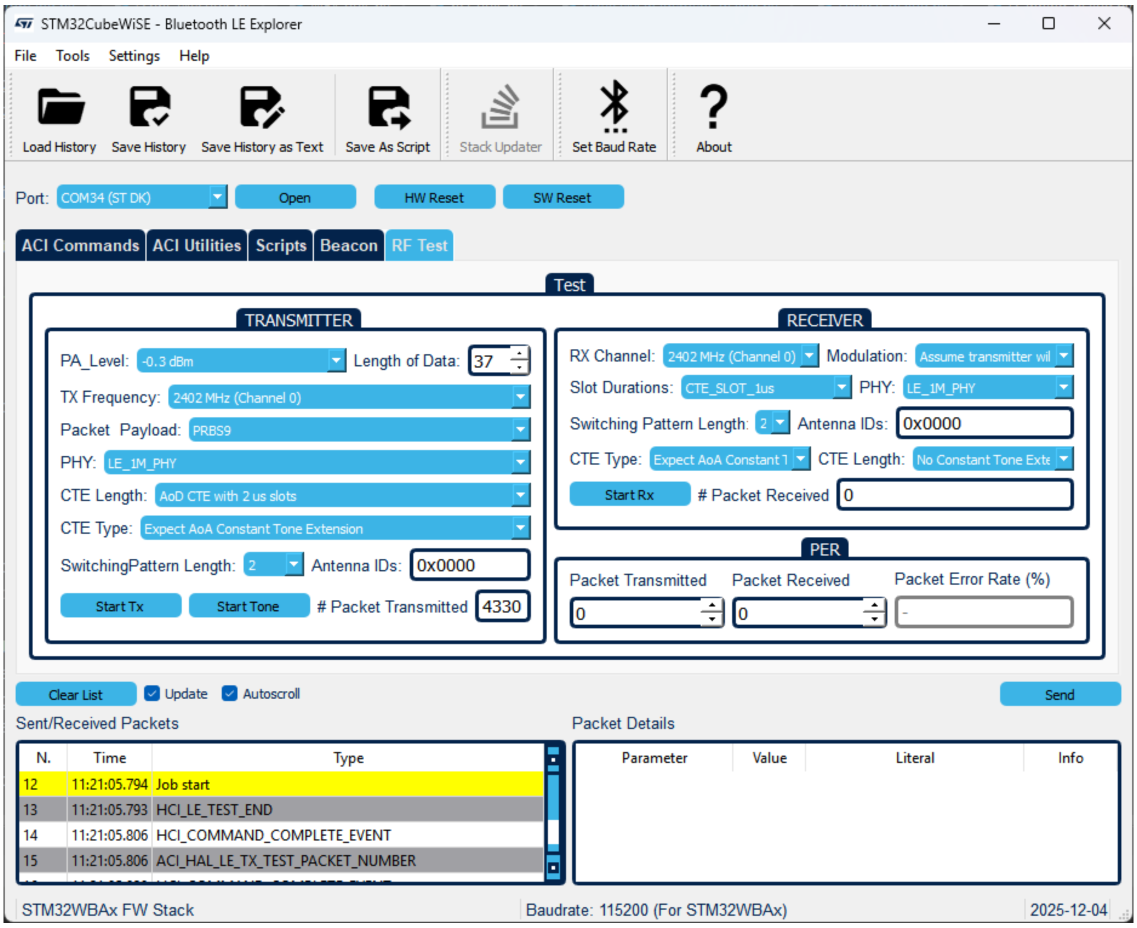
Stop PER test

1. In the GUI related to TX device:
   – Go to RF Test window, TRANSMITTER section
   – Click on "Stop Tx" button. The number of transmitted packets are displayed on #Packet Transmitted field.
2. In the GUI related to RX device:
   – Go to RF Test window, RECEIVER section
   – Click on "Stop Rx" button. The number of received packets are displayed on #PacketReceived field.

Get PER (packet error rate) value

1. In the GUI related to RX device:
   – Go to RF Test window, RECEIVER section
   – In the PER section, insert the number of transmitted packet from TX device in the packet transmitted field (read this value from TRANSMITTER section in the GUI related to TX device)
   – The PER (packet error rate) value is showed in the packet error rate field

Figure 10. PER test: TX device



Port: COM34 (ST DK)

TRANSMITTER

PA_Level: -0.3 dBm Length of Data: 37
TX Frequency: 2402 MHz (Channel 0)
Packet Payload: PRBS9
PHY: LE_1M_PHY
CTE Length: AoD CTE with 2 us slots
CTE Type: Expect AoA Constant Tone Extension
SwitchingPattern Length: 2 Antenna IDs: 0x0000
Packet Transmitted 4330

RECEIVER

RX Channel: 2402 MHz (Channel 0) Modulation: Assume transmitter wil
Slot Durations: CTE_SLOT_1us PHY: LE_1M_PHY
Switching Pattern Length: 2 Antenna IDs: 0x0000
CTE Type: Expect AoA Constant T CTE Length: No Constant Tone Exte
Packet Received 0

PER

Packet Transmitted	Packet Received	Packet Error Rate (%)
0	0	-

Sent/Received Packets

N.	Time	Type
12	11:21:05.794	Job start
13	11:21:05.793	HCI_LE_TEST_END
14	11:21:05.806	HCI_COMMAND_COMPLETE_EVENT
15	11:21:05.806	ACI_HAL_LE_TX_TEST_PACKET_NUMBER

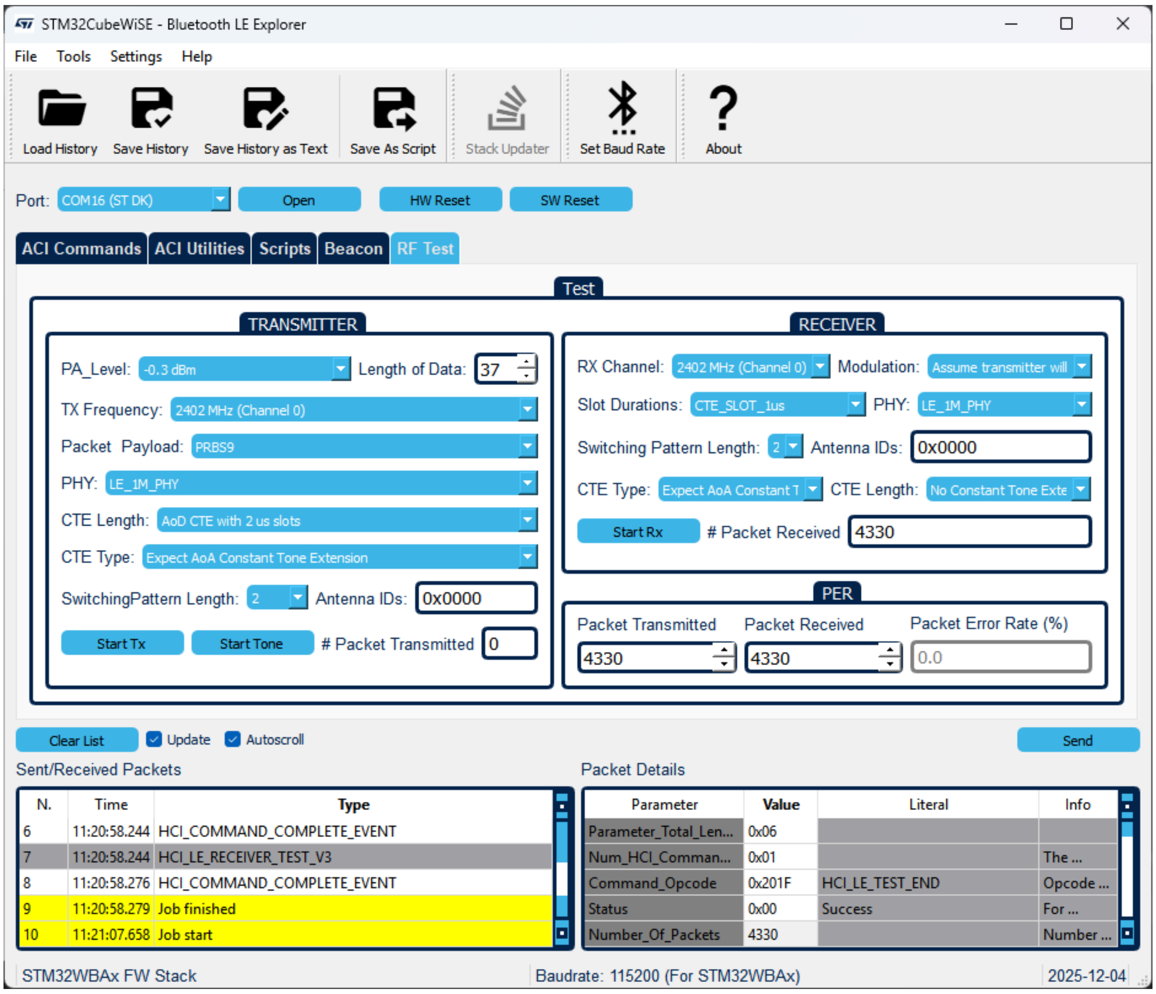
Packet Details

Parameter	Value	Literal	Info

STM32WBAx FW Stack Baudrate: 115200 (For STM32WBAx) 2025-12-04

DT79090V1

Figure 11. PER test: RX device



Port: COM16 (ST DK)

TRANSMITTER

PA_Level: -0.3 dBm Length of Data: 37
TX Frequency: 2402 MHz (Channel 0)
Packet Payload: PRBS9
PHY: LE_1M_PHY
CTE Length: AoD CTE with 2 us slots
CTE Type: Expect AoA Constant Tone Extension
SwitchingPattern Length: 2 Antenna IDs: 0x0000
Packet Transmitted: 0

RECEIVER

RX Channel: 2402 MHz (Channel 0) Modulation: Assume transmitter will
Slot Durations: CTE_SLOT_1us PHY: LE_1M_PHY
Switching Pattern Length: 2 Antenna IDs: 0x0000
CTE Type: Expect AoA Constant T CTE Length: No Constant Tone Exte
Packet Received: 4330

PER

Packet Transmitted	Packet Received	Packet Error Rate (%)
4330	4330	0.0

Sent/Received Packets

N.	Time	Type
6	11:20:58.244	HCI_COMMAND_COMPLETE_EVENT
7	11:20:58.244	HCI_LE_RECEIVER_TEST_V3
8	11:20:58.276	HCI_COMMAND_COMPLETE_EVENT
9	11:20:58.279	Job finished
10	11:21:07.658	Job start

Packet Details

Parameter	Value	Literal	Info
Parameter_Total_Len...	0x06		
Num_HCI_Comman...	0x01		The ...
Command_Opcode	0x201F	HCI_LE_TEST_END	Opcode ...
Status	0x00	Success	For ...
Number_Of_Packets	4330		Number ...

STM32WBAx FW Stack Baudrate: 115200 (For STM32WBAx) 2025-12-04

DT79091V1

3.7 Tools

The STM32CubeWiSE - Bluetooth® LE Explorer PC GUI application has some functions that can be accessed through the tools menu. These tools are described in this section.

3.7.1 Get version

The "Get Version..." tool is used to retrieve the device hardware version and Bluetooth® LE firmware version.

3.7.2 FUOTA bootloader

The "FUOTA bootloader ..." tool let you program new firmware to a remote supported STM32 Bluetooth® LE device using the firmware upgrade over the air protocol via Bluetooth® LE technology. Refer to the dedicated documentation available on https://www.st.com for further information:

- STM32WBA FUOTA wiki: https://wiki.st.com/stm32mcu/wiki/Connectivity:STM32WBA_FUOTA
- AN5977: “How to build a Bluetooth® LE application with STM32WB0 MCUs”.

3.7.3 Stack updater

The "Stack Updater..." tool can be used to update the firmware in the connected STM32 Bluetooth® LE devices provided that it has been already configured with a BLE_TransparentMode application which supports the updater capability.

To use this tool:

1. Go to "Tools → Stack Updater..."
2. Select the correct BLE_TransparentMode with updater capability for the selected device
3. Press "Update" to start the update procedure

If the procedure completes with no errors, the new firmware has been loaded into the device internal flash memory.

Note:
1. *STM32WBA BLE_TransparentMode does not support the stack updater capability.*
2. *BLE_TransparentMode with stack updater capability is available in STM32CubeWB0 software package.*

User is requested to load the following applications with this order:

1. BLE_TransparentMode_UART_Updater: the application which provides the updater capability
2. BLE_TransparentMode_UART_for_Updater: it is the BLE_TransparentMode which must be used for supporting the Stack Updater capability. It must be loaded at a specific offset (Flash base address + 0x1000)

Once the two applications are loaded, the BLE_TransparentMode_UART_for_Updater.bin file can be loaded through the "Stack updater" tool.

3.8 Settings

The "Settings" menu lets you configure the device/Bluetooth® LE stack version to be used from the PC GUI (when no device is actually connected to a PC USB port). It also lets you configure the PC GUI serial baud rate (serial UART communication only).

In order to use this function.

1. Go to settings → "Fw Stack Version" -> and select device/Bluetooth® LE stack version.
2. Go to settings → select "Set Baud Rate..." and choose the value (default value is 115200 for STM32WBA devices and 921600 for the STM32WB0 devices).

4 Script launcher utility

The script launcher is a standalone utility, which let you run a script without using the STM32CubeWiSE - Bluetooth® LE Explorer PC GUI script engine tool.

The script launcher utility (`STM32CubeWiSE-Bluetooth_LE_Explorer_Script_Launcher.exe`) is included in the STM32CubeWiSE - Bluetooth® LE Explorer software package in the "app" folder.

4.1 Requirements

To use the script launcher utility on a specific device, the corresponding platform must be connected to a PC USB port and loaded with the same pre-built binary file used for the STM32CubeWiSE - Bluetooth® LE Explorer PC GUI, as described in Section 1.1 Requirements.

Note: When using the script launcher utility on a specific device and related COM port, please make sure that the expected serial baud rate is properly set through the option -b:

- *STM32WBA serial baud rate: 115200 (default configuration);*
- *STM32WB0 serial baud rate: 921600.*

4.2 Script launcher utility options

To use the script launcher utility on a specific device, open a window command prompt shell and launch the relative `STM32CubeWiSE-Bluetooth_LE_Explorer_Script_Launcher.exe` (type –h to list all the supported options):

```
C:\Users\{UserName}\ST\ STM32CubeWiSE - Bluetooth LE Explorer x.x.x\app>
STM32CubeWiSE-Bluetooth_LE_Explorer_Script_Launcher.exe -h
usage: Script_launcher.py [-h] [-g] [-d] [-v {0,1,2}] [-p PORT]
                          [-b {921600,460800,256000,128000,115200,57600,56000,43000,38400,288
00,19200,9600,4800,2400,1200,600,300,110}]
                          [-s SCRIPT_FILE] [-z]
                          [LIST_PARAMS [LIST_PARAMS ...]]

STM32CubeWiSE - Bluetooth LE Explorer Script Launcher version 0.1.0.
positional arguments:
  LIST_PARAMS           list of script parameters

optional arguments:
  -h, --help            Show this help message and exit.
  -g, --get             get list of existing Com port and exit
  -d, --debug           debug script file
  -v {0,1,2}, --verbose {0,1,2}
                        Increase output verbosity, set debug level up to 3.
                        The default value is 0
  -p PORT, --port PORT  COM port
  -b {921600,460800,256000,128000,115200,57600,56000,43000,38400,28800,19200,9600,4800,2400,1
200,600,300,110}, --baudrate {921600,460800,256000,128000,115200,57600,56000,43000,38400,2880
0,19200,9600,4800,2400,1200,600,300,110}
                        set Baud Rate: 115200 (default).WARNING:115200 is
                        valid for STM32WBAx devices.
  -w {STM32WBAx,STM32WB0x}, --wb {STM32WBAx,STM32WB0x}
                        set STM32WBAx or STM32WB0x: STM32WBAx (default).
  -s SCRIPT_FILE, --script SCRIPT_FILE
                        script file name
  -z, --save            save two log files with the same name of script (txt
                        and csv) under log folder
```

Option to get the list of all available COMx ports (devices are connected to the PC USB)

> `STM32CubeWiSE-Bluetooth_LE_Explorer_Script_Launcher.exe -g`

List of available COM ports:

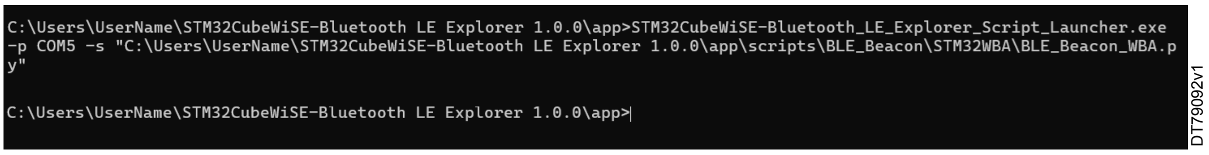
- COM5 (ST DK)

Options to run a script on a connected device

```
> STM32CubeWiSE-Bluetooth_LE_Explorer_Script_Launcher.exe -p COM5 -s
C:\Users\UserName\ST\STM32CubeWiSE - Bluetooth LE Explorer
x.y.z\app\scripts\BLE_Beacon\STM32WBA\BLE_Beacon_WBA.py
```

Note: COM5 is the platform virtual COM port.

Figure 12. Script launcher: run a script



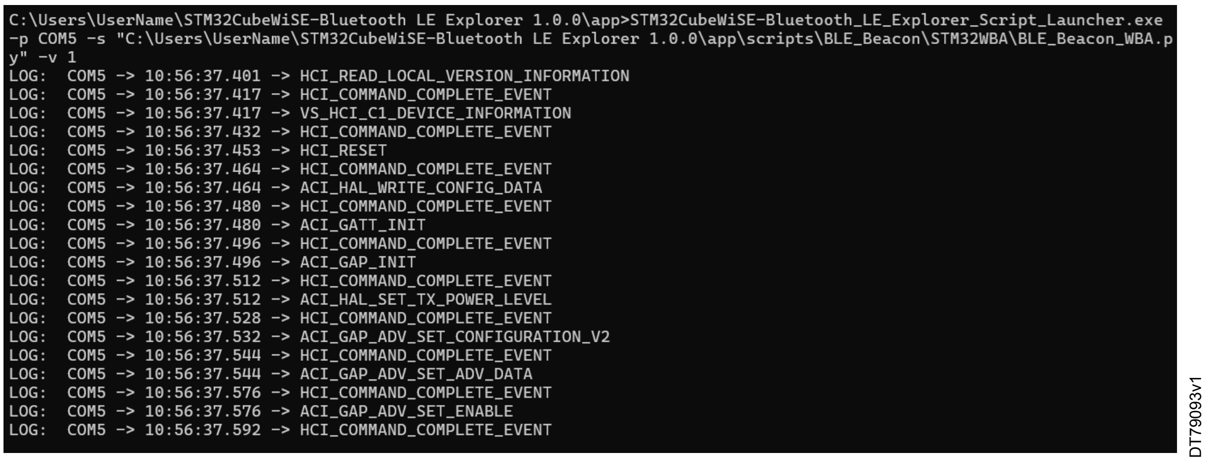
```
C:\Users\UserName\STM32CubeWiSE-Bluetooth LE Explorer 1.0.0\app>STM32CubeWiSE-Bluetooth_LE_Explorer_Script_Launcher.exe
-p COM5 -s "C:\Users\UserName\STM32CubeWiSE-Bluetooth LE Explorer 1.0.0\app\scripts\BLE_Beacon\STM32WBA\BLE_Beacon_WBA.p
y"

C:\Users\UserName\STM32CubeWiSE-Bluetooth LE Explorer 1.0.0\app>
```

Options to run a script on a connected device with log data

```
> STM32CubeWiSE-Bluetooth_LE_Explorer_Script_Launcher.exe -p COM5 -s
C:\Users\UserName\ST\STM32CubeWiSE - Bluetooth LE Explorer
x.y.z\app\scripts\BLE_Beacon\STM32WBA\BLE_Beacon_WBA.py -v 1
```

Note: COM10 is the platform virtual COM port.

Figure 13. Script launcher: run a script with log data



```
C:\Users\UserName\STM32CubeWiSE-Bluetooth LE Explorer 1.0.0\app>STM32CubeWiSE-Bluetooth_LE_Explorer_Script_Launcher.exe
-p COM5 -s "C:\Users\UserName\STM32CubeWiSE-Bluetooth LE Explorer 1.0.0\app\scripts\BLE_Beacon\STM32WBA\BLE_Beacon_WBA.p
y" -v 1
LOG:  COM5 -> 10:56:37.401 -> HCI_READ_LOCAL_VERSION_INFORMATION
LOG:  COM5 -> 10:56:37.417 -> HCI_COMMAND_COMPLETE_EVENT
LOG:  COM5 -> 10:56:37.417 -> VS_HCI_C1_DEVICE_INFORMATION
LOG:  COM5 -> 10:56:37.432 -> HCI_COMMAND_COMPLETE_EVENT
LOG:  COM5 -> 10:56:37.453 -> HCI_RESET
LOG:  COM5 -> 10:56:37.464 -> HCI_COMMAND_COMPLETE_EVENT
LOG:  COM5 -> 10:56:37.464 -> ACI_HAL_WRITE_CONFIG_DATA
LOG:  COM5 -> 10:56:37.480 -> HCI_COMMAND_COMPLETE_EVENT
LOG:  COM5 -> 10:56:37.480 -> ACI_GATT_INIT
LOG:  COM5 -> 10:56:37.496 -> HCI_COMMAND_COMPLETE_EVENT
LOG:  COM5 -> 10:56:37.496 -> ACI_GAP_INIT
LOG:  COM5 -> 10:56:37.512 -> HCI_COMMAND_COMPLETE_EVENT
LOG:  COM5 -> 10:56:37.512 -> ACI_HAL_SET_TX_POWER_LEVEL
LOG:  COM5 -> 10:56:37.528 -> HCI_COMMAND_COMPLETE_EVENT
LOG:  COM5 -> 10:56:37.532 -> ACI_GAP_ADV_SET_CONFIGURATION_V2
LOG:  COM5 -> 10:56:37.544 -> HCI_COMMAND_COMPLETE_EVENT
LOG:  COM5 -> 10:56:37.544 -> ACI_GAP_ADV_SET_ADV_DATA
LOG:  COM5 -> 10:56:37.576 -> HCI_COMMAND_COMPLETE_EVENT
LOG:  COM5 -> 10:56:37.576 -> ACI_GAP_ADV_SET_ENABLE
LOG:  COM5 -> 10:56:37.592 -> HCI_COMMAND_COMPLETE_EVENT
```

Revision history

Table 8. Document revision history

Date	Revision	Changes
19-Feb-2026	1	Initial release.

Contents

List of tables

List of figures