
Migrating from Bluetooth® Low Energy stack v3.x to v4.x on STM32WB0 MCUs

Introduction

This document describes the modifications from Bluetooth® Low Energy stack v3.x to v4.x on STM32WB0 series MCUs, and the related implications when migrating an application from Bluetooth® Low Energy stack v3.x to Bluetooth® Low Energy stack v4.x.

It provides the list of Bluetooth® Low Energy stack v4.x changes:

- Bluetooth® Low Energy stack modular configurations options
- Bluetooth® Low Energy stack initialization parameters
- Bluetooth® Low Energy stack error codes
- Bluetooth® Low Energy stack commands
- Bluetooth® Low Energy stack events

Further, it details the new Bluetooth® Low Energy security framework in terms of redefined and new commands and events introduced within the Bluetooth® Low Energy stack v4.x.

Note: Bluetooth® Low Energy stack v4.x is provided within the STM32_BLE middleware included on STM32CubeWB0 SW package. The Bluetooth® Low Energy stack v3.x is provided within the STSW-BNRGLP-DK SW package.

1 General information

This document applies to STM32WB0 Arm[®]-based MCUs.

For information on Bluetooth[®], refer to the www.bluetooth.com website

Note: Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



2 Reference documents

N°	Documents
[1]	How to build a Bluetooth® Low Energy application with STM32WB0 MCUs (AN5977)
[2]	https://wiki.st.com/stm32mcu/wiki/Utility:Sequencer

3 Modular configurations and initialization of Bluetooth® Low Energy stack v4.x vs v3.x

3.1 Bluetooth® Low Energy stack modular configuration options

Table 1 describes the available modular configurations options on Bluetooth® Low Energy stack v3.x and v4.x that may be selected by user based on the specific application scenario:

Table 1. Bluetooth® Low Energy stack modular configurations

Modular configuration option	3.x ⁽¹⁾	4.x ⁽²⁾	Associated symbol on Bluetooth® Low Energy stack v4.x (to be set to 0 or 1)
Controller privacy	Yes	Yes	CFG_BLE_CONTROLLER_PRIVACY_ENABLED
Controller scan role	Yes	Yes	CONTROLLER_SCAN_ENABLED ⁽³⁾
LE secure connections	Yes	Yes	CFG_BLE_SECURE_CONNECTIONS_ENABLED
Controller data length extension	Yes	Yes	CFG_BLE_CONTROLLER_DATA_LENGTH_EXTENSION_ENABLED
Controller LE 2M/Coded PHY	Yes	Yes	CFG_BLE_CONTROLLER_2M_CODED_PHY_ENABLED
Controller extended ADV/SCAN	Yes	Yes	CFG_BLE_CONTROLLER_EXT_ADV_SCAN_ENABLED
L2CAP connection oriented channels	Yes	Yes	CFG_BLE_L2CAP_COS_ENABLED
Controller periodic advertising	Yes	Yes	CFG_BLE_CONTROLLER_PERIODIC_ADV_ENABLED
Enhanced ATT	Yes	No	-
Controller periodic advertising with responses (PAWR)	No	Yes	CFG_BLE_CONTROLLER_PERIODIC_ADV_WR_ENABLED
Controller power control & path Loss	Yes	Yes	CFG_BLE_CONTROLLER_POWER_CONTROL_ENABLED
Connections support	Yes	Yes	CFG_BLE_CONNECTION_ENABLED
Controller LE Channel classification	Yes	Yes	CFG_BLE_CONTROLLER_CHAN_CLASS_ENABLED
Controller BIS support Broadcast isochronous streams	Yes	Yes	CFG_BLE_CONTROLLER_BIS_ENABLED
Controller CIS support Connected isochronous streams	Yes	Yes	CFG_BLE_CONTROLLER_CIS_ENABLED
Connection subrating	Yes	Yes	CFG_BLE_CONNECTION_SUBRATING_ENABLED
Controller constant tone extension ⁽⁴⁾	Yes	Yes	CFG_BLE_CONTROLLER_CTE_ENABLED

1. Bluetooth® Low Energy stack v3.2/3.2a.
2. Bluetooth® Low Energy stack v4.0 is the current version
3. Renamed option name on Bluetooth® Low Energy stack v4.x
4. Supported only by STM32WB05xZ and STM32WB09xE devices.

In the context of STM32CubeWB0 software package, the Bluetooth® Low Energy modular configurations options are defined within the file `app_conf.h` available on each Bluetooth® Low Energy application under `Core\Inc` folder.

This file is generated by the CubeMX tool, when building an STM32_BLE (STM32WB0 Bluetooth® Low Energy middleware) application. User can select the Bluetooth® Low Energy stack modular configuration options through the related CubeMX tool, in the STM32_BLE Application configuration - Modular Options tab.

3.2 Bluetooth® Low Energy stack initialization parameters

The Bluetooth® Low Energy stack v4.x stack initialization parameters have been modified to support new input parameters requested for handling the periodic advertising with response (PAwR) feature and the redesigned isochronous channel feature.

Table 2 describes the available Bluetooth® Low Energy stack initialization parameters on both Bluetooth® Low Energy stack v3.x and v4.x.

Table 2. Bluetooth® Low Energy stack initialization parameters

Type	Field/Parameter name	3.x ⁽¹⁾	4.x	Description
uint8_t*	BLEStartRamAddress	Yes	Yes	Start address of the RAM buffer for GATT database allocated according to TOTAL_BUFFER_SIZE (32-bit aligned RAM area).
uint32_t	TotalBufferSize	Yes	Yes	TOTAL_BUFFER_SIZE return value, used to check the MACRO correctness.
uint16_t	NumAttrRecords	Yes	Yes	Maximum number of attributes that can be stored in the GATT database.
uint8_t	MaxNumOfClientProcs	Yes	Yes	Maximum number of concurrent client procedures. This value shall be less or equal to NumOfLinks.
uint8_t	NumOfRadioTasks ⁽²⁾	Yes	Yes	Maximum number of simultaneous radio tasks. Radio controllers support up to 128 simultaneous radio tasks, but the actual usable max value depends on the available device RAM (NUM_LINKS used in the calculation of BLE_STACK_TOTAL_BUFFER_SIZE).
uint16_t	NumOfEATTChannels	Yes	Yes	Maximum number of simultaneous EATT active channels
uint16_t	NumBlockCount	Yes	Yes	Number of allocated memory blocks.
uint16_t	ATT_MTU	Yes	Yes	Maximum supported ATT_MTU size.
uint32_t	MaxConnEventLength	Yes	Yes	Maximum duration of the connection event when the device is in target mode in units of 625/256 us (~2.44 us).
uint16_t	SleepClockAccuracy	Yes	Yes	Sleep clock accuracy (ppm value).
uint8_t	NumOfAdvDataSet	Yes	Yes	Maximum number of advertising datasets. It is valid only when the advertising extension feature is enabled.
uint8_t	NumOfSubeventsPAwR	No	Yes	Maximum number of periodic advertising with responses subevents.
uint8_t	MaxPAwRSubeventDataCount	No	Yes	Maximum number of periodic advertising with responses subevents that data can be requested.
uint8_t	NumOfAuxScanSlots	Yes	Yes	Maximum number of slots for scanning on the secondary advertising channel, valid only when the advertising extension feature is enabled
uint8_t	NumOfSyncSlots	Yes	Yes	Maximum number of slots for synchronizing, valid only when periodic advertising and synchronizing feature is enabled
uint8_t	FilterAcceptListSizeLog2 ⁽³⁾	Yes	Yes	Two logarithms of the filter/resolving list size.
uint16_t	L2CAP_MPS	Yes	Yes	The maximum size of payload data in octets that the L2CAP layer entity can accept.
uint8_t	L2CAP_NumChannels	Yes	Yes	Maximum number of channels in LE credit based flow control mode.
uint8	CTE_MaxNumAntennalDs	Yes	Yes	Maximum number of antenna IDs in the antenna pattern used in CTE connection oriented mode

Type	Field/Parameter name	3.x ⁽¹⁾	4.x	Description
uint8	CTE_MaxNumIQSamples	Yes	Yes	Maximum number of IQ samples in the buffer used in CTE connection oriented mode.
uint8_t	NumOfSyncBIG	No	Yes	Maximum number of ISO synchronizer groups
uint8_t	NumOfBrcBIG	No	Yes	Maximum number of ISO broadcaster groups
uint8_t	NumOfSyncBIS	No	Yes	Maximum number of ISO synchronizer streams
uint8_t	NumOfBrcBIS	No	Yes	Maximum number of ISO broadcaster streams
uint8_t	NumOfCIG	No	Yes	Maximum number of connected isochronous groups (default = 2 if CIG is enabled; 0 if not enabled)
uint8_t	NumOfCIS	No	Yes	Maximum number of connected isochronous streams (default = 2 if CIG is enabled; 0 if not enabled)
uint16	isr0_fifo_size	Yes	Yes	Size of the internal FIFO used for critical controller events produced by the ISR (for example, Rx data packets)
uint16	isr1_fifo_size	Yes	Yes	Size of the internal FIFO used for noncritical controller events produced by the ISR (for example, advertising or IQ sampling reports)
uint16	user_fifo_size	Yes	Yes	Size of the internal FIFO used for controller and host events produced outside the ISR

1. *Bluetooth® Low Energy stack v3.2/3.2a.*
2. *On Bluetooth® Low Energy stack v3.x this parameter is called NumOfLinks*
3. *On Bluetooth® Low Energy stack v3.x this parameter is called WhiteListSizeLog2*

In the context of STM32CubeWB0 software package, the Bluetooth® Low Energy stack initialization parameters are defined within the file `app_conf.h` available on each Bluetooth® Low Energy application under `Core\Inc` folder.

This file is generated by the CubeMX tool, when building an STM32_BLE (STM32WB0 Bluetooth® Low Energy middleware) application. User can select the Bluetooth® Low Energy stack initialization parameter through the related CubeMX tool, in the STM32_BLE Application configuration - BLE stack tab.

4 Bluetooth® Low Energy stack v4.x command and event modifications

This section lists all the modifications that directly affected the commands and events of the Bluetooth® Low Energy stack v4.x.

4.1 New standard controller error codes

The following error codes in [Table 3](#) have been added to the `ble_status.h` definitions because they are used by the new features supported.

The documentation can be found in *Bluetooth Specification v.5.4, Vol. 1, Part F, Controller Error Codes*.

Table 3. Error codes

Controller error code (v4.x)	Value
BLE_ERROR_TOO_LATE (required by periodic advertising with responses)	(0x46)
BLE_ERROR_TOO_EARLY (required by periodic advertising with responses)	(0x47)

4.2 Bluetooth® Low Energy stack v4.x commands

4.2.1 New periodic advertising with response commands

Table 4. New commands related to PAwR added by Bluetooth® Low Energy stack v4.x

Command name (v4.x)	Short description
hci_le_extended_create_connection_v2	See <i>Bluetooth Spec v.5.4, Vol. 4, Part E, HCI, Sec. 7.8</i>
hci_le_set_extended_advertising_parameters_v2	
hci_le_set_periodic_sync_subevent	
hci_le_extended_create_connection_v2	
hci_le_set_periodic_advertising_parameters_v2	
hci_le_set_periodic_advertising_subevent_data	
hci_le_set_periodic_advertising_response_data	
ll_set_periodic_advertising_subevent_data_ptr	Specific commands used for advertising data management.
ll_set_periodic_advertising_response_data_ptr	
aci_gap_create_periodic_advertising_connection	Vendor specific, currently stubbed to void.

[Table 5](#) details the list of vendor specific commands added to the Bluetooth® Low Energy stack v4.x.

Table 5. List of new vendor-specific commands introduced by Bluetooth® Low Energy stack v4.x

Command name (v4.x)	Short description
aci_gap_pairing_resp	This command shall be given in response to an <code>aci_gap_pairing_event</code> , to allow or reject either the pairing request from the Central or the security request from the peripheral.
aci_gatt_clt_add_subscription_security_level	Set a minimum security level to accept server-initiated packets (notification, indication, multiple notifications).
aci_l2cap_cos_flow_control_credits_ind	Command to be issued when the device can receive additional K-frames in LE credit-based flow control mode.

4.2.2 Bluetooth® Low Energy stack v4.x renamed commands

Table 6. List of commands renamed on Bluetooth® Low Energy stack v4.x vs v3.x

New command name (v4.x)	Command name (v3.x)
hci_le_read_advertising_physical_channel_tx_power	hci_le_read_advertising_channel_tx_power
hci_le_read_filter_accept_list_size	hci_le_read_white_list_size
hci_le_clear_filter_accept_list	hci_le_clear_white_list
hci_le_add_device_to_filter_accept_list	hci_le_add_device_to_white_list
hci_le_remove_device_from_filter_accept_list	hci_le_remove_device_from_white_list
hci_le_read_remote_features	hci_le_read_remote_used_features
hci_le_enable_encryption	hci_le_start_encryption
hci_le_long_term_key_request_negative_reply	hci_le_long_term_key_requested_negative_reply
hci_le_remove_device_from_periodic_advertiser_list	hci_le_remove_device_from_periodic_advertising_list
hci_le_transmitter_test_v2	hci_le_enhanced_transmitter_test
aci_gap_set_security	aci_gap_slave_security_req
	aci_gap_send_pairing_req
aci_gap_add_devices_to_filter_accept_and_resolving_list	aci_gap_add_devices_to_white_and_resolving_list
aci_gap_set_security_requirements	aci_gap_set_authentication_requirement
aci_gap_configure_filter_accept_and_resolving_list	aci_gap_configure_white_and_resolving_list

4.2.3 Modified commands

Table 7. List of commands modified on Bluetooth® Low Energy stack v4.x

Modified command name (v4.x)	Description
aci_gap_init	Removed five parameters not directly used by the Bluetooth® Low Energy stack library
aci_gatt_clt_prepare_write_req	Added CID parameter to support both unenhanced and enhanced channels.
aci_gatt_clt_execute_write_req	
aci_gatt_clt_disc_all_primary_services	
aci_gatt_clt_disc_primary_service_by_uuid	
aci_gatt_clt_find_included_services	
aci_gatt_clt_disc_all_char_of_service	
aci_gatt_clt_disc_char_by_uuid	
aci_gatt_clt_disc_all_char_desc	
aci_gatt_clt_read	
aci_gatt_clt_read_using_char_uuid	
aci_gatt_clt_read_long	
aci_gatt_clt_read_multiple_char_value	
aci_gatt_clt_write_without_resp	
aci_gatt_clt_confirm_indication	
aci_gatt_srv_notify	
aci_gatt_srv_multi_notify	

Modified command name (v4.x)	Description
aci_gatt_srv_resp	Added CID parameter to support both unenhanced and enhanced channels.
aci_gatt_clt_write	
aci_gatt_clt_write_long	
aci_gatt_clt_write_char_reliable	

4.2.4

Removed commands

Table 8. List of commands removed on Bluetooth® Low Energy stack v4.x

Removed command name (v4.x)	Reason
aci_gap_set_event_mask	Not required/used.
aci_gap_set_periodic_advertising_configuration	These GAP commands to handle periodic advertising were simply wrappers of corresponding and standard HCI commands on Bluetooth® Low Energy stack v3.x.
aci_gap_set_periodic_advertising_enable	
aci_gap_periodic_advertising_create_sync	
aci_gap_periodic_advertising_create_sync_cancel	
aci_gap_periodic_advertising_terminate_sync	
aci_gap_add_device_to_periodic_advertiser_list	
aci_gap_remove_device_from_periodic_advertising_list	
aci_gap_clear_periodic_advertiser_list	
aci_gap_read_periodic_advertiser_list_size	
aci_gap_set_periodic_advertising_receive_enable	
aci_gap_periodic_advertising_sync_transfer	
aci_gap_periodic_advertising_set_info_transfer	
aci_gap_set_periodic_advertising_sync_transfer_parameters	
aci_gap_set_default_periodic_advertising_sync_transfer_parameters	
aci_gap_set_periodic_advertising_data	
aci_gatt_srv_init	Moved to GATT profile managed externally to the Bluetooth® Low Energy stack v4.x.
aci_gatt_eatt_srv_init	Specific GATT commands for EATT channels are replaced by corresponding GATT commands that have been modified by adding the CID parameter.
aci_gatt_eatt_clt_prepare_write_req	
aci_gatt_eatt_clt_execute_write_req	
aci_gatt_eatt_clt_disc_all_primary_services	
aci_gatt_eatt_clt_disc_primary_service_by_uuid	
aci_gatt_eatt_clt_find_included_services	
aci_gatt_eatt_clt_disc_all_char_of_service	
aci_gatt_eatt_clt_disc_char_by_uuid	
aci_gatt_eatt_clt_disc_all_char_desc	
aci_gatt_eatt_clt_read	
aci_gatt_eatt_clt_read_using_char_uuid	
aci_gatt_eatt_clt_read_long	
aci_gatt_eatt_clt_read_multiple_char_value	
aci_gatt_eatt_clt_write_without_resp	

Removed command name (v4.x)	Reason
aci_gatt_eatt_clt_confirm_indication	Specific GATT commands for EATT channels are replaced by corresponding GATT commands that have been modified by adding the CID parameter.
aci_gatt_eatt_srv_notify	
aci_gatt_eatt_srv_multi_notify	
aci_gatt_eatt_clt_read_multiple_var_len_char_value	
aci_gatt_eatt_srv_resp	
aci_gatt_eatt_clt_write	
aci_gatt_eatt_clt_write_long	
aci_gatt_eatt_clt_write_char_reliable	
aci_gap_allow_rebond	Discontinued

4.3 Bluetooth® Low Energy stack v4.x events

4.3.1 New events

The following new events in [Table 9](#) have been defined as required by the new periodic advertising with responses feature.

Table 9. List of HCI events related to PAWR feature added by Bluetooth® Low Energy stack v4.x

Event name (v4.x)	Short description
hci_le_periodic_advertising_subevent_data_request_event	See <i>Bluetooth Spec v.5.4, Vol. 4, Part E, HCI, Sec. 7.7</i>
hci_le_periodic_advertising_response_report_event	
hci_le_periodic_advertising_sync_established_v2_event	
hci_le_periodic_advertising_report_v2_event	
hci_le_periodic_advertising_sync_transfer_received_v2_event	
hci_le_periodic_advertising_subevent_data_request_event	
hci_le_periodic_advertising_response_report_event	
hci_le_enhanced_connection_complete_v2_event	
aci_hal_pawr_data_free_event	A new event used to notify the application when the stack has released the previously provided periodic advertising with responses subevent data pointer or response data pointer.
aci_gap_pairing_event	This new event is generated when a pairing process is started.

4.3.2 Modified events

Table 10. List of events modified from Bluetooth® Low Energy stack v4.x

Modified event name (v4.x)	Details
aci_gatt_srv_attribute_modified_event	Added CID parameter to support both unenhanced and enhanced channels.
aci_gatt_clt_notification_event	
aci_gatt_srv_read_event	
aci_gatt_srv_write_event	
aci_att_srv_prepare_write_req_event	

4.3.3 Removed events

Table 11. List of events removed from Bluetooth® Low Energy stack v4.x

Removed event name (v4.x)	Reason
aci_gap_peripheral_security_initated_event	Discontinued event
aci_gap_bond_lost_event	Replaced by new event aci_gap_pairing_event

5 Bluetooth® Low Energy stack v4.x vs v3.x security framework

The following section describes the key security changes available within the Bluetooth® Low Energy stack v4.x in terms of commands and events when compared with Bluetooth® Low Energy stack v3.x.

5.1 aci_gap_set_security_requirements()

It replaces the aci_gap_set_authentication_requirement() available on Bluetooth® Low Energy stack v3.x.

```
aci_gap_set_security_requirements (
    Bonding_Mode,
    MITM_Mode,
    SC_Support,
    KeyPress_Notification_Support,
    Min_Encryption_Key_Size,
    Max_Encryption_Key_Size,
    Pairing_Response
)
```

Description

This command shall be given to set security requirements at device level.

These security requirements are used only when responding to an incoming peripheral security request or pairing request received from a peer device.

If the command is given during pairing, the command returns BLE_STATUS_NOT_ALLOWED.

Removed parameters

Use_Fixed_Pin
Fixed_Pin

New parameters

Pairing_Response:

- If set to 0 (pairing response not required)
Pairing is always accepted even for bonded devices, and no user interaction is required, since the response is automatically handled by the stack library.
- If set to 1 (pairing response required for bonded devices only)
The pairing is always automatically accepted (no user interaction) except when the request comes from an already bonded device; in this case aci_gap_pairing_event is always notified, and the explicit pairing response is required (a sort of confirmation of rebonding; this mode is a replacement of the former bond lost event requiring explicit allow rebond command from the user).
- If set to 2 (explicit pairing response)
A pairing response is always required since aci_gap_pairing_event is always notified to the end-user when a pairing request from a Central or a peripheral security request from a peripheral is received.

Note: MITM mode parameter indicates the requirement to achieve MITM protection (that is, an authenticated security level); it is used when responding to an incoming peripheral security request or pairing request received from a peer device.

However, this parameter is not used when actively sending a pairing request or peripheral security request, since in these cases the MITM_mode bit declared into the authentication requirement field of the two aforementioned PDUs is derived from the security level parameter of the aci_gap_set_security.

This has an implication on how MITM is handled (that is “mandated” referring to the known MITM_Mandates requirement) at device level, because the MITM mandates is handled by referring to the requirements set into the security request and not to this parameter set at device level.

Returned values

- BLE_STATUS_SUCCESS - security requirements set without any error.
- BLE_STATUS_INVALID_PARAMS - some of the parameters are out of admitted range, according to BT specification, or custom API specification (Bonding and MITM values may be only 0 or 1, as well as KeyPress_Notification_Support; encryption key size value shall be in the range [0x07, 0x10], whereas the vendor specific pairing response mode may be equal to 0 or 1 or 2).
- BLE_ERROR_UNSUPPORTED_FEATURE - when KeyPress_Notification_Support is set to 1 but a secure connection feature is not supported (that is, excluded through modularity options).

5.2 aci_gap_set_security()

It replaces the aci_gap_peripheral_security_request() available on Bluetooth® Low Energy stack v3.x.

```
aci_gap_set_security(
    Connection_Handle,
    Security_Level,
    Force_Pairing
)
```

Description

A new unique command that shall be used to start a pairing procedure starting from BLEPS v4.0 and valid for both security roles Central-Initiator (CEI) and Peripheral-Responder (PER).

When submitted, and following the preliminary consistency checks, the following actions occur with respect to the peer device identified through the connection handle parameter:

- The device acting in the CEI role sends a pairing request.
- The device acting as the PER sends a peripheral security request.

Note:

- *This command replaces the following two commands existing in prior Bluetooth® Low Energy stack v.3.x:*
 - aci_gap_peripheral_security_request;
 - aci_gap_send_pairing_req.
- *This new command also brings a new parameter that does not exist in the two replaced commands, and that represents the minimum security level to be achieved.*
- *When submitted to a device acting as CEI, this command just enables encryption on the link if the peer is bonded with at least the specified security level, otherwise, it starts a pairing with the peer device.*

Parameters

Security_Level: indicates the minimum security level to be achieved.

- No security (no authentication and no encryption)
- Unauthenticated pairing with encryption
- Authenticated pairing with encryption
- Authenticated LE secure connections pairing with encryption using a 128-bit strength encryption key.

Force_Pairing (valid in case of Central device only).

- The bit b0 indicates whether a pairing request has to be sent (forced) in case the peer device was previously paired/bonded.
- The bit b1 indicates whether the link has to be reencrypted after the key exchange.

Returned values

- BLE_STATUS_SUCCESS - the security procedure (pairing) successfully initiated or the requested security level is already in place with the peer device (for example, requesting security Level 3 when the connection is already secured at level 3 because of previous pairing).
- BLE_STATUS_UNKNOWN_CONNECTION_ID - the connection handle specified is unknown or not valid.
- BLE_STATUS_INVALID_PARAMS when requested Security_Level parameter is invalid (<1 or >4) or Force_Pairing parameter value is invalid (>3).

- BLE_STATUS_DEV_IN_BLACKLIST - peer device temporarily locked in rejectlist because of previous pairing failure.
- BLE_STATUS_NOT_ALLOWED - current device status does not permit to start the pairing security procedure (for example, another pairing is pending with the same peer device, a peripheral security request is already pending with the same peer device, an encryption key refresh procedure is ongoing with the same peer device, or the initiator device is already serving a peripheral security request is from the same peer device).
- BLE_ERROR_CONTROLLER_BUSY - the PKA resource is temporarily busy with another pairing procedure.
- BLE_STATUS_SECURITY_REQUIREMENTS_NOT_ACHIEVABLE - current I/O capability settings do not enable to achieve the requested security level. For example, when requesting security Level 3 or 4 with just NoInput-NoOutput capabilities, or Level 4 with no SC support or encryption key size <16 octets.
- BLE_ERROR_HOST_BUSY_PAIRING - a maximum number of simultaneous pairing procedures supported has been achieved or SC pairing is requested when another SC pairing procedure is still ongoing.
- BLE_STATUS_INSUFFICIENT_RESOURCES - no free Tx packets in the SM Tx pool list to send the request.
- BLE_STATUS_FAILED - in case of failure when sending out the packet.

5.3 aci_gap_pairing_resp()

It is a new command available on Bluetooth® Low Energy stack v4.x.

```
aci_gap_pairing_resp(
    Connection_Handle,
    Accept
)
```

Description

This command shall be given in response to an aci_gap_paring_event, to allow or reject either the pairing request from the Central or the security request from the peripheral.

Parameters

Accept:

- 0 to REJECT;
- 1 to ACCEPT.

Effects

Effects when this command is submitted with Accept = 0 (that is, when pairing is rejected):

- On both Central and peripheral devices:
 - Pairing failed message is sent to the peer device with reason code = 0x05 pairing not supported.

Effects when this command is submitted with Accept = 1 (that is, when pairing is accepted):

- On a Central device: it triggers the sending of a PAIRING REQUEST PDU.
- On a peripheral device: it triggers the sending PAIRING RESPONSE PDU.

Return

- BLE_STATUS_SUCCESS - in case of successful response sent.
- BLE_STATUS_UNKNOWN_CONNECTION_ID - the connection handle specified is unknown or not valid.
- BLE_STATUS_INVALID_PARAMS - when Accept parameter value is invalid (>1).

5.4 aci_gap_pairing_event()

It is a new command available on Bluetooth® Low Energy stack v4.x.

```
aci_gap_pairing_event (
    Connection_Handle,
    Bonded
)
```

Description

This event is generated when a pairing process is started, and specifically when:

- CEI receives a security request from a PER, or
- PER receives a pairing request from the CEI.

The application shall respond with `aci_gap_pairing_resp` command to accept or reject the incoming pairing procedure notified through this event.

If the pairing is going to start with a nonbonded device, the bonded parameter is cleared. Instead, if the pairing process is going to start with an already bonded device, the event is raised with the bonded parameter set. This may happen either if the peer has lost the bond or if it is a malicious device.

If the `aci_gap_set_security_requirements` command was given with `Pairing_Response` set (pairing confirmation only for bonded devices), the event is raised only if pairing is going to be initiated with a bonded device.

Parameters

Bonded:

- 0: indicates that the peer device is not already bonded;
- 1: indicated that the peer device is already bonded.

5.5

`aci_gap_passkey_req_event()`

This is a modified event compared to the equivalent event on Bluetooth® Low Energy stack v3.x. A new parameter has been added.

```
aci_gap_passkey_req_event (
    Connection_Handle,
    Display_Input
)
```

New parameters

Display_Input:

- 0: A passkey shall be randomly generated and displayed on the device; then it must be provided to the stack through `ACI_GAP_PASSKEY_RESP`.
- 1: A passkey shall be asked of the user, who normally provides it with a keyboard, and provided to the stack through `ACI_GAP_PASSKEY_RESP`.

Appendix A STSW-BNRGLP-DK vs STM32CubeWB0 software packages

Bluetooth® Low Energy stack v3.x is supported within the STSW-BNRGLP-DK targeting the BlueNRG-LP, BlueNRG-LPS devices while the Bluetooth® Low Energy stack v4.x is supported within the STM32CubeWB0 software packages targeting the STM32WB09xE, STM32WB07xC and STM32WB05xZ devices.

The STM32WB07xC and STM32WB05xZ devices are respectively the rebranded BlueNRG-LP and BlueNRG-LPS devices within the STM32 brand.

The following table provides a comparison about the main characteristics of the STSW-BNRGLP-DK and STM32CubeWB0 software packages.

Table 12. STSW-BNRGLP-DK and STM32CubeWB0 software packages

What	STSW-BNRGLP-DK	STM32CubeWB0
Supported devices	BlueNRG-LP and BlueNRG-LPS	STM32WB09xE, STM32WB05xZ (BlueNRG-LPS), STM32WB07xC (BlueNRG-LP)
Software package structure	Application, Docs, Drivers, Firmware, Middleware, Projects, Utilities	Same as standard STM32Cube software packages: Documentation, Drivers, Firmware, Middleware, Projects, Utilities
LL, HAL drivers	LL & HAL drivers but with different naming convention and prefix (rf_driver_). 2.4 GHz LL/HAL drivers are included.	LL & HAL drivers with the same structure and naming convention as standard STM32Cube software packages. 2.4 GHz HAL radio driver is included inline with the STM32Cube standard framework
CMSIS	Standard CMSIS files	Standard CMSIS files
BSP	Folders and files for STEVAL-IDB011V1 (BlueNRG-LP) and STEVAL-IDB012V1 (BlueNRG-LPS) kits and resources (pressure, accelerometers)	BSP structure as standard STM32Cube software packages targeting NUCLEO-WB09KE, NUCLEO-WB05KZ, and NUCLEO-WB07CC kits.
Middleware	Bluetooth® Low Energy stack v3.2a framework. Centralized drivers for AES, NVM, PKA, RNG, and Bluetooth® Low Energy controller needed by Bluetooth® Low Energy stack	STM32_BLE Bluetooth® Low Energy middleware: Bluetooth® Low Energy stack v4.0 with new events manager and new Bluetooth® Low Energy features (commands and APIs). AES, NVM, PKA, RNG, and Bluetooth® Low Energy controller drivers are moved to the System folder on each STM32_BLE application.
Bluetooth® Low Energy applications	Simple project structure with main.c, rf_device_it.c (irq handler), and specific application specific files: one for GATT database (services and characteristics), and one for the application specific functionality (state machine, Bluetooth® Low Energy commands and events handling, etc.)	STM32_BLE Bluetooth® Low Energy application structure inline with STM32CubeWBA with System folder containing the drivers for AES, NVM, PKA, RNG, and Bluetooth® Low Energy controller needed by Bluetooth® Low Energy stack.

If a user needs to port an application built within the STSW-BNRGLP-DK software package (BlueNRG-LP/BlueNRG-LPS software package) to STM32CubeWB0, the following steps are suggested:

1. Identify the Bluetooth® Low Energy application scenario inline with the application to be ported (only broadcaster, peripheral & GATT server, Central & GATT client, peripheral, Central & GATT client server).

2. Open the STM32CubeMX tool, select the STM32WB0 series device and design the STM32_BLE application by following the steps described in the document [1]. The user must replicate the same application configurations available on the STSW-BNRGLP-DK Bluetooth® Low Energy application to be ported. The goal is to generate a project software framework providing:
 - a. System initialization and NVIC configuration
 - b. IP drivers initialization (RCC, GPIOs, PKA, RNG, RADIO, RADIO_TIMER)
 - c. Bluetooth® Low Energy application mode according to the user scenario (only broadcaster, peripheral & GATT server, Central & GATT client, peripheral, Central & GATT client server)
 - d. Bluetooth® Low Energy stack modular configurations
 - e. Bluetooth® Low Energy stack initialization parameters
 - f. Bluetooth® Low Energy application parameters (Tx power, ...)
 - g. Bluetooth® Low Energy application advertising parameters if applicable
 - h. Bluetooth® Low Energy service and characteristics definition if applicable
3. Once the STM32_BLE user application software framework has been defined, the user can generate the related project by selecting the preferred toolchain (EWARM, Keil®, STM32CubeIDE).
4. As a last step, the user must open the generated project and add the specific application specific parts from the original STSW-BNRGLP-DK Bluetooth® Low Energy application. It is recommended to add code within the USER defined sections supported from STM32CubeMX whenever it is possible (to keep consistency with CubeMX tool and ioc file).
5. Notice that STM32CubeMX generates STM32_BLE applications with the sequencer utility framework to use the sequencer. As a consequence, the generated code is tailored for using this framework and the user must follow some guidelines to integrate his application on it. Information and guidelines are provided on the online wiki sequencer page [2]. Referenced STM32_BLE applications on STM32CubeWB0 are using the sequencer.
6. If the user prefers a simplified approach with no sequencer, it is possible to remove the references to the sequencer utility and apply a simple while(1) approach with the user state machine without any task. An application example is provided on the STM32CubeWB0 software package (BLE_Peripheral_Lite).

Revision history

Table 13. Document revision history

Date	Version	Changes
24-Jun-2024	1	Initial release.
06-Nov-2024	2	Updated Table 5 . List of new vendor-specific commands introduced by Bluetooth® Low Energy stack v4.x

Contents

1	General information	2
2	Reference documents	3
3	Modular configurations and initialization of Bluetooth® Low Energy stack v4.x vs v3.x	4
3.1	Bluetooth® Low Energy stack modular configuration options	4
3.2	Bluetooth® Low Energy stack initialization parameters	5
4	Bluetooth® Low Energy stack v4.x command and event modifications	7
4.1	New standard controller error codes	7
4.2	Bluetooth® Low Energy stack v4.x commands	7
4.2.1	New periodic advertising with response commands	7
4.2.2	Bluetooth® Low Energy stack v4.x renamed commands	8
4.2.3	Modified commands	8
4.2.4	Removed commands	9
4.3	Bluetooth® Low Energy stack v4.x events	10
4.3.1	New events	10
4.3.2	Modified events	10
4.3.3	Removed events	11
5	Bluetooth® Low Energy stack v4.x vs v3.x security framework	12
5.1	aci_gap_set_security_requirements()	12
5.2	aci_gap_set_security()	13
5.3	aci_gap_pairing_resp()	14
5.4	aci_gap_pairing_event()	14
5.5	aci_gap_passkey_req_event()	15
Appendix A	STSW-BNRGLP-DK vs STM32CubeWB0 software packages	16
	Revision history	18

List of tables

Table 1.	Bluetooth® Low Energy stack modular configurations.	4
Table 2.	Bluetooth® Low Energy stack initialization parameters	5
Table 3.	Error codes	7
Table 4.	New commands related to PAwR added by Bluetooth® Low Energy stack v4.x	7
Table 5.	List of new vendor-specific commands introduced by Bluetooth® Low Energy stack v4.x	7
Table 6.	List of commands renamed on Bluetooth® Low Energy stack v4.x vs v3.x	8
Table 7.	List of commands modified on Bluetooth® Low Energy stack v4.x	8
Table 8.	List of commands removed on Bluetooth® Low Energy stack v4.x	9
Table 9.	List of HCI events related to PAwR feature added by Bluetooth® Low Energy stack v4.x	10
Table 10.	List of events modified from Bluetooth® Low Energy stack v4.x	10
Table 11.	List of events removed from Bluetooth® Low Energy stack v4.x	11
Table 12.	STSW-BNRGLP-DK and STM32CubeWB0 software packages	16
Table 13.	Document revision history	18

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.

© 2024 STMicroelectronics – All rights reserved