AN6142

Application note

Introduction to the STM32WB0 Bluetooth® LE wireless interface

Introduction

Bluetooth® LE is a wireless personal area network technology designed and marketed by the Bluetooth® special interest group (Bluetooth SIG), aimed at novel applications in the healthcare, fitness, beacons, security, and home entertainment industries.

Bluetooth® LE considerably reduces power consumption and cost compared to standard Bluetooth®, while maintaining a similar communication range.

Standard Host-Controller Interface (HCI) commands are defined in the Bluetooth® specification core, of which the Bluetooth® LE specification is a part.

This application note describes all commands and events provided by the Bluetooth® LE stack v4.x family supported by the STM32WB0 series devices. The Bluetooth® LE stack v4.x is available with the STM32CubeWB0 MCU Package.

This document describes the supported commands and events on both System on Chip (SoC) application context and on Network coprocessor application context.

Note: The availability of effective commands and events is linked to the Bluetooth® LE stack modular configuration options that are enabled on each specific application. For more details, refer to Section 6: Modular configuration options and supported APIs.

1 General information

This document applies to the Arm® Cortex® core-based STM32WB0 microcontrollers.

For more information on Bluetooth®, refer to http://www.bluetooth.com.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 ACI/HCI commands

2.1 HCI commands

This section describes the standard HCI commands.

Table 1. HCI commands opcodes

SoC command	Network coprocessor command	OpCode
hci_disconnect	hci_disconnect	0x0406
hci_read_remote_version_information	hci_read_remote_version_information	0x041D
hci_set_event_mask	hci_set_event_mask	0x0C01
-	hci_reset	0x0C03
hci_read_connection_accept_timeout	hci_read_connection_accept_timeout	0x0C15
hci_write_connection_accept_timeout	hci_write_connection_accept_timeout	0x0C16
hci_read_transmit_power_level	hci_read_transmit_power_level	0x0C2D
hci_read_afh_channel_assessment_mode	hci_read_afh_channel_assessment_mode	0x0C48
hci_write_afh_channel_assessment_mode	hci_write_afh_channel_assessment_mode	0x0C49
hci_set_event_mask_page_2	hci_set_event_mask_page_2	0x0C63
hci_read_authenticated_payload_timeout	hci_read_authenticated_payload_timeout	0x0C7B
hci_write_authenticated_payload_timeout	hci_write_authenticated_payload_timeout	0x0C7C
hci_read_local_version_information	hci_read_local_version_information	0x1001
hci_read_local_supported_commands	hci_read_local_supported_commands	0x1002
hci_read_local_supported_features	hci_read_local_supported_features	0x1003
hci_read_bd_addr	hci_read_bd_addr	0x1009
hci_read_rssi	hci_read_rssi	0x1405
hci_le_set_event_mask	hci_le_set_event_mask	0x2001
hci_le_read_buffer_size	hci_le_read_buffer_size	0x2002
hci_le_read_local_supported_features	hci_le_read_local_supported_features	0x2003
hci_le_set_random_address	hci_le_set_random_address	0x2005
hci_le_set_advertising_parameters	hci_le_set_advertising_parameters	0x2006
hci_le_read_advertising_physical_channel_tx_power	hci_le_read_advertising_physical_channel_tx_power	0x2007
-	hci_le_set_advertising_data	0x2008
-	hci_le_set_scan_response_data	0x2009
hci_le_set_advertising_enable	hci_le_set_advertising_enable	0x200A
hci_le_set_scan_parameters	hci_le_set_scan_parameters	0x200B
hci_le_set_scan_enable	hci_le_set_scan_enable	0x200C
hci_le_create_connection	hci_le_create_connection	0x200D
hci_le_create_connection_cancel	hci_le_create_connection_cancel	0x200E
hci_le_read_filter_accept_list_size	hci_le_read_filter_accept_list_size	0x200F
hci_le_clear_filter_accept_list	hci_le_clear_filter_accept_list	0x2010
hci_le_add_device_to_filter_accept_list	hci_le_add_device_to_filter_accept_list	0x2011
hci_le_remove_device_from_filter_accept_list	hci_le_remove_device_from_filter_accept_list	0x2012
hci_le_connection_update	hci_le_connection_update	0x2013

SoC command	Network coprocessor command	OpCode
hci_le_set_host_channel_classification	hci_le_set_host_channel_classification	0x2014
hci_le_read_channel_map	hci_le_read_channel_map	0x2015
hci_le_read_remote_features	hci_le_read_remote_features	0x2016
hci_le_encrypt	hci_le_encrypt	0x2017
hci_le_rand	hci_le_rand	0x2018
hci_le_enable_encryption	hci_le_enable_encryption	0x2019
hci_le_long_term_key_request_reply	hci_le_long_term_key_request_reply	0x201A
hci_le_long_term_key_request_negative_reply	hci_le_long_term_key_request_negative_reply	0x201B
hci_le_read_supported_states	hci_le_read_supported_states	0x201C
hci_le_set_data_length	hci_le_set_data_length	0x2022
hci_le_read_suggested_default_data_length	hci_le_read_suggested_default_data_length	0x2023
hci_le_write_suggested_default_data_length	hci_le_write_suggested_default_data_length	0x2024
hci_le_read_local_p256_public_key	hci_le_read_local_p256_public_key	0x2025
hci_le_generate_dhkey	hci_le_generate_dhkey	0x2026
hci_le_add_device_to_resolving_list	hci_le_add_device_to_resolving_list	0x2027
hci_le_remove_device_from_resolving_list	hci_le_remove_device_from_resolving_list	0x2028
hci_le_clear_resolving_list	hci_le_clear_resolving_list	0x2029
hci_le_read_resolving_list_size	hci_le_read_resolving_list_size	0x202A
hci_le_read_peer_resolvable_address	hci_le_read_peer_resolvable_address	0x202B
hci_le_read_local_resolvable_address	hci_le_read_local_resolvable_address	0x202C
hci_le_set_address_resolution_enable	hci_le_set_address_resolution_enable	0x202D
hci_le_set_resolvable_private_address_timeout	hci_le_set_resolvable_private_address_timeout	0x202E
hci_le_read_maximum_data_length	hci_le_read_maximum_data_length	0x202F
hci_le_read_phy	hci_le_read_phy	0x2030
hci_le_set_default_phy	hci_le_set_default_phy	0x2031
hci_le_set_phy	hci_le_set_phy	0x2032
hci_le_set_advertising_set_random_address	hci_le_set_advertising_set_random_address	0x2035
hci_le_set_extended_advertising_parameters	hci_le_set_extended_advertising_parameters	0x2036
-	hci_le_set_extended_advertising_data	0x2037
-	hci_le_set_extended_scan_response_data	0x2038
hci_le_set_extended_advertising_enable	hci_le_set_extended_advertising_enable	0x2039
-	hci_le_read_maximum_advertising_data_length	0x203A
hci_le_read_number_of_supported_advertising_sets	hci_le_read_number_of_supported_advertising_sets	0x203B
hci_le_remove_advertising_set	hci_le_remove_advertising_set	0x203C
hci_le_clear_advertising_sets	hci_le_clear_advertising_sets	0x203D
hci_le_set_periodic_advertising_parameters	hci_le_set_periodic_advertising_parameters	0x203E
-	hci_le_set_periodic_advertising_data	0x203F
hci_le_set_periodic_advertising_enable	hci_le_set_periodic_advertising_enable	0x2040
hci_le_set_extended_scan_parameters	hci_le_set_extended_scan_parameters	0x2041
hci_le_set_extended_scan_enable	hci_le_set_extended_scan_enable	0x2042
hci_le_extended_create_connection	hci_le_extended_create_connection	0x2043

SoC command	Network coprocessor command	OpCode
hci_le_periodic_advertising_create_sync	hci_le_periodic_advertising_create_sync	0x2044
hci_le_periodic_advertising_create_sync_cancel	hci_le_periodic_advertising_create_sync_cancel	0x2045
hci_le_periodic_advertising_terminate_sync	hci_le_periodic_advertising_terminate_sync	0x2046
hci_le_add_device_to_periodic_advertiser_list	hci_le_add_device_to_periodic_advertiser_list	0x2047
hci_le_remove_device_from_periodic_advertiser_list	hci_le_remove_device_from_periodic_advertiser_list	0x2048
hci_le_clear_periodic_advertiser_list	hci_le_clear_periodic_advertiser_list	0x2049
hci_le_read_periodic_advertiser_list_size	hci_le_read_periodic_advertiser_list_size	0x204A
hci_le_read_transmit_power	hci_le_read_transmit_power	0x204B
hci_le_read_rf_path_compensation	hci_le_read_rf_path_compensation	0x204C
hci_le_write_rf_path_compensation	hci_le_write_rf_path_compensation	0x204D
hci_le_set_privacy_mode	hci_le_set_privacy_mode	0x204E
hci_le_set_connectionless_cte_transmit_parameters	hci_le_set_connectionless_cte_transmit_parameters	0x2051
hci_le_set_connectionless_cte_transmit_enable	hci_le_set_connectionless_cte_transmit_enable	0x2052
hci_le_set_connectionless_iq_sampling_enable	hci_le_set_connectionless_iq_sampling_enable	0x2053
hci_le_set_connection_cte_receive_parameters	hci_le_set_connection_cte_receive_parameters	0x2054
hci_le_set_connection_cte_transmit_parameters	hci_le_set_connection_cte_transmit_parameters	0x2055
hci_le_connection_cte_request_enable	hci_le_connection_cte_request_enable	0x2056
hci_le_connection_cte_response_enable	hci_le_connection_cte_response_enable	0x2057
hci_le_read_antenna_information	hci_le_read_antenna_information	0x2058
hci_le_set_periodic_advertising_receive_enable	hci_le_set_periodic_advertising_receive_enable	0x2059
hci_le_periodic_advertising_sync_transfer	hci_le_periodic_advertising_sync_transfer	0x205A
hci_le_periodic_advertising_set_info_transfer	hci_le_periodic_advertising_set_info_transfer	0x205B
hci_le_set_periodic_advertising_sync_transfer_param eters	hci_le_set_periodic_advertising_sync_transfer_param eters	0x205C
hci_le_set_default_periodic_advertising_sync_transfe r_parameters	hci_le_set_default_periodic_advertising_sync_transfe r_parameters	0x205D
hci_le_read_buffer_size_v2	hci_le_read_buffer_size_v2	0x2060
hci_le_read_iso_tx_sync	hci_le_read_iso_tx_sync	0x2061
hci_le_set_cig_parameters	hci_le_set_cig_parameters	0x2062
hci_le_create_cis	hci_le_create_cis	0x2064
hci_le_remove_cig	hci_le_remove_cig	0x2065
hci_le_accept_cis_request	hci_le_accept_cis_request	0x2066
hci_le_reject_cis_request	hci_le_reject_cis_request	0x2067
hci_le_create_big	hci_le_create_big	0x2068
hci_le_terminate_big	hci_le_terminate_big	0x206A
hci_le_big_create_sync	hci_le_big_create_sync	0x206B
hci_le_big_terminate_sync	hci_le_big_terminate_sync	0x206C
hci_le_request_peer_sca	hci_le_request_peer_sca	0x206D
hci_le_setup_iso_data_path	hci_le_setup_iso_data_path	0x206E
hci_le_remove_iso_data_path	hci_le_remove_iso_data_path	0x206F
hci_le_set_host_feature	hci_le_set_host_feature	0x2074
hci_le_read_iso_link_quality	hci_le_read_iso_link_quality	0x2075

SoC command	Network coprocessor command	OpCode
hci_le_enhanced_read_transmit_power_level	hci_le_enhanced_read_transmit_power_level	0x2076
hci_le_read_remote_transmit_power_level	hci_le_read_remote_transmit_power_level	0x2077
hci_le_set_path_loss_reporting_parameters	hci_le_set_path_loss_reporting_parameters	0x2078
hci_le_set_path_loss_reporting_enable	hci_le_set_path_loss_reporting_enable	0x2079
hci_le_set_transmit_power_reporting_enable	hci_le_set_transmit_power_reporting_enable	0x207A
hci_le_set_data_related_address_changes	hci_le_set_data_related_address_changes	0x207C
hci_le_set_default_subrate	hci_le_set_default_subrate	0x207D
hci_le_subrate_request	hci_le_subrate_request	0x207E
hci_le_set_extended_advertising_parameters_v2	hci_le_set_extended_advertising_parameters_v2	0x207F
hci_le_set_periodic_advertising_subevent_data	hci_le_set_periodic_advertising_subevent_data	0x2082
hci_le_set_periodic_advertising_response_data	hci_le_set_periodic_advertising_response_data	0x2083
hci_le_set_periodic_sync_subevent	hci_le_set_periodic_sync_subevent	0x2084
hci_le_extended_create_connection_v2	hci_le_extended_create_connection_v2	0x2085
hci_le_set_periodic_advertising_parameters_v2	hci_le_set_periodic_advertising_parameters_v2	0x2086

2.1.1 hci_disconnect

```
tBleStatus hci_disconnect    ( uint16_t Connection_Handle,
                               uint8_t Reason
                              )
```

The hci_disconnect is used to terminate an existing connection. The Connection_Handle command parameter indicates which connection is to be disconnected. The Reason command parameter indicates the reason for ending the connection. The remote Controller receives the Reason command parameter in the *hci_disconnection_complete_event_rp0* event. All synchronous connections on a physical link should be disconnected before the ACL connection on the same physical connection is disconnected. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.1.6) It is important to leave a 100-ms blank window before sending any new command (including system hardware reset), since immediately after *hci_disconnection_complete_event_rp0* event, the system could save important information in the nonvolatile memory.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Reason

The reason for ending the connection. Values:

- 0x05: Authentication Failure
- 0x13: Remote User Terminated Connection
- 0x14: Remote Device Terminated Connection due to Low Resources
- 0x15: Remote Device Terminated Connection due to Power Off
- 0x1A: Unsupported Remote Feature
- 0x3B: Unacceptable Connection Parameters

Return values:

- *Value* indicating success or error code.

2.1.2 hci_le_accept_cis_request

```
tBleStatus hci_le_accept_cis_request    ( uint16_t Connection_Handle,
                                         )
```

The HCI_LE_Accept_CIS_Request command is used by the peripheral host to inform the Controller to accept the request for the CIS that is identified by the Connection_Handle. The command shall only be issued after an HCI_LE_CIS_Request event has occurred. The event contains the Connection_Handle of the CIS. If the Peripheral's Host issues this command with a Connection_Handle that does not exist, or the Connection_Handle is not for a CIS, the Controller shall return the error code Unknown Connection Identifier (0x02). If the peripheral host issues this command with a Connection_Handle for a CIS that has already been established or that already has an HCI_LE_Accept_CIS_Request or HCI_LE_Reject_CIS_Request command in progress, the Controller shall return the error code Command Disallowed (0x0C). If the Central's Host issues this command, the Controller shall return the error code Command Disallowed (0x0C).

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Return values:

- *Value* indicating success or error code.

2.1.3 hci_le_add_device_to_filter_accept_list

```
tBleStatus hci_le_add_device_to_filter_accept_list    ( uint8_t Address_Type,
                                                        uint8_t Address[6],
                                                       )
```

The LE_Add_Device_To_Filter_Accept_List command is used to add a single device to the Filter Accept list stored in the Controller. This command can be used at any time except when:

- The advertising filter policy uses the Filter Accept list and advertising is enabled.
- The scanning filter policy uses the Filter Accept list and scanning is enabled.
- The initiator filter policy uses the Filter Accept list and a create connection command is outstanding.

Parameters

Address_Type

Address_Type. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address

Address

Public Device Address or Random Device Address of the device to be added to the list.

Return values:

- *Value* indicating success or error code.

2.1.4 hci_le_add_device_to_periodic_advertiser_list

```
tBleStatus hci_le_add_device_to_periodic_advertiser_list    ( uint8_t Advertiser_Address_Type,
                                                              uint8_t Advertiser_Address[6],
                                                              uint8_t Advertising_SID
                                                             )
```

The LE_Add_Device_To_Periodic_Advertiser_List command is used to add a single device to the Periodic Advertiser list stored in the Controller. Any additions to the Periodic Advertiser list take effect immediately. If the device is already on the list, the Controller shall return the error code Invalid HCI Command Parameters (0x12). If the Host issues this command when an LE_Periodic_Advertising_Create_Sync command is pending, the Controller shall return the error code Command Disallowed (0x0C). When a Controller cannot add a device to the Periodic Advertiser list because the list is full, the Controller shall return the error code Memory Capacity Exceeded (0x07).

Parameters

Advertiser_Address_Type

Advertiser Address Type. Values:

- 0x00: Public Device Address or Public Identity Address
- 0x01: Random Device Address or Random (static) Identity Address

Advertiser_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the advertiser.

Advertising_SID

Advertising SID subfield in the ADI field used to identify the Periodic Advertising. Values:

- 0x00 ... 0x0F: Advertising SID subfield in the ADI field used to identify the Periodic Advertising

Return values:

- *Value* indicating success or error code.

2.1.5 hci_le_add_device_to_resolving_list

```
tBleStatus hci_le_add_device_to_resolving_list    ( uint8_t Peer_Identity_Address_Type,
                                                     uint8_t Peer_Identity_Address[6],
                                                     uint8_t Peer_IRK[16],
                                                     uint8_t Local_IRK[16]
                                                    )
```

The LE_Add_Device_To_Resolving_List command is used to add one device to the list of address translations used to resolve Resolvable Private Addresses in the Controller. This command cannot be used when address translation is enabled in the Controller and:

- Advertising is enabled
- Scanning is enabled
- Create connection command is outstanding.

This command can be used at any time when address translation is disabled in the Controller. When a Controller cannot add a device to the resolving list because the list is full, it shall respond with error code 0x07 (memory capacity exceeded). (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.38.)

Parameters

Peer_Identity_Address_Type

Identity address type. Values:

- 0x00: Public Identity Address
- 0x01: Random (static) Identity Address

Peer_Identity_Address

Public or Random (static) Identity address of the peer device.

Peer_IRK

IRK of the peer device.

Local_IRK

IRK of the local device.

Return values:

- *Value* indicating success or error code.

2.1.6 hci_le_big_create_sync

```
tBleStatus hci_le_big_create_sync     ( uint8_t BIG_Handle,
                                        uint16_t Sync_Handle,
                                        uint8_t Encryption,
                                        uint8_t Broadcast_Code[16],
                                        uint8_t MSE,
                                        uint16_t BIG_Sync_Timeout,
                                        uint8_t Num_BIS,
                                        uint8_t BIS[]
                                      )
```

The HCI_LE_BIG_Create_Sync command is used to synchronize to a BIG described in the periodic advertising train specified by the Sync_Handle parameter.

Parameters

BIG_Handle

Used to identify the BIG. Values:

- 0x00 ... 0xEF

Sync_Handle

Identifier of the periodic advertising train. Values:

- 0x0000 ... 0x00EF

Encryption

The encryption mode of the BIG. Values:

- 0x00: Unencrypted
- 0x01: Encrypted

Broadcast_Code

128-bit code used for deriving the session key for decrypting payloads of BISes in the BIG.

MSE

The MSE (Maximum Subevents) parameter is the maximum number of subevents that a Controller should use to receive data payloads in each interval for a BIS. The Host should set MSE to reduce the maximum continuous radio receiving time for a Synchronized Receiver with limited battery capacity. Values:

- 0x00: Any number of subevents
- 0x01 ... 0x1F

BIG_Sync_Timeout

The BIG_Sync_Timeout parameter specifies the maximum permitted time between successful receptions of BIS PDUs. If this time is exceeded, synchronization is lost. When the Controller establishes synchronization and if the BIG_Sync_Timeout set by the Host is less than 6 * ISO_Interval, the Controller shall set the timeout to 6 * ISO_Interval. Values:

- 0x000A (100 ms) ... 0x4000 (163840 ms)

Num_BIS

Total number of BISes to synchronize. Values:

- 0x01 ... 0x1F

BIS

List of indices of BISes.

Return values:

- *Value* indicating success or error code.

2.1.7 hci_le_big_terminate_sync

```
tBleStatus hci_le_big_terminate_sync ( uint8_t BIG_Handle )
```

The HCI_LE_BIG_Terminate_Sync command is used to stop synchronizing or cancel the process of synchronizing to the BIG identified by the BIG_Handle parameter. The command also terminates the reception of BISes in the BIG specified in the HCI_LE_BIG_Create_Sync command, destroys the associated connection handles of the BISes in the BIG, and removes the data paths for all BISes in the BIG.

Parameters

BIG_Handle

Used to identify the BIG. Values:

- 0x00 ... 0xEF

Return values:

- *Value* indicating success or error code.

2.1.8 hci_le_clear_advertising_sets

```
tBleStatus hci_le_clear_advertising_sets ( void )
```

The LE_Clear_Advertising_Sets command is used to remove all existing advertising sets from the Controller. If advertising is enabled on any advertising set, then the Controller shall return the error code Command Disallowed (0x0C). Note: All advertising sets are cleared on HCI reset.

Return values:

- *Value* indicating success or error code.

2.1.9 hci_le_clear_filter_accept_list

```
tBleStatus hci_le_clear_filter_accept_list ( void )
```

The LE_Clear_Filter_Accept_List command is used to clear the Filter Accept list stored in the Controller. This command can be used at any time except when:

- The advertising filter policy uses the Filter Accept list and advertising is enabled.
- The scanning filter policy uses the Filter Accept list and scanning is enabled.
- The initiator filter policy uses the Filter Accept list and an LE_Create_Connection command is outstanding.

Return values:

- *Value* indicating success or error code.

2.1.10 hci_le_clear_periodic_advertiser_list

```
tBleStatus hci_le_clear_periodic_advertiser_list ( void )
```

The LE_Clear_Periodic_Advertiser_List command is used to remove all devices from the list of Periodic Advertisers in the Controller. If this command is used when an LE_Periodic_Advertising_Create_Sync command is pending, the Controller shall return the error code Command Disallowed (0x0C).

Return values:

- *Value* indicating success or error code.

2.1.11 hci_le_clear_resolving_list

```
tBleStatus hci_le_clear_resolving_list ( void )
```

The LE_Clear_Resolving_List command is used to remove all devices from the list of address translations used to resolve Resolvable Private Addresses in the Controller. This command cannot be used when address translation is enabled in the Controller and:

- Advertising is enabled.
- Scanning is enabled.

- The Create connection command is outstanding.

This command can be used at any time when address translation is disabled in the Controller. (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.40.)

Return values:

- *Value* indicating success or error code.

2.1.12 hci_le_connection_cte_request_enable

```
tBleStatus hci_le_connection_cte_request_enable    ( uint16_t Connection_Handle,
                                                     uint8_t Enable,
                                                     uint16_t CTE_Request_Interval,
                                                     uint8_t Requested_CTE_Length,
                                                     uint8_t Requested_CTE_Type
                                                    )
```

The HCI_LE_Connection_CTE_Request_Enable command is used to request the Controller to start or stop initiating the Constant Tone Extension Request procedure on a connection identified by the Connection_Handle parameter. If the Host issues this command when the Controller is aware (for example, through a previous feature exchange) that the Link Layer of the peer device does not support the Connection CTE Response feature, the Controller shall return the error code Unsupported Remote Feature / Unsupported LMP Feature (0x1A). If the Host issues this command when the Controller is aware that the peer device's Link Layer does not support the requested CTE type, the Controller should return the error code Unsupported Remote Feature / Unsupported LMP Feature (0x1A). If Enable is set to 0x00, the remaining parameters shall be ignored.

The CTE_Request_Interval parameter defines whether the Constant Tone Extension Request procedure is initiated only once or periodically. In the case of periodic operation, the procedure is initiated every CTE_Request_Interval. However, the Controller may delay initiating the procedure beyond the requested interval (for example, in order to prioritize other activities). The Requested_CTE_Length parameter indicates the minimum length of the Constant Tone Extension and the Requested_CTE_Type parameter indicates the type of Constant Tone Extension that the Controller shall request from the remote device.

A request is active on a connection from when the Host issues a successful command with Enable set to 0x01 until the single procedure has been performed, the period specified by CTE_Request_Interval has ended, or a command with Enable set to 0x00 has succeeded, whichever happens first. If the Host issues this command with Enable set to 0x01 while a request is active for the specified connection, the Controller shall return the error code Command Disallowed (0x0C).

Note: *The failed command does not affect the behavior of the Link Layer in respect of the currently-active request.*

If the Host issues this command before issuing the HCI_LE_Set_Connection_CTE_Receive_Parameters command, at least once on the connection, the Controller shall return the error code Command Disallowed (0x0C). If the Host issues this command when the receiver PHY for the connection is not a PHY that allows Constant Tone Extensions, the Controller shall return the error code Command Disallowed (0x0C). If the Host sets CTE_Request_Interval to a non-zero value less than or equal to connPeripheralLatency, the Controller shall return the error code Command Disallowed (0x0C). If Enable is set to 0x01 and the receiver PHY for the connection changes to a PHY that does not allow Constant Tone Extensions, then the Controller shall automatically disable Constant Tone Extension requests as if the Host had issued this command with Enable set to 0x00.

Note: *If the PHY changes back to a PHY that allows Constant Tone Extensions, then the Controller does not automatically re-enable Constant Tone Extension requests.*

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Enable

If it is set to 0x00, the remaining parameters shall be ignored. Values:

- 0x00: Disable Constant Tone Extension Request for the connection (default)
- 0x01: Enable Constant Tone Extension Request for the connection

CTE_Request_Interval

Defines whether the Constant Tone Extension Request procedure is initiated only once or periodically. In the case of periodic operation, the procedure is initiated every CTE_Request_Interval. However, the Controller may delay initiating the procedure beyond the requested interval (e.g., in order to prioritize other activities). Values:

- 0x0000: Initiate the Constant Tone Extension Request procedure once, at the nearliest practical opportunity.
- 0x0001 ... 0xFFFF: Requested interval for initiating the Constant Tone Extension Request procedure in number of connection events.

Requested_CTE_Length

Indicates the minimum length of the Constant Tone Extension and the Requested_CTE_Type parameter indicates the type of Constant Tone Extension that the Controller shall request from the remote device. Values:

- 0x02 ... 0x14: Minimum length of the Constant Tone Extension being requested in 8 microseconds units

Requested_CTE_Type

Flags:

- 0x00: AoA Constant Tone Extension
- 0x01: AoD Constant Tone Extension with 1-microsecond slots
- 0x02: AoD Constant Tone Extension with 2-microsecond slots

Return values:

- *Value* indicating success or error code.

2.1.13 hci_le_connection_cte_response_enable

```
tBleStatus hci_le_connection_cte_response_enable   ( uint16_t Connection_Handle,
                                                     uint8_t Enable,
                                                    )
```

The HCI_LE_Connection_CTE_Response_Enable command is used to request the Controller to respond to LL_CTE_REQ PDUs with LL_CTE_RSP PDUs on the specified connection. If the Host issues this command before issuing the HCI_LE_Set_Connection_CTE_Transmit_Parameters command at least once on the connection, the Controller shall return the error code Command Disallowed (0x0C). If the Host issues this command when the transmitter PHY for the connection is not a PHY that allows Constant Tone Extensions, the Controller shall return the error code Command Disallowed (0x0C). If the transmitter PHY for the connection changes to a PHY that does not allow Constant Tone Extensions, then the Controller shall automatically disable Constant Tone Extension responses.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Enable

If it is set to 0x00, the remaining parameters shall be ignored. Values:

- 0x00: Disable Constant Tone Extension Request for the connection (default)
- 0x01: Enable Constant Tone Extension Request for the connection

Return values:

- *Value* indicating success or error code.

2.1.14 hci_le_connection_update

```
tBleStatus hci_le_connection_update    ( uint16_t Connection_Handle,
                                         uint16_t Connection_Interval_Min,
                                         uint16_t Connection_Interval_Max,
                                         uint16_t Max_Latency,
                                         uint16_t Supervision_Timeout,
                                         uint16_t Min_CE_Length,
                                         uint16_t Max_CE_Length
                                       )
```

The LE_Connection_Update command is used to change the Link Layer connection parameters of a connection. This command is supported only on central side. The Connection_Interval_Min and Connection_Interval_Max parameters are used to define the minimum and maximum allowed connection interval. The Connection_Interval_Min parameter shall not be greater than the Connection_Interval_Max parameter. The Max_Latency parameter shall define the maximum allowed connection latency. The Supervision_Timeout parameter shall define the link supervision timeout for the LE link. The Supervision_Timeout in milliseconds shall be larger than (1 + Max_Latency) * Connection_Interval_Max * 2, where Connection_Interval_Max is given in milliseconds. The Min_CE_Length and Max_CE_Length are information parameters providing the Controller with a hint about the expected minimum and maximum length of the connection events. The Min_CE_Length shall be less than or equal to the Max_CE_Length. The actual parameter values selected by the Link Layer may be different from the parameter values provided by the Host through this command. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.18.)

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Connection_Interval_Min

Minimum value for the connection event interval. This shall be less than or equal to Connection_Interval_Max. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Connection_Interval_Max

Maximum value for the connection event interval. This shall be greater than or equal to Connection_Interval_Min. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Max_Latency

Maximum Peripheral latency for the connection in number of connection events. Values:

- 0x0000 ... 0x01F3

Supervision_Timeout

Supervision timeout for the LE Link. It shall be a multiple of 10 ms and larger than (1 + connPeripheralLatency) * connInterval * 2. Time = N * 10 msec. Values:

- 0x000A (100 ms) ... 0x0C80 (32000 ms)

Min_CE_Length

The minimum length of connection event recommended for this LE connection. Time = N * 0.625 msec.

Max_CE_Length

The maximum length of connection event recommended for this LE connection. Time = N * 0.625 msec.

Return values:

- *Value* indicating success or error code.

2.1.15 hci_le_create_big

```
tBleStatus hci_le_create_big    ( uint8_t BIG_Handle,
                                  uint8_t Advertising_Handle,
                                  uint8_t Num_BIS,
                                  uint8_t SDU_Interval[3],
                                  uint16_t Max_SDU,
                                  uint16_t Max_Transport_Latency,
                                  uint8_t RTN,
                                  uint8_t PHY,
                                  uint8_t Packing,
                                  uint8_t Framing,
                                  uint8_t Encryption,
                                  uint8_t Broadcast_Code[16]
                                 )
```

The HCI_LE_Create_BIG command is used to create a BIG with one or more BISes (see [Vol 6] Part B, Section 4.4.6). All BISes in a BIG have the same value for all parameters. The BIG_Handle contains the identifier of the BIG. This parameter is allocated by the Host and used by the Controller and the Host to identify a BIG. The Advertising_Handle identifies the associated periodic advertising train of the BIG (see [Vol 6] Part B, Section 4.4.5.1). The Num_BIS parameter contains the total number of BISes in the BIG. The SDU_Interval parameter contains the time interval of the periodic SDUs. The Max_SDU parameter contains the maximum size of an SDU. The Max_Transport_Latency parameter is the maximum transport latency (in milliseconds) as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2. This includes pre-transmissions.

The RTN (Retransmission Number) parameter contains the number of times every PDU should be retransmitted, irrespective of which isochronous events the retransmissions occur in. This is a recommendation to the Controller which the Controller may ignore. The PHY parameter is a bit field that indicates the PHY used for transmission of PDUs of BISes in the BIG. The Host shall set at least one bit in this parameter and the Controller shall pick a PHY from the bits set. If the Host sets, in the PHY parameter, a bit for a PHY that the Controller does not support, including a bit that is reserved for future use, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). The Packing parameter is used to indicate the preferred method of arranging subevents of multiple BISes. The subevents can be arranged in Sequential or Interleaved arrangement. This is a recommendation to the Controller which it may ignore. This parameter shall be ignored when there is only one BIS in the BIG.

The Framing parameter indicates the format for sending BIS Data PDUs. If the Framing parameter is set to 1 then BIS Data PDUs shall be Framed and when set to 0 they may be unframed (see [Vol 6] Part G, Section 1). The Encryption parameter identifies the encryption mode of the BISes. If the Encryption parameter is set to 1 (encrypted), the Broadcast_Code is used in the encryption of payloads (see [Vol 6] Part B, Section 4.4.6.10). The Broadcast_Code parameter is used to generate the encryption key for encrypting payloads of all BISes. When the Encryption parameter is set to 0 (unencrypted), the Broadcast_Code parameter shall be set to zero by the Host and ignored by the Controller.

If the Controller cannot create all BISes of the BIG or if Num_BIS exceeds the maximum value supported by the Controller, it shall return the error code Connection Rejected due to Limited Resources (0x0D). If the Advertising_Handle does not identify a periodic advertising train or the periodic advertising train is associated with another BIG, the Controller shall return the error code Unknown Advertising Identifier (0x42). If the Host issues this command with a BIG_Handle for a BIG that is already created, the Controller shall return the error code Command Disallowed (0x0C). If the Host specifies an invalid combination of BIG parameters, the Controller shall return an error which should use the error code Unsupported Feature or Parameter Value (0x11).

Parameters

BIG_Handle

Used to identify the BIG. Values:

- 0x00 ... 0xEF

Advertising_Handle

Used to identify the periodic advertising train. Values:

- 0x00 ... 0xEF

Num_BIS

Total number of BISes in the BIG. Values:

- 0x01 ... 0x1F

SDU_Interval

The interval, in microseconds, of periodic SDUs. Values:

- 0x0000FF ... 0x0FFFFF

Max_SDU

Maximum size of an SDU, in octets. Values:

- 0x0001 ... 0x0FFF

Max_Transport_Latency

Maximum transport latency, in milliseconds. Values:

- 0x0005 ... 0x0FA0

RTN

The number of times that every BIS Data PDU should be retransmitted. Values:

- 0x00 ... 0x1E

PHY

Transmitter PHY of packets. Flags:

- 0x01: LE_1M_PHY_BIT
- 0x02: LE_2M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

Packing

Used to indicate the preferred method of arranging subevents of multiple BISes. Values:

- 0x00: Sequential
- 0x01: Interleaved

Framing

The format for sending BIS Data PDUs. Values:

- 0x00: Unframed
- 0x01: Framed

Encryption

The encryption mode of the BISes. Values:

- 0x00: Unencrypted
- 0x01: Encrypted

Broadcast_Code

128-bit code used for deriving the session key for decrypting payloads of BISes in the BIG.

Return values:

- *Value* indicating success or error code.

2.1.16 hci_le_create_cis

```
tBleStatus hci_le_create_cis    ( uint8_t          CIS_Count,
                                  CIS_Handles_t    CIS_Handles[]
                                 )
```

The HCI_LE_Create_CIS command is used by the Central's Host to create one or more CISes using the connections identified by the ACL_Connection_Handle arrayed parameter. The CIS_Count parameter is the total number of CISes created by this command. The CIS_Connection_Handle[i] parameter specifies the connection handle corresponding to the configuration of the CIS to be created and whose configuration is already stored in a CIG. The ACL_Connection_Handle[i] parameter specifies the connection handle of the ACL connection associated with each CIS to be created. The list of the ACL_Connection_Handles shall be in the same order as the list of the CIS_Connection_Handles e.g., CIS_Connection_Handle[1] connects to the Peripheral associated with the ACL_Connection_Handle[1].

If any ACL_Connection_Handle[i] is not the handle of an existing ACL connection or any CIS_Connection_Handle[i] is not the handle of a CIS or CIS configuration, the Controller shall return the error code Unknown Connection Identifier (0x02). If the Host attempts to create a CIS that has already been created, the Controller shall return the error code Connection Already Exists (0x0B). If two different elements of the CIS_Connection_Handle arrayed parameter identify the same CIS, the Controller shall return the error code Invalid HCI Command Parameters (0x12). If the Host issues this command before all the HCI_LE_CIS_Established events from the previous use of the command have been generated, the Controller shall return the error code Command Disallowed (0x0C). If the Host issues this command on an ACL_Connection_Handle where the Controller is the Peripheral, the Controller shall return the error code Command Disallowed (0x0C).

Note: The order of the CIS connection handles in this command does not relate to the order of connection handles in the return parameters of the HCI_LE_Set_CIG_Parameters command or the HCI_LE_Set_CIG_Parameters_Test command. If the Host issues this command when the Connected Isochronous Stream (Host Support) feature bit (see [Vol 6] Part B, Section 4.6.27) is not set, the Controller shall return the error code Command Disallowed (0x0C).

Parameters

CIS_Count

Total number of CISes to be created. Values:

- 0x01 ... 0x1F

CIS_Handles

See CIS_Handles_t.

Return values:

- *Value* indicating success or error code.

2.1.17 hci_le_create_connection

```
tBleStatus hci_le_create_connection    ( uint16_t LE_Scan_Interval,
                                         uint16_t LE_Scan_Window,
                                         uint8_t Initiator_Filter_Policy,
                                         uint8_t Peer_Address_Type,
                                         uint8_t Peer_Address[6],
                                         uint8_t Own_Address_Type,
                                         uint16_t Connection_Interval_Min,
                                         uint16_t Connection_Interval_Max,
                                         uint16_t Max_Latency,
                                         uint16_t Supervision_Timeout,
                                         uint16_t Min_CE_Length,
                                         uint16_t Max_CE_Length
                                       )
```

The LE_Create_Connection command is used to create a Link Layer connection to a connectable advertiser. The LE_Scan_Interval and LE_Scan_Window parameters are recommendations from the Host on how long (LE_Scan_Window) and how frequently (LE_Scan_Interval) the Controller should scan. The LE_Scan_Window parameter shall be set to a value smaller or equal to the value set for the LE_Scan_Interval parameter. If both are set to the same value, scanning should run continuously. The Initiator_Filter_Policy is used to determine whether the Filter Accept List is used. If the Filter Accept List is not used, the Peer_Address_Type and the Peer_Address parameters specify the address type and address of the advertising device to connect to. The Link Layer shall set the address in the CONNECT_REQ packets to either the Public Device Address or the Random Device Addressed based on the Own_Address_Type parameter.

The Connection_Interval_Min and Connection_Interval_Max parameters define the minimum and maximum allowed connection interval. The Connection_Interval_Min parameter shall not be greater than the Connection_Interval_Max parameter. The Max_Latency parameter defines the maximum allowed connection latency (see [Vol 6] Part B, Section 4.5.1). The Supervision_Timeout parameter defines the link supervision timeout for the connection. The Supervision_Timeout in milliseconds shall be larger than (1 + Max_Latency) * Connection_Interval_Max * 2, where Connection_Interval_Max is given in milliseconds. (See [Vol 6] Part B, Section 4.5.2).

The Min_CE_Length and Max_CE_Length parameters are informative parameters providing the Controller with the expected minimum and maximum length of the connection events. The Min_CE_Length parameter shall be less than or equal to the Max_CE_Length parameter. The Host shall not issue this command when another LE_Create_Connection is pending in the Controller; if this does occur the Controller shall return the Command Disallowed error code shall be used. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.12)

Parameters

LE_Scan_Interval

This is defined as the time interval from when the Controller started its last LE scan until it begins the subsequent LE scan. Time = N * 0.625 msec. Values:

- 0x0004 (2.500 ms) ... 0x4000 (10240.000 ms)

LE_Scan_Window

The duration of the LE scan. LE_Scan_Window shall be less than or equal to LE_Scan_Interval. Time = N * 0.625 msec. Values:

- 0x0004 (2.500 ms) ... 0x4000 (10240.000 ms)

Initiator_Filter_Policy

0x00 Filter Accept List is not used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be used. 0x01 Filter Accept List is used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be ignored. Values:

- 0x00: Filter Accept List not used
- 0x01: Filter Accept List used

Peer_Address_Type

0x00 Public Device Address 0x01 Random Device Address 0x02 Public Identity Address (Corresponds to Resolved Private Address) 0x03 Random (Static) Identity Address (Corresponds to Resolved Private Address) Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Public Identity Address
- 0x03: Random (Static) Identity Address

Peer_Address

Public Device Address, Random Device Address, Public Identity Address or Random (static) Identity Address of the advertising device.

Own_Address_Type

Own address type.

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Controller generates Resolvable Private Address based on the local IRK from resolving list. If resolving list contains no matching entry, use public address.
- 0x03: Controller generates Resolvable Private Address based on the local IRK from resolving list. If resolving list contains no matching entry, use random address from LE_Set_Random_Address.

Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Resolvable Private Address or Public Address
- 0x03: Resolvable Private Address or Random Address

Connection_Interval_Min

Minimum value for the connection event interval. This shall be less than or equal to Connection_Interval_Max. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Connection_Interval_Max

Maximum value for the connection event interval. This shall be greater than or equal to Connection_Interval_Min. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Max_Latency

Maximum Peripheral latency for the connection in number of connection events. Values:

- 0x0000 ... 0x01F3

Supervision_Timeout

Supervision timeout for the LE Link. It shall be a multiple of 10 ms and larger than (1 + connPeripheralLatency) * connInterval * 2. Time = N * 10 msec. Values:

- 0x000A (100 ms) ... 0x0C80 (32000 ms)

Min_CE_Length

The minimum length of connection event recommended for this LE connection. Time = N * 0.625 msec.

Max_CE_Length

The maximum length of connection event recommended for this LE connection. Time = N * 0.625 msec.

Return values:

- *Value* indicating success or error code.

2.1.18 hci_le_create_connection_cancel

```
tBleStatus hci_le_create_connection_cancel ( void )
```

The LE_Create_Connection_Cancel command is used to cancel the LE_Create_Connection command. This command shall only be issued after the LE_Create_Connection command has been issued, a Command Status event has been received for the LE Create Connection command and before the LE Connection Complete event. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.13.)

Return values:

- *Value* indicating success or error code.

2.1.19 hci_le_enable_encryption

```
tBleStatus hci_le_enable_encryption    ( uint16_t Connection_Handle,
                                          uint8_t  Random_Number[8],
                                          uint16_t Encrypted_Diversifier,
                                          uint8_t  Long_Term_Key[16]
                                         )
```

The LE_Enable_Encryption command is used to authenticate the given encryption key associated with the remote device specified by the connection handle, and once authenticated encrypts the connection. The parameters are as defined in [Vol 3] Part H, Section 2.4.4. If the connection is already encrypted then the Controller shall pause connection encryption before attempting to authenticate the given encryption key, and then re-encrypt the connection. While encryption is paused no user data shall be transmitted. On an authentication failure, the connection shall be automatically disconnected by the Link Layer. If this command succeeds, then the connection shall be encrypted. This command shall only be used when the local device's role is Central.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Random_Number

64-bit random number.

Encrypted_Diversifier

16-bit encrypted diversifier

Long_Term_Key

128-bit long term key

Return values:

- *Value* indicating success or error code.

2.1.20 hci_le_encrypt

```
tBleStatus hci_le_encrypt    ( uint8_t Key[16],
                                uint8_t Plaintext_Data[16],
                                uint8_t Encrypted_Data[16]
                               )
```

The LE_Encrypt command is used to request the Controller to encrypt the Plaintext_Data in the command using the Key given in the command and returns the Encrypted_Data to the Host. The AES-128 bit block cypher is defined in NIST Publication FIPS-197 (http://csrc.nist.gov/publications/fips/ fips197/fips-197.pdf). (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.22.)

Parameters

Key

128-bit key for the encryption of the data given in the command.

Plaintext_Data

128-bit data block that is requested to be encrypted.

`[out]` **Encrypted_Data**

128-bit encrypted data block.

Return values:

- *Value* indicating success or error code.

2.1.21 hci_le_enhanced_read_transmit_power_level

```
tBleStatus hci_le_enhanced_read_transmit_power_level ( uint16_t Connection_Handle,
                                                         uint8_t  PHY,
                                                         int8_t * Current_Transmit_Power_Level,
                                                         int8_t * Max_Transmit_Power_Level
                                                        )
```

Read the current and maximum transmit power levels of the local Controller on the ACL connection identified by the Connection_Handle parameter and the PHY indicated by the PHY parameter.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

PHY

PHY associated with the connection (not necessarily the currently used one). Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY_S8
- 0x04: LE_CODED_PHY_S2

`[out]` **Current_Transmit_Power_Level**

Current TX power level (dBm). Values:

- -127 ... 20
- 127: NA

`[out]` **Max_Transmit_Power_Level**

Maximum TX power level (dBm). Values:

- -127 ... 20

Return values:

- *Value* indicating success or error code.

2.1.22 hci_le_extended_create_connection

```
tBleStatus hci_le_extended_create_connection    ( uint8_t Advertising_Handle,
                                                   uint8_t Subevent,
                                                   uint8_t Initiator_Filter_Policy,
                                                   uint8_t Own_Address_Type,
                                                   uint8_t Peer_Address_Type,
                                                   uint8_t Peer_Address[6],
                                                   uint8_t Initiating_PHYs,
                                                   Extended_Create_Connection_Parameters_t
                                                   Extended_Create_Connection_Parameters[]
                                                  )
```

The LE_Extended_Create_Connection command is used to create a Link Layer connection to a connectable advertiser. LE_Extended_Create_Connection command can be used in place of LE_Create_Connection command.

Parameters

Advertising_Handle

Advertising_Handle identifying the periodic advertising train. Values:

- 0x00 ... 0xEF
- 0xFF: Not specified

Subevent

Subevent where the connection request is to be sent. Values:

- 0x00 ... 0x7F
- 0xFF: Not specified

Initiator_Filter_Policy

The Initiator_Filter_Policy parameter is used to determine whether the Filter Accept List is used. If the Filter Accept List is not used, the Peer_Address_Type and the Peer_Address parameters specify the address type and address of the advertising device to connect to. Values:

- 0x00: FILTER_ACCEPT_LIST_NOT_USED. Filter Accept List is not used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be used.
- 0x01: FILTER_ACCEPT_LIST_USED. Filter Accept List is used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be ignored.

Own_Address_Type

The Own_Address_Type parameter indicates the type of address being used in the connection request packets. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the public address.
- 0x03: Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the random address from the most recent successful LE_Set_Random_Address Command.

Peer_Address_Type

The Peer_Address_Type parameter indicates the type of address used in the connectable advertisement sent by the peer. Values:

- 0x00: Public Address. Public Device Address or Public Identity Address
- 0x01: Random Address. Random Device Address or Random (static) Identity Address

Peer_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected.

Initiating_PHYs

The Initiating_PHYs parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising channel and the PHYs for which connection parameters have been specified. The Host may enable one or more initiating PHYs. Flags:

- 0x01: LE_1M_PHY_BIT. Scan connectable advertisements on the LE 1M PHY. Connection parameters for the LE 1M PHY are provided.
- 0x02: LE_2M_PHY_BIT. Connection parameters for the LE 2M PHY are provided.
- 0x04: LE_CODED_PHY_BIT. Scan connectable advertisements on the LE Coded PHY. Connection parameters for the LE Coded PHY are provided.

Extended_Create_Connection_Parameters

See Extended_Create_Connection_Parameters_t.

Return values:

- *Value* indicating success or error code.

2.1.23 hci_le_extended_create_connection_v2

```
tBleStatus hci_le_extended_create_connection_v2 ( uint8_t Advertising_Handle,
                                                  uint8_t Subevent,
                                                  uint8_t Initiator_Filter_Policy,
                                                  uint8_t Own_Address_Type,
                                                  uint8_t Peer_Address_Type,
                                                  uint8_t Peer_Address[6],
                                                  uint8_t Initiating_PHYs,
                                                   Extended_Create_Connection_Parameters_t
                                                  Extended_Create_Connection_Parameters[]
                                                )
```

The LE_Extended_Create_Connection command is used to create a Link Layer connection to a connectable advertiser. LE_Extended_Create_Connection command can be used in place of LE_Create_Connection command.

Parameters

Advertising_Handle

Advertising_Handle identifying the periodic advertising train. Values:

- 0x00 ... 0xEF
- 0xFF: Not specified

Subevent

Subevent where the connection request is to be sent. Values:

- 0x00 ... 0x7F
- 0xFF: Not specified

Initiator_Filter_Policy

The Initiator_Filter_Policy parameter is used to determine whether the Filter Accept List is used. If the Filter Accept List is not used, the Peer_Address_Type and the Peer_Address parameters specify the address type and address of the advertising device to connect to. Values:

- 0x00: FILTER_ACCEPT_LIST_NOT_USED. Filter Accept List is not used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be used.
- 0x01: FILTER_ACCEPT_LIST_USED. Filter Accept List is used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be ignored.

Own_Address_Type

The Own_Address_Type parameter indicates the type of address being used in the connection request packets. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the public address.
- 0x03: Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the random address from the most recent successful LE_Set_Random_Address Command.

Peer_Address_Type

The Peer_Address_Type parameter indicates the type of address used in the connectable advertisement sent by the peer. Values:

- 0x00: Public Address. Public Device Address or Public Identity Address
- 0x01: Random Address. Random Device Address or Random (static) Identity Address

Peer_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected.

Initiating_PHYs

The Initiating_PHYs parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising channel and the PHYs for which connection parameters have been specified. The Host may enable one or more initiating PHYs. Flags:

- 0x01: LE_1M_PHY_BIT. Scan connectable advertisements on the LE 1M PHY. Connection parameters for the LE 1M PHY are provided.
- 0x02: LE_2M_PHY_BIT. Connection parameters for the LE 2M PHY are provided.
- 0x04: LE_CODED_PHY_BIT. Scan connectable advertisements on the LE Coded PHY. Connection parameters for the LE Coded PHY are provided.

Extended_Create_Connection_Parameters

See Extended_Create_Connection_Parameters_t.

Return values:

- *Value* indicating success or error code.

2.1.24 hci_le_generate_dhkey

```
tBleStatus hci_le_generate_dhkey ( uint8_t Remote_P256_Public_Key[64] )
```

The LE_Generate_DHKey command is used to initiate generation of a Diffie-Hellman key in the Controller for use over the LE transport. This command takes the remote P-256 public key as input. The Diffie-Hellman key generation uses the private key generated by LE_Read_Local_P256_Public_Key command. (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.37.)

Parameters

Remote_P256_Public_Key

The remote P-256 public key: X, Y format

- Octets 31-0: X coordinate octets
- 63-32: Y coordinate

Little Endian Format

Return values:

- *Value* indicating success or error code.

2.1.25 hci_le_long_term_key_request_negative_reply

```
tBleStatus hci_le_long_term_key_request_negative_reply ( uint16_t Connection_Handle )
```

The LE_Long_Term_Key_Request_Negative_Reply command is used to reply to an LE Long Term Key Request event from the Controller if the Host cannot provide a Long Term Key for this Connection_Handle. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.26.)

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:

- 0x0000 ... 0x0EFF

Return values:

- *Value* indicating success or error code.

2.1.26 hci_le_long_term_key_request_reply

```
tBleStatus hci_le_long_term_key_request_reply    ( uint16_t Connection_Handle,
                                                    uint8_t  Long_Term_Key[16]
                                                  )
```

The LE_Long_Term_Key_Request_Reply command is used to reply to an LE Long Term Key Request event from the Controller, and specifies the Long_Term_Key parameter that shall be used for this Connection_Handle. The Long_Term_Key is used as defined in [Vol 6] Part B, Section 5.1.3. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.25.)

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:

- 0x0000 ... 0x0EFF

Long_Term_Key

128-bit long-term key.

Return values:

- *Value* indicating success or error code.

2.1.27 hci_le_periodic_advertising_create_sync

```
tBleStatus hci_le_periodic_advertising_create_sync    ( uint8_t  Options,
                                                         uint8_t  Advertising_SID,
                                                         uint8_t   Advertiser_Address_Type,
                                                         uint8_t   Advertiser_Address[6],
                                                         uint16_t Skip,
                                                         uint16_t Sync_Timeout,
                                                         uint8_t  Sync_CTE_Type
                                                        )
```

The HCI_LE_Periodic_Advertising_Create_Sync command is used to synchronize with a periodic advertising train from an advertiser and begin receiving periodic advertising packets. This command may be issued whether or not scanning is enabled and scanning may be enabled and disabled (see the LE Set Extended Scan Enable command) while this command is pending. However, synchronization can only occur when scanning is enabled. While scanning is disabled, no attempt to synchronize takes place.

The Options parameter is used to determine whether the Periodic Advertiser List is used, whether HCI_LE_Periodic_Advertising_Report events for this periodic advertising train are initially enabled or disabled, and whether duplicate reports are filtered or not. If the Periodic Advertiser List is not used, the Advertising_SID, Advertiser Address_Type, and Advertiser Address parameters specify the periodic advertising device to listen to; otherwise they shall be ignored. The Advertising_SID parameter, if used, specifies the value that must match the Advertising SID subfield in the ADI field of the received advertisement for it to be used to synchronize.

The Skip parameter specifies the maximum number of consecutive periodic advertising events that the receiver may skip after successfully receiving a periodic advertising packet. The Sync_Timeout parameter specifies the maximum permitted time between successful receives. If this time is exceeded, synchronization is lost. The Sync_CTE_Type parameter specifies whether to only synchronize to periodic advertising with certain types of Constant Tone Extension (a value of 0 indicates that the presence or absence of a Constant Tone Extension is irrelevant).

If the periodic advertising has the wrong type of Constant Tone Extension, then:

- If bit 0 of Options is set, the Controller shall ignore this address and SID and continue to search for other periodic advertisements.
- Otherwise, the Controller shall cancel the synchronization with the error code Unsupported Remote Feature (0x1A).

If the periodic advertiser changes the type of Constant Tone Extension after the scanner has synchronized with the periodic advertising, the scanner's Link Layer shall remain synchronized. If the Host sets all the non-reserved bits of the Sync_CTE_Type parameter to 1, the Controller shall return the error code Command Disallowed (0x0C). Irrespective of the value of the Skip parameter, the Controller should stop skipping packets before the Sync_Timeout would be exceeded. If the Host issues this command when another HCI_LE_Periodic_Advertising, the Create_Sync command is pending, the Controller shall return the error code Command Disallowed (0x0C). If the Host issues this command with bit 0 of Options not set and with Advertising_SID, Advertiser_Address_Type, and Advertiser_Address the same as those of a periodic advertising train that the Controller is already synchronized to, the Controller shall return the error code Connection Already Exists (0x0B). If the Host issues this command and the Controller has insufficient resources to handle any more periodic advertising trains, the Controller shall return the error code Memory Capacity Exceeded (0x07).

If bit 1 of Options is set to 1 and the Controller supports the Periodic Advertising ADI Support feature, then the Controller shall ignore bit 2. If bit 1 of Options is set to 0, bit 2 is set to 1, and the Controller does not support the Periodic Advertising ADI Support feature, then the Controller shall return an error which should use the error code Unsupported Feature or Parameter Value (0x11). If bit 1 of the Options parameter is set to 1 and the Controller does not support the HCI_LE_Set_Periodic_Advertising_Receive_Enable command, the Controller shall return the error code Connection Failed to be Established / Synchronization Timeout (0x3E).

Parameters

Options

The Options parameter is a bitmask used to determine whether the Periodic Advertiser List is used, whether HCI_LE_Periodic_Advertising_Report events for this periodic advertising train are initially enabled or disabled, and whether duplicate reports are filtered or not. If bit 0 is 0: use the Advertising_SID, Advertiser_Address_Type, and Advertiser_Address parameters to determine which advertiser to listen to. If bit 0 is 1: use the Periodic Advertiser List to determine which advertiser to listen to. If bit 1 is 0, reporting is initially enabled, otherwise it is enabled. If bit 2 is 0, duplicate filtering is initially disabled, otherwise it is enabled. Flags:

- 0x01: USE_PERIODIC_ADV_LIST
- 0x02: DISABLE_REPORTING
- 0x04: ENABLE_DUPLICATE_FILTERING

Advertising_SID

The Advertising_SID parameter, if used, specifies the value that must match the Advertising SID subfield in the ADI field of the received advertisement for it to be used to synchronize. Values:

- 0x00 ... 0x0F: Advertising SID subfield in the ADI field used to identify the Periodic Advertising

Advertiser_Address_Type

Advertising address type. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address

Advertiser_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the advertiser.

Skip

The Skip parameter specifies the number of consecutive periodic advertising packets that the receiver may skip after successfully receiving a periodic advertising packet. Values:

- 0x0000 ... 0x01F3

Sync_Timeout

Synchronization timeout for the periodic advertising train. Time = N*10 ms. Values:

- 0x000A (100 ms) ... 0x4000 (163840 ms)

Sync_CTE_Type

The Sync_CTE_Type parameter specifies whether to only synchronize to periodic advertising with certain types of Constant Tone Extension (a value of 0 indicates that the presence or absence of a Constant Tone Extension is irrelevant). If the periodic advertising has the wrong type of Constant Tone Extension then: - If bit 0 of Options is set, the Controller shall ignore this address and SID and continue to search for other periodic advertisements. - Otherwise, the Controller shall cancel the synchronization with the error code Unsupported Remote Feature/ Unsupported LMP Feature (0x1A). Flags:

- 0x01: Do not sync to packets with an AoA Constant Tone Extension
- 0x02: Do not sync to packets with an AoD Constant Tone Extension with 1 microsecondslots
- 0x04: Do not sync to packets with an AoD Constant Tone Extension with 2 microsecondsslots
- 0x08: Do not sync to packets with a type 3 Constant Tone Extension (currentlyreserved for future use)
- 0x10: Do not sync to packets without a Constant Tone Extension

Return values:

- *Value* indicating success or error code.

2.1.28 hci_le_periodic_advertising_create_sync_cancel

```
tBleStatus hci_le_periodic_advertising_create_sync_cancel ( void )
```

The LE_Periodic_Advertising_Create_Sync_Cancel command is used to cancel the LE_Periodic_Advertising_Create_Sync command while it is pending. If the Host issues this command while no LE_Periodic_Advertising_Create_Sync command is pending, the Controller shall return the error code Command Disallowed (0x0C).

Return values:

- *Value* indicating success or error code.

2.1.29 hci_le_periodic_advertising_set_info_transfer

```
tBleStatus hci_le_periodic_advertising_set_info_transfer    ( uint16_t Connection_Handle,
                                                               uint16_t Service_Data,
                                                               uint8_t Advertising_Handle
                                                              )
```

The HCI_LE_Periodic_Advertising_Set_Info_Transfer command is used to instruct the Controller to send synchronization information about the periodic advertising in an advertising set to a connected device. The Advertising_Handle parameter identifies the advertising set. If the parameters in the advertising set have changed since the periodic advertising was first enabled, the current parameters, not the original ones, are sent. The Service_Data parameter is a value provided by the Host to identify the periodic advertising train to the peer device. It is not used by the Controller. The connected device is identified by the Connection_Handle parameter. If the advertising set corresponding to the Advertising_Handle parameter does not exist, the Controller shall return the error code Unknown Advertising Identifier (0x42).

If periodic advertising is not currently in progress for the advertising set, the Controller shall return the error code Command Disallowed (0x0C). If the Connection_Handle parameter does not identify a current connection, the Controller shall return the error code Unknown Connection Identifier (0x02). If the remote device has not indicated support for the Periodic Advertising Sync Transfer - Recipient feature, the Controller shall return the error code Unsupported Remote Feature / Unsupported LMP Feature (0x1A). Note: This command may complete before the periodic advertising synchronization information is sent. No indication is given as to how the recipient handled the information.

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:

- 0x0000 ... 0x0EFF

Service_Data

It is a value provided by the Host to identify the periodic advertising train to the peer device. It is not used by the Controller.

Advertising_Handle

It is used to identify an advertising set. Values:

- 0x00 ... 0xEF

Return values:

- *Value* indicating success or error code.

2.1.30 hci_le_periodic_advertising_sync_transfer

```
tBleStatus hci_le_periodic_advertising_sync_transfer    ( uint16_t Connection_Handle,
                                                           uint16_t Service_Data,
                                                           uint16_t Sync_Handle
                                                          )
```

The HCI_LE_Periodic_Advertising_Sync_Transfer command is used to instruct the Controller to send synchronization information about the periodic advertising train identified by the Sync_Handle parameter to a connected device. The Service_Data parameter is a value provided by the Host for use by the Host of the peer device. It is not used by the Controller. The connected device is identified by the Connection_Handle parameter. If the periodic advertising train corresponding to the Sync_Handle parameter does not exist, the Controller shall return the error code Unknown Advertising Identifier (0x42). If the Connection_Handle parameter does not identify a current connection, the Controller shall return the error code Unknown Connection Identifier (0x02). If the remote device has not indicated support for the Periodic Advertising Sync Transfer - Recipient feature, the Controller shall return the error code Unsupported Remote Feature / Unsupported LMP Feature (0x1A). Note: This command may complete before the periodic advertising synchronization information is sent. No indication is given as to how the recipient handled the information.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Service_Data

The Service_Data parameter is a value provided by the Host for use by the Host of the peer device. It is not used by the Controller.

Sync_Handle

Sync handle that identifies the synchronization information about the periodic advertising train. Values:

- 0x0000 ... 0x0EFF

Return values:

- *Value* indicating success or error code.

2.1.31 hci_le_periodic_advertising_terminate_sync

```
tBleStatus hci_le_periodic_advertising_terminate_sync ( uint16_t Sync_Handle )
```

The LE_Periodic_Advertising_Terminate_Sync command is used to stop reception of the periodic advertising identified by the Sync_Handle parameter. If the Host issues this command when another LE_Periodic_Advertising_Create_Sync command is pending (see below), the Controller shall return the error code Command Disallowed (0x0C). If the periodic advertising corresponding to the Sync_Handle parameter does not exist, then the Controller shall return the error code Unknown Advertising Identifier (0x42).

Parameters

Sync_Handle

It is used to identify the periodic advertiser. Values:

- 0x0000 ... 0x0EFF: Sync_Handle to be used to identify the periodic advertiser

Return values:

- *Value* indicating success or error code.

2.1.32 hci_le_rand

```
tBleStatus hci_le_rand ( uint8_t Random_Number[8] )
```

The LE_Rand command is used to request the Controller to generate 8 octets of random data to be sent to the Host. The Random_Number shall be generated according to [Vol 2] Part H, Section 2 if the LE Feature (LL Encryption) is supported. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.23.)

Parameters

`[out]` **Random_Number**

Random_Number

Return values:

- *Value* indicating success or error code.

2.1.33 hci_le_read_advertising_physical_channel_tx_power

```
tBleStatus hci_le_read_advertising_physical_channel_tx_power ( int8_t * Transmit_Power_Level
)
```

The LE_Read_Advertising_Physical_Channel_Tx_Power command is used by the Host to read the transmit power level used for LE advertising channel packets. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.6.)

Parameters

`[out]` **Transmit_Power_Level**

Size: 1 Octet (signed integer)

Units: dBm Accuracy: +/- 4 dBm.

Values:

- -20 ... 10

Return values:

- *Value* indicating success or error code.

2.1.34 hci_le_read_antenna_information

```
tBleStatus hci_le_read_antenna_information    ( uint8_t * Supported_Switching_Sampling_Rates,
                                                uint8_t * Num_Antennae,
                                                uint8_t * Max_Switching_Pattern_Length,
                                                uint8_t * Max_CTE_Length
                                              )
```

The HCI_LE_Read_Antenna_Information command allows the Host to read the switching rates, the sampling rates, the number of antennae, and the maximum length of a transmitted Constant Tone Extension supported by the Controller.

Parameters

`[out]` **Supported_Switching_Sampling_Rates**

Flags:

- 0x00: 1 microsecond switching supported for AoD transmission
- 0x02: 1 microsecond switching supported for AoD reception
- 0x04: 1 microsecond switching and sampling supported for AoA reception

`[out]` **Num_Antennae**

Values:

- 0x01 ... 0x4B: The number of antennae supported by the Controller

`[out]` **Max_Switching_Pattern_Length**

Values:

- 0x02 ... 0x4B: Maximum length of antenna switching pattern supported by the Controller

`[out]` **Max_CTE_Length**

Values:

- 0x02 ... 0x14: Maximum length of a transmitted Constant Tone Extension supported in 8 microseconds units

Return values:

- *Value* indicating success or error code.

2.1.35 hci_le_read_buffer_size

```
tBleStatus hci_le_read_buffer_size    ( uint16_t * HC_LE_ACL_Data_Packet_Length,
                                        uint8_t  * HC_Total_Num_LE_ACL_Data_Packets
                                       )
```

The LE_Read_Buffer_Size command is used to read the maximum size of the data portion of HCI LE ACL Data Packets sent from the Host to the Controller. The Host segments the data transmitted to the Controller according to these values, so that the HCI Data Packets contains data with up to this size. The LE_Read_Buffer_Size command also returns the total number of HCI LE ACL Data Packets that can be stored in the data buffers of the Controller. The LE_Read_Buffer_Size command must be issued by the Host before it sends any data to an LE Controller (see Section 4.1.1). If the Controller returns a length value of zero, the Host shall use the Read_Buffer_Size command to determine the size of the data buffers.

Note: Both the Read_Buffer_Size and LE_Read_Buffer_Size commands may return buffer length and number of packets parameter values that are nonzero.

The HC_LE_ACL_Data_Packet_Length return parameter shall be used to determine the size of the L2CAP PDU segments contained in ACL Data Packets, which are transferred from the Host to the Controller to be broken up into packets by the Link Layer. Both the Host and the Controller shall support command and event packets, where the data portion (excluding header) contained in the packets is 255 octets in size. The HC_Total_Num_LE_ACL_Data_Packets return parameter contains the total number of HCI ACL Data Packets that can be stored in the data buffers of the Controller. The Host determines how the buffers are to be divided between different Connection Handles.

Note: The HC_LE_ACL_Data_Packet_Length return parameter does not include the length of the HCI Data Packet header. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.2.)

Parameters

`[out]` **HC_LE_ACL_Data_Packet_Length**

0x0000: No dedicated LE Buffer exists. Use the HCI_Read_Buffer_Size command. 0x001B - 0xFFFF Maximum length (in octets) of the data portion of each HCI ACL data packet. Values:

- 0x0000: NO_BUFFER
- 0x001B ... 0xFFFF

`[out]` **HC_Total_Num_LE_ACL_Data_Packets**

0x00: No dedicated LE Buffer exists. Use the HCI_Read_Buffer_Size command. 0x01 - 0xFF: Total number of HCI ACL Data Packets that can be stored in the data buffers of the Controller. Values:

- 0x00: NO_BUFFER
- 0x01 ... 0xFF

Return values:

- *Value* indicating success or error code.

2.1.36 hci_le_read_buffer_size_v2

```
tBleStatus hci_le_read_buffer_size_v2    ( uint16_t * HC_LE_ACL_Data_Packet_Length,
                                           uint8_t * HC_Total_Num_LE_ACL_Data_Packets,
                                           uint16_t * ISO_Data_Packet_Length,
                                           uint8_t * Total_Num_ISO_Data_Packets
                                          )
```

This command is used to read the maximum size of the data portion of ACL data packets and isochronous data packets sent from the Host to the Controller. The Host shall segment the data transmitted to the Controller according to these values so that the HCI Data packets and isochronous data packets contains data up to this size. The HCI_LE_Read_Buffer_Size command also returns the total number of HCI LE ACL Data packets and isochronous data packets that can be stored in the data buffers of the Controller. The HCI_LE_Read_Buffer_Size command shall be issued by the Host before it sends any data to an LE Controller (see Section 4.1.1). If the Controller supports HCI ISO Data packets, it shall return non-zero values for the ISO_Data_Packet_Length and Total_Num_ISO_Data_Packets parameters.

Parameters

`[out]` **HC_LE_ACL_Data_Packet_Length**

0x0000: No dedicated LE Buffer exists. Use the HCI_Read_Buffer_Size command. 0x001B - 0xFFFF Maximum length (in octets) of the data portion of each HCI ACL data packet. Values:

- 0x0000: NO_BUFFER
- 0x001B ... 0xFFFF

`[out]` **HC_Total_Num_LE_ACL_Data_Packets**

0x00: No dedicated LE Buffer exists. Use the HCI_Read_Buffer_Size command. 0x01 - 0xFF: Total number of HCI ACL Data Packets that can be stored in the data buffers of the Controller. Values:

- 0x00: NO_BUFFER
- 0x01 ... 0xFF

`[out]` **ISO_Data_Packet_Length**

0x0000: No dedicated ISO Buffer exists. 0x0001 to 0xFFFF: The maximum length (in octets) of the data portion of each HCI ISO data packet. Values:

- 0x0000: NO_BUFFER
- 0x0001 ... 0xFFFF

`[out]` **Total_Num_ISO_Data_Packets**

0x00: No dedicated ISO Buffer exists. 0x01 to 0xFF: The total number of HCI ISO data packets that can be stored in the ISO buffers of the Controller. Values:

- 0x00: NO_BUFFER
- 0x01 ... 0xFF

Return values:

- *Value* indicating success or error code.

2.1.37 hci_le_read_channel_map

```
tBleStatus hci_le_read_channel_map    ( uint16_t Connection_Handle,
                                          uint8_t  LE_Channel_Map[5]
                                        )
```

The LE_Read_Channel_Map command returns the current Channel_Map for the specified Connection_Handle. The returned value indicates the state of the Channel_Map specified by the last transmitted or received Channel_Map (in a CONNECT_REQ or LL_CHANNEL_MAP_REQ message) for the specified Connection_Handle, regardless of whether the Central has received an acknowledgement. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.20.)

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

`[out]` **LE_Channel_Map**

This parameter contains 37 1-bit fields. The nth such field (in the range 0 to 36) contains the value for the link layer channel index n. Channel n is unused = 0. Channel n is used = 1. The most significant bits are reserved and shall be set to 0.

Return values:

- *Value* indicating success or error code.

2.1.38 hci_le_read_filter_accept_list_size

```
tBleStatus hci_le_read_filter_accept_list_size ( uint8_t * Filter_Accept_List_Size )
```

The LE_Read_Filter_Accept_List_Size command is used to read the total number of Filter Accept list entries that can be stored in the Controller.

Parameters

`[out]` Filter_Accept_List_Size

Total number of Filter Accept List entries that can be stored in the Controller.

Return values:

- *Value* indicating success or error code.

2.1.39 hci_le_read_iso_link_quality

```
tBleStatus hci_le_read_iso_link_quality    ( uint16_t Connection_Handle,
                                              uint32_t * Tx_UnACKed_Packets,
                                              uint32_t * Tx_Flushed_Packets,
                                              uint32_t * Tx_Last_Subevent_Packets,
                                              uint32_t * Retransmitted_Packets,
                                              uint32_t * CRC_Error_Packets,
                                              uint32_t * Rx_Unreceived_Packets,
                                              uint32_t * Duplicate_Packets
                                            )
```

This command returns the values of various counters related to link quality that are associated with the isochronous stream specified by the Connection_Handle parameter. This command may be issued on both the central and peripheral if the connection handle identifies a CIS and on the Synchronized Receiver if the connection handle identifies a BIS.

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:

- 0x0000 ... 0x0EFF

`[out]` Tx_UnACKed_Packets

Value of the Tx_UnACKed_Packets counter. Incremented when the Link Layer does not receive an acknowledgment for a CIS Data PDU that it transmitted at least once by its flush point (see Core 5.2 [Vol 6] Part B, Section 4.5.13.5).

`[out]` Tx_Flushed_Packets

Value of the Tx_Flushed_Packets counter. Incremented when the Link Layer does not transmit a specific payload by its flush point.

`[out]` Tx_Last_Subevent_Packets

Value of the Tx_Last_Subevent_Packets counter. Incremented when the Link Layer transmits a CIS Data PDU in the last subevent of a CIS event.

`[out]` Retransmitted_Packets

Value of the Retransmitted_Packets counter. Incremented when the Link Layer retransmits a CIS Data PDU.

`[out]` CRC_Error_Packets

Value of the CRC_Error_Packets counter. Incremented when the Link Layer receives a packet with a CRC error.

`[out]` Rx_Unreceived_Packets

Value of the Rx_Unreceived_Packets counter. Incremented when the Link Layer does not receive a specific payload by its flush point (on a CIS) or the end of the event it is associated with (on a BIS; see Core v5.2 [Vol 6] Part B, Section 4.4.6.6).

`[out]` Duplicate_Packets

Value of the Duplicate_Packets counter. Incremented when the Link Layer receives a retransmission of a CIS Data PDU.

Return values:

- *Value* indicating success or error code.

2.1.40 hci_le_read_iso_tx_sync

```
tBleStatus hci_le_read_iso_tx_sync    ( uint16_t Connection_Handle,
                                        uint16_t * Packet_Sequence_Number,
                                        uint32_t * TX_Time_Stamp,
                                        uint8_t  Time_Offset[3]
                                       )
```

Transmitted SDU identified by the Packet_Sequence_Number on a CIS or BIS identified by the Connection_Handle parameter on the Central or Peripheral. The Packet_Sequence_Number parameter contains the sequence number of a transmitted SDU. The TX_Time_Stamp and Time_Offset parameters are described in [Vol 6] Part G, Section 3.3 and [Vol 6] Part G, Section 3.1 respectively. When the Connection_Handle identifies a CIS or BIS that is transmitting unframed PDUs, the value of Time_Offset returned shall be zero. If the Host issues this command with a connection handle that does not exist, or the connection handle is not associated with a CIS or BIS, the Controller shall return the error code Unknown Connection Identifier (0x02). If the Host issues this command on an existing connection handle for a CIS or BIS that is not configured for transmitting SDUs, the Controller shall return the error code Command Disallowed (0x0C). If the Host issues this command before an SDU has been transmitted by the Controller, the Controller shall return the error code Command Disallowed (0x0C).

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:

- 0x0000 ... 0x0EFF

`[out]` **Packet_Sequence_Number**

The packet sequence number of an SDU.

`[out]` **TX_Time_Stamp**

The CIG reference point or BIG anchor point of a transmitted SDU derived using the Controller's free running reference clock (in microseconds).

`[out]` **Time_Offset**

The time offset, in microseconds, that is associated with a transmitted SDU.

Return values:

- *Value* indicating success or error code.

2.1.41 hci_le_read_local_p256_public_key

```
tBleStatus hci_le_read_local_p256_public_key ( void )
```

The LE_Read_Local_P-256_Public_Key command is used to return the local P-256 public key from the Controller. The Controller shall generate a new P-256 public/private key pair upon receipt of this command. (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.36.)

Return values:

- *Value* indicating success or error code.

2.1.42 hci_le_read_local_resolvable_address

```
tBleStatus hci_le_read_local_resolvable_address    ( uint8_t Peer_Identity_Address_Type,
                                                     uint8_t Peer_Identity_Address[6],
                                                     uint8_t Local_Resolvable_Address[6]
                                                    )
```

The LE_Read_Local_Resolvable_Address command is used to get the current local Resolvable Private Address being used for the corresponding peer Identity Address. The local's resolvable address being used may change after the command is called. This command can be used at any time. When a Controller cannot find a Resolvable Private Address associated with the Peer Identity Address, it shall respond with error code 0x02 (Unknown Connection Identifier). (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.43.)

Parameters

Peer_Identity_Address_Type

Identity address type. Values:

- 0x00: Public Identity Address
- 0x01: Random (static) Identity Address

Peer_Identity_Address

Public or Random (static) Identity address of the peer device.

`[out]` **Local_Resolvable_Address**

Resolvable Private Address being used by the local device.

Return values:

- *Value* indicating success or error code.

2.1.43 hci_le_read_local_supported_features

```
tBleStatus hci_le_read_local_supported_features ( uint8_t LE_Features[8] )
```

This command requests the list of the supported LE features for the Controller. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.3.)

Parameters

`[out]` **LE_Features**

Bit Mask List of LE features. See Core v4.1, Vol. 6, Part B, Section 4.6.

Return values:

- *Value* indicating success or error code.

2.1.44 hci_le_read_maximum_data_length

```
tBleStatus hci_le_read_maximum_data_length    ( uint16_t * supportedMaxTxOctets,
                                                uint16_t * supportedMaxTxTime,
                                                uint16_t * supportedMaxRxOctets,
                                                uint16_t * supportedMaxRxTime
                                              )
```

The LE_Read_Maximum_Data_Length command allows the Host to read the Controller maximum supported payload octets and packet duration times for transmission and reception (supportedMaxTxOctets and supportedMaxTxTime, supportedMaxRxOctets, and supportedMaxRxTime, see [Vol 6] Part B, Section 4.5.10).

Parameters

`[out]` **supportedMaxTxOctets**

Maximum number of payload octets that the local Controller supports for transmission of a single Link Layer Data Channel PDU. Range 0x001B-0x00FB (0x0000 - 0x001A and 0x00FC - 0xFFFF reserved for future use).

`[out]` **supportedMaxTxTime**

Maximum time, in microseconds, that the local Controller supports for transmission of a single Link Layer Data Channel PDU. Range 0x0148-0x0848 (0x0000 - 0x0147 and 0x0849 - 0xFFFF reserved for future use).

`[out]` **supportedMaxRxOctets**

Maximum number of payload octets that the local Controller supports for reception of a single Link Layer Data Channel PDU. Range 0x001B-0x00FB (0x0000 - 0x001A and 0x00FC - 0xFFFF Reserved for future use).

`[out]` **supportedMaxRxTime**

Maximum time, in microseconds, that the local Controller supports for reception of a single Link Layer Data Channel PDU. Range 0x0148-0x0848 (0x0000 - 0x0147 and 0x0849 - 0xFFFF Reserved for future use).

Return values:

- *Value* indicating success or error code.

2.1.45 hci_le_read_number_of_supported_advertising_sets

```
tBleStatus hci_le_read_number_of_supported_advertising_sets ( uint8_t * Num_Supported_Adverti
sing_Sets )
```

The LE_Read_Number_of_Supported_Advertising_Sets command is used to read the maximum number of advertising sets supported by the advertising Controller at the same time.

Note: *The number of advertising sets that can be supported is not fixed and the Controller can change it at any time because the memory used to store advertising sets can also be used for other purposes.*

Parameters

`[out]` **Num_Supported_Advertising_Sets**

Maximum number of advertising sets supported by the advertising Controller at the same time. Values:

- 0x01 ... 0xF0: Number of advertising sets supported at the same time

Return values:

- *Value* indicating success or error code.

2.1.46 hci_le_read_peer_resolvable_address

```
tBleStatus hci_le_read_peer_resolvable_address    ( uint8_t Peer_Identity_Address_Type,
                                                    uint8_t Peer_Identity_Address[6],
                                                    uint8_t Peer_Resolvable_Address[6]
                                                   )
```

The LE_Read_Peer_Resolvable_Address command is used to get the current peer Resolvable Private Address being used for the corresponding peer Public and Random (static) Identity Address. The peer's resolvable address being used may change after the command is called. This command can be used at any time. When a Controller cannot find a Resolvable Private Address associated with the Peer Identity Address, it shall respond with error code 0x02 (Unknown Connection Identifier). (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.42.)

Parameters

Peer_Identity_Address_Type

Identity address type. Values:

- 0x00: Public Identity Address
- 0x01: Random (static) Identity Address

`[out]` **Peer_Identity_Address**

Public or Random (static) Identity address of the peer device.

Peer_Resolvable_Address

Resolvable Private Address being used by the peer device.

Return values:

- *Value* indicating success or error code.

2.1.47 hci_le_read_periodic_advertiser_list_size

```
tBleStatus hci_le_read_periodic_advertiser_list_size ( uint8_t * Periodic_Advertiser_List_Size )
```

The LE_Read_Periodic_Advertiser_List_Size command is used to read the total number of Periodic Advertiser list entries that can be stored in the Controller.

Note: *The number of entries that can be stored is not fixed and the Controller can change it at any time (for instance, because the memory used to store the list can also be used for other purposes).*

Parameters

`[out]` **Periodic_Advertiser_List_Size**

Total number of Periodic Advertiser list entries that can be stored in the Controller Values:

- 0x1F ... 0xFF: Total number of Periodic Advertiser list entries that can be stored in the Controller

Return values:

- *Value* indicating success or error code.

2.1.48 hci_le_read_phy

```
tBleStatus hci_le_read_periodic_advertiser_list_size    ( uint16_t Connection_Handle,
                                                          uint8_t  * TX_PHY,
                                                          uint8_t  * RX_PHY
                                                         )
```

The LE_Read_PHY command is used to read the current transmitter PHY and receiver PHY on the connection identified by the Connection_Handle.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

`[out]` **TX_PHY**

Transmitter PHY for the connection. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY

`[out]` **RX_PHY**

Receiver PHY for the connection. Values:

- 0x01: The receiver PHY for the connection is LE 1M.
- 0x02: The receiver PHY for the connection is LE 2M.
- 0x03: The receiver PHY for the connection is LE Coded.

All other values: reserved for future use

Return values:

- *Value* indicating success or error code.

2.1.49 hci_le_read_remote_features

```
tBleStatus hci_le_read_remote_features ( uint16_t Connection_Handle )
```

This command requests a list of the used LE features from the remote device. This command shall return a list of the used LE features. For details see [Vol 6] Part B, Section 4.6. This command may be issued on both the central and peripheral. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.21.)

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Return values:

- *Value* indicating success or error code.

2.1.50 hci_le_read_remote_transmit_power_level

```
tBleStatus hci_le_read_remote_transmit_power_level    ( uint16_t Connection_Handle,
                                                        uint8_t   PHY
                                                       )
```

Read the transmit power level used by the remote Controller on the ACL connection that is identified by the Connection_Handle parameter and the PHY indicated by the PHY parameter. Initiate a Power Control Request procedure to obtain the remote transmit power level if no prior value is available or used.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

PHY

PHY associated with the connection (not necessarily the currently used one). Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY_S8
- 0x04: LE_CODED_PHY_S2

Return values:

- *Value* indicating success or error code.

2.1.51 hci_le_read_resolving_list_size

```
tBleStatus hci_le_read_resolving_list_size ( uint8_t * Resolving_List_Size )
```

The LE_Read_Resolving_List_Size command is used to read the total number of address translation entries in the resolving list that can be stored in the Controller. (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.41.)

Parameters

`[out]` **Resolving_List_Size**

Number of address translation entries in the resolving list.

Return values:

- *Value* indicating success or error code.

2.1.52 hci_le_read_rf_path_compensation

```
tBleStatus hci_le_read_rf_path_compensation    ( int16_t * RF_TX_Path_Compensation_Value,
                                                 int16_t * RF_RX_Path_Compensation_Value
                                                )
```

The HCI_LE_Read_RF_Path_Compensation command is used to read the RF Path Compensation Values parameter used in the Tx Power Level and RSSI calculation.

Parameters

`[out]` **RF_TX_Path_Compensation_Value**

`[out]` **RF_RX_Path_Compensation_Value**

Return values:

- *Value* indicating success or error code.

2.1.53 hci_le_read_suggested_default_data_length

```
tBleStatus hci_le_read_suggested_default_data_length    ( uint16_t * SuggestedMaxTxOctets,
                                                          uint16_t * SuggestedMaxTxTime
                                                         )
```

The LE_Read_Suggested_Default_Data_Length command allows the Host to read the Host preferred values for the Controller maximum transmitted number of payload octets and maximum packet transmission time to be used for new connections (connInitialMaxTxOctets and connInitialMaxTxTime - see ([Vol 6] Part B, Section 4.5.10).

Parameters

`[out]` **SuggestedMaxTxOctets**

The Host suggested value for the Controller maximum transmitted number of payload octets to be used for new connections: connInitialMaxTxOctets.

Range 0x001B-0x00FB (0x0000 - 0x001A and 0x00FC - 0xFFFF reserved for future use)

Default: 0x001B

`[out]` **SuggestedMaxTxTime**

The Host suggested value for the Controller maximum packet transmission time to be used for new connections - connInitialMaxTx-Time.

Range 0x0148-0x0848 (0x0000 - 0x0147 and 0x0849 - 0xFFFF reserved for future use)

Default: 0x0148

Return values:

- *Value* indicating success or error code.

2.1.54 hci_le_read_supported_states

```
tBleStatus hci_le_read_supported_states ( uint8_t LE_States[8] )
```

The LE_Read_Supported_States command reads the states and state combinations that the link layer supports. See [Vol 6] Part B, Section 1.1.1. LE_States is an 8-octet bit field. If a bit is set to 1 then this state or state combination is supported by the Controller. Multiple bits in LE_States may be set to 1 to indicate support for multiple state and state combinations. All the Advertising type with the Initiate State combinations shall be set only if the corresponding Advertising types and Central Role combination are set. All the Scanning types and the Initiate State combinations shall be set only if the corresponding Scanning types and Central Role combination are set. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.27.)

Parameters

`[out]` **LE_States**

State or state combination is supported by the Controller. See Core v4.1, Vol.2, part E, Ch. 7.8.27.

Return values:

- *Value* indicating success or error code.

2.1.55 hci_le_read_transmit_power

```
tBleStatus hci_le_read_transmit_power    ( int8_t * Min_Tx_Power,
                                           int8_t * Max_Tx_Power
                                          )
```

The HCI_LE_Read_Transmit_Power command is used to read the minimum and maximum transmit powers supported by the Controller.

Parameters

`[out]` **Min_Tx_Power**

Minimum supported TX power (units: dBm). Values:

- -127 ... 20

`[out]` **Max_Tx_Power**

Maximum supported TX power (units: dBm). Values:

- -127 ... 20

Return values:

- *Value* indicating success or error code.

2.1.56 hci_le_reject_cis_request

```
tBleStatus hci_le_reject_cis_request    ( uint16_t Connection_Handle,
                                          uint8_t Reason
                                         )
```

The HCI_LE_Reject_CIS_Request command is used by the Peripheral's Host to inform the Controller to reject the request for the CIS that is identified by the Connection_Handle. The command shall only be issued after an HCI_LE_CIS_Request event has occurred. The event contains the Connection_Handle of the CIS. When this command succeeds, the Controller shall delete the Connection_Handle of the requested CIS. The Reason command parameter indicates the reason for rejecting the CIS request. If the Peripheral's Host issues this command with a Connection_Handle that is not for a CIS, the Controller shall return the error code Unknown Connection Identifier (0x02). If the Peripheral's Host issues this command with a Connection_Handle for a CIS that has already been established or that already has an HCI_LE_- Accept_CIS_Request or HCI_LE_Reject_CIS_Request command in progress, the Controller shall return the error code Command Disallowed (0x0C). If the Central's Host issues this command, the Controller shall return the error code Command Disallowed (0x0C).

Parameters

Connection_Handle

Reason the CIS request was rejected. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

Reason

Reason the CIS request was rejected. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

Return values:

- *Value* indicating success or error code.

2.1.57 hci_le_remove_advertising_set

```
tBleStatus hci_le_remove_advertising_set ( uint8_t Advertising_Handle )
```

The LE_Remove_Advertising_Set command is used to remove an advertising set from the Controller. If the advertising set corresponding to the Advertising_Handle parameter does not exist, then the Controller shall return the error code Unknown Advertising Identifier (0x42). If advertising on the advertising set is enabled, then the Controller shall return the error code Command Disallowed (0x0C).

Parameters

Advertising_Handle

It is used to identify an advertising set. Values:

- 0x00 ... 0xEF: Used to identify an advertising set

Return values:

- *Value* indicating success or error code.

2.1.58 hci_le_remove_cig

```
tBleStatus hci_le_remove_cig ( uint8_t CIG_ID )
```

The HCI_LE_Remove_CIG command is used by the Central's Host to remove the CIG identified by CIG_ID. The CIG_ID parameter contains the identifier of the CIG. This command shall delete the CIG_ID and also delete the Connection_Handles of the CIS configurations stored in the CIG. This command shall also remove the isochronous data paths that are associated with the Connection_Handles of the CIS configurations, which is equivalent to issuing the HCI_LE_Remove_ISO_Data_Path command (see Section 7.8.109). If the Host tries to remove a CIG which is in the active state, then the Controller shall return the error code Command Disallowed (0x0C). If the Host issues this command with a CIG_ID that does not exist, the Controller shall return the error code Unknown Connection Identifier (0x02).

Parameters

CIG_ID

Identifier of a CIG. Values:

- 0x00 ... 0xEF

Return values:

- *Value* indicating success or error code.

2.1.59 hci_le_remove_device_from_filter_accept_list

```
tBleStatus hci_le_remove_device_from_filter_accept_list    ( uint8_t Address_Type,
                                                             uint8_t Address[6]
                                                            )
```

The LE_Remove_Device_From_Filter_Accept_List command is used to remove a single device from the Filter Accept list stored in the Controller. This command can be used at any time except when:

- The advertising filter policy uses the Filter Accept list and advertising is enabled.
- The scanning filter policy uses the Filter Accept list and scanning is enabled.
- The initiator filter policy uses the Filter Accept list and a create connection command is outstanding.

Parameters

Address_Type

Address type. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address

Address

Public Device Address or Random Device Address of the device to be removed from the Filter Accept List.

Return values:

- *Value* indicating success or error code.

2.1.60 hci_le_remove_device_from_periodic_advertiser_list

```
tBleStatus hci_le_remove_device_from_filter_accept_list    ( uint8_t Advertiser_Address_Type,
                                                             uint8_t Advertiser_Address[6],
                                                             uint8_t Advertising_SID
                                                            )
```

The LE_Remove_Device_From_Periodic_Advertiser_List command is used to remove one device from the list of Periodic Advertisers stored in the Controller. Removals from the Periodic Advertisers List take effect immediately. If the Host issues this command when an LE_Periodic_Advertising_Create_Sync command is pending, the Controller shall return the error code Command Disallowed (0x0C). When a Controller cannot remove a device from the Periodic Advertiser list because it is not found, the Controller shall return the error code Unknown Advertising Identifier (0x42).

Parameters

Advertiser_Address_Type

Advertising Address type Values:
- 0x00: Public Device Address or Public Identity Address
- 0x01: Random Device Address or Random (static) Identity Address

Advertiser_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the advertiser.

Advertising_SID

It is used to identify the Periodic Advertising. Values:
- 0x00 ... 0x0F: Advertising SID subfield in the ADI field used to identify the Periodic Advertising All other values Reserved for future

Return values:
- *Value* indicating success or error code.

2.1.61 hci_le_remove_device_from_resolving_list

```
tBleStatus hci_le_remove_device_from_resolving_list    ( uint8_t Peer_Identity_Address_Type,
                                                         uint8_t Peer_Identity_Address[6]
                                                        )
```

The LE_Remove_Device_From_Resolving_List command is used to remove one device from the list of address translations used to resolve Resolvable Private Addresses in the controller. This command cannot be used when address translation is enabled in the Controller and:
- Advertising is enabled.
- Scanning is enabled.
- The Create connection command is outstanding.

This command can be used at any time when address translation is disabled in the Controller. When a Controller cannot remove a device from the resolving list because it is not found, it shall respond with error code 0x02 (Unknown Connection Identifier). (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.39.)

Parameters

Peer_Identity_Address_Type

Identity address type. Values:
- 0x00: Public Identity Address
- 0x01: Random (static) Identity Address

Peer_Identity_Address

Public or Random (static) Identity address of the peer device.

Return values:
- *Value* indicating success or error code.

2.1.62 hci_le_remove_iso_data_path

```
tBleStatus hci_le_remove_iso_data_path    ( uint16_t Connection_Handle,
                                            uint8_t  Data_Path_Direction
                                           )
```

The HCI_LE_Remove_ISO_Data_Path command is used to remove the input and/or output data path(s) associated with a CIS or BIS identified by the Connection_Handle parameter.

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:

- 0x0000 ... 0x0EFF

Data_Path_Direction

The Data_Path_Direction parameter specifies which directions are to have the data path removed. Flags:

- 0x01: Input
- 0x02: Output

Return values:

- *Value* indicating success or error code.

2.1.63 hci_le_request_peer_sca

```
tBleStatus hci_le_request_peer_sca ( uint16_t Connection_Handle )
```

This command is used to read the Sleep Clock Accuracy (SCA) of the peer device. The Connection_Handle parameter is the connection handle of the ACL connection. If the Host sends this command with a Connection_Handle that does not exist, or the Connection_Handle is not for an ACL the Controller shall return the error code Unknown Connection Identifier (0x02). If the Host sends this command and the peer device does not support the Sleep Clock Accuracy Updates feature, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11) in the HCI_LE_Request_Peer_SCA_Complete event. If the Host issues this command when the Controller is aware (e.g., through a previous feature exchange) that the peer device's Link Layer does not support the Sleep Clock Accuracy Updates feature, the Controller shall return the error code Unsupported Remote Feature (0x1A). When the HCI_LE_Request_Peer_SCA command has completed, the HCI_LE_Request_Peer_SCA_Complete event shall be generated.

Parameters

Connection_Handle

Connection handle of the ACL. Values:

- 0x0000 ... 0x0EFF

Return values:

- *Value* indicating success or error code.

2.1.64 hci_le_set_address_resolution_enable

```
tBleStatus hci_le_set_address_resolution_enable ( uint8_t Address_Resolution_Enable )
```

The LE_Set_Address_Resolution_Enable command is used to enable resolution of Resolvable Private Addresses in the Controller. This causes the Controller to use the resolving list whenever the Controller receives a local or peer Resolvable Private Address. This command can be used at any time except when:

- Advertising is enabled.
- Advertising is enabled.
- Create connection command is outstanding (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.44).

Parameters

Address_Resolution_Enable

Enable/disable address resolution in the controller. 0x00: Address Resolution in controller disabled (default), 0x01: Address Resolution in controller enabled. Values:

- 0x00: Address Resolution in controller disabled (default)
- 0x01: Address Resolution in controller enabled

Return values:

- *Value* indicating success or error code.

2.1.65 hci_le_set_advertising_enable

```
tBleStatus hci_le_set_advertising_enable ( uint8_t Advertising_Enable )
```

The LE_Set_Advertise_Enable command is used to request the Controller to start or stop advertising. The Controller manages the timing of advertisements as per the advertising parameters given in the LE_Set_Advertising_Parameters command. The Controller shall continue advertising until the Host issues an LE_Set_Advertise_Enable command with Advertising_Enable set to 0x00 (Advertising is disabled) or until a connection is created or until the Advertising is timed out due to high duty cycle Directed Advertising. In these cases, advertising is then disabled. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.9.)

Parameters

Advertising_Enable

Enable/disable advertise. Default is 0 (disabled). Values:

- 0x00: Disable
- 0x01: Enable

Return values:

- *Value* indicating success or error code.

2.1.66 hci_le_set_advertising_parameters

```
tBleStatus hci_le_set_advertising_parameters       ( uint16_t Advertising_Interval_Min,
                                                      uint16_t Advertising_Interval_Max,
                                                      uint8_t  Advertising_Type,
                                                      uint8_t  Own_Address_Type,
                                                      uint8_t  Peer_Address_Type,
                                                      uint8_t  Peer_Address[6],
                                                      uint8_t  Advertising_Channel_Map,
                                                      uint8_t  Advertising_Filter_Policy
                                                     )
```

The LE_Set_Advertising_Parameters command is used by the Host to set the advertising parameters. The Advertising_Interval_Min shall be less than or equal to the Advertising_Interval_Max. The Advertising_Interval_Min and Advertising_Interval_Max should not be the same value to enable the Controller to determine the best advertising interval given other activities. For high duty cycle directed advertising, i.e. when Advertising_Type is 0x01 (ADV_DIRECT_IND, high duty cycle), the Advertising_Interval_Min and Advertising_Interval_Max parameters are not used and shall be ignored. The Advertising_Type is used to determine the packet type that is used for advertising when advertising is enabled. Own_Address_Type parameter indicates the type of address being used in the advertising packets.

If Own_Address_Type equals 0x02 or 0x03, the Peer_Address parameter contains the peer's Identity Address and the Peer_Address_Type parameter contains the Peer's Identity Type (that is, 0x00 or 0x01). These parameters are used to locate the corresponding local IRK in the resolving list; this IRK is used to generate the own address used in the advertisement. If directed advertising is performed, i.e. when Advertising_Type is set to 0x01 (ADV_DIRECT_IND, high duty cycle) or 0x04 (ADV_DIRECT_IND, low duty cycle mode), then the Peer_Address_Type and Peer_Address shall be valid. If Own_Address_Type equals 0x02 or 0x03, the Controller generates the peer's Resolvable Private Address using the peer's IRK corresponding to the peer's Identity Address contained in the Peer_Address parameter and peer's Identity Address Type (that is. 0x00 or 0x01) contained in the Peer_Address_Type parameter.

The Advertising_Channel_Map is a bit field that indicates the advertising channels that shall be used when transmitting advertising packets. At least one channel bit shall be set in the Advertising_Channel_Map parameter. The Advertising_Filter_Policy parameter shall be ignored when directed advertising is enabled. The Host shall not issue this command when advertising is enabled in the Controller; if it is the Command Disallowed error code shall be used. If the advertising interval range provided by the Host (Advertising_Interval_Min, Advertising_Interval_Max) is outside the advertising interval range supported by the Controller, then the Controller shall return the Unsupported Feature or Parameter Value (0x11) error code.

Parameters

Advertising_Interval_Min

Minimum advertising interval for undirected and low duty cycle directed advertising. Time = N * 0.625 msec. Values:

- 0x0020 (20.000 ms) ... 0x4000 (10240.000 ms)

Advertising_Interval_Max

Maximum advertising interval. Time = N * 0.625 msec. Values:

- 0x0020 (20.000 ms) ... 0x4000 (10240.000 ms)

Advertising_Type

Advertising type. Values:

- 0x00: ADV_IND (Connectable undirected advertising)
- 0x01: ADV_DIRECT_IND, high duty cycle (Connectable high duty cycle directed advertising)
- 0x02: ADV_SCAN_IND (Scannable undirected advertising)
- 0x03: ADV_NONCONN_IND (Non connectable undirected advertising)
- 0x04: ADV_DIRECT_IND, low duty cycle (Connectable low duty cycle directed advertising)

Own_Address_Type

Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Resolvable Private Address or Public Address. Controller generates Resolvable Private Address based on the local IRK from resolving list.
- 0x03: Resolvable Private Address or Random Address. Controller generates Resolvable Private Address based on the local IRK from resolving list. If resolving list contains no matching entry, use random address from LE_Set_Random_Address.

Peer_Address_Type

Peer Address type. Values:

- 0x00: Public Device Address or Public Identity Address
- 0x01: Random Device Address or Random (static) Identity Address

Peer_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected.

Advertising_Channel_Map

Advertising channel map. Default: 00000111b (all channels enabled). Flags:

- 0x01: ch 37
- 0x02: ch 38
- 0x04: ch 39

Advertising_Filter_Policy

This parameter is ignored when directed advertising is enabled. Values:

- 0x00: HCI_ADV_FILTER_NONE. Process scan and connection requests from all devices (that is, the Filter Accept List is not in use).
- 0x01: HCI_ADV_FILTER_ACCEPT_LIST_SCAN. Process connection requests from all devices and scan requests only from devices that are in the Filter Accept List.
- 0x02: HCI_ADV_FILTER_ACCEPT_LIST_CONNECT. Process scan requests from all devices and connection requests only from devices that are in the Filter Accept List.
- 0x03: HCI_ADV_FILTER_ACCEPT_LIST_SCAN_CONNECT. Process scan and connection requests only from devices in the Filter Accept List.

All other values are reserved for future use

Return values:

- *Value* indicating success or error code.

2.1.67 hci_le_set_advertising_set_random_address

```
tBleStatus hci_le_set_advertising_set_random_address ( uint8_t Advertising_Handle,
                                                       uint8_t Advertising_Random_Address[6]
                                                      )
```

The LE_Set_Advertising_Set_Random_Address command is used by the Host to set the random device address specified by the Random_Address parameter. This address is used in the Controller (see [Vol 6] Part B, Section 1.3.2) for the advertiser's address contained in the advertising PDUs for the advertising set specified by the Advertising_Handle parameter. If the Host issues this command while an advertising set using connectable advertising is enabled, the Controller shall return the error code Command Disallowed (0x0C). The Host may issue this command at any other time. If this command is used to change the address, the new random address shall take effect for advertising no later than the next successful LE Extended Set Advertising Enable Command and for periodic advertising no later than the next successful LE Periodic Advertising Enable Command.

Parameters

Advertising_Handle

It is used to identify an advertising set. Values:

- 0x00 ... 0xEF

Advertising_Random_Address

Random Device Address as defined by [Vol 6] Part B, Section 1.3.2.

Return values:

- *Value* indicating success or error code.

2.1.68 hci_le_set_cig_parameters

```
tBleStatus hci_le_set_cig_parameters     ( uint8_t  CIG_ID,
                                           uint8_t  SDU_Interval_C_To_P[3],
                                           uint8_t  SDU_Interval_P_To_C[3],
                                           uint8_t  Worst_Case_SCA,
                                           uint8_t  Packing,
                                           uint8_t  Framing,
                                           uint16_t Max_Transport_Latency_C_To_P,
                                           uint16_t Max_Transport_Latency_P_To_C,
                                           uint8_t  CIS_Count,
                                           CIS_Param_t CIS_Param[],
                                           uint16_t Connection_Handle[]
                                          )
```

The HCI_LE_Set_CIG_Parameters command is used by a Central's Host to create a CIG and to set the parameters of one or more CISes that are associated with a CIG in the Controller. The CIG_ID parameter identifies a CIG. This parameter is allocated by the Central's Host and passed to the Peripheral's Host through the Link Layers during the process of creating a CIS. If the CIG_ID does not exist, then the Controller shall first create a new CIG. Once the CIG is created (whether through this command or previously), the Controller shall modify or add CIS configurations in the CIG that is identified by the CIG_ID and update all the parameters that apply to the CIG. The SDU_Interval_C_To_P parameter specifies the time interval between the start of consecutive SDUs from the Central's Host for all the CISes in the CIG. This parameter shall be ignored for all CISes that are unidirectional from Peripheral to Central.

The SDU_Interval_P_To_C parameter specifies the time interval between the start of consecutive SDUs from the Peripheral's Host for all the CISes in the CIG. This parameter shall be ignored for all CISes that are unidirectional from Central to Peripheral. The Worst_Case_SCA parameter shall be the worst-case sleep clock accuracy of all the Peripherals that participate in the CIG. The Host should get the sleep clock accuracy from all the Peripherals before issuing this command. If the Host cannot get the sleep clock accuracy from all the Peripherals, it shall set the Worst_Case_SCA parameter to zero.

Note: The Worst_Case_SCA parameter can be used by the Link Layer to better allow for clock drift when scheduling the CISes in the CIG. For example, if a CIS has more than two subevents, the Link Layer of the Central can set the timing of the subevents such that the worst case drift in the Peripheral's clock does not exceed 2 x Sub_Interval. This prevents the Peripheral from synchronizing its timing to the wrong subevent (adjacent subevents cannot be on the same channel).

The Packing parameter indicates the preferred method of arranging subevents of multiple CISes. The subevents can be arranged in Sequential or Interleaved arrangement (see [Vol 6] Part B, Section 4.5.14.2). This is a recommendation to the Controller which the Controller may ignore. This parameter shall be ignored when there is only one CIS in the CIG. The Framing parameter indicates the format of the CIS Data PDUs of the specified CISes. If the Framing parameter is set to 1 then the CIS Data PDUs of the specified CISes shall be framed. If the Framing parameter is set to 0 the CIS Data PDUs of a given CIS may be either unframed or framed (determined separately for each specified CIS) (see [Vol 6] Part G, Section 1). The Max_Transport_Latency_C_To_P parameter contains the maximum transport latency from the Central to the Peripheral, in milliseconds, as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2. This parameter shall be ignored for all CISes that are unidirectional from Peripheral to Central.

The Max_Transport_Latency_P_To_C parameter contains the maximum transport latency from the Peripheral to the Central, in milliseconds, as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2. This parameter shall be ignored for all CISes that are unidirectional from Central to Peripheral. The CIS_Count parameter indicates the number of CIS configurations being modified or added by this command. The Controller shall set the CIS_Count return parameter equal to this. The CIS_ID[i] parameter identifies a CIS and is set by the Central's Host and passed to the Peripheral's Host through the Link Layers during the process of establishing a CIS. The Max_SDU_C_To_P[i] parameter identifies the maximum size of an SDU from the Central's Host. If the CIS is unidirectional from Peripheral to Central, this parameter shall be set to 0. If a CIS configuration that is being modified has a data path set in the Central to Peripheral direction and the Host has specified that Max_SDU_C_To_P[i] shall be set to zero, the Controller shall return the error code Command Disallowed (0x0C). The Max_SDU_P_To_C[i] parameter identifies the maximum size of an SDU from the Peripheral's Host.

If the CIS is unidirectional from Central to Peripheral, this parameter shall be set to 0. If a CIS configuration that is being modified has a data path set in the Peripheral to Central direction and the Host has specified that Max_SDU_P_To_C[i] shall be set to zero, the Controller shall return the error code Command Disallowed (0x0C). The PHY_C_To_P[i] parameter identifies which PHY to use for transmission from the Central to the Peripheral. The Host shall set at least one bit in this parameter and the Controller shall pick a PHY from the bits that are set. The PHY_P_To_C[i] parameter identifies which PHY to use for transmission from the Peripheral to the Central. The Host shall set at least one bit in this parameter and the Controller shall pick a PHY from the bits that are set.

The RTN_C_To_P[i] (Retransmission Number) parameter contains the number of times that a CIS Data PDU should be retransmitted from the Central to Peripheral before being acknowledged or flushed (irrespective of which isochronous events the retransmission opportunities occur in). If the CIS is unidirectional from Peripheral to Central, this parameter shall be ignored. Otherwise, this parameter is a recommendation to the Controller which the Controller may ignore. The RTN_P_To_C[i] parameter contains the number of times that a CIS Data PDU should be retransmitted from the Peripheral to Central before being acknowledged or flushed (irrespective of which isochronous events the retransmission opportunities occur in). If the CIS is unidirectional from Central to Peripheral, this parameter shall be ignored. Otherwise, this parameter is a recommendation to the Controller which the Controller may ignore.

If the Status return parameter is non-zero, then the state of the CIG and its CIS configurations shall not be changed by the command. If the CIG did not already exist, it shall not be created. If the Status return parameter is zero, then the Controller shall set the Connection_Handle arrayed return parameter to the connection handle(s) corresponding to the CIS configurations specified in the CIS_IDs command parameter, in the same order. If the same CIS_ID is being reconfigured, the same connection handle shall be returned.

The connection handle of a CIS shall refer to the CIS when it exists and to the configuration of the CIS stored in a CIG when the CIG exists but the CIS with that CIS_ID does not. If the Host issues this command when the CIG is not in the configurable state, the Controller shall return the error code Command Disallowed (0x0C). If the Host attempts to create a CIG or set parameters that exceed the maximum supported resources in the Controller, the Controller shall return the error code Memory Capacity Exceeded (0x07).

If the Host attempts to set CIS parameters that exceed the maximum supported connections in the Controller, the Controller shall return the error code Connection Limit Exceeded (0x09). If the Host sets, in the PHY_C_To_P[i] or PHY_P_To_C[i] parameters, a bit for a PHY that the Controller does not support, including a bit that is reserved for future use, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Controller does not support asymmetric PHYs and the Host sets PHY_C_To_P[i] to a different value than PHY_P_To_C[i], the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Host specifies an invalid combination of CIS parameters, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11).

Parameters

CIG_ID

Used to identify the CIG. Values:

- 0x00 ... 0xEF

SDU_Interval_C_To_P

The interval, in microseconds, of periodic SDUs. Values:

- 0x0000FF ... 0x0FFFFF

SDU_Interval_P_To_C

The interval, in microseconds, of periodic SDUs. Values:

- 0x0000FF ... 0x0FFFFF

Worst_Case_SCA

Worst-case sleep clock accuracy of all peripherals. Values:

- 0x00: 251 ppm to 500 ppm
- 0x01: 151 ppm to 250 ppm
- 0x02: 101 ppm to 150 ppm
- 0x03: 76 ppm to 100 ppm
- 0x04: 51 ppm to 75 ppm
- 0x05: 31 ppm to 50 ppm
- 0x06: 21 ppm to 30 ppm
- 0x07: 0 ppm to 20 ppm

Packing

Preferred method of arranging subevents of multiple CISes. Values:

- 0x00: Sequential
- 0x01: Interleaved

Framing

Format of the CIS Data PDUs of the specified CISes. Values:

- 0x00: Unframed
- 0x01: Framed

Max_Transport_Latency_C_To_P

Maximum transport latency, in milliseconds, from the Central's Controller to the Peripheral's Controller. Values:

- 0x0005 ... 0x0FA0

Max_Transport_Latency_P_To_C

Maximum transport latency, in milliseconds, from the Peripheral's Controller to the Central's Controller. Values:

- 0x0005 ... 0x0FA0

CIS_Count

Total number of CIS configurations in the CIG being added or modified. Values:

- 0x00 ... 0x1F

CIS_Param

See CIS_Param_t

`[out]` **Connection_Handle**

Connection handle of the CIS in the CIG.

Return values:

- *Value* indicating success or error code.

2.1.69 hci_le_set_connection_cte_receive_parameters

```
tBleStatus hci_le_set_connection_cte_receive_parameters ( uint16_t Connection_Handle,
                                                           uint8_t  Sampling_Enable,
                                                           uint8_t  Slot_Durations,
                                                           uint8_t  Switching_Pattern_Length,
                                                           uint8_t  Antenna_IDs[]
                                                          )
```

The HCI_LE_Set_Connection_CTE_Receive_Parameters command is used to enable or disable sampling received Constant Tone Extension fields on the connection identified by the Connection_Handle parameter and to set the antenna switching pattern and switching and sampling slot durations to be used. If the Sampling_Enable parameter is set to 0x01, the Controller shall sample Constant Tone Extensions on the specified connection and report the samples to the Host. If it is set to 0x00, the Controller shall cease sampling on the specified connection; the remaining parameters shall be ignored. If Slot_Durations is set to 0x01 and the Controller does not support 1 microsecond switching and sampling, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11).

The Slot_Durations, Switching_Pattern_Length, and Antenna_IDs parameters are only used when receiving an AoA Constant Tone Extension and do not affect the reception of an AoD Constant Tone Extension. If Switching_Pattern_Length is greater than the maximum length of switching pattern supported by the Controller, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Controller determines that any of the Antenna_IDs[i] values do not identify an antenna in the device's antenna array, it shall return the error code Unsupported Feature or Parameter Value (0x11). Note: Some Controllers may be unable to determine which values do or do not identify an antenna.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Sampling_Enable

Values:

- 0x00: Connection IQ sampling is disabled (default)
- 0x01: Connection IQ sampling is enabled

Slot_Durations

Sampling rate used by the Controller. Values:

- 0x01: CTE_SLOT_1us
- 0x02: CTE_SLOT_2us

Switching_Pattern_Length

Values:

- 0x02 ... 0x4B: The number of Antenna IDs in the pattern.

Antenna_IDs

List of Antenna IDs in the pattern.

Return values:

- *Value* indicating success or error code.

2.1.70 hci_le_set_connection_cte_transmit_parameters

```
tBleStatus hci_le_set_connection_cte_transmit_parameters ( uint16_t Connection_Handle,
                                                           uint8_t  CTE_Type,
                                                           uint8_t  Switching_Pattern_Length,
                                                           uint8_t  Antenna_IDs[]
                                                          )
```

The HCI_LE_Set_Connection_CTE_Transmit_Parameters command is used to set the antenna switching pattern and permitted Constant Tone Extension types used for transmitting Constant Tone Extensions requested by the peer device on the connection identified by the Connection_Handle parameter. If the Host issues this command when Constant Tone Extension responses have been enabled on the connection, the Controller shall return the error code Command Disallowed (0x0C). If the CTE_Types parameter has a bit set for a type of Constant Tone Extension that the Controller does not support, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11).

The Switching_Pattern_Length and Antenna_IDs[i] parameters are only used when transmitting an AoD Constant Tone Extension and shall be ignored when CTE_Types does not have a bit set for an AoD Constant Tone Extension; they do not affect the transmission of an AoA Constant Tone Extension. If Switching_Pattern_Length is greater than the maximum length of switching pattern supported by the Controller, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Controller determines that any of the Antenna_IDs[i] values do not identify an antenna in the device's antenna array, it shall return the error code Unsupported Feature or Parameter Value (0x11).

Note: Some Controllers may be unable to determine which values do or do not identify an antenna.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CTE_Type

Flags:

- 0x01: Allow AoA Constant Tone Extension Response
- 0x02: Allow AoD Constant Tone Extension Response with 1 microsecond slots
- 0x04: Allow AoD Constant Tone Extension Response with 2 microseconds slots

Switching_Pattern_Length

Values:

- 0x02 ... 0x4B: The number of Antenna IDs in the pattern.

Antenna_IDs

List of Antenna IDs in the pattern.

Return values:

- *Value* indicating success or error code.

2.1.71 hci_le_set_connectionless_cte_transmit_enable

```
tBleStatus hci_le_set_connectionless_cte_transmit_enable     ( uint8_t Advertising_Handle,
                                                                uint8_t CTE_Enable
                                                               )
```

The HCI_LE_Set_Connectionless_CTE_Transmit_Enable command is used to request that the Controller enables or disables the use of Constant Tone Extensions in any periodic advertising on the advertising set identified by Advertising_Handle. In order to start sending periodic advertisements containing a Constant Tone Extension, the Host must also enable periodic advertising using the HCI_LE_Set_Periodic_Advertising_Enable command (see Section 7.8.63).

Note: Periodic advertising can only be enabled when advertising is enabled on the same advertising set, but can continue after advertising has been disabled.

If the Host issues this command before it has issued the HCI_LE_Set_Periodic_Advertising_Parameters command (see Section 7.8.61) for the advertising set, the Controller shall return the error code Command Disallowed (0x0C). Once enabled, the Controller shall continue advertising with Constant Tone Extensions until either one of the following occurs:

- The Host issues an HCI_LE_Set_Connectionless_CTE_Transmit_Enable command with CTE_Enable set to 0x00 (disabling Constant Tone Extensions but allowing periodic advertising to continue).
- The Host issues an HCI_LE_Set_Periodic_Advertising_Enable command (see Section 7.8.63) with Enable set to 0x00 (disabling periodic advertising).

If periodic advertising is re-enabled, it shall continue to contain Constant Tone Extensions. If the Host issues this command before it has issued the HCI_LE_Set_Connectionless_CTE_Transmit_Parameters command for the advertising set, the Controller shall return the error code Command Disallowed (0x0C). If the periodic advertising is on a PHY that does not allow Constant Tone Extensions, the Controller shall return the error code Command Disallowed (0x0C).

Parameters

Advertising_Handle

Identifier for the advertising set in which Constant Tone Extension is being enabled or disabled. Values:

- 0x00 ... 0xEF

CTE_Enable

It enables or disables the use of Constant Tone Extensions. Values:

- 0x00: Advertising with Constant Tone Extension is disabled (default)
- 0x01: Advertising with Constant Tone Extension is enabled

Return values:

- *Value* indicating success or error code.

2.1.72 hci_le_set_connectionless_cte_transmit_parameters

```
tBleStatus hci_le_set_connectionless_cte_transmit_parameters  ( uint8_t Advertising_Handle,
                                                                uint8_t CTE_Length,
                                                                uint8_t CTE_Type,
                                                                uint8_t CTE_Count,
                                                                uint8_t Switching_Pattern_Length,
                                                                uint8_t Antenna_IDs[]
                                                              )
```

The HCI_LE_Set_Connectionless_CTE_Transmit_Parameters command is used to set the type, length, and antenna switching pattern for the transmission of Constant Tone Extensions in any periodic advertising on the advertising set identified by the Advertising_Handle parameter. The CTE_Count parameter specifies how many packets with a Constant Tone Extension are to be transmitted in each periodic advertising event. If the number of packets that would otherwise be transmitted is less than this, the Controller shall transmit sufficient AUX_CHAIN_IND PDUs with no AdvData to make up the number. However, if a change in circumstances since this command was issued means that the Controller can no longer schedule all of these packets, it should transmit as many as possible.

If the Host issues this command when Constant Tone Extensions have been enabled in the advertising set, the Controller shall return the error code Command Disallowed (0x0C). The Switching_Pattern_Length and Antenna_IDs[i] parameters are only used when transmitting an AoD Constant Tone Extension and shall be ignored if CTE_Type specifies an AoA Constant Tone Extension.

If the CTE_Length parameter is greater than the maximum length of Constant Tone Extension supported, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Host requests a type of Constant Tone Extension that the Controller does not support, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Controller is unable to schedule CTE_Count packets in each event, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11).

If the advertising set corresponding to the Advertising_Handle parameter does not exist, the Controller shall return the error code Unknown Advertising Identifier (0x42). If Switching_Pattern_Length is greater than the maximum length of switching pattern supported by the Controller (see Section 7.8.87), the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Controller determines that any of the Antenna_IDs[i] values do not identify an antenna in the device's antenna array, it shall return the error code Unsupported Feature or Parameter Value (0x11).

Note: Some Controllers may be unable to determine which values do or do not identify an antenna.

Parameters

Advertising_Handle

Identifier for the advertising set in which Constant Tone Extension is being enabled or disabled. Values:

- 0x00 ... 0xEF

CTE_Length

Values:

- 0x02 ... 0x14: Constant Tone Extension length in 8 microseconds units

CTE_Type

Values:

- 0x00: AoA Constant Tone Extension
- 0x01: AoD Constant Tone Extension with 1 microsecond slots
- 0x02: AoD Constant Tone Extension with 2 microseconds slots

CTE_Count

The CTE_Count parameter specifies how many packets with a Constant Tone Extension are to be transmitted in each periodic advertising event. If the number of packets that would otherwise be transmitted is less than this, the Controller shall transmit sufficient AUX_CHAIN_IND PDUs with no AdvData to make up the number. However, if a change in circumstances since this command was issued means that the Controller can no longer schedule all of these packets, it should transmit as many as possible. Values:

- 0x01 ... 0x10: The number of Constant Tone Extensions to transmit in each periodic advertising interval

Switching_Pattern_Length

Values:

- 0x02 ... 0x4B: The number of Antenna IDs in the pattern.

Antenna_IDs

List of Antenna IDs in the pattern.

Return values:

- *Value* indicating success or error code.

2.1.73 hci_le_set_connectionless_iq_sampling_enable

```
tBleStatus hci_le_set_connectionless_iq_sampling_enable ( uint16_t Sync_Handle,
                                                          uint8_t  Sampling_Enable,
                                                          uint8_t  Slot_Durations,
                                                          uint8_t  Max_Sampled_CTEs,
                                                          uint8_t  Switching_Pattern_Length,
                                                          uint8_t  Antenna_IDs[]
                                                        )
```

The HCI_LE_Set_Connectionless_IQ_Sampling_Enable command is used to request that the Controller enables or disables capturing IQ samples from the Constant Tone Extension of periodic advertising packets in the periodic advertising train identified by the Sync_Handle parameter. If that periodic advertising train does not exist, then the Controller shall return the error code Unknown Advertising Identifier (0x42). The Max_Sampled_CTEs parameter specifies the maximum number of Constant Tone Extensions in each periodic advertising event that the Controller should collect and report IQ samples from. The Controller should sample all Constant Tone Extensions up to this number.

If the Sampling_Enable parameter is set to 0x01 (sampling is enabled), the Controller starts attempting to capture IQ samples from the periodic advertisements. Once sampling has been enabled, the Controller shall continue taking IQ samples until the Host issues an HCI_LE_Set_Connectionless_IQ_Enable command with Sampling_Enable set to 0x00 (sampling is disabled) or synchronization with the periodic advertising train is lost. If Sampling_Enable is set to 0x00, Slot_Durations, Max_Sampled_CTEs, Switching_Pattern_Length, and Antenna_IDs shall be ignored.

The command is also used to set the antenna switching pattern and switching and sampling slot durations to be used while receiving the Constant Tone Extension. If Slot_Durations is set to 0x01 and the Controller does not support 1 microsecond switching and sampling, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). The Slot_Durations, Switching_Pattern_Length, and Antenna_IDs parameters are only used when receiving an AoA Constant Tone Extension and do not affect the reception of an AoD Constant Tone Extension.

If Switching_Pattern_Length is greater than the maximum length of switching pattern supported by the Controller, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Controller determines that any of the Antenna_IDs[i] values do not identify an antenna in the device's antenna array, it shall return the error code Unsupported Feature or Parameter Value (0x11). Note: Some Controllers may be unable to determine which values do or do not identify an antenna. If Sampling_Enable is set to 0x01 and the periodic advertising is on a PHY that does not allow Constant Tone Extensions, the Controller shall return the error code Command Disallowed (0x0C).

Parameters

Sync_Handle

Sync handle that identifies the synchronization information about the periodic advertising train. Values:

- 0x0000 ... 0x0EFF

Sampling_Enable

If the Sampling_Enable parameter is set to 0x01 (sampling is enabled), the Controller starts attempting to capture IQ samples from the periodic advertisements. Values:

- 0x00: DISABLE
- 0x01: ENABLE

Slot_Durations

Sampling rate used by the Controller. Values:

- 0x01: CTE_SLOT_1us
- 0x02: CTE_SLOT_2us

Max_Sampled_CTEs

It specifies the maximum number of Constant Tone Extensions in each periodic advertising event that the Controller should collect and report IQ samples from. The Controller should sample all Constant Tone Extensions up to this number. Values:

- 0x00: REPORT_ALL_CTES
- 0x01 ... 0x10

Switching_Pattern_Length

The number of Antenna IDs in the pattern. Values:

- 0x02 ... 0x4B

Antenna_IDs

List of Antenna IDs in the pattern.

Return values:

- *Value* indicating success or error code.

2.1.74 hci_le_set_data_length

```
tBleStatus hci_le_set_data_length     ( uint16_t Connection_Handle,
                                        uint16_t TxOctets,
                                        uint16_t TxTime
                                       )
```

The LE_Set_Data_Length command allows the Host to suggest maximum transmission packet size and maximum packet transmission time (connMaxTxOctets and connMaxTxTime - see [Vol 6] Part B, Section 4.5.10) to be used for a given connection. The Controller may use smaller or larger values based on local information.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

TxOctets

Preferred maximum number of payload octets that the local Controller should include in a single Link Layer Data Channel PDU. Range 0x001B-0x00FB (0x0000 - 0x001A and 0x00FC - 0xFFFF reserved for future use). Default: 27 bytes. Values:

- 0x001B ... 0x00FB

TxTime

Preferred maximum number of microseconds that the local Controller should use to transmit a single Link Layer Data Channel PDU. Range 0x0148-0x0848 (0x0000 - 0x0147 and 0x0849 - 0xFFFF reserved for future use). Default: 328 bytes. Values:

- 0x0148 ... 0x0848

Return values:

- *Value* indicating success or error code.

2.1.75 hci_le_set_data_related_address_changes

```
tBleStatus hci_le_set_data_related_address_changes     ( uint8_t Advertising_Handle,
                                                         uint8_t Change_Reasons
                                                        )
```

The HCI_LE_Set_Data_Related_Address_Changes command specifies circumstances when the Controller shall refresh any Resolvable Private Address used by the advertising set identified by the Advertising_Handle parameter, whether or not the address timeout period has been reached. This command may be used while advertising is enabled. The Change_Reasons parameter specifies the reason(s) for refreshing addresses. The default when an advertising set is created, or if legacy advertising commands (see Section 3.1.1) are used, is for all bits to be clear. If extended advertising commands (see Section 3.1.1) are being used and the advertising set corresponding to the Advertising_Handle parameter does not exist, or if no command specified in Table 3.2 has been used, then the Controller shall return the error code Unknown Advertising Identifier (0x42). If legacy advertising commands are being used, the Controller shall ignore the Advertising_Handle parameter.

Parameters

Advertising_Handle

Used to identify an advertising set. Values:

- 0x00 ... 0xEF

Change_Reasons

Bitmap associated with the reasons to refresh the Resolvable Private Addresses used by the advertising set. If bit 0 is set, change the address whenever the advertising data changes. If bit 1 is set, change the address whenever the scan response data changes. Flags:

- 0x01: ADV_DATA_CHANGES
- 0x02: SCAN_RESP_DATA_CHANGES

Return values:

- *Value* indicating success or error code.

2.1.76 hci_le_set_default_periodic_advertising_sync_transfer_parameters

```
tBleStatus hci_le_set_default_periodic_advertising_sync_transfer_parameters  ( uint8_t  Mode,
                                                                               uint16_t Skip,
                                                                               uint16_t Sync_Timeout,
                                                                               uint8_t  CTE_Type
                                                                              )
```

The HCI_LE_Set_Default_Periodic_Advertising_Sync_Transfer_Parameters command is used to specify the initial value for the mode, skip, timeout, and Constant Tone Extension type (set by the HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters command; see Section 7.8.91) to be used for all subsequent connections over the LE transport. The Mode parameter specifies the initial action to be taken. If Mode is 0x00, the Controller ignores the information. Otherwise, it notifies the Host and synchronize to the periodic advertising. Mode also specifies whether periodic advertising reports are initially enabled or disabled.

The Skip parameter specifies the number of consecutive periodic advertising packets that the receiver may skip after successfully receiving a periodic advertising packet. The Sync_Timeout parameter specifies the maximum permitted time between successful receives. If this time is exceeded, synchronization is lost. The CTE_Type parameter specifies whether to only synchronize to periodic advertising with certain types of Constant Tone Extension. If the periodic advertiser changes the type of the Constant Tone Extension after the Controller has synchronized with the periodic advertising, it shall remain synchronized.

Note: *A value of 0 (that is, all bits clear) indicates that the presence or absence of a Constant Tone Extension is irrelevant. This command does not affect any existing connection.*

Parameters

Mode

The action to be taken when periodic advertising synchronization information is received. If 0, no attempt is made to synchronize to the periodic advertising and no HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. If 1, an HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events are disabled. If 2, an HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events is enabled with duplicate filtering disabled. If 3, an HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events are enabled with duplicate filtering enabled. Values:

- 0x00: NO_SYNC
- 0x01: REPORTS_DISABLED
- 0x02: REPORTS_ENABLED
- 0x03: REPORTS_ENABLED_WITH_DUPLICATE_FILTERING

Skip

The number of periodic advertising packets that can be skipped after a successful receive. Values:

- 0x0000 ... 0x01F3

Sync_Timeout

Synchronization timeout for the periodic advertising train. Time = N*10 ms. Values:

- 0x000A (100 ms) ... 0x4000 (163840 ms)

CTE_Type

It specifies whether to only synchronize to periodic advertising with certain types of Constant Tone Extension. If bit 0 is set: do not sync to packets with an AoA Constant Tone Extension. If bit 1 is set: Do not sync to packets with an AoD Constant Tone Extension with 1 us slots. If bit 2 is set: Do not sync to packets with an AoD Constant Tone Extension with 2 us slots. If bit 3 is set: Do not sync to packets without a Constant Tone Extension. Flags:

- 0x01: DO_NOT_SYNC_WITH_AOA
- 0x02: DO_NOT_SYNC_WITH_AOD_1US
- 0x04: DO_NOT_SYNC_WITH_AOD_2US
- 0x10: DO_NOT_SYNC_WITHOUT_CTE

Return values:

- *Value* indicating success or error code.

2.1.77 hci_le_set_default_phy

```
tBleStatus hci_le_set_default_phy      ( uint8_t ALL_PHYS,
                                          uint8_t TX_PHYS,
                                          uint8_t RX_PHYS
                                        )
```

The LE_Set_Default_PHY command allows the Host to specify its preferred values for the transmitter PHY and receiver PHY to be used for all subsequent connections over the LE transport.

Parameters

ALL_PHYS

The ALL_PHYS parameter is a bit field that allows the Host to specify, for each direction, whether it has no preference among the PHYs that the Controller supports in a given direction or whether it has specified particular PHYs that it prefers in the TX_PHYS or RX_PHYS parameter. Bits: 0: The Host has no preference among the transmitter PHYs supported by the Controller 1: The Host has no preference among the receiver PHYs supported by the Controller Flags:

- 0x01: No preference for TX
- 0x02: No preference for RX

TX_PHYS

The TX_PHYS parameter is a bit field that indicates the transmitter PHYs that the Host prefers the Controller to use. If the ALL_PHYS parameter specifies that the Host has no preference, the TX_PHYS parameter is ignored; otherwise at least one bit shall be set to 1. Bits: 0: The Host prefers to use the LE 1M transmitter PHY (possibly among others) 1: The Host prefers to use the LE 2M transmitter PHY (possibly among others) 2: The Host prefers to use the LE Coded transmitter PHY (possibly among others) 3-7: Reserved for future use Flags:

- 0x01: LE_1M_PHY_BIT
- 0x02: LE_2M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

RX_PHYS

The RX_PHYS parameter is a bit field that indicates the receiver PHYs that the Host prefers the Controller to use. If the ALL_PHYS parameter specifies that the Host has no preference, the RX_PHYS parameter is ignored; otherwise at least one bit shall be set to 1. Bits: 0: The Host prefers to use the LE 1M receiver PHY (possibly among others) 1: The Host prefers to use the LE 2M receiver PHY (possibly among others) 2: The Host prefers to use the LE Coded receiver PHY (possibly among others) 3-7: Reserved for future use Flags:

- 0x01: LE_1M_PHY_BIT
- 0x02: LE_2M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

Return values:

- *Value* indicating success or error code.

2.1.78 hci_le_set_default_subrate

```
tBleStatus hci_le_set_default_subrate     ( uint16_t Subrate_Min,
                                              uint16_t Subrate_Max,
                                              uint16_t Max_Latency,
                                              uint16_t Continuation_Number,
                                              uint16_t Supervision_Timeout
                                             )
```

The HCI_LE_Set_Default_Subrate command is used by the Host to set the initial values for the acceptable parameters for subrating requests, as defined by the HCI_LE Subrate_Request command, for all future ACL connections where the Controller is the Central. This command does not affect any existing connection. The parameters have the same meanings and restrictions as those in the HCI_LE_Subrate_Request command.

Parameters

Subrate_Min

Minimum subrate factor allowed in requests by a Peripheral. Values:

- 0x0001 ... 0x01F4

Subrate_Max

Maximum subrate factor allowed in requests by a Peripheral. Values:

- 0x0001 ... 0x01F4

Max_Latency

Maximum Peripheral latency allowed in requests by a Peripheral, in units of subrated connection intervals. Values:

- 0x0000 ... 0x01F3

Continuation_Number

Minimum number of underlying connection events to remain active after a packet containing a Link Layer PDU with a non- zero Length field is sent or received in requests by a Peripheral. Values:

- 0x0000 ... 0x01F3

Supervision_Timeout

Maximum supervision timeout allowed in requests by a Peripheral. Time = N x 10 ms. Values:

- 0x000A (100 ms) ... 0x0C80 (32000 ms)

Return values:

- *Value* indicating success or error code.

2.1.79 hci_le_set_event_mask

```
tBleStatus hci_le_set_event_mask ( uint8_t LE_Event_Mask[8] )
```

The LE_Set_Event_Mask command is used to control which LE events are generated by the HCI for the Host. If the bit in the LE_Event_Mask is set to a one, then the event associated with that bit is enabled. The Host has to deal with each event that is generated by an LE Controller. The event mask allows the Host to control which events interrupt it. For LE events to be generated, the LE Meta-Event bit in the Event_Mask shall also be set. If that bit is not set, then LE events shall not be generated, regardless of how the LE_Event_Mask is set. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.1.)

Parameters

LE_Event_Mask

LE event mask. Default: 0x000000000000001F.

Flags:

- 0x0000 0000 0000 0000: No LE events specified
- 0x0000 0000 0000 0001: LE Connection Complete Event
- 0x0000 0000 0000 0002: LE Advertising Report Event
- 0x0000 0000 0000 0004: LE Connection Update Complete Event
- 0x0000 0000 0000 0008: LE Read Remote Used Features Complete Event
- 0x0000 0000 0000 0010: LE Long Term Key Request Event
- 0x0000 0000 0000 0020: LE Remote Connection Parameter Request Event
- 0x0000 0000 0000 0040: LE Data Length Change Event
- 0x0000 0000 0000 0080: LE Read Local P-256 Public Key Complete Event
- 0x0000 0000 0000 0100: LE Generate DHKey Complete Event
- 0x0000 0000 0000 0200: LE Enhanced Connection Complete Event
- 0x0000 0000 0000 0400: LE Directed Advertising Report Event
- 0x0000 0000 0000 0800: LE PHY Update Complete event
- 0x0000 0000 0000 1000: LE Extended Advertising Report event
- 0x0000 0000 0000 2000: LE Periodic Advertising Sync Established event
- 0x0000 0000 0000 4000: LE Periodic Advertising Report event
- 0x0000 0000 0000 8000: LE Periodic Advertising Sync Lost event
- 0x0000 0000 0001 0000: LE Scan Timeout event
- 0x0000 0000 0002 0000: LE Advertising Set Terminated event
- 0x0000 0000 0004 0000: LE Scan Request Received event
- 0x0000 0000 0008 0000: LE Channel Selection Algorithm event
- 0x0000 0000 0010 0000: LE Connectionless IQ Report event
- 0x0000 0000 0020 0000: LE Connection IQ Report event
- 0x0000 0000 0040 0000: LE CTE Request Failed event
- 0x0000 0000 0080 0000: LE Periodic Advertising Sync Transfer Received event
- 0x0000 0000 0100 0000: LE CIS Established event
- 0x0000 0000 0200 0000: LE CIS Request event
- 0x0000 0000 0400 0000: LE Create BIG Complete event
- 0x0000 0000 0800 0000: LE Terminate BIG Complete event
- 0x0000 0000 1000 0000: LE BIG Sync Established event
- 0x0000 0000 2000 0000: LE BIG Sync Lost event
- 0x0000 0000 4000 0000: LE Request Peer SCA Complete event
- 0x0000 0000 8000 0000: LE Path Loss Threshold event
- 0x0000 0001 0000 0000: LE Transmit Power Reporting event
- 0x0000 0002 0000 0000: LE BIGInfo Advertising Report event
- 0x0000 0004 0000 0000: LE Subrate Change event
- 0x0000 0008 0000 0000: LE Periodic Advertising Sync Established event [v2]
- 0x0000 0010 0000 0000: LE Periodic Advertising Report event [v2]
- 0x0000 0020 0000 0000: LE Periodic Advertising Sync Transfer Received event [v2]
- 0x0000 0040 0000 0000: LE Periodic Advertising Subevent Data Request event
- 0x0000 0080 0000 0000: LE Periodic Advertising Response Report event
- 0x0000 0100 0000 0000: LE Enhanced Connection Complete event [v2]

Return values:

- *Value* indicating success or error code.

2.1.80 hci_le_set_extended_advertising_enable

```
tBleStatus hci_le_set_extended_advertising_enable   ( uint8_t Enable,
                                                      uint8_t Number_of_Sets,
                                                      Advertising_Set_Parameters_t
                                                      Advertising_Set_Parameters[]
                                                     )
```

The LE_Set_Extended_Advertising_Enable command is used to request the Controller to enable or disable one or more advertising sets using the advertising sets identified by the Advertising_Handle[i] parameter. The Controller manages the timing of advertisements in accordance with the advertising parameters given in the LE_Set_Extended_Advertising_Parameters command.

Parameters

Enable

Enables or disables one or more advertising sets using the advertising sets identified by the Advertising_Handle[i] parameter. Values:

- 0x00: Disable
- 0x01: Enable

Number_of_Sets

The Number_of_Sets parameter is the number of advertising sets contained in the parameter arrays. Values:

- 0x00: Disable all advertising sets
- 0x01 ... 0x3F: Number of advertising sets to enable or disable

Advertising_Set_Parameters

See Advertising_Set_Parameters_t.

Return values:

- *Value* indicating success or error code.

2.1.81 hci_le_set_extended_advertising_parameters

```
tBleStatus hci_le_set_extended_advertising_parameters ( uint8_t  Advertising_Handle,
                                                        uint16_t Advertising_Event_Properties,
                                                        uint8_t  Primary_Advertising_Interval_Min[3],
                                                        uint8_t  Primary_Advertising_Interval_Max[3],
                                                        uint8_t  Primary_Advertising_Channel_Map,
                                                        uint8_t  Own_Address_Type,
                                                        uint8_t  Peer_Address_Type,
                                                        uint8_t  Peer_Address[6],
                                                        uint8_t  Advertising_Filter_Policy,
                                                        int8_t   Advertising_Tx_Power,
                                                        uint8_t  Primary_Advertising_PHY,
                                                        uint8_t  Secondary_Advertising_Max_Skip,
                                                        uint8_t  Secondary_Advertising_PHY,
                                                        uint8_t  Advertising_SID,
                                                        uint8_t  Scan_Request_Notification_Enable,
                                                        int8_t   * Selected_Tx_Power
                                                       )
```

The LE_Set_Extended_Advertising_Parameters command is used by the Host to set the advertising parameters. The Advertising_Handle parameter identifies the advertising set whose parameters are being configured. The Advertising_Event_Properties parameter describes the type of advertising event that is being configured and its basic properties. The type shall be one supported by the Controller.

Parameters

Advertising_Handle

The Advertising_Handle parameter identifies the advertising set whose parameters are being configured. Values:

- 0x00 ... 0xEF

Advertising_Event_Properties

The Advertising_Event_Properties parameter describes the type of advertising event that is being configured and its basic properties. The type shall be one supported by the Controller. Bits: 0 Connectable advertising 1 Scannable advertising 2 Directed advertising 3 High Duty Cycle Directed Connectable advertising (<= 3.75 ms Advertising Interval) 4 Use legacy advertising PDUs 5 Omit advertiser's address from all PDUs ("anonymous advertising") 6 Include TxPower in the extended header of the advertising PDU Flags:

- 0x0001: Connectable
- 0x0002: Scannable
- 0x0004: Directed
- 0x0008: HDC Directed Connectable
- 0x0010: Legacy
- 0x0020: Anonymous
- 0x0040: TxPower in ext header

Primary_Advertising_Interval_Min

Minimum advertising interval for undirected and low duty cycle directed advertising. Time = N * 0.625 ms. Time Range: 20 ms to 10,485.759375 s. Values:

- 0x000020 (20.000 ms) ... 0xFFFFFF (10485759.375 ms)

Primary_Advertising_Interval_Max

Maximum advertising interval for undirected and low duty cycle directed advertising. Time = N * 0.625 ms. Time Range: 20 ms to 10,485.759375 s. Values:

- 0x000020 (20.000 ms) ... 0xFFFFFF (10485759.375 ms)

Primary_Advertising_Channel_Map

The Primary_Advertising_Channel_Map is a bit field that indicates the advertising channels that shall be used when transmitting advertising packets. At least one channel bit shall be set in the Primary_Advertising_Channel_Map parameter. Flags:

- 0x01: CH_37
- 0x02: CH_38
- 0x04: CH_39

Own_Address_Type

The Own_Address_Type parameter specifies the type of address being used in the advertising packets. For random addresses, the address is specified by the LE_Set_Advertising_Set_Random_Address command. 0x00 Public Device Address 0x01 Random Device Address 0x02 Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the public address. 0x03 Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the random address from LE_Set_Advertising_Set_Random_Address. All other values Reserved for future use. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Resolvable Private Address/Public Address
- 0x03: Resolvable Private Address/Random Address

Peer_Address_Type

Peer address type. Values:

- 0x00: Public Device Address or Public Identity Address
- 0x01: Random Device Address or Random (static) Identity Address

Peer_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected.

Advertising_Filter_Policy

This parameter is ignored when directed advertising is enabled. Values:

- 0x00: HCI_ADV_FILTER_NONE. Process scan and connection requests from all devices (that is, the Filter Accept List is not in use).
- 0x01: HCI_ADV_FILTER_ACCEPT_LIST_SCAN. Process connection requests from all devices and scan requests only from devices that are in the Filter Accept List.
- 0x02: HCI_ADV_FILTER_ACCEPT_LIST_CONNECT. Process scan requests from all devices and connection requests only from devices that are in the Filter Accept List.
- 0x03: HCI_ADV_FILTER_ACCEPT_LIST_SCAN_CONNECT. Process scan and connection requests only from devices in the Filter Accept List.

All other values are reserved for future use

Advertising_Tx_Power

Units: dBm The Advertising_Tx_Power parameter indicates the maximum power level at which the advertising packets are to be transmitted on the advertising channels. The Controller shall choose a power level lower than or equal to the one specified by the Host. Values:

- -127 ... 126
- 127: No preference

Primary_Advertising_PHY

The Primary_Advertising_PHY parameter indicates the PHY on which the advertising packets are transmitted on the primary advertising channel. If legacy advertising PDUs are being used, the Primary_Advertising_PHY shall indicate the LE 1M PHY. Values:

- 0x01: LE_1M_PHY
- 0x03: LE_CODED_PHY

Secondary_Advertising_Max_Skip

The Secondary_Advertising_Max_Skip parameter is the maximum number of advertising events that can be skipped before the AUX_ADV_IND can be sent. 0x00 AUX_ADV_IND shall be sent prior to the next advertising event 0x01-0xFF Maximum advertising events the Controller can skip before sending the AUX_ADV_IND packets on the secondary advertising channel. Values:

- 0x00 ... 0xFF

Secondary_Advertising_PHY

The Secondary_Advertising_PHY parameter indicates the PHY on which the advertising packets are transmitted on the secondary advertising channel. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY

Advertising_SID

The Advertising_SID parameter specifies the value to be transmitted in the Advertising SID subfield of the ADI field of the Extended Header of those advertising channel PDUs that have an ADI field. If the advertising set only uses PDUs that do not contain an ADI field, Advertising_SID is ignored. Values:

- 0x00 ... 0x0F

Scan_Request_Notification_Enable

The Scan_Request_Notification_Enable parameter indicates whether the Controller shall send notifications upon the receipt of a scan request PDU that is in response to an advertisement from the specified advertising set that contains its device address and is from a scanner that is allowed by the advertising filter policy. Values:

- 0x00: Scan request notifications disabled
- 0x01: Scan request notifications enabled

`[out]` **Selected_Tx_Power**

Units: dBm. The Selected_Tx_Power return parameter indicates the transmit power selected by the Controller. The Controller shall not change the transmit power for this advertising set without being directed to by the Host. Values:

- -127 ... 126

Return values:

- *Value* indicating success or error code.

2.1.82 hci_le_set_extended_advertising_parameters_v2

```
tBleStatus hci_le_set_extended_advertising_parameters_v2 ( uint8_t  Advertising_Handle,
                                                           uint16_t Advertising_Event_Properties,
                                                           uint8_t  Primary_Advertising_Interval_Min[3],
                                                           uint8_t  Primary_Advertising_Interval_Max[3],
                                                           uint8_t  Primary_Advertising_Channel_Map,
                                                           uint8_t  Own_Address_Type,
                                                           uint8_t  Peer_Address_Type,
                                                           uint8_t  Peer_Address[6],
                                                           uint8_t  Advertising_Filter_Policy,
                                                           int8_t   Advertising_Tx_Power,
                                                           uint8_t  Primary_Advertising_PHY,
                                                           uint8_t  Secondary_Advertising_Max_Skip,
                                                           uint8_t  Secondary_Advertising_PHY,
                                                           uint8_t  Advertising_SID,
                                                           uint8_t  Scan_Request_Notification_Enable,
                                                           uint8_t  Primary_Advertising_PHY_Options,
                                                           uint8_t  Secondary_Advertising_PHY_Options,
                                                           int8_t   * Selected_Tx_Power
                                                          )
```

The LE_Set_Extended_Advertising_Parameters command is used by the Host to set the advertising parameters. The Advertising_Handle parameter identifies the advertising set whose parameters are being configured. The Advertising_Event_Properties parameter describes the type of advertising event that is being configured and its basic properties. The type shall be one supported by the Controller.

Parameters

Advertising_Handle

The Advertising_Handle parameter identifies the advertising set whose parameters are being configured. Values:

- 0x00 ... 0xEF

Advertising_Event_Properties

The Advertising_Event_Properties parameter describes the type of advertising event that is being configured and its basic properties. The type shall be one supported by the Controller. Bits: 0 Connectable advertising 1 Scannable advertising 2 Directed advertising 3 High Duty Cycle Directed Connectable advertising (<= 3.75 ms Advertising Interval) 4 Use legacy advertising PDUs 5 Omit advertiser's address from all PDUs ("anonymous advertising") 6 Include TxPower in the extended header of the advertising PDU Flags:

- 0x0001: Connectable
- 0x0002: Scannable
- 0x0004: Directed
- 0x0008: HDC Directed Connectable
- 0x0010: Legacy
- 0x0020: Anonymous
- 0x0040: TxPower in ext header

Primary_Advertising_Interval_Min

Minimum advertising interval for undirected and low duty cycle directed advertising. Time = N * 0.625 ms. Time Range: 20 ms to 10,485.759375 s. Values:

- 0x0000 0020 (20.000 ms) ... 0x00FF FFFF (10485759.375 ms)

Primary_Advertising_Interval_Max

Maximum advertising interval for undirected and low duty cycle directed advertising. Time = N * 0.625 ms. Time Range: 20 ms to 10,485.759375 s. Values:

- 0x0000 0020 (20.000 ms) ... 0x00FF FFFF (10485759.375 ms)

Primary_Advertising_Channel_Map

The Primary_Advertising_Channel_Map is a bit field that indicates the advertising channels that shall be used when transmitting advertising packets. At least one channel bit shall be set in the Primary_Advertising_Channel_Map parameter. Flags:

- 0x01: CH_37
- 0x02: CH_38
- 0x04: CH_39

Own_Address_Type

The Own_Address_Type parameter specifies the type of address being used in the advertising packets. For random addresses, the address is specified by the LE_Set_Advertising_Set_Random_Address command. 0x00 Public Device Address 0x01 Random Device Address 0x02 Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the public address. 0x03 Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the random address from LE_Set_Advertising_Set_Random_Address. All other values Reserved for future use. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Resolvable Private Address/Public Address
- 0x03: Resolvable Private Address/Random Address

Peer_Address_Type

Peer address type. Values:

- 0x00: Public Device Address or Public Identity Address
- 0x01: Random Device Address or Random (static) Identity Address

Peer_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected.

Advertising_Filter_Policy

This parameter is ignored when directed advertising is enabled. Values:

- 0x00: HCI_ADV_FILTER_NONE. Process scan and connection requests from all devices (that is, the Filter Accept List is not in use).
- 0x01: HCI_ADV_FILTER_ACCEPT_LIST_SCAN. Process connection requests from all devices and scan requests only from devices that are in the Filter Accept List.
- 0x02: HCI_ADV_FILTER_ACCEPT_LIST_CONNECT. Process scan requests from all devices and connection requests only from devices that are in the Filter Accept List.
- 0x03: HCI_ADV_FILTER_ACCEPT_LIST_SCAN_CONNECT. Process scan and connection requests only from devices in the Filter Accept List.

All other values are reserved for future use

Advertising_Tx_Power

Units: dBm The Advertising_Tx_Power parameter indicates the maximum power level at which the advertising packets are to be transmitted on the advertising channels. The Controller shall choose a power level lower than or equal to the one specified by the Host. Values:

- -127 ... 126
- 127: No preference

Primary_Advertising_PHY

The Primary_Advertising_PHY parameter indicates the PHY on which the advertising packets are transmitted on the primary advertising channel. If legacy advertising PDUs are being used, the Primary_Advertising_PHY shall indicate the LE 1M PHY. Values:

- 0x01: LE_1M_PHY
- 0x03: LE_CODED_PHY

Secondary_Advertising_Max_Skip

The Secondary_Advertising_Max_Skip parameter is the maximum number of advertising events that can be skipped before the AUX_ADV_IND can be sent. 0x00 AUX_ADV_IND shall be sent prior to the next advertising event 0x01-0xFF Maximum advertising events the Controller can skip before sending the AUX_ADV_IND packets on the secondary advertising channel. Values:

- 0x00 ... 0xFF

Secondary_Advertising_PHY

The Secondary_Advertising_PHY parameter indicates the PHY on which the advertising packets are transmitted on the secondary advertising channel. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY

Advertising_SID

The Advertising_SID parameter specifies the value to be transmitted in the Advertising SID subfield of the ADI field of the Extended Header of those advertising channel PDUs that have an ADI field. If the advertising set only uses PDUs that do not contain an ADI field, Advertising_SID is ignored. Values:

- 0x00 ... 0x0F

Scan_Request_Notification_Enable

The Scan_Request_Notification_Enable parameter indicates whether the Controller shall send notifications upon the receipt of a scan request PDU that is in response to an advertisement from the specified advertising set that contains its device address and is from a scanner that is allowed by the advertising filter policy. Values:

- 0x00: Scan request notifications disabled
- 0x01: Scan request notifications enabled

Primary_Advertising_PHY_Options

Preference or requirements on coding scheme when transmitting on Primary Advertising Physical Channel. Values:

- 0x00: CODED_PHY_NO_PREFERENCE
- 0x01: CODED_PHY_S2_PREFERRED
- 0x02: CODED_PHY_S8_PREFERRED
- 0x03: CODED_PHY_S2_REQUIRED
- 0x04: CODED_PHY_S8_REQUIRED

Secondary_Advertising_PHY_Options

Preference or requirements on coding scheme when transmitting on Secondary Advertising Physical Channel. Values:

- 0x00: CODED_PHY_NO_PREFERENCE
- 0x01: CODED_PHY_S2_PREFERRED
- 0x02: CODED_PHY_S8_PREFERRED
- 0x03: CODED_PHY_S2_REQUIRED
- 0x04: CODED_PHY_S8_REQUIRED

`[out]` **Selected_Tx_Power**

Units: dBm. The Selected_Tx_Power return parameter indicates the transmit power selected by the Controller. The Controller shall not change the transmit power for this advertising set without being directed to by the Host. Values:

- -127 ... 126

Return values:

- *Value* indicating success or error code.

2.1.83 hci_le_set_extended_scan_enable

```
tBleStatus hci_le_set_extended_scan_enable     ( uint8_t  Enable,
                                                 uint8_t  Filter_Duplicates,
                                                 uint16_t Duration,
                                                 uint16_t Period
                                                )
```

The LE_Set_Extended_Scan_Enable command is used to enable or disable scanning. The Enable parameter determines whether scanning is enabled or disabled. If it is disabled, the remaining parameters are ignored. The Filter_Duplicates parameter controls whether the Link Layer should filter out duplicate advertising reports (filtering duplicates enabled) to the Host or if the Link Layer should generate advertising reports for each packet received (filtering duplicates disabled). See [Vol 6] Part B, Section 4.4.3.5.

If the Filter_Duplicates parameter is set to 0x00, all advertisements received from advertisers shall be sent to the Host in advertising report events. If the Filter_Duplicates parameter is set to 0x01, duplicate advertisements should not be sent to the Host in advertising report events until scanning is disabled. If the Filter_Duplicates parameter is set to 0x02, duplicate advertisements in a single scan period should not be sent to the Host in advertising report events; this setting shall only be used if Period is non-zero. If Filter_Duplicates is set to 0x2 and Period to zero, the Controller shall return the Invalid error code HCI Command Parameters (0x12).

If the Duration parameter is zero or both the Duration parameter and Period parameter are non-zero, the Controller shall continue scanning until scanning is disabled by the Host issuing an LE_Set_Extended_Scan_Enable command with the Enable parameter set to 0x00 (Scanning is disabled).

The Period parameter is ignored when the Duration parameter is zero. If the Duration parameter is non-zero and the Period parameter is zero, the Controller shall continue scanning until the duration specified in the Duration parameter has expired. If both the Duration and Period parameters are non-zero and the Duration parameter is greater than or equal to the Period parameter, the Controller shall return the error code Invalid HCI Command Parameters (0x12).

When the Duration and Period parameters are non-zero, the Controller shall scan for the duration of the Duration parameter within a scan period specified by the Period parameter. After the scan period has expired, a new scan period shall begin and scanning shall begin again for the duration specified. The scan periods continue until the Host disables scanning. If the LE_Set_Extended_Scan_Enable command is sent while scanning is enabled, the timers used for duration and period are reset to the new parameter values and a new scan period is started. Any change to the Filter_Duplicates setting or the random address shall take effect.

Note: *Disabling scanning when it is disabled has no effect.*

Note: *The duration of a scan period refers to the time spent scanning on both the primary and secondary advertising channels. However, expiry of the duration does not prevent the Link Layer from scanning for and receiving auxiliary packets of received advertisements. If the scanning parameters' Own_Address_Type parameter is set to 0x01 or 0x03 and the random address for the device has not been initialized, the Controller shall return the error code Invalid HCI Command Parameters (0x12).*

Parameters

Enable

The Enable parameter determines whether scanning is enabled or disabled. If it is disabled, the remaining parameters are ignored. Values:

- 0x00: Scanning disabled
- 0x01: Scanning enabled

Filter_Duplicates

The Filter_Duplicates parameter controls whether the Link Layer should filter out duplicate advertising reports (filtering duplicates enabled) to the Host or if the Link Layer should generate advertising reports for each packet received (filtering duplicates disabled). See [Vol 6] Part B, Section 4.4.3.5. Values:

- 0x00: Duplicate filtering disabled
- 0x01: Duplicate filtering enabled
- 0x02: Duplicate filtering enabled, reset for each scan period

Duration

Scan duration. Time = N * 10 ms. Time Range: 10 ms to 655.35 s. Values:

- 0x0000 (0.000 ms) : Scan continuously until explicitly disable
- 0x0001 (0.625 ms) ... 0xFFFF (40959.375 ms) : Scan duration

Period

Time interval from when the Controller started its last Scan_Duration until it begins the subsequent Scan_Duration. Time = N * 1.28 sec. Time Range: 1.28 s to 83,884.8 s. Values:

- 0x0000: Periodic scanning disabled
- 0x0001 ... 0xFFFF: Time interval from when the Controller started its last Scan_Duration until it begins the subsequent Scan_Duration

Return values:

- *Value* indicating success or error code.

2.1.84 hci_le_set_extended_scan_parameters

```
tBleStatus hci_le_set_extended_scan_parameters  ( uint8_t Own_Address_Type,
                                                  uint8_t Scanning_Filter_Policy,
                                                  uint8_t Scanning_PHYs,
                                                  Extended_Scan_Parameters_t Extended_Scan_Parameters[]
                                                 )
```

The LE_Set_Extended_Scan_Parameters command is used to set the extended scan parameters to be used on the advertising channels. The Scanning_PHYs parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising channel. The Host may enable one or more scanning PHYs. The Scan_Type[i], Scan_Interval[i], and Scan_Window[i] parameters array elements are ordered in the same order as the set bits in the Scanning_PHY parameter, starting from bit 0. The number of array elements is determined by the number of bits set in the Scanning_PHY parameter.

The Scan_Type[i] parameter specifies the type of scan to perform. The Scan_Interval[i] and Scan_Window[i] parameters are recommendations from the Host on how long (Scan_Window[i]) and how frequently (Scan_Interval[i]) the Controller should scan (see [Vol 6] Part B, Section 4.5.3); however, the frequency and length of the scan is implementation specific. If the requested scan cannot be supported by the implementation, the Controller shall return the error code Invalid HCI Command Parameters (0x12). The Own_Address_Type parameter indicates the type of address being used in the scan request packets. If the Host issues this command when scanning is enabled in the Controller, the Controller shall return the error code Command Disallowed (0x0C).

Parameters

Own_Address_Type

The Own_Address_Type parameter indicates the type of address being used in the scan request packets. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the public address.
- 0x03: Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the random address from LE_Set_Random_Address.

Scanning_Filter_Policy

Values:

- 0x00: Accept all advertisement packets. Directed advertising packets which are not addressed for this device shall be ignored.
- 0x01: Ignore devices not in the Filter Accept List Only. Directed advertising packets which are not addressed for this device shall be ignored
- 0x02: Accept all (use resolving list). Accept all undirected advertisement packets. Directed advertisement packets where initiator address is a RPA and Directed advertisement packets addressed to this device shall be accepted.
- 0x03: Ignore devices not in the Filter Accept List (use resolving list). Accept all undirected advertisement packets from devices that are in the Filter Accept List. Directed advertisement packets where initiator address is RPA and Directed advertisement packets addressed to this device shall be accepted.

Scanning_PHYs

The Scanning_PHYs parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising channel. The Host may enable one or more scanning PHYs. Flags:

- 0x01: LE_1M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

Extended_Scan_Parameters

See Extended_Scan_Parameters_t.

Return values:

- *Value* indicating success or error code.

2.1.85 hci_le_set_host_channel_classification

```
tBleStatus hci_le_set_host_channel_classification ( uint8_t LE_Channel_Map[5] )
```

The HCI_LE_Set_Host_Channel_Classification command allows the Host to specify a channel classification for the data, secondary advertising, periodic, and isochronous physical channels based on its local information. This classification persists until overwritten with a subsequent HCI_LE_Set_Host_Channel_Classification command or until the Controller is reset using the HCI_Reset command. If this command is used, the Host should send it within 10 seconds of knowing that the channel classification has changed. The interval between two successive commands sent shall be at least one second.

Parameters

LE_Channel_Map

This parameter contains 37 1-bit fields. The nth such field (in the range 0 to 36) contains the value for the link layer channel index n. Channel n is bad = 0. Channel n is unknown = 1. The most significant bits are reserved and shall be set to 0. At least one channel shall be marked as unknown. Flags:

- 0x0000000000 ... 0x1FFFFFFFFF

Return values:

- *Value* indicating success or error code.

2.1.86 hci_le_set_host_feature

```
tBleStatus hci_le_set_host_feature    ( uint8_t Bit_Number,
                                         uint8_t Bit_Value
                                        )
```

The HCI_LE_Set_Host_Feature command is used by the Host to set or clear a bit controlled by the Host in the Link Layer FeatureSet stored in the Controller (see [Vol 6] Part B, Section 4.6). The Bit_Number parameter specifies the bit position in the FeatureSet. The Bit_Value parameter specifies whether the feature is enabled or disabled. If Bit_Number specifies a feature bit that is not controlled by the Host, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If Bit_Value is set to 0x01 and Bit_Number specifies a feature bit that requires support of a feature that the Controller does not support, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Host issues this command while the Controller has a connection to another device, the Controller shall return the error code Command Disallowed (0x0C).

Parameters

Bit_Number

Bit position in the FeatureSet. Values:

- 0x00 ... 0x3F

Bit_Value

If 0, the Host feature is disabled, if 1 the Host feature is enabled. Values:

- 0x00: DISABLED
- 0x01: ENABLED

Return values:

- *Value* indicating success or error code.

2.1.87 hci_le_set_path_loss_reporting_enable

```
tBleStatus hci_le_set_path_loss_reporting_enable    ( uint16_t Connection_Handle,
                                                      uint8_t  Enable
                                                     )
```

Enable or disable path loss reporting for the ACL connection identified by the Connection_Handle parameter. Initiate a new Power Control Request procedure to obtain the remote transmit power level if no prior value is available or used and no prior Power Control Request procedure has been initiated. Path loss reporting is disabled when the connection is first created.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Enable

Enable (1) or disable (0) reporting. Values:

- 0x00: DISABLE
- 0x01: ENABLE

Return values:

- *Value* indicating success or error code.

2.1.88 hci_le_set_path_loss_reporting_parameters

```
tBleStatus hci_le_set_path_loss_reporting_parameters    ( uint16_t Connection_Handle,
                                                          uint8_t  High_Threshold,
                                                          uint8_t  High_Hysteresis,
                                                          uint8_t  Low_Threshold,
                                                          uint8_t  Low_Hysteresis,
                                                          uint16_t Min_Time_Spent
                                                         )
```

Set the path loss threshold reporting parameters for the ACL connection identified by the Connection_Handle parameter.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

High_Threshold

High threshold for the path loss. Units: dB. Values:

- 0 ... 254
- 255: UNUSED

High_Hysteresis

Hysteresis value for the high threshold. Units: dB.

Low_Threshold

Low threshold for the path loss. Units: dB.

Low_Hysteresis

Hysteresis value for the low threshold. Units: dB.

Min_Time_Spent

Minimum time in number of connection events to be observed once the path crosses the threshold before an event is generated.

Return values:

- *Value* indicating success or error code.

2.1.89 hci_le_set_periodic_advertising_enable

```
tBleStatus hci_le_set_periodic_advertising_enable    ( uint8_t Enable,
                                                        uint8_t Advertising_Handle
                                                       )
```

The HCI_LE_Set_Periodic_Advertising_Enable command is used to request the Controller to enable or disable the periodic advertising for the advertising set specified by the Advertising_Handle parameter (ordinary advertising is not affected). If the advertising set is not currently enabled (see the HCI_LE_Set_Extended_Advertising_Enable command), the periodic advertising is not started until the advertising set is enabled. Once the advertising set has been enabled, the Controller shall continue periodic advertising until the Host issues an HCI_LE_Set_Periodic_Advertising_Enable command with bit 0 of Enable set to 0 (periodic advertising is disabled). Disabling the advertising set has no effect on the periodic advertising once the advertising set has been enabled.

The Controller manages the timing of advertisements in accordance with the advertising parameters given in the HCI_LE_Set_Periodic_Advertising_Parameters command. If the advertising set corresponding to the Advertising_Handle parameter does not exist, the Controller shall return the error code Unknown Advertising Identifier (0x42). If bit 0 of Enable is set to 1 (periodic advertising is enabled) and the advertising set contains partial periodic advertising data, the Controller shall return the error code Command Disallowed (0x0C). If bit 0 of Enable is set to 1 and the Host has not issued the HCI_LE_Set_Periodic_Advertising_Parameters command for the advertising set, the Controller shall either use vendor-specified parameters or return the error code Command Disallowed (0x0C).

If bit 0 of Enable is set to 1 and the length of the periodic advertising data is greater than the maximum that the Controller can transmit within the chosen periodic advertising interval, the Controller shall return the error code Packet Too Long (0x45). If advertising on the LE Coded PHY, the S=8 coding shall be assumed. If bit 0 of Enable is set to 1 and the advertising set identified by the Advertising_Handle specified scannable, connectable, legacy, or anonymous (0x0C). If bit 0 of Enable is set to 0 and the Controller supports the Periodic Advertising ADI Support feature, then the Controller shall ignore bit 1. If bit 1 of Enable is set to 1 and the Controller does not support the Periodic Advertising ADI Support feature, the Controller shall return an error which should use the error code Unsupported Feature or Parameter Value (0x11).

Enabling periodic advertising when it is already enabled can cause the random address to change. Disabling periodic advertising when it is already disabled has no effect.

Parameters

Enable

It is used to enabled advertising and include ADI field. If bit 0 is set, enable periodic advertising. if bit 1 is set, Include the ADI field in AUX_SYNC_IND PDUs. Flags:

- 0x01: ENABLE_PERIODIC_ADV
- 0x02: INCLUDE_ADI_FIELD

Advertising_Handle

Used to identify an advertising set. Values:

- 0x00 ... 0xEF

Return values:

- *Value* indicating success or error code.

2.1.90 hci_le_set_periodic_advertising_parameters

```
tBleStatus hci_le_set_periodic_advertising_parameters ( uint8_t  Advertising_Handle,
                                                        uint16_t Periodic_Advertising_Interval_Min,
                                                        uint16_t Periodic_Advertising_Interval_Max,
                                                        uint16_t Periodic_Advertising_Properties
                                                       )
```

The LE_Set_Periodic_Advertising_Parameters command is used by the Host to set the parameters for periodic advertising. The Advertising_Handle parameter identifies the advertising set whose periodic advertising parameters are being configured. If the corresponding advertising set does not already exist, then the Controller shall return the error code Unknown Advertising Identifier (0x42). The Periodic_Advertising_Interval_Min parameter shall be less than or equal to the Periodic_Advertising_Interval_Max parameter. The Periodic_Advertising_Interval_Min and Periodic_Advertising_Interval_Max parameters should not be the same value to enable the Controller to determine the best advertising interval given other activities.

The Periodic_Advertising_Properties parameter indicates which fields should be included in the advertising packet. If the advertising set identified by the Advertising_Handle specified anonymous advertising, the Controller shall return the error code Invalid HCI Parameters (0x12). If the Host issues this command when periodic advertising is enabled for the specified advertising set, the Controller shall return the error code Command Disallowed (0x0C). If the Advertising_Handle does not identify an advertising set that is already configured for periodic advertising and the Controller is unable to support more periodic advertising at present, the Controller shall return the error code Memory Capacity Exceeded (0x07).

Parameters

Advertising_Handle

It is used to identify an advertising set. Values:

- 0x00 ... 0xEF: Used to identify a periodic advertisement

Periodic_Advertising_Interval_Min

Minimum advertising interval for periodic advertising. Time = N * 1.25 ms. Time Range: 7.5ms to 81.91875 s. Values:

- 0x0006 (7.50 ms) ... 0xFFFF (NaN)

Periodic_Advertising_Interval_Max

Maximum advertising interval for periodic advertising. Time = N * 1.25 ms. Time Range: 7.5ms to 81.91875 s. Values:

- 0x0006 (7.50 ms) ... 0xFFFF (NaN)

Periodic_Advertising_Properties

The Periodic_Advertising_Properties parameter indicates which fields should be included in the advertising packet. Flags:

- 0x0040: Include TxPower in the advertising PDU

Return values:

- *Value* indicating success or error code.

2.1.91 hci_le_set_periodic_advertising_parameters_v2

```
tBleStatus hci_le_set_periodic_advertising_parameters_v2 ( uint8_t  Advertising_Handle,
                                                           uint16_t Periodic_Advertising_Interval_Min,
                                                           uint16_t Periodic_Advertising_Interval_Max,
                                                           uint16_t Periodic_Advertising_Properties,
                                                           uint8_t  Num_Subevents,
                                                           uint8_t  Subevent_Interval,
                                                           uint8_t  Response_Slot_Delay,
                                                           uint8_t  Response_Slot_Spacing,
                                                           uint8_t  Num_Response_Slots
                                                          )
```

The HCI_LE_Set_Periodic_Advertising_Parameters command is used by the Host to set the parameters for periodic advertising. The Advertising_Handle parameter identifies the advertising set whose periodic advertising parameters are being configured. If the corresponding advertising set does not already exist, then the Controller shall return the error code Unknown Advertising Identifier (0x42). The Periodic_Advertising_Interval_Min parameter shall be less than or equal to the Periodic_Advertising_Interval_Max parameter. The Periodic_Advertising_- Interval_Min and Periodic_Advertising_Interval_Max parameters should not be the same value to enable the Controller to determine the best advertising interval given other activities.

If the periodic advertising interval range provided by the Host (Periodic_Advertising_Interval_Min, Periodic_Advertising_Interval_Max) does not overlap with the periodic advertising interval range supported by the Controller, then the Controller shall return an error which should use the error code Unsupported Feature or Parameter Value (0x11).

The Periodic_Advertising_Properties parameter indicates which fields should be included in the advertising packet. The Num_Subevents parameter identifies the number of subevents that shall be transmitted for each periodic advertising event. If the Num_Subevents parameter value is 0x00, then the Subevent_Interval, Response_Slot_Delay, Response_Slot_Spacing, and Num_Response_Slots parameters shall be ignored. The Subevent_Interval parameter identifies the time between the subevents of PAwR. The Subevent_Interval shall be less than or equal to the Periodic_Advertising_Interval_Min divided by the Num_Subevents of the advertising set.

The Response_Slot_Delay parameter identifies the time between the start of the advertising packet at the start of a subevent and the start of the first response slot. The Response_Slot_Delay shall be less than the Subevent_Interval. The Response_Slot_Spacing parameter identifies the time between the start of two consecutive response slots. The Response_Slot_Spacing shall be less than or equal to 10 x (Subevent_Interval - Response_Slot_Delay) / Num_Response_Slots. If the Num_Response_Slots parameter is set to 1, then the Controller shall ignore the Response_Slot_Spacing parameter. The Num_Response_Slots parameter identifies the number of response slots in a subevent. If the Num_Response_Slots parameter value is 0x00, then the Response_Slot_Delay and Response_Slot_Spacing parameters shall be ignored.

If the advertising set identified by the Advertising_Handle specified scannable, connectable, legacy, or anonymous advertising, the Controller shall return the error code Invalid HCI Command Parameters (0x12). If the Host issues this command when periodic advertising is enabled for the specified advertising set, the Controller shall return the error code Command Disallowed (0x0C). If the Advertising_Handle does not identify an advertising set that is already configured for periodic advertising and the Controller is unable to support more periodic advertising at present, the Controller shall return the error code Memory Capacity Exceeded (0x07).

If the advertising set already contains periodic advertising data and the length of the data is greater than the maximum that the Controller can transmit within a periodic advertising interval of Periodic_Advertising_Interval_Max, the Controller shall return the error code Packet Too Long (0x45). If advertising on the LE Coded PHY, the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel.

Parameters

Advertising_Handle

Used to identify an advertising set. Values:

- 0x00 ... 0xEF: Used to identify a periodic advertisement

Periodic_Advertising_Interval_Min

Minimum advertising interval for periodic advertising. Time = N * 1.25 ms. Time Range: 7.5ms to 81.91875 s. Values:

- 0x0006 (7.50 ms) ... 0xFFFF (NaN)

Periodic_Advertising_Interval_Max

Maximum advertising interval for periodic advertising. Time = N * 1.25 ms. Time Range: 7.5ms to 81.91875 s. Values:

- 0x0006 (7.50 ms) ... 0xFFFF (NaN)

Periodic_Advertising_Properties

The Periodic_Advertising_Properties parameter indicates which fields should be included in the advertising packet. Flags:

- 0x0040

Num_Subevents

Number of subevents. Values:

- 0x00 ... 0x80

Subevent_Interval

Interval between subevents. Time = N x 1.25 ms. Values:

- 0x06 (7.50 ms) ... 0xFF (318.75 ms)

Response_Slot_Delay

Time between the advertising packet in a subevent and the first response slot. Time = N x 1.25 ms. Values:

- 0x00 (NaN) : No response slots
- 0x01 (1.25 ms) ... 0xFE (317.50 ms)

Response_Slot_Spacing

Time between response slots. Time = N x 0.125 ms. Values:

- 0x00 (0.000 ms) : No response slots
- 0x02 (0.250 ms) ... 0xFF (31.875 ms)

Num_Response_Slots

Number of subevent response slots. Values:

- 0x00 ... 0xFF

Return values:

- *Value* indicating success or error code.

2.1.92 hci_le_set_periodic_advertising_receive_enable

```
tBleStatus hci_le_set_periodic_advertising_receive_enable    ( uint16_t Sync_Handle,
                                                               uint8_t  Enable
                                                              )
```

The HCI_LE_Set_Periodic_Advertising_Receive_Enable command enables or disables reports for the periodic advertising train identified by the Sync_Handle parameter. The Enable parameter determines whether reporting and duplicate filtering are enabled or disabled. If the value is the same as the current state, the command has no effect. If the periodic advertising train corresponding to the Sync_Handle parameter does not exist, the Controller shall return the error code Unknown Advertising Identifier (0x42).

Parameters

Sync_Handle

Sync_Handle identifying the periodic advertising train. Values:

- 0x0000 ... 0x0EFF

Enable

Bit 0 to enable reporting. Bit 1 to enable duplicate filtering. Flags:

- 0x01: ENABLE_REPORTING
- 0x02: ENABLE_DUPLICATE_FILTERING

Return values:

- *Value* indicating success or error code.

2.1.93 hci_le_set_periodic_advertising_sync_transfer_parameters

```
tBleStatus hci_le_set_periodic_advertising_sync_transfer_parameters ( uint16_t Connection_Handle,
                                                                      uint8_t  Mode,
                                                                      uint16_t Skip,
                                                                      uint16_t Sync_Timeout,
                                                                      uint8_t  CTE_Type
                                                                    )
```

The HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters command is used to specify how the Controller processes periodic advertising synchronization information received from the device identified by the Connection_Handle parameter (the "transfer mode").

The Mode parameter specifies the action to be taken when periodic advertising synchronization information is received. If Mode is 0x00, the Controller ignores the information. Otherwise, it notifies the Host and synchronizes to the periodic advertising. Mode also specifies whether periodic advertising reports are initially enabled or disabled and whether duplicates are filtered.

The Skip parameter specifies the number of consecutive periodic advertising packets that the receiver may skip after successfully receiving a periodic advertising packet. The Sync_Timeout parameter specifies the maximum permitted time between successful receives. If this time is exceeded, synchronization is lost. Irrespective of the value of the Skip parameter, the Controller should stop skipping packets before the Sync_Timeout would be exceeded.

The CTE_Type parameter specifies whether to only synchronize to periodic advertising with certain types of Constant Tone Extension. If the periodic advertiser changes the type of the Constant Tone Extension after the Controller has synchronized with the periodic advertising, it shall remain synchronized.

Note: *A value of 0 (that is, all bits clear) indicates that the presence or absence of a Constant Tone Extension is irrelevant.*

This command does not affect any processing of any periodic advertising synchronization information already received from the peer device, whether or not the Controller has yet synchronized to the periodic advertising train it describes. The parameter values provided by this command override those provided via the HCI_LE_Set_Default_Per iodic_Advertising_Sync_Transfer_Parameterscommand or any preferences previously set using the HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters command on the same connection. If the Connection_Handle parameter does not identify a current connection, the Controller shall return the error code Unknown Connection Identifier (0x02).

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Mode

The action to be taken when periodic advertising synchronization information is received. If 0, no attempt is made to synchronize to the periodic advertising and no HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. If 1, an HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events are disabled. If 2, an HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events are enabled with duplicate filtering disabled. If 3, an HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events are enabled with duplicate filtering enabled.

Values:

- 0x00: NO_SYNC
- 0x01: REPORTS_DISABLED
- 0x02: REPORTS_ENABLED
- 0x03: REPORTS_ENABLED_WITH_DUPLICATE_FILTERING

Skip

The number of periodic advertising packets that can be skipped after a successful receive. Values:

- 0x0000 ... 0x01F3

Sync_Timeout

Synchronization timeout for the periodic advertising train. Time = N*10 ms. Values:

- 0x000A (100 ms) ... 0x4000 (163840 ms)

CTE_Type

It specifies whether to only synchronize to periodic advertising with certain types of Constant Tone Extension. If bit 0 is set: do not sync to packets with an AoA Constant Tone Extension. If bit 1 is set: Do not sync to packets with an AoD Constant Tone Extension with 1 us slots. If bit 2 is set: Do not sync to packets with an AoD Constant Tone Extension with 2 us slots. If bit 3 is set: Do not sync to packets without a Constant Tone Extension. Flags:

- 0x01: DO_NOT_SYNC_WITH_AOA
- 0x02: DO_NOT_SYNC_WITH_AOD_1US
- 0x04: DO_NOT_SYNC_WITH_AOD_2US
- 0x10: DO_NOT_SYNC_WITHOUT_CTE

Return values:

- *Value* indicating success or error code.

2.1.94 hci_le_set_periodic_sync_subevent

```
tBleStatus hci_le_set_periodic_sync_subevent    ( uint16_t Sync_Handle,
                                                  uint16_t Periodic_Advertising_Properties,
                                                  uint8_t  Num_Subevents,
                                                  uint8_t  Subevent[]
                                                 )
```

The HCI_LE_Set_Periodic_Sync_Subevent command is used to instruct the Controller to synchronize with a subset of the subevents within a PAwR train identified by the Sync_Handle parameter, listen for packets sent by the peer device and pass any received data up to the Host. If the Controller is synchronized with any subevents that are not in the subset of subevents in this command, then the Controller shall no longer synchronize with those subevents. The Periodic_Advertising_Properties parameter indicates which fields should be included in the AUX_SYNC_SUBEVENT_RSP PDUs. The Num_Subevents parameter identifies the number of values in the subevents parameter. The Subevents arrayed parameter identifies the subevents that the Controller shall synchronize with.

Parameters

Sync_Handle

Sync_Handle identifying the PAwR train. Values:

- 0x0000 ... 0x0EFF

Periodic_Advertising_Properties

If bit 6 is set, include TxPower in the advertising PDU. Flags:

- 0x0040: TX_POWER

Num_Subevents

Number of subevent data in the command. Values:

- 0x01 ... 0x0F

Subevent

The subevent to synchronize with.

Return values:

- *Value* indicating success or error code.

2.1.95 hci_le_set_phy

```
tBleStatus hci_le_set_phy    ( uint16_t Connection_Handle,
                               uint8_t  ALL_PHYS,
                               uint8_t  TX_PHYS,
                               uint8_t  RX_PHYS,
                               uint16_t PHY_options
                              )
```

The LE_Set_PHY command is used to set the PHY preferences for the connection identified by the Connection_Handle. The Controller might not be able to make the change (e.g. because the peer does not support the requested PHY) or may decide that the current PHY is preferable.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

ALL_PHYS

The ALL_PHYS parameter is a bit field that allows the Host to specify, for each direction, whether it has no preference among the PHYs that the Controller supports in a given direction or whether it has specified particular PHYs that it prefers in the TX_PHYS or RX_PHYS parameter. Bits: 0: The Host has no preference among the transmitter PHYs supported by the Controller 1: The Host has no preference among the receiver PHYs supported by the Controller Flags:

- 0x01: No preference for TX
- 0x02: No preference for RX

TX_PHYS

The TX_PHYS parameter is a bit field that indicates the transmitter PHYs that the Host prefers the Controller to use. If the ALL_PHYS parameter specifies that the Host has no preference, the TX_PHYS parameter is ignored; otherwise at least one bit shall be set to 1. Bits: 0: The Host prefers to use the LE 1M transmitter PHY (possibly among others) 1: The Host prefers to use the LE 2M transmitter PHY (possibly among others) 2: The Host prefers to use the LE Coded transmitter PHY (possibly among others) 3-7: Reserved for future use Flags:

- 0x01: LE_1M_PHY_BIT
- 0x02: LE_2M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

RX_PHYS

The RX_PHYS parameter is a bit field that indicates the receiver PHYs that the Host prefers the Controller to use. If the ALL_PHYS parameter specifies that the Host has no preference, the RX_PHYS parameter is ignored; otherwise at least one bit shall be set to 1. Bits: 0: The Host prefers to use the LE 1M receiver PHY (possibly among others) 1: The Host prefers to use the LE 2M receiver PHY (possibly among others) 2: The Host prefers to use the LE Coded receiver PHY (possibly among others) 3-7: Reserved for future use Flags:

- 0x01: LE_1M_PHY_BIT
- 0x02: LE_2M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

PHY_options

The PHY_options parameter is a bit field that allows the Host to specify options for PHYs. The default value for a new connection shall be all zero bits. The Controller may override any preferred coding for transmitting on the LE Coded PHY. The Host may specify a preferred coding even if it prefers not to use the LE Coded transmitter PHY since the Controller may override the PHY preference. 0 = the Host has no preferred coding when transmitting on the LE Coded PHY 1 = the Host prefers that S=2 coding be used when transmitting on the LE Coded PHY 2 = the Host prefers that S=8 coding be used when transmitting on the LE Coded PHY Values:

- 0: No preferred LE Coded PHY
- 1: S=2 preferred on LE Coded PHY
- 2: S=8 preferred on LE Coded PHY

Return values:

- *Value* indicating success or error code.

2.1.96 hci_le_set_privacy_mode

```
tBleStatus hci_le_set_privacy_mode    ( uint8_t Peer_Identity_Address_Type,
                                        uint8_t Peer_Identity_Address[6],
                                        uint8_t Privacy_Mode
                                       )
```

The HCI_LE_Set_Privacy_Mode command is used to allow the Host to specify the privacy mode to be used for a given entry on the resolving list. The effect of this setting is specified in [Vol 6] Part B, Section 4.7. When an entry on the resolving list is removed, the mode associated with that entry shall also be removed. This command cannot be used when address translation is enabled in the Controller and: Advertising is enabled Scanning is enabled Create connection command is outstanding This command can be used at any time when address translation is disabled in the Controller. If the device is not on the resolving list, the Controller shall return the error code Unknown Connection Identifier (0x02).

Parameters

Peer_Identity_Address_Type

Peer Address type Values:

- 0x00: Public Identity Address
- 0x01: Random (static) Identity Address

Peer_Identity_Address

Public Identity Address or Random (static) Identity Address of the advertiser

Privacy_Mode

0x00 Use Network Privacy Mode for this peer device (default) 0x01 Use Device Privacy Mode for this peer device. Values:

- 0x00: Network Privacy Mode
- 0x01: Device Privacy Mode

Return values:

- *Value* indicating success or error code.

2.1.97 hci_le_set_random_address

```
tBleStatus hci_le_set_random_address ( uint8_t Random_Address[6] )
```

The LE_Set_Random_Address command is used by the Host to set the LE Random Device Address in the Controller (see [Vol 6] Part B, Section 1.3). (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.4.)

Parameters

Random_Address

Random Device Address

Return values:

- *Value* indicating success or error code.

2.1.98 hci_le_set_resolvable_private_address_timeout

```
tBleStatus hci_le_set_random_address ( uint16_t RPA_Timeout )
```

The LE_Set_Resolvable_Private_Address_Timeout command set the length of time the controller uses a Resolvable Private Address before a new resolvable private address is generated and starts being used. This timeout applies to all addresses generated by the controller. (See Bluetooth Specification v.4.2, Vol. 2, Part E, 7.8.45.)

Parameters

RPA_Timeout

RPA_Timeout measured in seconds. Range for N: 0x0001 - 0xA1B8 (1 sec - approximately 11.5 hours). Default: N= 0x0384 (900 secs or 15 minutes). Values:

- 0x0001 ... 0xA1B8

Return values:

- *Value* indicating success or error code.

2.1.99 hci_le_set_scan_enable

```
tBleStatus hci_le_set_scan_enable     ( uint8_t LE_Scan_Enable,
                                         uint8_t Filter_Duplicates
                                        )
```

The LE_Set_Scan_Enable command is used to start scanning. Scanning is used to discover advertising devices nearby. The Filter_Duplicates parameter controls whether the Link Layer shall filter duplicate advertising reports to the Host, or if the Link Layer should generate advertising reports for each packet received. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.11.)

Parameters

LE_Scan_Enable

Enable/disable scan. Default is 0 (disabled). Values:

- 0x00: Scanning disabled
- 0x01: Scanning enabled

Filter_Duplicates

Enable/disable duplicate filtering. Values:

- 0x00: Duplicate filtering disabled
- 0x01: Duplicate filtering enabled

Return values:

- *Value* indicating success or error code.

2.1.100 hci_le_set_scan_parameters

```
tBleStatus hci_le_set_scan_parameters     ( uint8_t  LE_Scan_Type,
                                            uint16_t LE_Scan_Interval,
                                            uint16_t LE_Scan_Window,
                                            uint8_t  Own_Address_Type,
                                            uint8_t  Scanning_Filter_Policy
                                           )
```

The LE_Set_Scan_Parameters command is used to set the scan parameters. The LE_Scan_Type parameter controls the type of scan to perform. The LE_Scan_Interval and LE_Scan_Window parameters are recommendations from the Host on how long (LE_Scan_Window) and how frequently (LE_Scan_Interval) the Controller should scan (See [Vol 6] Part B, Section 4.5.3). The LE_Scan_Window parameter shall always be set to a value smaller or equal to the value set for the LE_Scan_Interval parameter. If they are set to the same value scanning should be run continuously. The Own_Address_Type parameter determines the address used (Public or Random Device Address) when performing active scan. The Host shall not issue this command when scanning is enabled in the Controller; if it is the Command Disallowed error code shall be used. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.10.)

Parameters

LE_Scan_Type

Passive or active scanning. With active scanning SCAN_REQ packets are sent. Values:

- 0x00: Passive Scanning
- 0x01: Active scanning

LE_Scan_Interval

This is defined as the time interval from when the Controller started its last LE scan until it begins the subsequent LE scan. Time = N * 0.625 msec. Values:

- 0x0004 (2.500 ms) ... 0x4000 (10240.000 ms)

LE_Scan_Window

The duration of the LE scan. LE_Scan_Window shall be less than or equal to LE_Scan_Interval. Time = N * 0.625 msec. Values:

- 0x0004 (2.500 ms) ... 0x4000 (10240.000 ms)

Own_Address_Type

Own address type. - 0x00: Public Device Address - 0x01 Random Device Address - 0x02: Controller generates Resolvable Private Address based on the local IRK from resolving list. If resolving list contains no matching entry, use public address. - 0x03: Controller generates Resolvable Private Address based on the local IRK from resolving list. If resolving list contains no matching entry, use random address from LE_Set_Random_Address. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Resolvable Private Address or Public Address
- 0x03: Resolvable Private Address or Random Address

Scanning_Filter_Policy

See Scanning filter policy in Bluetooth Core specification. Values:

- 0x00: Basic unfiltered scanning filter policy
- 0x01: Basic filtered scanning filter policy
- 0x02: Extended unfiltered scanning filter policy
- 0x03: Extended filtered scanning filter policy

Return values:

- *Value* indicating success or error code.

2.1.101 hci_le_set_transmit_power_reporting_enable

```
tBleStatus hci_le_set_transmit_power_reporting_enable     ( uint16_t Connection_Handle,
                                                            uint8_t  Local_Enable,
                                                            uint8_t  Remote_Enable
                                                           )
```

Enable or disable the reporting of transmit power level changes in the local and remote Controllers for the ACL connection identified by the Connection_Handle parameter. Initiate a new Power Control Request procedure to obtain the remote transmit power level if Remote_Enable is 0x01, and no prior value is available or used, and no prior Power Control Request procedure has been initiated.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Local_Enable

Enable (1) or disable (0) local transmit power reports. Values:

- 0x00: DISABLE
- 0x01: ENABLE

Remote_Enable

Enable (1) or disable (0) remote transmit power reports. Values:

- 0x00: DISABLE
- 0x01: ENABLE

Return values:

- *Value* indicating success or error code.

2.1.102 hci_le_setup_iso_data_path

```
tBleStatus hci_le_setup_iso_data_path     ( uint16_t Connection_Handle,
                                            uint8_t  Data_Path_Direction,
                                            uint8_t  Data_Path_ID,
                                            uint8_t  Codec_ID[5],
                                            uint8_t  Controller_Delay[3],
                                            uint8_t  Codec_Configuration_Length,
                                            uint8_t  Codec_Configuration[]
                                           )
```

The HCI_LE_Setup_ISO_Data_Path command is used to identify and create the isochronous data path between the Host and the Controller for an established CIS or BIS identified by the Connection_Handle parameter. This command can also be used to configure a codec for each data path.

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:

- 0x0000 ... 0x0EFF

Data_Path_Direction

The Data_Path_Direction parameter specifies the direction for which the data path is being configured. The input and output directions are defined from the perspective of the Controller, so "input" refers to data flowing from the Host to the Controller. Values:

- 0x00: Input
- 0x01: Output

Data_Path_ID

The Data_Path_ID parameter specifies the data transport path used. When set to 0x00, the data path shall be over the HCI transport. When set to 0xFF the path shall be disabled. When set to a value in the range 0x01 to 0xFE, the data path shall use a vendor- specific transport interface (e.g., a PCM interface) with logical transport numbers. The meanings of these logical transport numbers are vendor-specific. Values:

- 0x00: HCI
- 0x01 ... 0xFE
- 0xFF: Disabled

Codec_ID

The Codec_ID parameter specifies the coding format used over the air. Octet 0: See Assigned Numbers for Coding Format. Octets 1 to 2: Company ID, see Assigned Numbers for Company Identifier. Shall be ignored if octet 0 is not 0xFF. Octets 3 to 4: Vendor-defined codec ID. Shall be ignored if octet 0 is not 0xFF.

Controller_Delay

Controller delay in microseconds. When Data_Path_Direction is set to 0x00 (input), the Controller_Delay parameter specifies the delay at the data source from the reference time of an SDU to the CIG reference point (see Bluetooth Core v5.2 [Vol 6] Part B, Section 4.5.14.1) or BIG anchor point (see Core v5.2 [Vol 6] Part B, Section 4.4.6.4). When Data_Path_Direction is set to 0x01 (output), Controller_Delay specifies the delay from the CIG synchronization point or BIG synchronization point to the point in time at which the Controller begins to transfer the corresponding data to the data path interface. Values:

- 0x000000 ... 0x3D0900

Codec_Configuration_Length

Length of codec configuration.

Codec_Configuration

The Codec_Configuration parameter specifies codec- specific configuration information for the specified direction.

Return values:

- *Value* indicating success or error code.

2.1.103 hci_le_subrate_request

```
tBleStatus hci_le_subrate_request     ( uint16_t Connection_Handle,
                                        uint16_t Subrate_Min,
                                        uint16_t Subrate_Max,
                                        uint16_t Max_Latency,
                                        uint16_t Continuation_Number,
                                        uint16_t Supervision_Timeout
                                      )
```

The HCI_LE_Subrate_Request command is used by a Central or a Peripheral to request a change to the subrating factor and/or other parameters (see Core Vol 6 Part B, Section 4.5.1) applied to an existing connection using the Connection Subrate Update procedure. The Subrate_Min and Subrate_Max parameters specify the range of acceptable subrating factors being requested. The Max_Latency parameter specifies the maximum Peripheral latency in units of subrated connection events. The same maximum shall apply irrespective of the subrating factor actually chosen. The Continuation_Number parameter specifies the number of underlying connection intervals to remain active after a packet (other than an empty packet) is transmitted or received. The Supervision_Timeout parameter specifies the link supervision timeout for the connection. The Supervision_Timeout, in milliseconds, shall be greater than 2 x current connection interval x Subrate_Max x (Max_Latency + 1).

If this command is issued on the Central, the following rules shall apply when the Controller initiates the Connection Subrate Update procedure (see Core Vol 6 Part B, Section 5.1.19): * The Peripheral latency shall be less than or equal to Max_Latency. * The subrate factor shall be between Subrate_Min and Subrate_Max. * The continuation number shall be equal to the lesser of Continuation_- Number and (subrate factor - 1). * The connection supervision timeout shall be equal to Supervision_Timeout. If this command is issued on the Central, it also sets the acceptable parameters for requests from the Peripheral (see Core Vol 6 Part B, Section 5.1.20). The acceptable parameters set by this command override those provided via the HCI_LE_Set_Default_Subrate command or any values set by previous uses of this command on the same connection.

If this command is issued on the Peripheral, the following rules shall apply when the Controller initiates the Connection Subrate Request procedure: * The Peripheral latency shall be less than or equal to Max_Latency. * The minimum and maximum subrate factors shall be between Subrate_Min and Subrate_Max. * The continuation number shall be equal to the lesser of Continuation_Number and (maximum subrate factor - 1). * The connection supervision timeout shall be equal to Supervision_Timeout. If the Connection_Handle parameter does not identify a current ACL connection, the Controller shall return the error code Unknown Connection Identifier (0x02).

If the Host issues this command with parameters such that Subrate_Max x (Max_Latency + 1) is greater than 500 or the current connection interval x Subrate_ Max x (Max_Latency + 1) is greater than or equal to half the Supervision_ Timeout parameter, the Controller shall return the error code Invalid HCI Command Parameters (0x12). If the Host issues this command with Subrate_Max less than Subrate_Min, the Controller shall return the error code Invalid HCI Command Parameters (0x12). If the Host issues this command with Continuation_Number greater than or equal to Subrate_Max, then the Controller shall return the error code Invalid HCI Command Parameters (0x12). If the Central's Host issues this command when the Connection Subrating (Host Support) bit is not set in the Peripheral's FeatureSet, the Controller shall return the error code Unsupported Remote Feature (0x1A).

Parameters

Connection_Handle

Connection handle of the ACL. Values:

- 0x0000 ... 0x0EFF

Subrate_Min

Minimum subrate factor to be applied to the underlying connection interval. Values:

- 0x0001 ... 0x01F4

Subrate_Max

Maximum subrate factor to be applied to the underlying connection interval. Values:

- 0x0001 ... 0x01F4

Max_Latency

Maximum Peripheral latency for the connection in units of subrated connection intervals. Values:

- 0x0000 ... 0x01F3

Continuation_Number

Minimum number of underlying connection events to remain active after a packet containing a Link Layer PDU with a non-zero Length field is sent or received. Values:

- 0x0000 ... 0x01F3

Supervision_Timeout

Supervision timeout for this connection. Time = N x 10 ms. Values:

- 0x000A (100 ms) ... 0x0C80 (32000 ms)

Return values:

- *Value* indicating success or error code.

2.1.104 hci_le_terminate_big

```
tBleStatus hci_le_terminate_big     ( uint8_t BIG_Handle,
                                      uint8_t Reason
                                     )
```

The HCI_LE_Terminate_BIG command is used to terminate a BIG identified by the BIG_Handle parameter. The command also terminates the transmission of all BISes of the BIG, destroys the associated connection handles of the BISes in the BIG and removes the data paths for all BISes in the BIG.

Parameters

BIG_Handle

Used to identify the BIG. Values:

- 0x00 ... 0xEF

Reason

Reason for disconnection. See Error Codes.

Return values:

- *Value* indicating success or error code.

2.1.105 hci_le_write_rf_path_compensation

```
tBleStatus hci_le_write_rf_path_compensation     ( int16_t RF_TX_Path_Compensation_Value,
                                                   int16_t RF_RX_Path_Compensation_Value
                                                  )
```

The HCI_LE_Write_RF_Path_Compensation command is used to indicate the RF path gain or loss between the RF transceiver and the antenna contributed by intermediate components. A positive value means a net RF path gain and a negative value means a net RF path loss. The RF Tx Path Compensation Value parameter shall be used by the Controller to calculate radiative Tx Power Level used in HCI commands, HCI events, Advertising physical channel PDUs, and Link Layer Control PDUs using the following equation: Radiative Tx Power Level = Tx Power Level at RF transceiver output + RF Tx Path Compensation Value. For example, if the Tx Power Level is +4 (dBm) at RF transceiver output and the RF Path Compensation Value is -1.5 (dB), the radiative Tx Power Level is +4+(-1.5) = 2.5 (dBm). The RF Rx Path Compensation Value parameter shall be used by the Controller to calculate the RSSI value reported to the Host.

Parameters

RF_TX_Path_Compensation_Value

RF_RX_Path_Compensation_Value

Return values:

- *Value* indicating success or error code.

2.1.106 hci_le_write_suggested_default_data_length

```
tBleStatus hci_le_write_suggested_default_data_length     ( uint16_t SuggestedMaxTxOctets,
                                                            uint16_t SuggestedMaxTxTime
                                                           )
```

The LE_Write_Suggested_Default_Data_Length command allows the Host to specify its preferred values for the Controller maximum transmission number of payload octets and maximum packet transmission time to be used for new connections (connInitialMaxTxOctets and connInitialMaxTxTime - see [Vol 6] Part B, Section 4.5.10). The Controller may use smaller or larger values based on local information.

Parameters

SuggestedMaxTxOctets

The Host suggested value for the Controller maximum transmitted number of payload octets to be used for new connections - connInitialMaxTxOctets. Range 0x001B - 0x00FB (0x0000 - 0x001A and 0x00FC - 0xFFFF reserved for future use). Values:

- 0x001B ... 0x00FB

SuggestedMaxTxTime

The Host suggested value for the Controller maximum packet transmission time to be used for new connections - connInitialMaxTx-Time. Range 0x0148 - 0x0848 (0x0000 - 0x0147 and 0x0849 - 0xFFFF reserved for future use). Values:

- 0x0148 ... 0x0848

Return values:

- *Value* indicating success or error code.

2.1.107 hci_read_afh_channel_assessment_mode

```
tBleStatus hci_read_afh_channel_assessment_mode ( uint8_t * AFH_Channel_Assessment_Mode )
```

The HCI_Read_AFH_Channel_Assessment_Mode command reads the value for the AFH_Channel_Assessment_Mode parameter. The AFH_Channel_Assessment_Mode parameter controls whether the Controller's channel assessment scheme is enabled or disabled.

Parameters

`[out]` **AFH_Channel_Assessment_Mode**

Enable or disable channel assessment scheme. Values:

- 0x00: DISABLED
- 0x01: ENABLED

Return values:

- *Value* indicating success or error code.

2.1.108 hci_read_authenticated_payload_timeout

```
tBleStatus hci_read_authenticated_payload_timeout ( uint16_t Connection_Handle,
                                                    uint16_t * Authenticated_Payload_Timeout
                                                   )
```

This command reads the Authenticated_Payload_Timeout parameter in the Primary Controller on the specified Connection_Handle.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

`[out]` **Authenticated_Payload_Timeout**

Maximum amount of time specified between packets authenticated by a MIC. Time = N * 10 ms. Values:

- 0x0001 (10 ms) ... 0xFFFF (655350 ms)

Return values:

- *Value* indicating success or error code.

2.1.109 hci_read_bd_addr

```
tBleStatus hci_read_bd_addr ( uint8_t BD_ADDR[6] )
```

On an LE Controller, this command shall read the Public Device Address as defined in [Vol 6] Part B, Section 1.3, Device Address. If this Controller does not have a Public Device Address, the value 0x000000000000 shall be returned. On an LE Controller, the public address shall be the same as the BD_ADDR. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.4.6.)

Parameters

`[out]` BD_ADDR

Bluetooth Device Address of the Device.

Return values:

- *Value* indicating success or error code.

2.1.110 hci_read_connection_accept_timeout

```
tBleStatus hci_read_connection_accept_timeout ( uint16_t * Connection_Accept_Timeout )
```

The HCI_Read_Connection_Accept_Timeout command reads the value for the Connection Accept Timeout configuration parameter, which allows the Controller to automatically deny a connection request after a specified period has occurred, and to refuse a new connection.

Parameters

`[out]` Connection_Accept_Timeout

Connection Accept Timeout. Interval Length = N * 0.625 ms. Values:

- 0x0001 (0.625 ms) ... 0xB540 (29000.000 ms)

Return values:

- *Value* indicating success or error code.

2.1.111 hci_read_local_supported_commands

```
tBleStatus hci_read_local_supported_commands ( uint8_t Supported_Commands[64] )
```

This command reads the list of HCI commands supported for the local Controller. This command shall return the Supported_Commands configuration parameter. It is implied that if a command is listed as supported, the feature underlying that command is also supported. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.4.2.)

Parameters

`[out]` Supported_Commands

Bit mask for each HCI Command. If a bit is 1, the Controller supports the corresponding command and the features required for the command. Unsupported or undefined commands shall be set to 0.

Return values:

- *Value* indicating success or error code.

2.1.112 hci_read_local_supported_features

```
tBleStatus hci_read_local_supported_features ( uint8_t LMP_Features[8] )
```

This command requests a list of the supported features for the local Controller. This command returns a list of the LMP features. For more details, see Part C, Link Manager Protocol Specification on page 227. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.4.3.)

Parameters

`[out]` LMP_Features

Bit Mask List of LMP features.

Return values:

- *Value* indicating success or error code.

2.1.113 hci_read_local_version_information

```
tBleStatus hci_read_local_version_information ( uint8_t  * HCI_Version,
                                                 uint16_t * HCI_Revision,
                                                 uint8_t  * LMP_PAL_Version,
                                                 uint16_t * Manufacturer_Name,
                                                 uint16_t * LMP_PAL_Subversion
                                               )
```

This command reads the values for the version information for the local Controller. The HCI Version information defines the version information of the HCI layer. The LMP/PAL Version information defines the version of the LMP or PAL. The Manufacturer_Name information indicates the manufacturer of the local device. The HCI Revision and LMP/PAL Subversion are implementation dependent. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.4.1.)

Parameters

`[out]` **HCI_Version**

See Bluetooth Assigned Numbers (https://www.bluetooth.org/en-us/specification/assigned-numbers).

`[out]` **HCI_Revision**

Revision of the Current HCI in the BR/EDR Controller.

`[out]` **LMP_PAL_Version**

Version of the Current LMP or PAL in the Controller. See Bluetooth Assigned Numbers (https://www.bluetooth.org/en-us/specification/assigned-numbers).

`[out]` **Manufacturer_Name**

Manufacturer Name of the BR/EDR Controller. See Bluetooth Assigned Numbers (https://www.bluetooth.org/en-us/specification/assigned-numbers).

`[out]` **LMP_PAL_Subversion**

Subversion of the Current LMP or PAL in the Controller. This value is implementation dependent.

Return values:

- *Value* indicating success or error code.

2.1.114 hci_read_remote_version_information

```
tBleStatus hci_read_remote_version_information ( uint16_t Connection_Handle )
```

This command obtains the values for the version information for the remote device identified by the Connection_Handle parameter. The Connection_Handle must be a Connection_Handle for an ACL or LE connection. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.1.23.)

Parameters

Connection_Handle

Specifies which Connection_Handle's version information to get. Values:

- 0x0000 ... 0x0EFF

Return values:

- *Value* indicating success or error code.

2.1.115 hci_read_rssi

```
tBleStatus hci_read_rssi      ( uint16_t Connection_Handle,
                                int8_t * RSSI )
```

This command reads the Received Signal Strength Indication (RSSI) value from a Controller. For an LE transport, a Connection_Handle is used as the Handle command parameter and return parameter. The meaning of the RSSI metric is an absolute receiver signal strength value in dBm to +/- 6 dB accuracy. If the RSSI cannot be read, the RSSI metric shall be set to 127. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.5.4.)

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

`[out]` **RSSI**

N Size: 1 Octet (signed integer). Units: dBm. Values:

- -127 ... 20
- 127: RSSI not available

Return values:

- *Value* indicating success or error code.

2.1.116 hci_read_transmit_power_level

```
tBleStatus hci_read_transmit_power_level     ( uint16_t Connection_Handle,
                                               uint8_t  Type,
                                               int8_t * Transmit_Power_Level
                                              )
```

This command reads the values for the Transmit_Power_Level parameter for the specified Connection_Handle. The Connection_Handle shall be a Connection_Handle for an ACL connection. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.3.35.)

Parameters

Connection_Handle

Specifies which Connection_Handle's Transmit Power Level setting to read. Values:

- 0x0000 ... 0x0EFF

Type

Current or maximum transmit power level. Values:

- 0x00: Read Current Transmit Power Level.
- 0x01: Read Maximum Transmit Power Level.

`[out]` **Transmit_Power_Level**

Size: 1 Octet (signed integer). Units: dBm. Values:

- -30 ... 20

Return values:

- *Value* indicating success or error code.

2.1.117 hci_set_event_mask

```
tBleStatus hci_set_event_mask ( uint8_t Event_Mask[8] )
```

The Set_Event_Mask command is used to control which events are generated by the HCI for the Host. If the bit in the Event_Mask is set to a one, then the event associated with that bit is enabled. For an LE Controller, the LE Meta Event bit in the Event_Mask shall enable or disable all LE events in the LE Meta Event (see Section 7.7.65). The Host has to deal with each event that occurs. The event mask allows the Host to control how much it is interrupted. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.3.1.)

Parameters

Event_Mask

Event mask. Default: 0x0000 1FFF FFFF FFFF.

Flags:

- 0x0000 0000 0000 0000: No events specified
- 0x0000 0000 0000 0001: Inquiry Complete Event
- 0x0000 0000 0000 0002: Inquiry Result Event
- 0x0000 0000 0000 0004: Connection Complete Event
- 0x0000 0000 0000 0008: Connection Request Event
- 0x0000 0000 0000 0010: Disconnection Complete Event
- 0x0000 0000 0000 0020: Authentication Complete Event
- 0x0000 0000 0000 0040: Remote Name Request Complete Event
- 0x0000 0000 0000 0080: Encryption Change Event
- 0x0000 0000 0000 0100: Change Connection Link Key Complete Event
- 0x0000 0000 0000 0200: Central Link Key Complete Event
- 0x0000 0000 0000 0400: Read Remote Supported Features Complete Event
- 0x0000 0000 0000 0800: Read Remote Version Information Complete Event
- 0x0000 0000 0000 1000: QoS Setup Complete Event
- 0x0000 0000 0000 8000: Hardware Error Event
- 0x0000 0000 0001 0000: Flush Occurred Event
- 0x0000 0000 0002 0000: Role Change Event
- 0x0000 0000 0008 0000: Mode Change Event
- 0x0000 0000 0010 0000: Return Link Keys Event
- 0x0000 0000 0020 0000: PIN Code Request Event
- 0x0000 0000 0040 0000: Link Key Request Event
- 0x0000 0000 0080 0000: Link Key Notification Event
- 0x0000 0000 0100 0000: Loopback Command Event
- 0x0000 0000 0200 0000: Data Buffer Overflow Event
- 0x0000 0000 0400 0000: Max Slots Change Event
- 0x0000 0000 0800 0000: Read Clock Offset Complete Event
- 0x0000 0000 1000 0000: Connection Packet Type Changed Event
- 0x0000 0000 2000 0000: QoS Violation Event
- 0x0000 0000 4000 0000: Page Scan Mode Change Event
- 0x0000 0000 8000 0000: Page Scan Repetition Mode Change Event
- 0x0000 0001 0000 0000: Flow Specification Complete Event
- 0x0000 0002 0000 0000: Inquiry Result with RSSI Event
- 0x0000 0004 0000 0000: Read Remote Extended Features Complete Event
- 0x0000 0800 0000 0000: Synchronous Connection Complete Event
- 0x0000 1000 0000 0000: Synchronous Connection Changed Event
- 0x0000 2000 0000 0000: Sniff Subrating Event
- 0x0000 4000 0000 0000: Extended Inquiry Result Event
- 0x0000 8000 0000 0000: Encryption Key Refresh Complete Event
- 0x0001 0000 0000 0000: IO Capability Request Event
- 0x0002 0000 0000 0000: IO Capability Request Reply Event

- 0x0004 0000 0000 0000: User Confirmation Request Event
- 0x0008 0000 0000 0000: User Passkey Request Event
- 0x0010 0000 0000 0000: Remote OOB Data Request Event
- 0x0020 0000 0000 0000: Simple Pairing Complete Event
- 0x0080 0000 0000 0000: Link Supervision Timeout Changed Event
- 0x0100 0000 0000 0000: Enhanced Flush Complete Event
- 0x0400 0000 0000 0000: User Passkey Notification Event
- 0x0800 0000 0000 0000: Keypress Notification Event
- 0x1000 0000 0000 0000: Remote Host Supported Features Notification Event
- 0x2000 0000 0000 0000: LE Meta-Event

Return values:

- *Value* indicating success or error code.

2.1.118 hci_set_event_mask_page_2

```
tBleStatus hci_set_event_mask_page_2 ( uint8_t Event_Mask_Page_2[8] )
```

The HCI_Set_Event_Mask_Page_2 command is used to control which events are generated by the HCI for the Host. The Event_Mask_Page_2 is a logical extension to the Event_Mask parameter of the HCI_Set_Event_Mask command. If the bit in the Event_Mask_Page_2 is set to a one, then the event associated with that bit shall be enabled. The event mask allows the Host to control how much it is interrupted. The Controller shall ignore those bits which are reserved for future use or represent events which it does not support. If the Host sets any of these bits to 1, the Controller shall act as if they were set to 0.

Parameters

Event_Mask_Page_2

For the complete list of bits that can be set, see Core v5.1, Vol 2, part E, chapter 7.3.69. The only bit that is not ignored is Bit 23: Authenticated Payload Timeout Expired event. Flags:

- 0x0000000000800000: AUTHENTICATED_PAYLOAD_TIMEOUT_EXPIRED_EVENT

Return values:

- *Value* indicating success or error code.

2.1.119 hci_tx_acl_data

```
tBleStatus hci_tx_acl_data    ( uint16_t Connection_Handle,
                                uint8_t  PB_Flag,
                                uint8_t  BC_Flag,
                                uint16_t Data_Length,
                                uint8_t  * PDU_Data
                               )
```

API used to send HCI ACL Data Packets to exchange data between the Host and Controller.

Note: *The API name is available in link layer only mode.*

Parameters

Connection_Handle

Connection handle for which the command is given. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF reserved for future use).

PB_Flag

Packet boundary flag

BC_Flag

Broadcast flag

Data_Length

Length of PDU data in octets

PDU_Data

PDU data pointer

Return values:

- *Value* indicating success or error code.

2.1.120 hci_tx_iso_data

```
tBleStatus hci_tx_iso_data    ( uint16_t Connection_Handle,
                                 uint8_t  PB_Flag,
                                 uint8_t  TS_Flag,
                                 uint16_t ISO_Data_Load_Length,
                                 uint8_t  * ISO_Data_Load
                               )
```

Function to send isochronous data to the Controller.

Parameters

Connection_Handle

Connection handle for which the command is given. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF reserved for future use).

PB_Flag

Packet boundary flag

TS_Flag

Timestamp flag. Set if the ISO_Data_Load field contains a Time_Stamp field.

ISO_Data_Load_Length

Length of ISO_Data_Load in octets

ISO_Data_Load

The format of the ISO_Data_Load is described in Core v5.3 Vol. 4, part E, section 5.4.5.

Return values:

- *Value* indicating success or error code.

2.1.121 hci_write_afh_channel_assessment_mode

```
tBleStatus hci_write_afh_channel_assessment_mode ( uint8_t AFH_Channel_Assessment_Mode )
```

The HCI_Write_AFH_Channel_Assessment_Mode command writes the value for the AFH_Channel_Assessment_Mode parameter. The AFH_Channel_Assessment_Mode parameter controls whether the Controller's channel assessment scheme is enabled or disabled.

Parameters

AFH_Channel_Assessment_Mode

Values:

- 0x00: DISABLED
- 0x01: ENABLED

Return values:

- *Value* indicating success or error code.

2.1.122 hci_write_authenticated_payload_timeout

```
tBleStatus hci_write_authenticated_payload_timeout ( uint16_t Connection_Handle,
                                                     uint16_t Authenticated_Payload_Timeout )
```

This command writes the Authenticated_Payload_Timeout parameter in the Primary Controller for the specified Connection_Handle. The Authenticated_Payload_Timeout shall be equal to or greater than connInterval * (1 + connPeripheralLatency). The Link Layer uses this parameter to determine when to use the LE ping sequence.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Authenticated_Payload_Timeout

Maximum amount of time specified between packets authenticated by a valid MIC. Time = N * 10 ms. Values:

- 0x0001 (10 ms) ... 0xFFFF (655350 ms)

Return values:

- *Value* indicating success or error code.

2.1.123 hci_write_connection_accept_timeout

```
tBleStatus hci_write_connection_accept_timeout ( uint16_t Connection_Accept_Timeout )
```

The HCI_Write_Connection_Accept_Timeout command writes the value for the Connection Accept Timeout configuration parameter, which allows the Controller to automatically deny a connection request after a specified period has occurred, and to refuse a new connection.

Parameters

Connection_Accept_Timeout

Connection Accept Timeout. Interval Length = N * 0.625 ms. Default: 0x1FA0. Time = 5.06 s. Mandatory range for Controller: 0x00A0 to 0xB540. Values:

- 0x0001 (0.625 ms) ... 0xB540 (29000.000 ms)

Return values:

- *Value* indicating success or error code.

2.1.124 hci_reset

```
tBleStatus hci_reset(void)
```

The reset command resets the Link Layer on an LE Controller. The reset command shall not affect the used HCI transport layer since the HCI transport layers may have reset mechanisms of their own. After the reset is completed, the current operational state is lost, the Controller enters standby mode and the Controller automatically reverts to the default values for the parameters for which default values are defined in the specification.

Note: *The reset command does not necessarily perform a hardware reset. This is implementation defined. The Host shall not send additional HCI commands before the command complete event related to the reset command has been received (refer to Bluetooth Specification v.4.1, Vol. 2, Part E, 7.3.2).*

Parameters

None.

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.1.125 hci_le_set_advertising_data

```
tBleStatus hci_le_set_advertising_data(uint8_t Advertising_Data_Length,
                                                  uint8_t Advertising_Data[31])
```

The LE_Set_Advertising_Data command is used to set the data used in advertising packets that have a data field. Only the significant part of the Advertising_Data is transmitted in the advertising packets, as defined in [Vol 3] Part C, Section 11 (see Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.7).

Parameters

Advertising_Data_Length

The number of significant octets in the following data field. Values:

- 0 ... 31

Advertising_Data

31 octets of data formatted as defined in [Vol 3] Part C, Section 11.

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.1.126 hci_le_set_scan_response_data

```
tBleStatus hci_le_set_scan_response_data(uint8_t Scan_Response_Data_Length,
                                         uint8_t Scan_Response_Data[31])
```

This command is used to provide data used in scanning packets that have a data field. Only the significant part of the Scan_Response_Data is transmitted in the scanning packets, as defined in [Vol 3] Part C, Section 11 (see Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.8).

Parameters

Scan_Response_Data_Length

The number of significant octets in the following data field. Values:

- 0 ... 31

param Scan_Response_Data

31 octets of data formatted as defined in [Vol 3] Part C, Section 11.

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.1.127 hci_le_set_extended_advertising_data

```
tBleStatus hci_le_set_extended_advertising_data(uint8_t Advertising_Handle,
                                                uint8_t Operation,
                                                uint8_t Fragment_Preference,
                                                uint8_t Advertising_Data_Length,
                                                uint8_t Advertising_Data[])
```

The LE_Set_Extended_Advertising_Data command is used to set the data used in advertising PDUs that have a data field. This command may be issued at any time after an advertising set identified by the Advertising_Handle parameter has been created using the LE Set Extended Advertising Parameters Command (see section 7.8.53), regardless of whether advertising in that set is enabled or disabled.

Parameters

Advertising_Handle

It is used to identify an advertising set. Values:

- 0x00 ... 0xEF

Operation

The Host may set the advertising data in one or more operations using this parameter. Values:

- 0x00: Intermediate fragment of fragmented extended advertising data
- 0x01: First fragment of fragmented extended advertising data
- 0x02: Last fragment of fragmented extended advertising data
- 0x03: Complete extended advertising data
- 0x04: Unchanged data (just update the Advertising DID)

All other values: Reserved for future use.

Fragment_Preference

The Fragment_Preference parameter provides a hint to the Controller as to whether advertising data should be fragmented. Values:

- 0x00: The Controller may fragment
- 0x01: The Controller should not fragment or should minimize fragmentation

Advertising_Data_Length

The number of octets in the Advertising Data parameter. Values:

- 0 ... 251

Advertising_Data formatted as defined in [Vol 3] Part C, Section 11

Advertising_Data

Advertising data formatted as defined in [Vol 3] Part C, Section 11

Note: This parameter has a variable length.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.1.128 hci_le_set_extended_scan_response_data

```
tBleStatus hci_le_set_extended_scan_response_data(uint8_t Advertising_Handle,
                                                  uint8_t Operation,
                                                  uint8_t Fragment_Preference,
                                                  uint8_t Scan_Response_Data_Length,
                                                  uint8_t Scan_Response_Data[])
```

The LE_Set_Extended_Scan_Response_Data command is used to provide scan response data used in scanning response PDUs. This command may be issued at any time after the advertising set identified by the Advertising_Handle parameter has been created using the LE Set Extended Advertising Parameters Command (see Section 7.8.53) regardless of whether advertising in that set is enabled or disabled.

Parameters

Advertising_Handle

It is used to identify an advertising set. Values:

- 0x00 ... 0xEF

Operation

The Host may set the scan data in one or more operations using this parameter. Values:

- 0x00: Intermediate fragment of fragmented scan response data
- 0x01: First fragment of fragmented scan response data
- 0x02: Last fragment of fragmented scan response data
- 0x03: Complete scan response data

All other values: Reserved for future use.

Fragment_Preference

The Fragment_Preference parameter provides a hintto the Controller as to whether advertising data should be fragmented. Values:

- 0x00: The Controller may fragment all scan response data
- 0x01: The Controller should not fragment or should minimize fragmentation of scan response data

Scan_Response_Data_Length

The number of octets in the Scan_Response Data parameter. Values:

- 0x00 ... 0xFB

Scan_Response

Scan response data formatted as defined in [Vol 3] Part C, Section 11

Note: This parameter has a variable length.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.1.129 hci_le_read_maximum_advertising_data_length

```
tBleStatus hci_le_read_maximum_advertising_data_length(uint16_t *Maximum_Advertising_Data_Len
gth)
```

The LE_Read_Maximum_Advertising_Data_Length command is used to read the maximum length of data supported by the Controller for use as advertisement data or scan response data in an advertising event or as periodic advertisement data.

Note: The maximum amount may be fragmented across multiple PDUs (see [Vol 6] Part B, Section 2.3.4.9).

Parameters

`[out]` Maximum_Advertising_Data_Length

Maximum length of data supported by the Controller for use as advertisement data or scan response data in an advertising event or as periodic advertisement data. Values:

- 0x001F ... 0x0672

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.1.130 hci_le_set_periodic_advertising_data

```
tBleStatus hci_le_set_periodic_advertising_data(uint8_t Advertising_Handle,
                                                uint8_t Operation,
                                                uint8_t Advertising_Data_Length,
                                                uint8_t Advertising_Data[])
```

The LE_Set_Periodic_Advertising_Data command is used to set the data used in periodic advertising PDUs. This command may be issued at any time after the advertising set identified by the Advertising_Handle parameter has been configured for periodic advertising using the LE_Set_Periodic_Advertising_Parameters Command (see Section 7.8.61), regardless of whether advertising in that set is enabled or disabled. If the advertising set has not been configured for periodic advertising, then the Controller shall return the error code Command Disallowed (0x0C). If advertising is currently enabled for the specified advertising set, the Controller shall use the new data in subsequent periodic advertising events for this advertising set. If a periodic advertising event is in progress when this command is issued, the Controller may use the old or new data for that event. If periodic advertising is currently disabled for the specified advertising set, the data shall be kept by the Controller and used once periodic advertising is enabled for that set. The data shall be discarded when the advertising set is removed. Only the significant part of the periodic advertising data should be transmitted in the advertising packets as defined in [Vol 3] Part C, Section 11. The Host may set the periodic advertising data in one or more operations using the Operation parameter in the command. If the combined length of the data exceeds the capacity of the advertising set identified by the Advertising_Handle parameter (see Section 7.8.57 LE Read Maximum Advertising Data Length Command) or the amount of memory currently available, all the data shall be discarded and the Controller shall return the error code Memory Capacity Exceeded (0x07). If Operation indicates the start of new data (values 0x01 or 0x03), then any existing partial or complete data shall be discarded. If the Advertising_Data_Length parameter is 0, then Operation shall be 0x03;this indicates that any existing partial or complete data shall be deleted and no new data provided.

Parameters

Advertising_Handle

It is used to identify an advertising set. Values:

- 0x00 ... 0xEF

Operation

The Host may set the periodic advertising data in one or more operations using the Operation parameter in the command. Values:

- 0x00: Intermediate fragment of fragmented periodic advertising data
- 0x01: First fragment of fragmented periodic advertising data
- 0x02: Last fragment of fragmented periodic advertising data
- 0x03: Complete periodic advertising data

Advertising_Data_Length

The number of octets in the Advertising Data parameter. Values:

- 0 ... 252

Advertising_Data

Periodic advertising data formatted as defined in [Vol 3] Part C, Section 11

Note: This parameter has a variable length.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.2 HCI test commands

This section describes the standard HCI commands for testing.

Table 2. **HCI tests commands opcodes**

SoC command	Network coprocessor command	OpCode
hci_le_receiver_test	hci_le_receiver_test	0x201D
hci_le_transmitter_test	hci_le_transmitter_test	0x201E
hci_le_test_end	hci_le_test_end	0x201F

SoC command	Network coprocessor command	OpCode
hci_le_receiver_test_v2	hci_le_receiver_test_v2	0x2033
hci_le_transmitter_test_v2	hci_le_transmitter_test_v2	0x2034
hci_le_receiver_test_v3	hci_le_receiver_test_v3	0x204F
hci_le_transmitter_test_v3	hci_le_transmitter_test_v3	0x2050
hci_le_set_cig_parameters_test	hci_le_set_cig_parameters_test	0x2063
hci_le_create_big_test	hci_le_create_big_test	0x2069
hci_le_iso_transmit_test	hci_le_iso_transmit_test	0x2070
hci_le_iso_receive_test	hci_le_iso_receive_test	0x2071
hci_le_iso_read_test_counters	hci_le_iso_read_test_counters	0x2072
hci_le_iso_test_end	hci_le_iso_test_end	0x2073
hci_le_transmitter_test_v4	hci_le_transmitter_test_v4	0x207B

2.2.1 hci_le_receiver_test

```
tBleStatus hci_le_receiver_test ( uint8_t  RX_Frequency )
```

This command is used to start a test where the DUT receives test reference packets at a fixed interval. The tester generates the test reference packets. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.28.)

Parameters

RX_Frequency

Values:

- 0x00 ... 0x27: N = (F - 2402) / 2. Frequency Range: 2402 MHz to 2480 MHz.

Return values:

- *Value* indicating success or error code.

2.2.2 hci_le_receiver_test_v2

```
tBleStatus hci_le_receiver_test_v2    ( uint8_t RX_Channel,
                                        uint8_t PHY,
                                        uint8_t Modulation_index
                                       )
```

This command is used to start a test where the DUT receives test reference packets at a fixed interval. The tester generates the test reference packets.

Parameters

RX_Channel

Frequency Range: 2402 MHz to 2480 MHz; N = (F - 2402) / 2. Values:

- 0x00 ... 0x27: Frequency Range: 2402 MHz to 2480 MHz

PHY

PHY to be used by the receiver. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY

Modulation_index

The Modulation_Index parameter specifies whether or not the Controller should assume the receiver has a stable modulation index. Values:

- 0x00: Standard modulation index
- 0x01: Stable modulation index

Return values:

- *Value* indicating success or error code.

2.2.3 hci_le_receiver_test_v3

```
tBleStatus hci_le_receiver_test_v3    ( uint8_t RX_Channel,
                                        uint8_t PHY,
                                        uint8_t Modulation_Index,
                                        uint8_t Expected_CTE_Length,
                                        uint8_t Expected_CTE_Type,
                                        uint8_t Slot_Durations,
                                        uint8_t Switching_Pattern_Length,
                                        uint8_t Antenna_IDs[]
                                       )
```

This command is used to start a test where the DUT receives test reference packets at a fixed interval. The tester generates the test reference packets. The RX_Channel and PHY parameters specify the RF channel and PHY to be used by the receiver. If the Host sets the PHY parameter to a PHY that the Controller does not support, including a value that is reserved for future use, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11).

The Modulation_Index parameter specifies whether or not the Controller should assume the receiver has a stable modulation index. The Expected_CTE_Length and Expected_CTE_Type parameters specify the expected length and type of the Constant Tone Extensions in received test reference packets. When receiving on a PHY that allows Constant Tone Extensions, if the Constant Tone Extension in a received test reference packet does not match both of these, the DUT shall discard that packet. If Expected_CTE_Length is not zero and PHY specifies a PHY that does not allow Constant Tone Extensions, the Controller shall return the error code Command Disallowed (0x0C).

If the Slot_Durations parameter is set to 0x01 and the Controller does not support 1 microsecond switching and sampling, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). Slot_Durations, Switching_Pattern_Length, and Antenna_IDs[i] are only used when expecting an AoA Constant Tone Extension and shall be ignored when expecting an AoD Constant Tone Extension. If the Controller determines that any of the Antenna_IDs[i] values do not identify an antenna in the device's antenna array, it shall return the error code Unsupported Feature or Parameter Value (0x11).

Note: *Some Controllers may be unable to determine which values do or do not identify an antenna.*

Parameters

RX_Channel

Values:

- 0x00 ... 0x27: N = (F - 2402) / 2. Frequency Range: 2402 MHz to 2480 MHz.

PHY

PHY to be used by the receiver. Values:
- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY

Modulation_index

Values:
- 0x00: Assume transmitter has a standard modulation index
- 0x01: Assume transmitter has a stable modulation index

Expected_CTE_Length

Values:
- 0x00: No Constant Tone Extension expected (default)
- 0x02 ... 0x14: Expected length of the Constant Tone Extension in 8 microseconds units.

Expected_CTE_Type

Values:
- 0x00: Expect AoA Constant Tone Extension
- 0x01: Expect AoD Constant Tone Extension with 1 microsecond slots
- 0x02: Expect AoD Constant Tone Extension with 2 microseconds slots

Slot_Durations

Sampling rate used by the Controller. Values:
- 0x01: CTE_SLOT_1us
- 0x02: CTE_SLOT_2us

Switching_Pattern_Length

Values:
- 0x02 ... 0x4B: The number of Antenna IDs in the pattern.

Antenna_IDs

List of Antenna IDs in the pattern

Return values:
- *Value* indicating success or error code.

2.2.4 hci_le_test_end

```
tBleStatus hci_le_test_end ( uint16_t * Number_Of_Packets )
```

This command is used to stop any test which is in progress. The Number_Of_Packets for a transmitter test shall be reported as 0x0000. The Number_Of_Packets is an unsigned number and contains the number of received packets. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.30.)

Parameters

Number_Of_Packets

Number of packets received.

Return values:
- *Value* indicating success or error code.

2.2.5 hci_le_transmitter_test

```
tBleStatus hci_le_transmitter_test    ( uint8_t TX_Frequency,
                                          uint8_t Length_Of_Test_Data,
                                          uint8_t Packet_Payload
                                         )
```

This command is used to start a test where the DUT generates test reference packets at a fixed interval. The Controller shall transmit at maximum power. An LE Controller supporting the LE_Transmitter_Test command shall support Packet_Payload values 0x00, 0x01 and 0x02. An LE Controller may support other values of Packet_Payload. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.8.29.)

Parameters

TX_Frequency

N = (F - 2402) / 2. Frequency Range: 2402 MHz to 2480 MHz. Values:

- 0x00 ... 0x27

Length_Of_Test_Data

Length in bytes of payload data in each packet. Supported ranges:

- (0x00, 0x25) if data length extension is disabled
- (0x00, 0xFF) if data length extension is enabled

Values:

- 0x00 ... 0xFF

Packet_Payload

Content of the Payload of the test reference packets. Values:

- 0x00: PRBS9 sequence '11111111100000111101...' (in transmission order)
- 0x01: Repeated '11110000' (in transmission order) sequence
- 0x02: Repeated '10101010' (in transmission order) sequence
- 0x03: PRBS15 sequence
- 0x04: Repeated '11111111' (in transmission order) sequence
- 0x05: Repeated '00000000' (in transmission order) sequence
- 0x06: Repeated '00001111' (in transmission order) sequence
- 0x07: Repeated '01010101' (in transmission order) sequence

Return values:

- *Value* indicating success or error code.

2.2.6 hci_le_transmitter_test_v2

```
tBleStatus hci_le_transmitter_test_v2    ( uint8_t TX_Channel,
                                             uint8_t Length_Of_Test_Data,
                                             uint8_t Packet_Payload
                                             uint8_t PHY
                                            )
```

This command is used to start a test where the DUT generates test reference packets at a fixed interval. The Controller shall transmit at maximum power. An LE Controller supporting the LE_Enhanced Transmitter_Test command shall support Packet_Payload values 0x00, 0x01 and 0x02. An LE Controller supporting the LE Coded PHY shall also support Packet_Payload value 0x04. An LE Controller may support other values of Packet_Payload.

Parameters

TX_Channel

Frequency Range: 2402 MHz to 2480 MHz; N = (F-2402) / 2. Values:

- 0x00 ... 0x27: Frequency Range: 2402 MHz to 2480 MHz.

Length_Of_Test_Data

Length in bytes of payload data in each packet. Supported ranges:

- (0x00, 0x25) if data length extension is disabled
- (0x00, 0xFF) if data length extension is enabled

Values:

- 0x00 ... 0xFF

Packet_Payload

Content of the Payload of the test reference packets. Values:

- 0x00: PRBS9 sequence '11111111100000111101...' (in transmission order)
- 0x01: Repeated '11110000' (in transmission order) sequence
- 0x02: Repeated '10101010' (in transmission order) sequence
- 0x03: PRBS15 sequence
- 0x04: Repeated '11111111' (in transmission order) sequence
- 0x05: Repeated '00000000' (in transmission order) sequence
- 0x06: Repeated '00001111' (in transmission order) sequence
- 0x07: Repeated '01010101' (in transmission order) sequence

PHY

PHY to be used by the transmitter. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY_S8
- 0x04: LE_CODED_PHY_S2

Return values:

- *Value* indicating success or error code.

2.2.7 hci_le_transmitter_test_v3

```
tBleStatus hci_le_transmitter_test_v3    ( uint8_t TX_Channel,
                                           uint8_t Test_Data_Length,
                                           uint8_t Packet_Payload,
                                           uint8_t PHY,
                                           uint8_t CTE_Length,
                                           uint8_t CTE_Type,
                                           uint8_t Switching_Pattern_Length,
                                           uint8_t Antenna_IDs[]
                                          )
```

This command is used to start a test where the DUT generates test reference packets at a fixed interval. The Controller shall transmit at the power level indicated by the Transmit_Power_Level parameter. The TX_Channel and PHY parameters specify the RF channel and PHY to be used by the transmitter. If the Host sets the PHY parameter to a PHY that the Controller does not support, including a value that is reserved for future use, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). The Test_Data_Length and Packet_Payload parameters specify the length and contents of the Payload of the test reference packets.

An LE Controller supporting the HCI_LE_Transmitter_Test command shall support Packet_Payload values 0x00, 0x01 and 0x02. An LE Controller supporting the LE Coded PHY shall also support Packet_Payload value 0x04. An LE Controller may support other values of Packet_Payload. The CTE_Length and CTE_Type parameters specify the length and type of the Constant Tone Extension in the test reference packets. If the CTE_Type parameter is set to 0x01 and the Controller does not support 1 microsecond switching, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If CTE_Length is not zero and PHY specifies a PHY that does not allow Constant Tone Extensions, the Controller shall return the error code Command Disallowed (0x0C).

The Switching_Pattern_Length and Antenna_IDs[i] parameters specify the antenna switching pattern. They are only used when transmitting an AoD Constant Tone Extension and shall be ignored when transmitting an AoA Constant Tone Extension. If the Controller determines that any of the Antenna_IDs[i] values do not identify an antenna in the device's antenna array, it shall return the error code Unsupported Feature or Parameter Value (0x11). Note: Some Controllers may be unable to determine which values do or do not identify an antenna. The Transmit_Power_Level parameter specifies the transmit power level to be used by the transmitter. If the parameter is set to a value other than 0x7E or 0x7F, then the Controller shall make the requested change or shall make the nearest change that it is capable of doing.

Parameters

TX_Channel

N = (F - 2402) / 2. Frequency Range: 2402 MHz to 2480 MHz. Values:

- 0x00 ... 0x27

Length_Of_Test_Data

Length in bytes of payload data in each packet. Supported ranges:

- (0x00, 0x25) if data length extension is disabled
- (0x00, 0xFF) if data length extension is enabled

Values:

- 0x00 ... 0xFF

Packet_Payload

Content of the Payload of the test reference packets. Values:

- 0x00: PRBS9 sequence '11111111100000111101...' (in transmission order)
- 0x01: Repeated '11110000' (in transmission order) sequence
- 0x02: Repeated '10101010' (in transmission order) sequence
- 0x03: PRBS15 sequence
- 0x04: Repeated '11111111' (in transmission order) sequence
- 0x05: Repeated '00000000' (in transmission order) sequence
- 0x06: Repeated '00001111' (in transmission order) sequence
- 0x07: Repeated '01010101' (in transmission order) sequence

CTE_Length

Values:

- 0x00: No Constant Tone Extension expected (default)
- 0x02 ... 0x14: Expected length of the Constant Tone Extension in 8 microseconds units.

CTE_Type

Values:

- 0x00: Expect AoA Constant Tone Extension
- 0x01: Expect AoD Constant Tone Extension with 1 microsecond slots
- 0x02: Expect AoD Constant Tone Extension with 2 microseconds slots

Switching_Pattern_Length

The number of Antenna IDs in the pattern. Values:

- 0x02 ... 0x4B

Antenna_IDs

List of Antenna IDs in the pattern.

Return values:

- *Value* indicating success or error code.

2.2.8 hci_le_transmitter_test_v4

```
tBleStatus hci_le_transmitter_test_v4    ( uint8_t TX_Channel,
                                            uint8_t Test_Data_Length,
                                            uint8_t Packet_Payload,
                                            uint8_t PHY,
                                            uint8_t CTE_Length,
                                            uint8_t CTE_Type,
                                            uint8_t Switching_Pattern_Length,
                                            uint8_t Antenna_IDs[],
                                            int8_t  Transmit_Power_Level
                                          )
```

This command is used to start a test where the DUT generates test reference packets at a fixed interval. The Controller shall transmit at the power level indicated by the Transmit_Power_Level parameter. The TX_Channel and PHY parameters specify the RF channel and PHY to be used by the transmitter. If the Host sets the PHY parameter to a PHY that the Controller does not support, including a value that is reserved for future use, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). The Test_Data_Length and Packet_Payload parameters specify the length and contents of the Payload of the test reference packets.

An LE Controller supporting the HCI_LE_Transmitter_Test command shall support Packet_Payload values 0x00, 0x01 and 0x02. An LE Controller supporting the LE Coded PHY shall also support Packet_Payload value 0x04. An LE Controller may support other values of Packet_Payload. The CTE_Length and CTE_Type parameters specify the length and type of the Constant Tone Extension in the test reference packets. If the CTE_Type parameter is set to 0x01 and the Controller does not support 1 microsecond switching, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If CTE_Length is not zero and PHY specifies a PHY that does not allow Constant Tone Extensions, the Controller shall return the error code Command Disallowed (0x0C).

The Switching_Pattern_Length and Antenna_IDs[i] parameters specify the antenna switching pattern. They are only used when transmitting an AoD Constant Tone Extension and shall be ignored when transmitting an AoA Constant Tone Extension. If the Controller determines that any of the Antenna_IDs[i] values do not identify an antenna in the device's antenna array, it shall return the error code Unsupported Feature or Parameter Value (0x11). Note: Some Controllers may be unable to determine which values do or do not identify an antenna. The Transmit_Power_Level parameter specifies the transmit power level to be used by the transmitter. If the parameter is set to a value other than 0x7E or 0x7F, then the Controller shall make the requested change or shall make the nearest change that it is capable of doing.

Parameters

TX_Channel

N = (F - 2402) / 2. Frequency Range: 2402 MHz to 2480 MHz. Values:
- 0x00 ... 0x27

Length_Of_Test_Data

Length in bytes of payload data in each packet. Supported ranges:
- (0x00, 0x25) if data length extension is disabled
- (0x00, 0xFF) if data length extension is enabled

Values:
- 0x00 ... 0xFF

Packet_Payload

Content of the Payload of the test reference packets. Values:
- 0x00: PRBS9 sequence '11111111100000111101...' (in transmission order)
- 0x01: Repeated '11110000' (in transmission order) sequence
- 0x02: Repeated '10101010' (in transmission order) sequence
- 0x03: PRBS15 sequence
- 0x04: Repeated '11111111' (in transmission order) sequence
- 0x05: Repeated '00000000' (in transmission order) sequence
- 0x06: Repeated '00001111' (in transmission order) sequence
- 0x07: Repeated '01010101' (in transmission order) sequence

PHY

PHY to be used by the transmitter. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY_S8
- 0x04: LE_CODED_PHY_S2

CTE_Length

Values:

- 0x00: No Constant Tone Extension expected (default)
- 0x02 ... 0x14: Expected length of the Constant Tone Extension in 8 microseconds units.

CTE_Type

Values:

- 0x00: Expect AoA Constant Tone Extension
- 0x01: Expect AoD Constant Tone Extension with 1 microsecond slots
- 0x02: Expect AoD Constant Tone Extension with 2 microseconds slots

Switching_Pattern_Length

The number of Antenna IDs in the pattern. Values:

- 0x02 ... 0x4B

Antenna_IDs

List of Antenna IDs in the pattern.

Transmit_Power_Level

Values:

- -127 ... 20: Set transmitter to the specified or the nearest transmit power level.
- 126: Set transmitter to minimum transmit power level
- 127: Set transmitter to maximum transmit power level

Return values:

- *Value* indicating success or error code.

2.2.9 hci_le_create_big_test

```
tBleStatus hci_le_create_big_test    ( uint8_t BIG_Handle,
                                       uint8_t Advertising_Handle,
                                       uint8_t Num_BIS,
                                       uint8_t SDU_Interval[3],
                                       uint16_t ISO_Interval,
                                       uint8_t NSE,
                                       uint16_t Max_SDU,
                                       uint16_t Max_PDU,
                                       uint8_t PHY,
                                       uint8_t Packing,
                                       uint8_t Framing,
                                       uint8_t BN,
                                       uint8_t IRC,
                                       uint8_t  PTO,
                                       uint8_t Encryption,
                                       uint8_t Broadcast_Code[16]
                                      )
```

The HCI_LE_Create_BIG_Test command should only be used for testing purposes. The command is used to create one or more BISes of a BIG. All BISes in the BIG have the same values for all parameters.

Parameters

BIG_Handle

Used to identify the BIG. Values:

- 0x00 ... 0xEF

Advertising_Handle

Used to identify the periodic advertising train. Values:

- 0x00 ... 0xEF

Num_BIS

Total number of BISes in the BIG. Values:

- 0x01 ... 0x1F

SDU_Interval

The interval, in microseconds, of periodic SDUs. Values:

- 0x0000FF ... 0x0FFFFF

ISO_Interval

The time between consecutive BIG anchor points. Time = N

- 1.25 ms Time Range: 5 ms to 4 s Values:
- 0x0004 (5.00 ms) ... 0x0C80 (4000.00 ms)

Max_SDU

Maximum size of an SDU, in octets. Values:

- 0x0001 ... 0x0FFF

Max_PDU

Maximum size, in octets, of payload Values:

- 0x0001 ... 0x00FB

PHY

Transmitter PHY of packets. Flags:

- 0x01: LE_1M_PHY_BIT
- 0x02: LE_2M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

Packing

Used to indicate the preferred method of arranging subevents of multiple BISes. Values:

- 0x00: Sequential
- 0x01: Interleaved

Framing

The format for sending BIS Data PDUs. Values:

- 0x00: Unframed
- 0x01: Framed

BN

The number of new payloads in each interval for each BIS. Values:

- 0x01 ... 0x07

IRC

The number of times the scheduled payload(s) are transmitted in a given event. Values:

- 0x01 ... 0x0F

PTO

Offset used for pre-transmissions. Values:

- 0x00 ... 0x0F

Encryption

The encryption mode of the BISes. Values:

- 0x00: Unencrypted
- 0x01: Encrypted

Broadcast_Code

128-bit code used for deriving the session key for decrypting payloads of BISes in the BIG.

Return values:

- *Value* indicating success or error code.

2.2.10 hci_le_iso_read_test_counters

```
tBleStatus hci_le_iso_read_test_counters    ( uint16_t Connection_Handle,
                                              uint32_t * Received_Packet_Count,
                                              uint32_t * Missed_Packet_Count,
                                              uint32_t * Failed_Packet_Count
                                             )
```

The HCI_LE_ISO_Read_Test_Counters command should only be used in the ISO Test mode and only for testing purposes. The command is used to read the test counters (see Core 5.2 [Vol 6] Part B, Section 7) in the Controller, which is configured in ISO Receive Test mode for a CIS or BIS specified by the Connection_Handle. Reading the test counters does not reset the test counters.

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:

- 0x0000 ... 0x0EFF

`[out]` **Received_Packet_Count**

Number in the Received_Packet_Count.

`[out]` **Missed_Packet_Count**

Number in the Missed_Packet_Count.

`[out]` **Failed_Packet_Count**

Number in the Failed_Packet_Count.

Return values:

- *Value* indicating success or error code.

2.2.11 hci_le_iso_receive_test

```
tBleStatus hci_le_iso_receive_test    ( uint16_t Connection_Handle,
                                        uint8_t  Payload_Type
                                       )
```

The HCI_LE_ISO_Receive_Test command should only be used in the ISO test mode and only for testing purposes. The command is used to configure an established CIS or a synchronized BIG specified by the Connection_Handle parameter to receive payloads. When using this command for a BIS, the Host shall synchronize with a BIG using the HCI_LE_BIG_Create_Sync command before invoking this command.

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:
- 0x0000 ... 0x0EFF

Payload_Type

The Payload_Type parameter defines the configuration of SDUs in the payload. Values:
- 0x00: ZERO_LENGTH_PAYLOAD
- 0x01: VARIABLE_LENGTH_PAYLOAD
- 0x02: MAXIMUM_LENGTH_PAYLOAD

Return values:
- *Value* indicating success or error code.

2.2.12 hci_le_iso_test_end

```
tBleStatus hci_le_iso_test_end    ( uint16_t Connection_Handle,
                                    uint32_t * Received_Packet_Count,
                                    uint32_t * Missed_Packet_Count,
                                    uint32_t * Failed_Packet_Count
                                   )
```

The HCI_LE_ISO_Test_End command should only be used in the ISO Test mode and only for testing purposes. The command is used to terminate the ISO Transmit and/or Receive Test mode for a CIS or BIS specified by the Connection_Handle parameter but does not terminate the CIS or BIS. When the Host terminates the ISO Test mode for a CIS or BIS that is set to ISO Transmit Test mode only, the test counters in the return parameters shall be set to zero. When the Host terminates the ISO Test mode for a CIS or BIS that is set to the ISO Receive Test mode, the return parameters contain the values of the test counters as defined in [Vol 6] Part B, Section 7.

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:
- 0x0000 ... 0x0EFF

`[out]` **Received_Packet_Count**

Number in the Received_Packet_Count.

`[out]` **Missed_Packet_Count**

Number in the Missed_Packet_Count.

`[out]` **Failed_Packet_Count**

Number in the Failed_Packet_Count.

Return values:
- *Value* indicating success or error code.

2.2.13 hci_le_iso_transmit_test

```
tBleStatus hci_le_iso_transmit_test    ( uint16_t Connection_Handle,
                                         uint8_t  Payload_Type
                                        )
```

The HCI_LE_ISO_Transmit_Test command should only be used in the ISO Test mode and only for testing purposes. The command is used to configure an established CIS or BIS specified by the Connection_Handle parameter, and transmit test payloads which are generated by the Controller. When using this command for a CIS, the Host shall set up a CIG before invoking this command. When using this command for a BIS, the Host shall create the BIG before invoking this command. This command applies to all BISes in the BIG.

Parameters

Connection_Handle

Connection handle of the CIS or BIS. Values:

- 0x0000 ... 0x0EFF

Payload_Type

The Payload_Type parameter defines the configuration of SDUs in the payload. Values:

- 0x00: ZERO_LENGTH_PAYLOAD
- 0x01: VARIABLE_LENGTH_PAYLOAD
- 0x02: MAXIMUM_LENGTH_PAYLOAD

Return values:

- *Value* indicating success or error code.

2.2.14 hci_le_set_cig_parameters_test

```
tBleStatus hci_le_set_cig_parameters_test     ( uint8_t  CIG_ID,
                                                 uint8_t  SDU_Interval_C_To_P[3],
                                                 uint8_t  SDU_Interval_P_To_C[3],
                                                 uint8_t  FT_C_To_P,
                                                 uint8_t  FT_P_To_C,
                                                 uint16_t ISO_Interval,
                                                 uint8_t  Worst_Case_SCA,
                                                 uint8_t  Packing,
                                                 uint8_t  Framing,
                                                 uint8_t  CIS_Count,
                                                 CIS_Param_Test_t CIS_Param_Test[],
                                                 uint16_t Connection_Handle[]
                                                )
```

The HCI_LE_Set_CIG_Parameters_Test command should only be used for testing purposes. The command is used by a Central's Host to create a CIG and to set the parameters of one or more CISes that are associated with a CIG in the Controller. The CIG_ID parameter identifies a CIG. This parameter is allocated by the Central's Host and passed to the Peripheral's Host through the Link Layers during the process of creating a CIS. If the CIG_ID does not exist, then the Controller shall first create a new CIG. Once the CIG is created (whether through this command or previously), the Controller shall modify or add CIS configurations in the CIG that is identified by the CIG_ID and update all the parameters that apply to the CIG. The SDU_Interval_C_To_P parameter specifies the time interval of periodic SDUs from the Central's Host.

The SDU_Interval_P_To_C parameter specifies the time interval of periodic SDUs from the Peripheral's Host. The FT_C_To_P parameter identifies the maximum time for a payload from the Central to Peripheral to be transmitted and re-transmitted, after which it is flushed (see [Vol 6] Part B, Section 4.5.13.5). This parameter is expressed in multiples of ISO_Interval. The FT_P_To_C parameter identifies the maximum time for a payload from the Peripheral to Central to be transmitted and re-transmitted, after which it is flushed (see[Vol 6] Part B, Section 4.5.13.5). This parameter is expressed in multiples of ISO_Interval. The ISO_Interval parameter specifies the time between two consecutive CIS anchor points.

The CIS_Count parameter contains the number of CIS configurations being added or modified by this command. The Controller shall set the CIS_Count return parameter equal to this. The CIS_ID[i] parameter identifies the CIS and is set by the Central's Host and passed to the Peripheral's Host through the Link Layers during the process of establishing a CIS. The Worst_Case_SCA parameter is the worst-case sleep clock accuracy of all the Peripherals that participate in the CIG. The Host should get the sleep clock accuracy from all the Peripherals before issuing this command. In case the Host cannot get the sleep clock accuracy from all the Peripherals, it shall set the Worst_Case_SCA parameter to zero.

Note: *The Worst_Case_SCA parameter can be used by the Link Layer to better allow for clock drift when scheduling the CISes in the CIG. For example, if a CIS has more than two subevents, the Link Layer of the Central can set the timing of the subevents such that the worst case drift in the Peripheral's clock does not exceed 2 x Sub_Interval. This prevents the Peripheral from synchronizing its timing to the wrong subevent (adjacent subevents cannot be on the same channel).*

The Packing parameter is used to indicate the preferred method of arranging subevents of multiple CISes. The subevents can be arranged in Sequential or Interleaved arrangement. This is a recommendation to the Controller which it may ignore. This parameter shall be ignored when there is only one CIS in the CIG. The Framing parameter indicates the format of the CIS Data PDUs of all the CISes. If the Framing parameter is set to 1 then the CIS Data PDUs of the specified CISes shall be framed, and when set to 0 they shall be unframed (see [Vol 6] Part G, Section 1). The CIS_ID[i] parameter is used to identify a CIS. The NSE[i] parameter identifies the maximum number of subevents for each CIS in a CIG event. The Max_SDU_C_To_P[i] parameter identifies the maximum size of SDU from the Central's Host. If the CIS is unidirectional from Peripheral to Central, this parameter shall be set to 0.

If a CIS configuration that is being modified has a data path set in the Central to Peripheral direction and the Host has specified that Max_SDU_C_To_P[i] shall be set to zero, the Controller shall return the error code Command Disallowed (0x0C). The minimum value of the Max_SDU_Size parameter in the ISO Transmit Test mode when the Payload_Type = 1 or 2 shall be 4 octets. The Max_SDU_P_To_C[i] parameter identifies the maximum size of SDU from the Peripheral's Host. If the CIS is unidirectional from Central to Peripheral, this parameter shall be set to 0.

If a CIS configuration that is being modified has a data path set in the Peripheral to Central direction and the Host has specified that Max_SDU_P_To_C[i] shall be set to zero, the Controller shall return the error code Command Disallowed (0x0C).The minimum value of the Max_SDU parameter in the ISO Transmit Test mode when the Payload_Type = 1 or 2 shall be 4 octets. The Max_PDU_C_To_P[i] parameter identifies the maximum size PDU from the Central to Peripheral. The Max_PDU_P_To_C[i] parameter identifies the maximum size PDU from the Peripheral to Central. The PHY_C_To_P[i] parameter identifies the PHY to be used for transmission of packets from the Central to the Peripheral. The Host shall set only one bit in this parameter and the Controller shall use the PHY set by the Host. The PHY_P_To_C[i] parameter identifies the PHY to be used for transmission of packets from the Peripheral to the Central. The Host shall set only one bit in this parameter and the Controller shall use the PHY set by the Host. The BN_C_To_P[i] parameter identifies the burst number for Central to Peripheral (see [Vol 6] Part B, Section 4.5.13).

If the CIS is unidirectional from Peripheral to Central, this parameter shall be set to zero. The BN_P_To_C[i] parameter identifies the burst number for Peripheral to Central (see [Vol 6] Part B, Section 4.5.13). If the CIS is unidirectional from Central to Peripheral, this parameter shall be set to zero. If the Status return parameter is non-zero, then the state of the CIG and its CIS configurations shall not be changed by the command. If the CIG did not already exist, it shall not be created. If the Status return parameter is zero, then the Controller shall set the Connection_Handle arrayed return parameter to the connection handle(s) corresponding to the CIS configurations specified in the CIS_IDs command parameter, in the same order.

If the same CIS_ID is reconfigured, the same connection handle shall be returned. If the Host issues this command when the CIG is not in the configurable state, the Controller shall return the error code Command Disallowed (0x0C). If the Host attempts to create a CIG or set parameters that exceed the maximum supported resources in the Controller, the Controller shall return the error code Memory Capacity Exceeded (0x07). If the Host attempts to set CIS parameters that exceed the maximum supported connections in the Controller, the Controller shall return the error code Connection Limit Exceeded (0x09).

If the Host attempts to set an invalid combination of CIS parameters, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Host sets, in the PHY_C_To_P[i] or PHY_P_To_C[i] parameters, a bit for a PHY that the Controller does not support, including a bit that is reserved for future use, the Controller shall return the error code Unsupported Feature or Parameter Value (0x11). If the Controller does not support asymmetric PHYs and the Host sets PHY_C_To_P[i] to a different value than PHY_P_To_C[i], the Controller shall return the error code Unsupported Feature or Parameter Value (0x11).

Parameters

CIG_ID

Used to identify the CIG. Values:

- 0x00 ... 0xEF

SDU_Interval_C_To_P

The interval, in microseconds, of periodic SDUs. Values:

- 0x0000FF ... 0x0FFFFF

SDU_Interval_P_To_C

The interval, in microseconds, of periodic SDUs. Values:

- 0x0000FF ... 0x0FFFFF

FT_C_To_P

The flush timeout in multiples of ISO_Interval for each payload sent from the Central to Peripheral. Values:

- 0x01 ... 0xFF

FT_P_To_C

The flush timeout in multiples of ISO_Interval for each payload sent from the Peripheral to Central. Values:

- 0x01 ... 0xFF

ISO_Interval

Time between consecutive CIS anchor points. Time = N * 1.25 ms. Values:

- 0x0004 (5.00 ms) ... 0x0C80 (4000.00 ms)

Worst_Case_SCA

Worst-case sleep clock accuracy of all peripherals. Values:

- 0x00: 251 ppm to 500 ppm
- 0x01: 151 ppm to 250 ppm
- 0x02: 101 ppm to 150 ppm
- 0x03: 76 ppm to 100 ppm
- 0x04: 51 ppm to 75 ppm
- 0x05: 31 ppm to 50 ppm
- 0x06: 21 ppm to 30 ppm
- 0x07: 0 ppm to 20 ppm

Packing

Preferred method of arranging subevents of multiple CISes. Values:

- 0x00: Sequential
- 0x01: Interleaved

Framing

Format of the CIS Data PDUs of the specified CISes. Values:

- 0x00: Unframed
- 0x01: Framed

CIS_Count

Total number of CIS configurations in the CIG being added or modified. Values:

- 0x00 ... 0x1F

CIS_Param

See CIS_Param_t

`[out]` **Connection_Handle**

Connection handle of the CIS in the CIG.

Return values:

- *Value* indicating success or error code.

2.3 HAL commands

This section describes the supported HAL commands.

Table 3. **HAL commands opcodes**

SoC command	Network coprocessor command	OpCode
aci_hal_get_fw_build_number	aci_hal_get_fw_build_number	0xFC00

SoC command	Network coprocessor command	OpCode
-	aci_hal_get_firmware_details	0xFC01
-	aci_hal_get_firmware_details_v2	0xFC02
aci_hal_write_config_data	aci_hal_write_config_data	0xFC0C
aci_hal_read_config_data	aci_hal_read_config_data	0xFC0D
aci_hal_set_tx_power_level	aci_hal_set_tx_power_level	0xFC0F
aci_hal_le_tx_test_packet_number	aci_hal_le_tx_test_packet_number	0xFC14
aci_hal_tone_start	aci_hal_tone_start	0xFC15
aci_hal_tone_stop	aci_hal_tone_stop	0xFC16
aci_hal_get_link_status	aci_hal_get_link_status	0xFC17
aci_hal_set_radio_activity_mask	aci_hal_set_radio_activity_mask	0xFC18
aci_hal_set_le_power_control	aci_hal_set_le_power_control	0xFC1C
-	aci_hal_updater_start	0xFC20
-	aci_hal_updater_reboot	0xFC21
-	aci_hal_get_updater_version	0xFC22
-	aci_hal_get_updater_bufsize	0xFC23
-	aci_hal_updater_erase_blue_flag	0xFC24
-	aci_hal_updater_reset_blue_flag	0xFC25
-	aci_hal_updater_erase_sector	0xFC26
-	aci_hal_updater_prog_data_blk	0xFC27
-	aci_hal_updater_read_data_blk	0xFC28
-	aci_hal_updater_calc_crc	0xFC29
-	aci_hal_updater_hw_version	0xFC2A
-	aci_hal_transmitter_test_packets	0xFC2B
-	aci_hal_transmitter_test_packets_v2	0xFC2C
-	aci_hal_write_radio_reg	0xFC35
-	aci_hal_read_radio_reg	0xFC36
aci_hal_set_antenna_switch_parameters	aci_hal_set_antenna_switch_parameters	0xFC37
aci_hal_peripheral_latency_enable	aci_hal_peripheral_latency_enable	0xFC38
aci_hal_get_evt_fifo_max_level	aci_hal_get_evt_fifo_max_level	0xFC60
-	aci_test_tx_notification_start	0xFE00
-	aci_test_tx_write_command_start	0xFE01
-	aci_test_rx_start	0xFE02
-	aci_test_stop	0xFE03
-	aci_test_report	0xFE04
ll_set_legacy_advertising_data_ptr	-	0xFE80
ll_set_legacy_scan_reponse_data_ptr	-	0xFE81
ll_set_advertising_data_ptr	-	0xFE82
ll_set_scan_reponse_data_ptr	-	0xFE83
ll_get_advertising_info	-	0xFE84
ll_set_periodic_advertising_data_ptr	-	0xFE85
aci_hal_get_anchor_point	-	0xFE86

SoC command	Network coprocessor command	OpCode
ll_set_periodic_advertising_subevent_data_ptr	-	0xFE87
ll_set_periodic_advertising_response_data_ptr	-	0xFE88

2.3.1 aci_hal_get_anchor_point

```
tBleStatus aci_hal_get_anchor_point    ( uint16_t Connection_Handle,
                                          uint16_t * Event_Counter,
                                          uint32_t * Anchor_Point
                                        )
```

Function to get the value of the last anchor point for the given connection.

Parameters

Connection_Handle

`[out]` **Event_Counter**

`[out]` **Anchor_Point**

Return values:

- *Value* indicating success or error code.

2.3.2 aci_hal_get_evt_fifo_max_level

```
tBleStatus aci_hal_get_evt_fifo_max_level    ( uint16_t * ISR0_FIFO_Max_Level,
                                                uint16_t * ISR1_FIFO_Max_Level,
                                                uint16_t * User_FIFO_Max_Level
                                              )
```

This command can be used to get the maximum used size of the stack's internal FIFO queues: isr1_fifo, isr2_fifo and user_fifo. These values can be used to chose the maximum correct size for the queues, which can be set through the BLE_STACK_Init() function. If one of these queues reaches the maximum size, an hci_hardware_error_event_rp0 is raised, with error code 0x03.

Parameters

`[out]` **ISR0_FIFO_Max_Level**

Maximum size reached by the FIFO used for critical controller events produced by the ISR (e.g. rx data packets). See isr0_fifo_size parameter field of BLE_STACK_InitTypeDef structure, used by BLE_STACK_Init().

`[out]` **ISR1_FIFO_Max_Level**

Maximum size reached by the FIFO used for non-critical controller events produced by the ISR (e.g. advertising or IQ sampling reports). See isr1_fifo_size parameter field of BLE_STACK_InitTypeDef structure, used by BLE_STACK_Init().

`[out]` **User_FIFO_Max_Level**

Maximum size reached by the FIFO used for controller and host events produced outside the ISR. See user_fifo_size parameter field of BLE_STACK_InitTypeDef structure, used by BLE_STACK_Init().

Return values:

- *Value* indicating success or error code.

2.3.3 aci_hal_get_fw_build_number

```
tBleStatus aci_hal_get_fw_build_number ( uint16_t * Build_Number )
```

This command returns the build number associated with the firmware version currently running.

Parameters

`[out]` **Build_Number**

Build number of the firmware.

Return values:

- *Value* indicating success or error code.

2.3.4 aci_hal_get_link_status

```
tBleStatus aci_hal_get_link_status    ( uint8_t  Bank_index,
                                        uint8_t  Link_Status[8],
                                        uint16_t Link_Connection_Handle[16/2]
                                       )
```

This command returns the status of the Bluetooth® LE links managed by the device.

Parameters

Bank_index

Index that identifies the link bank. Each bank is made by 8 links. Set Bank_Index to 0 to retrieve the status of the first 8 links, Bank_Index 1 to retrieve the status of the second 8 links, and so on. Values:

- 0x00 ... 0x15

`[out]` **Link_Status**

Array of link status (8 links). Each link status is 1 byte.

- 0x00: Idle
- 0x01: Advertising
- 0x02: Connected as peripheral
- 0x03: Scanning
- 0x04: Initiating
- 0x05: Connected as central
- 0x06: TX test mode
- 0x07: RX test mode

`[out]` **Link_Connection_Handle**

Array of connection handles (2 bytes) for 8 links.

Return values:

- *Value* indicating success or error code.

2.3.5 aci_hal_le_tx_test_packet_number

```
tBleStatus aci_hal_le_tx_test_packet_number ( uint32_t * Number_Of_Packets )
```

This command returns the number of packets sent in Direct Test Mode. When the Direct TX test is started, a 32-bit counter is used to count how many packets have been transmitted. This command can be used to check how many packets have been sent during the Direct TX test. The counter starts from 0 and counts upwards. The counter can wrap and start from 0 again. The counter is not cleared until the next Direct TX test starts.

Parameters

`[out]` **Number_Of_Packets**

Number of packets sent during the last Direct TX test.

Return values:

- *Value* indicating success or error code.

2.3.6 aci_hal_peripheral_latency_enable

```
tBleStatus aci_hal_peripheral_latency_enable    ( uint16_t Connection_Handle,
                                                   uint8_t Enable
                                                  )
```

Command used on the peripheral to force the Link Layer to keep the peripheral latency disabled. Peripheral latency is enabled by default on a connection where peripheral latency has been set to a value greater than zero by the Central when connection is established. Disabling the peripheral latency is useful to immediately reduce latency from Central to Peripheral. Note that when peripheral latency is enabled, the Link Layer uses the first available connection event to transfer the queued data, so there is no need to force the disabling of the peripheral latency to reduce latency from Peripheral to Central.

This command returns BLE_ERROR_UNKNOWN_CONNECTION_ID if the connection handle does not exist. BLE_ERROR_COMMAND_DISALLOWED is returned if the command is given when the device is not in the peripheral role on the connection handle.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Enable

Enable or disable the peripheral latency. Default value is Enabled (1). Values:

- 0x00: DISABLED
- 0x01: ENABLED

Return values:

- *Value* indicating success or error code.

2.3.7 aci_hal_read_config_data

```
tBleStatus aci_hal_read_config_data    ( uint8_t Offset,
                                          uint8_t * Data_Length,
                                          uint8_t Data[]
                                         )
```

This command requests the value in the low level configure data structure. The number of read bytes changes for different offsets.

Parameters

Offset

Offset of the element in the configuration data structure which has to be read. Values:

- 0x00: CONFIG_DATA_PUBADDR_OFFSET. Bluetooth® public address; value length returned: 6 bytes
- 0x08: CONFIG_DATA_ER_OFFSET. Encryption root key used to derive LTK and CSRK; value length returned: 16 bytes.
- 0x18: CONFIG_DATA_IR_OFFSET. Identity root key used to derive LTK and CSRK; value length returned: 16 bytes.
- 0x2C: LL_WITHOUT_HOST. Link layer without host (for certification purposes); value length returned: 1 byte.
- 0x80: CONFIG_DATA_STORED_STATIC_RANDOM_ADDRESS. The static random address stored in NVM. Value length returned: 6 bytes (read-only).

`[out]` **Data_Length**

Length of Data in octets

`[out]` **Data**

Data field associated with Offset parameter

Return values:

- *Value* indicating success or error code.

2.3.8 aci_hal_set_le_power_control

```
tBleStatus aci_hal_set_le_power_control    ( uint8_t  Enable,
                                              uint8_t PHY,
                                              int8_t  RSSI_Target,
                                              uint8_t RSSI_Hysteresis,
                                              int8_t  Initial_TX_Power,
                                              uint8_t RSSI_Filtering_Coefficient
                                             )
```

This command is used to enable or disable the LE Power Control feature and procedure for a given PHY on the later established connections. It also provides the parameters that let the Controller initiate the LE Power Control procedure. In particular, the procedure is initiated when the current (average) RSSI (say Curr_Avg_RSSI) gets: 1) less than (RSSI_Target - RSSI_Hysteresis) and the Controller requests the peer to increase its TX power level for the given PHY by (RSSI_Target - Curr_Avg_RSSI); 2) greater than (RSSI_Target + RSSI_Hysteresis) and the Controller requests the peer to decrease its TX power level for the given PHY by (Curr_Avg_RSSI - RSSI_Target).

The Controller starts transmitting on the connections for which the power control is enabled and for the given PHY using the Initial_Tx_Power value. It changes its TX power based on the requests or feedbacks from the peer:

1. If the peer initiates an LE Power Control procedure and requests to increase or decrease the TX power of a given delta, the TX power is increased or reduced by the requested delta within the acceptable limits.
2. If the peer reports that it can accept a TX power reduction of a given delta, the TX power is reduced by the reported delta within the acceptable limits.

If this command is not issued, the Controller uses the parameter default values.

Parameters

Enable

Enable or disable LE power control on following connections. Default: 1. Values:

- 0x00: DISABLE
- 0x01: ENABLE

PHY

PHY on which the power control must be enabled or disabled. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY_S8
- 0x04: LE_CODED_PHY_S2

RSSI_Target

Target RSSI in dBm. Default: -55 dBm.

RSSI_Hysteresis

Hysteresis applied on the target RSSI in dB. Default: 15 dB.

Initial_TX_Power

Initial TX power in dBm. Default: max TX power supported by the platform.

RSSI_Filtering_Coefficient

Coefficient used for the filtering of the RSSI samples and the calculation of the average RSSI. Allowed values are from 0 (fast moving average, low accuracy, max weight of last RSSI) to 4 (slow moving average, high accuracy, min weight of last RSSI). Default: 2. Values:

- 0x00 ... 0x04

Return values:

- *Value* indicating success or error code.

2.3.9 aci_hal_set_radio_activity_mask

```
tBleStatus aci_hal_set_radio_activity_mask ( uint16_t Radio_Activity_Mask )
```

This command set the bitmask associated to aci_hal_end_of_radio_activity_event_rp0. Only the radio activities enabled in the mask are reported to application by aci_hal_end_of_radio_activity_event_rp0.

Parameters

Radio_Activity_Mask

Bitmask of radio events. Flags:

- 0x0001: Idle
- 0x0002: Advertising
- 0x0004: Connection event peripheral
- 0x0008: Scanning
- 0x0010: Connection request
- 0x0020: Connection event central
- 0x0040: TX test mode
- 0x0080: RX test mode

Return values:

- *Value* indicating success or error code.

2.3.10 aci_hal_set_tx_power_level

```
tBleStatus aci_hal_set_tx_power_level     ( uint8_t En_High_Power,
                                              uint8_t PA_Level
                                             )
```

This command sets the TX power level of the device for all the radio activities, unless explicitly defined by other commands. The combination of En_High_Power and PA_Level parameters determines the output power level (dBm). When the system starts up or reboots, the default TX power level is used, which is En_High_Power = 0 and PA_Level = 31. The En_High_Power is used to change the SMPS level. When the parameter is set to 0, SMPS level is set to 1.4V. When the parameter is set to 1, SMPS level is set to 1.9V. However, if SMPS is disabled, no action is done on the SMPS level, since the voltage must be applied externally on the VFBSD pin. To reach 8 dBm of TX power, En_High_Power must be set to 1. If SMPS is not used, this voltage level must be applied to the VFBSD pin. On devices other then STM32WB09, setting En_High_Power to 1 also bypasses the LDO_TRANSFO during transmission.

On STM32WB09 devices, PA_Level 32 is used to specify the configuration to reach 8 dBm, to be used only when En_High_Power is set to 1. In this configuration, LDO_TRANSFO is bypassed during transmission, by setting TxHp bit. The output power for other PA levels remain almost equal to the case when En_High_Power is set to 0, since LDO_TRANSFO is not bypassed (TxHp bit set to 0). On devices other than STM32WB09, (that is, where TxHp bit is not present), PA level 32 cannot be used. The 8 dBm configuration can be selected by using PA_Level 31, with En_High_Power set to 1. However, when setting En_High_Power to 1, also the output power of other PA levels less than 31 changes (especially for the higher ones) since LDO_TRANSFO is always bypassed during transmission for all the PA levels. Note that, if En_High_Power = 1 and SMPS is disabled, when LDO_TRANSFO is bypassed the actual output power depends on the voltage applied to the VFBSD pin.

In case the voltage is not 1.9V, it is recommended to change the TX output levels specified in the high_power_pa_level_table. The real output power may also depend on PCB layout and associated components. After this command is given, the new output power is used only for new Link Layer state machines, that is, for new Bluetooth® activities, that is, new advertising sets, new connections, new scanning procedures. For current activities the output power does not change. The only exception is AUX_SCAN_REQ and AUX_CONNECT_REQ PDUs, for which the new output power also takes affect immediately for the current scanning procedure.

Parameters

En_High_Power

Enable High Power mode, by changing SMPS level. High power mode should be enabled only to reach the maximum output power. Normal power (0x00) is the default. Values:

- 0x00: Normal Power
- 0x01: High Power

PA_Level

Power amplifier output level.

Important: *PA_Level 32 is available only for STM32WB09, to select 8 dBm of output power.*

Values:

- 0: -54/-54 dBm
- 1: -21/-19 dBm
- 2: -20/-18 dBm
- 3: -19/-17 dBm
- 4: -17/-16 dBm
- 5: -16/-15 dBm
- 6: -15/-14 dBm
- 7: -14/-13 dBm
- 8: -13/-12 dBm
- 9: -12/-11 dBm
- 10: -11/-10 dBm
- 11: -10/-9 dBm
- 12: -9/-8 dBm
- 13: -8/-7 dBm
- 14: -7/-6 dBm
- 15: -6/-5 dBm
- 16: -6/-4 dBm
- 17: -4/-3 dBm
- 18: -3/-3 dBm
- 19: -3/-2 dBm
- 20: -2/-1 dBm
- 21: -2/+0 dBm
- 22: -1/+1 dBm
- 23: -1/+2 dBm
- 24: +0/+3 dBm
- 25: +0/+8 dBm
- 26: +1/+8 dBm
- 27: +2/+8 dBm
- 28: +3/+8 dBm
- 29: +4/+8 dBm
- 30: +5/+8 dBm
- 31: +6/+8 dBm
- 32: +8 dBm

Return values:

- *Value* indicating success or error code.

2.3.11 aci_hal_write_config_data

```
tBleStatus aci_hal_write_config_data    ( uint8_t Offset,
                                          uint8_t Length,
                                          uint8_t Value[]
                                        )
```

This command writes a value to a low level configure data structure. It is useful to set up directly some low-level parameters for the system in the runtime.

Note: *This command shall not be called if a command different than Stack Init, HCI_RESET, ACI_HAL_WRITE_CONFIG_DATA, or ACI_HAL_READ_CONFIG_DATA has already been called.*

Parameters

Offset

Offset of the element in the configuration data structure which has to be written. The valid offsets are:

- 0x00: CONFIG_DATA_PUBADDR_OFFSET. Bluetooth® public address; value length to be written: 6 bytes.
- 0x08: CONFIG_DATA_ER_OFFSET. Encryption root key used to derive LTK and CSRK; value length to be written: 16 bytes.
- 0x18: CONFIG_DATA_IR_OFFSET. Identity root key used to derive LTK and CSRK; value length to be written: 16 bytes.
- 0x2C: LL_WITHOUT_HOST. Link layer without host (for certification purposes); value length to be written: 1 byte.
- 0x2E: CONFIG_DATA_STATIC_RANDOM_ADDRESS. If set, the stack uses this address as the static random address instead of the one stored in NVM.
- 0x2F: CONFIG_DATA_SCAN_CH_MAP. The value is a bit field that indicates the advertising channel indices that shall be used when scanning. At least one channel bit shall be set.
  - Bit 0: channel 37
  - Bit 1: channel 38
  - Bit 2: channel 39
- 0xD0: CONFIG_DATA_DEBUG_KEY. Use debug key for Secure connection: 1 byte.
- 0xD1: CONFIG_DATA_DLE. Set the maximum allowed parameter values for Data Length Extension: 8 bytes; 2 bytes for each of the following parameters: supportedMaxTxOctets, supportedMaxTxTime, supportedMaxRxOctets, and supportedMaxRxTime, in little-endian order (default 251, 2120, 251, 2120).

Length

Length of data to be written

Value

Data to be written

Return values:

- *Value* indicating success or error code.

2.3.12 ll_get_advertising_info

```
tBleStatus ll_get_advertising_info    ( uint16_t Advertising_Handle,
                                        uint8_t  * Adv_Enabled,
                                        uint8_t  * Periodic_Adv_Configured,
                                        uint8_t  * Periodic_Adv_Enabled,
                                        uint16_t * Advertising_Event_Properties
                                      )
```

Retrieves info about an existing advertising set.

Parameters

Advertising_Handle

Used to identify an advertising set. Values:

- 0x0000 ... 0x00EF

`[out]` **Adv_Enabled**

It is 1 (TRUE) if advertising is enabled for the given advertising handle, 0 (FALSE) otherwise.

`[out]` **Periodic_Adv_Configured**

It is 1 (TRUE) if periodic advertising has been configured for the given advertising handle, 0 (FALSE) otherwise.

`[out]` **Periodic_Adv_Enabled**

It is 1 (TRUE) if periodic advertising has been enabled for the given advertising handle, 0 (FALSE) otherwise.
Note: periodic advertising may be enabled but not started; in this case Adv_Enabled is false and Periodic_Adv_Enable is true.

`[out]` **Advertising_Event_Properties**

Advertising event properties that have been previously set for the advertising set.

Return values:

- *Value* indicating success or error code.

2.3.13 ll_set_advertising_data_ptr

```
tBleStatus ll_set_advertising_data_ptr    ( uint16_t Advertising_Handle,
                                              uint8_t  Operation,
                                              uint16_t Advertising_Data_Length,
                                              uint8_t  Advertising_Data[]
                                             )
```

Set data pointer for extended advertising data.

Parameters

Advertising_Handle

Used to identify an advertising set. This parameter is only meaningful if Extended Advertising Feature is enabled. Values:

- 0x0000 ... 0x00EF

Operation

If set to Unchanged data, just update the Advertising DID. Values:

- 0x03: Complete data
- 0x04: Unchanged data

Advertising_Data_Length

Length of advertising data.

Advertising_Data

Pointer to the buffer containing properly formatted advertising data (see Core v5.2 Vol 3, part C, chapter 11). Its content must not change, until an aci_hal_adv_scan_resp_data_update_event_rp0 is received, which informs the application that the buffer is no more used by the Bluetooth® stack.

Return values:

- *Value* indicating success or error code.

2.3.14 ll_set_legacy_advertising_data_ptr

```
tBleStatus ll_set_legacy_advertising_data_ptr    ( uint8_t Data_Length,
                                                    uint8_t * Data
                                                   )
```

Set data pointer for legacy advertising data.

Parameters

Data_Length

Data length

Data

Pointer to the data buffer

Return values:

- *Value* indicating success or error code.

2.3.15 ll_set_legacy_scan_reponse_data_ptr

```
tBleStatus ll_set_legacy_scan_reponse_data_ptr    ( uint8_t Data_Length,
                                                    uint8_t * Data
                                                  )
```

Set data pointer for legacy advertising data.

Parameters

Data_Length

Data length

Data

Pointer to the data buffer

Return values:

- *Value* indicating success or error code.

2.3.16 ll_set_periodic_advertising_data_ptr

```
tBleStatus ll_set_periodic_advertising_data_ptr    ( uint16_t Advertising_Handle,
                                                     uint8_t  Operation,
                                                     uint16_t Advertising_Data_Length,
                                                     uint8_t  Advertising_Data[]
                                                   )
```

Set data pointer for periodic extended advertising data.

Parameters

Advertising_Handle

Used to identify an advertising set. This parameter is only meaningful if Extended Advertising Feature is enabled. Values:

- 0x0000 ... 0x00EF

Operation

If set to Unchanged data, just update the Advertising DID. Values:

- 0x03: Complete data
- 0x04: Unchanged data

Advertising_Data_Length

Length of periodic advertising data.

Advertising_Data

Pointer to the buffer containing properly formatted periodic advertising data (see Core v5.2 Vol 3, part C, chapter 11). Its content must not change, until an aci_hal_adv_scan_resp_data_update_event_rp0 is received, which informs the application that the buffer is no more used by the Bluetooth® stack.

Return values:

- *Value* indicating success or error code.

2.3.17 ll_set_periodic_advertising_response_data_ptr

```
tBleStatus ll_set_periodic_advertising_response_data_ptr    ( uint16_t Sync_Handle,
                                                              uint16_t Request_Event,
                                                              uint8_t  Request_Subevent,
                                                              uint8_t  Response_Subevent,
                                                              uint8_t  Response_Slot,
                                                              uint8_t  Response_Data_Length,
                                                              uint8_t  * Response_Data
                                                            )
```

The LL_Set_Periodic_Advertising_Response_Data_Ptr command is used by the Host to set the data for a response slot in a specific subevent of the PAwR identified by the Sync_Handle. The data for a response slot shall be transmitted only once. The Request_Event parameter identifies the periodic advertising event in which the periodic advertising packet that the Host is responding to was received. The Request_Subevent parameter identifies the subevent in which the periodic advertising packet that the Host is responding to was received. The Response_Subevent parameter identifies the subevent that the response shall be sent in. The Response_Slot parameter identifies the response slot in the subevent identified by the Response_Subevent parameter in which this response data is to be transmitted. The Response_Data_Length specifies the length of the Response_Data that is significant. The Response_Data contains the advertising data to be transmitted in the response slot.

If the Response_Data_Length is greater than the maximum that the Controller can transmit within the response slot, then the Response_Data shall be discarded and the Controller shall return the error code Packet Too Long (0x45). If advertising on the LE Coded PHY, then the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. If the response slot identified by the Response_Slot parameter has passed by the time this command is received by the Controller, the Controller shall return the error code Too Late (0x46) and discard the Response_Data parameter.

Parameters

Sync_Handle

Sync_Handle identifying the PAwR train. Values:

- 0x0000 ... 0x0EFF

Request_Event

The value of paEventCounter (see [Vol 6] Part B, Section 4.4.2.1) for the periodic advertising packet that the Host is responding to.

Request_Subevent

Used to identify the subevent of the PAwR train. Values:

- 0x00 ... 0x7F

Response_Subevent

Used to identify the response slot of the PAwR train.

Response_Slot

Used to identify the response slot of the PAwR train.

Response_Data_Length

The number of octets in the Response_Data parameter. Values:

- 0x00 ... 0xFB

Response_Data

Pointer to response data formatted as defined in [Vol 3] Part C, Section 11.

Return values:

- *Value* indicating success or error code.

2.3.18 ll_set_periodic_advertising_subevent_data_ptr

```
tBleStatus ll_set_periodic_advertising_subevent_data_ptr ( uint8_t Advertising_Handle,
                                                            uint8_t Num_Subevents,
                                                            Subevent_Data_Ptr_Parameters_t
                                                            Subevent_Data_Ptr_Parameters[]
                                                          )
```

Parameters

Advertising_Handle

It is used to identify an advertising set. Values:

- 0x00 ... 0xEF: Used to identify an advertising set

Request_Event

The value of paEventCounter (see [Vol 6] Part B, Section 4.4.2.1) for the periodic advertising packet that the Host is responding to.

Num_Subevents

Number of subevent data in the command. Values:

- 0x01 ... 0x0F

Subevent_Data_Ptr_Parameters

See Subevent_Data_Ptr_Parameters_t

Return values:

- *Value* indicating success or error code.

2.3.19 ll_set_scan_reponse_data_ptr

```
tBleStatus ll_set_scan_reponse_data_ptr    ( uint16_t Advertising_Handle,
                                              uint16_t Scan_Response_Data_Length,
                                              uint8_t  Scan_Response_Data[]
                                            )
```

Set data pointer for extended scan response data.

Parameters

Advertising_Handle

Used to identify an advertising set. This parameter is only meaningful if Extended Advertising Feature is enabled. Values:

- 0x0000 ... 0x00EF

Scan_Response_Data_Length

Length of scan response data. If the advertising set uses scannable legacy advertising PDUs maximum length is 31 octets.

Scan_Response_Data

Pointer to the buffer containing properly formatted scan response data (see Core v5.1 Vol 3, part C, chapter 11). Its content must not change, until an aci_hal_adv_scan_resp_data_update_event_rp0 is received, which informs the application that the buffer is no more used by the Bluetooth® stack.

Subevent_Data_Ptr_Parameters

See Subevent_Data_Ptr_Parameters_t

Return values:

- *Value* indicating success or error code.

2.3.20 aci_hal_get_firmware_details

```
tBleStatus aci_hal_get_firmware_details(uint8_t *DTM_version_major,
                                        uint8_t *DTM_version_minor,
                                        uint8_t *DTM_version_patch,
                                        uint8_t *DTM_variant,
                                        uint16_t *DTM_Build_Number,
                                        uint8_t *BTLE_Stack_version_major,
                                        uint8_t *BTLE_Stack_version_minor,
                                        uint8_t *BTLE_Stack_version_patch,
                                        uint8_t *BTLE_Stack_development,
                                        uint16_t *BTLE_Stack_variant,
                                        uint16_t *BTLE_Stack_Build_Number)
```

This command return information regarding the version of the network coprocessor firmware and Bluetooth® LE stack library associated. The information returned includes values that can be retrieved with existing commands (refer to hci_read_local_version_information and aci_hal_get_fw_build_number). The aim is to have a single command that returns all version information details for a network coprocessor application (also known as BLE_TransparentMode or DTM application).

Parameters

`[out]` **DTM_version_major**

Major version number of the DTM application part.

`[out]` **DTM_version_minor**

Minor version number of the DTM application part.

`[out]` **DTM_version_patch**

Patch version number of the DTM application part.

`[out]` **DTM_variant**

Transport layer mode (numbers not defined reserved for future use). Values:

- 0x01: UART
- 0x02: SPI

`[out]` **DTM_Build_Number**

Build number for DTM application part.

`[out]` **BTLE_Stack_version_major**

Major version number of Bluetooth® LE stack.

`[out]` **BTLE_Stack_version_minor**

Minor version number of Bluetooth® LE stack.

`[out]` **BTLE_Stack_version_patch**

Patch version number of Bluetooth® LE stack.

`[out]` **BTLE_Stack_development**

Specific variant build. Values:

- 0x00: Official release
- 0x01: Internal development release

`[out]` **BTLE_Stack_variant**

Bitmask of Bluetooth® LE stack v4.x or later variants (modular configurations options and link layer only). Flags:

- 0x0001: CONTROLLER_PRIVACY_ENABLED
- 0x0002: SECURE_CONNECTIONS_ENABLED
- 0x0004: CONTROLLER_SCAN_ENABLED
- 0x0008: CONTROLLER_DATA_LENGTH_EXTENSION_ENABLED
- 0x0010: LINK LAYER ONLY
- 0x0020: CONTROLLER_2M_CODED_PHY_ENABLED
- 0x0040: CONTROLLER_EXT_ADV_SCAN_ENABLED
- 0x0080: L2CAP_COS_ENABLED
- 0x0100: CONTROLLER_PERIODIC_ADV_ENABLED
- 0x0200: CONTROLLER_CTE_ENABLED
- 0x0400: CONTROLLER_POWER_CONTROL_ENABLED
- 0x0800: CONNECTION_ENABLED

`[out]` **BTLE_Stack_Build_Number**

Build number for Bluetooth® LE stack.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.21 aci_hal_get_firmware_details_v2

```
tBleStatus aci_hal_get_firmware_details_v2(uint8_t *DTM_version_major,
                                           uint8_t *DTM_version_minor,
                                           uint8_t *DTM_version_patch,
                                           uint8_t *DTM_variant,
                                           uint16_t *DTM_Build_Number,
                                           uint8_t *BTLE_Stack_version_major,
                                           uint8_t *BTLE_Stack_version_minor,
                                           uint8_t *BTLE_Stack_version_patch,
                                           uint8_t *BTLE_Stack_development,
                                           uint32_t *BTLE_Stack_variant,
                                           uint16_t *BTLE_Stack_Build_Number)
```

This command return information regarding the version of the network coprocessor firmware and Bluetooth® LE stack library associated. The information returned includes values that can be retrieved with existing commands (refer to hci_read_local_version_information and aci_hal_get_fw_build_number). The aim is to have a single command that returns all version information details for a network coprocessor application (also known as BLE_TransparentMode or DTM application).

Parameters

`[out]` **DTM_version_major**

Major version number of the DTM application part.

`[out]` **DTM_version_minor**

Minor version number of the DTM application part.

`[out]` **DTM_version_patch**

Patch version number of the DTM application part.

`[out]` **DTM_variant**

Transport layer mode (numbers not defined reserved for future use). Values:
- 0x01: UART
- 0x02: SPI

`[out]` **DTM_Build_Number**

Build number for DTM application part.

`[out]` **BTLE_Stack_version_major**

Major version number of Bluetooth® LE stack.

`[out]` **BTLE_Stack_version_minor**

Minor version number of Bluetooth® LE stack.

`[out]` **BTLE_Stack_version_patch**

Patch version number of Bluetooth® LE stack.

`[out]` **BTLE_Stack_development**

Specific variant build. Values:
- 0x00: Official release
- 0x01: Internal development release

`[out]` **BTLE_Stack_variant**

Bitmask of Bluetooth® LE stack v4.x or later variants (modular configurations options and link layer only). Flags:
- 0x00000001: CONTROLLER_PRIVACY_ENABLED
- 0x00000002: SECURE_CONNECTIONS_ENABLED
- 0x00000004: CONTROLLER_SCAN_ENABLED
- 0x00000008: CONTROLLER_DATA_LENGTH_EXTENSION_ENABLED
- 0x00000010: LINK LAYER ONLY
- 0x00000020: CONTROLLER_2M_CODED_PHY_ENABLED
- 0x00000040: CONTROLLER_EXT_ADV_SCAN_ENABLED
- 0x00000080: L2CAP_COS_ENABLED
- 0x00000100: CONTROLLER_PERIODIC_ADV_ENABLED
- 0x00000200: CONTROLLER_CTE_ENABLED
- 0x00000400: CONTROLLER_POWER_CONTROL_ENABLED
- 0x00000800: CONNECTION_ENABLED
- 0x00010000: CONTROLLER_CHAN_CLASS_ENABLED
- 0x00020000: CONTROLLER_BIS_ENABLED
- 0x00080000: CONNECTION_SUBRATING_ENABLED
- 0x00100000: CONTROLLER_CIS_ENABLED
- 0x00400000: CONTROLLER_PERIODIC_ADV_WR

`[out]` **BTLE_Stack_Build_Number**

Build number for Bluetooth® LE stack.

Return values:
- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.3.22 aci_hal_updater_start

```
tBleStatus aci_hal_updater_start(void)
```

This command is only implemented together with the normal application. The updater does not support this command. If this command is called, the system reboots and enters updater mode.

Parameters

None.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.23 aci_hal_updater_reboot

```
tBleStatus aci_hal_updater_reboot(void)
```

This command reboots the system. This command does not set the BLUE flag, which must be done by another command.

Parameters

None.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.24 aci_hal_get_updater_version

```
tBleStatus aci_hal_get_updater_version(uint8_t *Version)
```

This command returns the version of the updater.

Parameters

`[out]` Version

Updater version

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.25 aci_hal_get_updater_bufsize

```
tBleStatus aci_hal_get_updater_bufsize(uint8_t *Buffer_Size)
```

It returns the maximum buffer size. This value limits the size of the data. blocks that could be used on the command aci_hal_updater_prog_data_blk.

Parameters

`[out]` Version

Updater version

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.26 aci_hal_updater_erase_blue_flag

```
tBleStatus aci_hal_updater_erase_blue_flag(void)
```

This command erases the BLUE flag in the Flash. After this operation, the updater cannot jump to the firmware until the BLUE flag is set to a valid value with aci_hal_updater_reset_blue_flag. This command is strongly recommended when the updater wants to upgrade the firmware application.

Parameters

None.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.27 aci_hal_updater_reset_blue_flag

```
tBleStatus aci_hal_updater_reset_blue_flag(void)
```

Resets the BLUE flag to its proper value. This command must be called when the firmware upgrade is finished. So that after reboot, the update may jump to the firmware application.

Parameters

None.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.28 aci_hal_updater_erase_sector

```
tBleStatus aci_hal_updater_erase_sector(uint32_t Address)
```

This command erases one sector of the Flash memory. One sector is 2 KB. After erasing, the sector is all 0xFF.

Parameters

Address

Address on sector

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.29 aci_hal_updater_prog_data_blk

```
tBleStatus aci_hal_updater_prog_data_blk(uint32_t Address,
                                         uint16_t Data_Length,
                                         uint8_t Data[])
```

This command writes a block of data to the Flash, starting from the given base address.

Parameters

Address

Base address

Data_Length

Length of data in octets

Data

Data to be written

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.30 aci_hal_updater_read_data_blk

```
tBleStatus aci_hal_updater_read_data_blk(uint32_t Address,
                                         uint16_t Data_Length,
                                         uint8_t Data[])
```

This command reads a block of data from the Flash, starting from the given base address. It is only allowed to read from the IFR flash. The Base Address must be bigger than 0x10020000.

Parameters

Address

Base address

Data_Length

Length of data in octets

[out] Data

Read data

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.31 aci_hal_updater_calc_crc

```
tBleStatus aci_hal_updater_calc_crc(uint32_t Address,
                                    uint8_t Num_Of_Sectors,
                                    uint32_t *crc)
```

It calculates the CRC32 of one or more Flash sectors. One Flash sector is 2 KB.

Parameters

Address

Base address

Num_Of_Sectors

Number of sectors

[out] crc

Crc value

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.32 aci_hal_updater_hw_version

```
tBleStatus aci_hal_updater_hw_version(uint8_t *HW_Version)
```

It gives device ID and cut version. See DIE_ID register.

Parameters

`[out]` **HW_Version**

It is the content of Die ID register

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.33 aci_hal_transmitter_test_packets

```
tBleStatus aci_hal_transmitter_test_packets(uint8_t TX_Frequency,
                                            uint8_t Length_Of_Test_Data,
                                            uint8_t Packet_Payload,
                                            uint16_t Number_Of_Packets,
                                            uint8_t PHY)
```

This command is equivalent to the corresponding Bluetooth® LE standard command HCI_LE_TRANSMITTER_TEST, with an additional parameter to specify the number of packets to be transmitted. An HCI_COMMAND_STATUS_EVENT is sent after the command is received. An ACI_HAL_LE_TEST_END_EVENT is generated when the number of specified packets have been sent.

Parameters

TX_Frequency

N = (F - 2402) / 2. Frequency range : 2402 MHz to 2480 MHz. Values:

- 0x00 ... 0x27

Length_Of_Test_Data

Length in bytes of payload data in each packet. Supported ranges:

- (0x00,0x25): Bluetooth® LE stack version < 2.1
- (0x00,0xFF): Bluetooth® LE stack version >= 2.1 and extended packet length.

Packet_Payload Type of packet payload. Values:

- 0x01: Pattern of alternating bits '11110000'
- 0x02: Pattern of alternating bits '10101010'
- 0x03: Pseudo-Random bit sequence 15
- 0x04: Pattern of All '1' bits
- 0x05: Pattern of All '0' bits
- 0x06: Pattern of alternating bits '00001111'
- 0x07: Pattern of alternating bits '0101'

Number_Of_Packets

Number of packets to be sent

PHY

PHY to be used by the transmitter. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY_S8
- 0x04: LE_CODED_PHY_S2

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.34 aci_hal_transmitter_test_packets_v2

```
tBleStatus aci_hal_transmitter_test_packets_v2(uint8_t TX_Channel,
                                               uint8_t Test_Data_Length,
                                               uint8_t Packet_Payload,
                                               uint16_t Number_Of_Packets,
                                               uint8_t PHY,
                                               uint8_t CTE_Length,
                                               uint8_t CTE_Type,
                                               uint8_t Switching_Pattern_Length,
                                               uint8_t Antenna_IDs[])
```

This command is equivalent to the corresponding Bluetooth® LE standard command HCI_LE_TRANSMITTER_TEST_V3 hci_le_transmitter_test_v3, with an additional parameter to specify the number of packets to be transmitted. An HCI_COMMAND_STATUS_EVENT is sent after the command is received. An ACI_HAL_LE_TEST_END_EVENT is generated when the number of specified packet have been sent.

Parameters

TX_Channel

N = (F - 2402) / 2. Frequency range: 2402 MHz to 2480 MHz. Values:

- 0x00 ... 0x27

Test_Data_Length

Length in bytes of payload data in each packet.

Packet_Payload

Content of the Payload of the test reference packets. Values:

- 0: PRBS9 sequence '11111111100000111101...' (in transmission order)
- 1:Repeated '11110000' (in transmission order) sequence
- 2: Repeated '10101010' (in transmission order) sequence
- 3: PRBS15 sequence
- 4: Repeated '11111111' (in transmission order) sequence
- 5: Repeated '00000000' (in transmission order) sequence
- 6: Repeated '00001111' (in transmission order) sequence
- 7: Repeated '01010101' (in transmission order) sequence

Number_Of_Packets

Number of packets to be sent

PHY

PHY to be used by the transmitter. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY_S8
- 0x04: LE_CODED_PHY_S2

CTE_Length

Expected length of the Constant Tone Extension in 8-microseconds units. If 0, no Constant Tone Extension expected. Values:

- 0x00
- 0x02 ... 0x14

CTE_Type

Type of the Constant Tone Extension in the test reference packets. 0: AoA; 1: AoD with 1 µs slots; 2: AoD with 2 µs slots. Values:

- 0x00: CTE_AOA
- 0x01: CTE_AOD_1us
- 0x02: CTE_AOD_2us

Switching_Pattern_Length

The number of Antenna IDs in the pattern.Values:

- 0x02 ... 0x4B

Antenna_IDs

List of Antenna IDs in the pattern

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.3.35 aci_hal_write_radio_reg

```
tBleStatus aci_hal_write_radio_reg(uint32_t Start_Address,
                                   uint8_t Num_Bytes,
                                   uint8_t Data[])
```

It writes a device register.

Parameters

Start_Address

Register address

Num_Bytes

Length of Data in octets

Data

Data to be written

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.3.36 aci_hal_read_radio_reg

```
tBleStatus aci_hal_read_radio_reg(uint32_t Start_Address,
                                  uint8_t Num_Bytes,
                                  uint8_t *Data_Length,
                                  uint8_t Data[])
```

It reads the device register

Parameters

Start_Address

Register address

Num_Bytes

Length of Data in octets

`[out]` **Data_Length**

Length of Data in octets

`[out]` **Data**

Read Data

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.37 aci_test_tx_notification_start

```
tBleStatus aci_test_tx_notification_start(uint16_t Connection_Handle,
                                          uint16_t Service_Handle,
                                          uint16_t Char_Handle,
                                          uint16_t Value_Length)
```

Test command for burst notifications

Parameters

Connection_Handle

Connection handle to notify

Service_Handle

Handle of service to which the characteristic belongs. Values:

- 0x0001 ... 0xFFFF

Char_Handle

Handle of the characteristic. Values:

- 0x0001 ... 0xFFFF

Value_Length

Length of the characteristic to be notified. Only ATT_MTU - 3 bytes are sent with notifications.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.38 aci_test_tx_write_command_start

```
tBleStatus aci_test_tx_write_command_start(uint16_t Connection_Handle,
                                           uint16_t Attr_Handle,
                                           uint16_t Value_Length)
```

Test command for burst writes

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Attr_Handle

Handle of the attribute to be written. Values:

- 0x0001 ... 0xFFFF

Value_Length

Length of the characteristic to be written with write commands. Only ATT_MTU - 3 bytes are written.

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.3.39 aci_test_rx_start

```
tBleStatus aci_test_rx_start(uint16_t Connection_Handle,
                             uint16_t Attr_Handle,
                             uint8_t Notifications_WriteCmds)
```

Test command for burst reception

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Attr_Handle

Handle of the attribute to be written. Values:

- 0x0001 ... 0xFFFF

Notifications_WriteCmds

It defines which burst test to start. Values:

- 0x00: Notifications
- 0x01: Write Commands

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.3.40 aci_test_stop

```
tBleStatus aci_test_stop(uint8_t TX_RX)
```

It stops the burst test.

Parameters

TX_RX

It defines the test type to stop. Values:

- 0x00: TX
- 0x01: RX

Attr_Handle

Handle of the attribute to be written. Values:

- 0x0001 ... 0xFFFF

Notifications_WriteCmds

It defines which burst test to start. Values:
- 0x00: Notifications
- 0x01: Write Commands

Return values:
- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.3.41 aci_test_report

```
tBleStatus aci_test_report(uint32_t *TX_Packets,
                           uint32_t *RX_Packets,
                           uint16_t *RX_Data_Length,
                           uint32_t *RX_Sequence_Errors)
)
```

It stops the burst test.

Parameters

TX_Packets

Number of transmitted packets

RX_Packets

Number of received packets

RX_Data_Length

RX data length

RX_Sequence_Errors

Number of RX sequence errors

Return values:
- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.4 GAP commands

This section describes the supported GAP commands.

Table 4. **GAP commands opcodes**

SoC command	Network coprocessor command	OpCode
aci_gap_init	aci_gap_init	0xFC81
aci_gap_set_io_capability	aci_gap_set_io_capability	0xFC85
aci_gap_set_security_requirements	aci_gap_set_security_requirements	0xFC86
aci_gap_passkey_resp	aci_gap_passkey_resp	0xFC88
aci_gap_profile_init	aci_gap_profile_init	0xFC8A
aci_gap_set_security	aci_gap_set_security	0xFC8D
aci_gap_get_security_level	aci_gap_get_security_level	0xFC90
aci_gap_set_le_event_mask	aci_gap_set_le_event_mask	0xFC92
aci_gap_terminate	aci_gap_terminate	0xFC93

SoC command	Network coprocessor command	OpCode
aci_gap_clear_security_db	aci_gap_clear_security_db	0xFC94
aci_gap_pairing_resp	aci_gap_pairing_resp	0xFC96
aci_gap_create_connection	aci_gap_create_connection	0xFC9C
aci_gap_terminate_proc	aci_gap_terminate_proc	0xFC9D
aci_gap_start_connection_update	aci_gap_start_connection_update	0xFC9E
aci_gap_resolve_private_addr	aci_gap_resolve_private_addr	0xFCA0
aci_gap_get_bonded_devices	aci_gap_get_bonded_devices	0xFCA3
aci_gap_is_device_bonded	aci_gap_is_device_bonded	0xFCA4
aci_gap_numeric_comparison_value_confirm_yesno	aci_gap_numeric_comparison_value_confirm_yesno	0xFCA5
aci_gap_passkey_input	aci_gap_passkey_input	0xFCA6
aci_gap_get_oob_data	aci_gap_get_oob_data	0xFCA7
aci_gap_set_oob_data	aci_gap_set_oob_data	0xFCA8
aci_gap_remove_bonded_device	aci_gap_remove_bonded_device	0xFCAA
aci_gap_set_advertising_configuration	aci_gap_set_advertising_configuration	0xFCAB
aci_gap_set_advertising_enable	aci_gap_set_advertising_enable	0xFCAC
aci_gap_set_advertising_data	aci_gap_set_advertising_data_nwk	0xFCAD
aci_gap_set_scan_response_data	aci_gap_set_scan_response_data_nwk	0xFCAE
aci_gap_set_scan_configuration	aci_gap_set_scan_configuration	0xFCAF
aci_gap_set_connection_configuration	aci_gap_set_connection_configuration	0xFCB0
aci_gap_start_procedure	aci_gap_start_procedure	0xFCB1
aci_gap_discover_name	aci_gap_discover_name	0xFCB2
aci_gap_add_devices_to_filter_accept_and_resolving_list	aci_gap_add_devices_to_filter_accept_and_resolving_list	0xFCB3
aci_gap_configure_filter_accept_and_resolving_list	aci_gap_configure_filter_accept_and_resolving_list	0xFCB4
aci_gap_remove_advertising_set	aci_gap_remove_advertising_set	0xFCBA
aci_gap_clear_advertising_sets	aci_gap_clear_advertising_sets	0xFCBB
aci_gap_create_periodic_advertising_connection	aci_gap_create_periodic_advertising_connection	0xFCCA
aci_gap_encrypt_adv_data	aci_gap_encrypt_adv_data_nwk	0xFCCB
aci_gap_decrypt_adv_data	aci_gap_decrypt_adv_data_nwk	0xFCCC

2.4.1 aci_gap_add_devices_to_filter_accept_and_resolving_list

```
tBleStatus aci_gap_add_devices_to_filter_accept_and_resolving_list ( uint8_t Lists,
                                                                     uint8_t Clear_Lists,
                                                                     uint8_t Num_of_List_Entries,
                                                                     List_Entry_t List_Entry[]
                                                                   )
```

Add specific device addresses to the Filter Accept and/or resolving list.

Parameters

Lists

Select to which list the device addresses are added: Filter Accept List, resolving list or both. Flags:

- 0x01: Filter Accept List
- 0x02: Resolving List

Clear_Lists

Clear the selected lists before adding the device addresses. Values:

- 0x00: Do not clear
- 0x01: Clear before adding

Num_of_List_Entries

Number of devices that have to be added to the Filter Accept List. Values:

- 0x00 ... 0xFF

List_Entry

See List_Entry_t

Return values:

- *Value* indicating success or error code.

2.4.2 aci_gap_clear_advertising_sets

```
aci_gap_clear_advertising_sets ( void )
```

The GAP_Clear_Advertising_Sets command is used to remove all existing advertising sets from the Controller. If advertising is enabled on any advertising set, then the Controller shall return the error code Command Disallowed (0x0C).

Note: *All advertising sets are cleared on HCI reset.*

Return values:

- *Value* indicating success or error code.

2.4.3 aci_gap_clear_security_db

```
aci_gap_clear_security_db ( void )
```

Clear the security database. All devices in the security database are removed.

Attention: *It is strongly recommended not to use this command during intense radio activity (that is, during advertising or connection with short intervals (less than 30 ms) or during scanning), since it triggers an erase of a flash memory sector. After this command, all devices previously recorded in the bonding table and connected when the command has been submitted remain connected, preserving authentication and encryption of the link.*

Return values:

- *Value* indicating success or error code.

2.4.4 aci_gap_configure_filter_accept_and_resolving_list

```
aci_gap_configure_filter_accept_and_resolving_list ( uint8_t Lists )
```

Clear the specified lists and add all bonded devices.

Parameters

Lists

Select to which list the device addresses are added: Filter Accept List, resolving list or both. Flags:

- 0x01: Filter Accept List
- 0x02: Resolving List

Return values:

- *Value* indicating success or error code.

2.4.5 aci_gap_create_connection

```
tBleStatus aci_gap_create_connection    ( uint8_t Initiating_PHY,
                                           uint8_t Peer_Address_Type,
                                           uint8_t Peer_Address[6]
                                          )
```

Creates a direct connection to a device.

Parameters

Initiating_PHY

PHYs that are used for initiating the connection. Flags:

- 0x01: LE_1M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

Peer_Address_Type

The Peer_Address_Type parameter indicates the type of address used in the connectable advertisement sent by the peer. Values:

- 0x00: Public Device Address or Public Identity Address
- 0x01: Random Device Address or Random (static) Identity Address

Peer_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected.

List_Entry

See List_Entry_t

Return values:

- *Value* indicating success or error code.

2.4.6 aci_gap_create_periodic_advertising_connection

```
tBleStatus aci_gap_create_periodic_advertising_connection ( uint8_t  Advertising_Handle,
                                                             uint8_t  Subevent,
                                                             uint8_t  Initiator_Filter_Policy,
                                                             uint8_t  Own_Address_Type,
                                                             uint8_t  Peer_Address_Type,
                                                             uint8_t  Peer_Address[6],
                                                             uint16_t Connection_Interval_Min,
                                                             uint16_t Connection_Interval_Max,
                                                             uint16_t Max_Latency,
                                                             uint16_t Supervision_Timeout,
                                                             uint16_t Min_CE_Length,
                                                             uint16_t Max_CE_Length
                                                            )
```

This command is used to create a connection between a periodic advertiser and a synchronized device. See LE Extended Create Connection [v2] command.

Parameters

Advertising_Handle

Advertising_Handle identifying the periodic advertising train. Values:

- 0x00 ... 0xEF
- 0xFF: Not specified

Subevent

Subevent where the connection request is to be sent. Values:

- 0x00 ... 0x7F
- 0xFF: Not specified

Initiator_Filter_Policy

The Initiator_Filter_Policy parameter is used to determine whether the Filter Accept List is used. If the Filter Accept List is not used, the Peer_Address_Type and the Peer_Address parameters specify the address type and address of the device to connect to. Values:

- 0x00: FILTER_ACCEPT_LIST_NOT_USED. Filter Accept List is not used to determine which device to connect to. Peer_Address_Type and Peer_Address shall be used.
- 0x01: FILTER_ACCEPT_LIST_USED. Filter Accept List is used to determine which device to connect to. Peer_Address_Type and Peer_Address shall be ignored.

Own_Address_Type

The Own_Address_Type parameter indicates the type of address being used in the connection request packets. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address
- 0x02: Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the public address.
- 0x03: Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the random address from the most recent successful LE_Set_Random_Address Command.

Peer_Address_Type

The Peer_Address_Type parameter indicates the type of address used in the connectable advertisement sent by the peer. Values:

- 0x00: Public Device Address or Public Identity Address
- 0x01: Random Device Address or Random (static) Identity Address

Peer_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected.

Connection_Interval_Min

Minimum value for the connection interval. This shall be less than or equal to Connection_Interval_Max. Time = N x 1.25 ms. Values:

- 0x0006 ... 0x0C80

Connection_Interval_Max

Maximum value for the connection interval. This shall be greater than or equal to Connection_Interval_Min. Time = N x 1.25 ms. Values:

- 0x0006 ... 0x0C80

Max_Latency

Maximum peripheral latency for the connection in number of connection events. Values:

- 0x0000 ... 0x01F3

Supervision_Timeout

Supervision timeout for the LE Link. Time = N x 10 ms. Values:

- 0x000A ... 0x0C80

Min_CE_Length

The minimum length of connection event recommended for this LE connection.

Max_CE_Length

The maximum length of connection event recommended for this LE connection.

Return values:

- *Value* indicating success or error code.

2.4.7 aci_gap_decrypt_adv_data

```
tBleStatus aci_gap_decrypt_adv_data    ( uint8_t  Session_Key[16],
                                         uint8_t  IV[8],
                                         uint8_t  Encrypted_Data_Length,
                                         uint8_t  * Encrypted_Data,
                                         uint32_t * Decrypted_Data
                                        )
```

Decrypt encrypted advertising data.

Parameters

Session_Key

The shared session key.

IV

The initialization vector.

Encrypted_Data_Length

Length of encrypted data.

Encrypted_Data

Encrypted data.

`[out]` **Decrypted_Data**

Pointer to the buffer that contains decrypted data.

Return values:

- *Value* indicating success or error code.

2.4.8 aci_gap_discover_name

```
tBleStatus aci_gap_discover_name    ( uint8_t PHYs,
                                      uint8_t Peer_Address_Type,
                                      uint8_t Peer_Address[6]
                                     )
```

Creates a direct connection to a device and read the name characteristic.

Parameters

PHYs

PHYs that is used for initiating the connection. Flags:

- 0x01: LE_1M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

Peer_Address_Type

The Peer_Address_Type parameter indicates the type of address used in the connectable advertisement sent by the peer. Values:

- 0x00: Public Device Address or Public Identity Address
- 0x01: Random Device Address or Random (static) Identity Address

Peer_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected.

Return values:

- *Value* indicating success or error code.

2.4.9 aci_gap_encrypt_adv_data

```
tBleStatus aci_gap_encrypt_adv_data    ( uint8_t  Session_Key[16],
                                         uint8_t  IV[8],
                                         uint8_t  Data_Length,
                                         uint32_t * Data,
                                         uint8_t  * Encrypted_Data
                                        )
```

This command is used by the application to encrypt data used in advertising packets.

Parameters

Session_Key

The shared session key.

IV

The initialization vector.

Data_Length

Length of data.

Data

Plain data to be encrypted.

`[out]` **Encrypted_Data**

Pointer to the buffer that contains encrypted data.

Return values:

- *Value* indicating success or error code.

2.4.10 aci_gap_get_bonded_devices

```
tBleStatus aci_gap_get_bonded_devices    ( uint8_t Offset,
                                           uint8_t Max_Num_Of_Addresses,
                                           uint8_t * Num_of_Addresses,
                                           Bonded_Device_Entry_t Bonded_Device_Entry[]
                                          )
```

This command returns the identity addresses of the bonded devices.

Parameters

Offset

Index of the first record to be returned.

Max_Num_Of_Addresses

Used to specify the maximum number of devices to be returned.

`[out]` **Num_of_Addresses**

The number of bonded devices returned by this command.

`[out]` **Bonded_Device_Entry**

See Bonded_Device_Entry_t.

Return values:

- *Value* indicating success or error code.

2.4.11 aci_gap_get_oob_data

```
tBleStatus aci_gap_get_oob_data    ( uint8_t OOB_Data_Type,
                                     uint8_t * Address_Type,
                                     uint8_t Address[6],
                                     uint8_t * OOB_Data_Len,
                                     uint8_t OOB_Data[16]
                                    )
```

This command can be used to get the local OOB authentication for LE Secure connections or the remote Temporary Key for Legacy Pairing set through aci_gap_set_oob_data(). This command is particularly useful in case of LE Secure Connections to retrieve the local OOB data generated with aci_gap_set_oob_data(). This data should then be sent to the peer through the OOB channel.

Parameters

OOB_Data_Type

OOB Data type. Values:

- 0x00: SM_TK. Legacy Pairing (LP) v.4.1 TK (Temporary Key)
- 0x01: SM_RANDOM_VALUE. Secure Connections (SC) v.4.2 Random value r used for generation of Confirm value
- 0x02: SM_CONFIRM_VALUE. Secure Connections (SC) v.4.2 Confirm value C generated through AES-CMAC-128 based cryptographic function: C=f4(PKx, PKx, r, 0)

`[out]` **Address_Type**

Identity address type. Values:

- 0x00: Public Identity Address
- 0x01: Random (static) Identity Address

`[out]` **Address**

Public or Random (static) address of this device

`[out]` **OOB_Data_Len**

Length of OOB Data carried by next data field

`[out]` **OOB_Data**

OOB Data to be exported via OOB

Return values:

- *Value* indicating success or error code.

2.4.12 aci_gap_get_security_level

```
tBleStatus aci_gap_get_security_level    ( uint16_t Connection_Handle,
                                           uint8_t  * Security_Mode,
                                           uint8_t  * Security_Level
                                          )
```

This command can be used to get the current security settings of the device.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

`[out]` **Security_Mode**

Security mode. Values:
- 0x01: Security Mode 1
- 0x02: Security Mode 2

`[out]` **Security_Level**

Security Level. Values:
- 0x01: Security Level 1
- 0x02: Security Level 2
- 0x03: Security Level 3
- 0x04: Security Level 4

Return values:
- *Value* indicating success or error code.

2.4.13 aci_gap_init

```
tBleStatus aci_gap_init    ( uint8_t Privacy_Type,
                             uint8_t Identity_Address_Type
                            )
```

Initialize the GAP layer.

Warning: *A section of the flash memory is used by this procedure. When this section is empty, data are written inside. This normally happens once during the lifetime of the device, when the command is executed for the first time (unless the section is erased). Do not power off the device while this function is writing into the flash memory.*

Parameters

Privacy_Type

Specify if privacy is enabled or not and which one. Values:
- 0x00: Privacy disabled
- 0x01: Privacy host enabled
- 0x02: Privacy controller enabled

Identity_Address_Type

Specify which address has to be used as Identity Address. Values:
- 0x00: Public Address. The public address is used as identity address
- 0x01: Static Random Address. The static random address is used as identity address

Return values:
- *Value* indicating success or error code.

2.4.14 aci_gap_profile_init

```
tBleStatus aci_gap_profile_init    (    uint8_t     Role,
        uint8_t     Privacy_Type,
        uint16_t *     Dev_Name_Char_Handle,
        uint16_t *     Appearance_Char_Handle,
        uint16_t *     Periph_Pref_Conn_Param_Char_Handle
    )
```

Register the GAP service with the GATT. This command should be called after aci_gap_init command. Standard characteristics are added to the GAP service depending on the specified role.

Parameters

Role

Bitmap of allowed roles. Ignored in stack v3.2 and earlier. Flags:

- 0x01: Peripheral
- 0x02: Broadcaster
- 0x04: Central
- 0x08: Observer

Privacy_Type

Specify if privacy is enabled or not and which one. Values:

- 0x00: Privacy disabled
- 0x01: Privacy host enabled
- 0x02: Privacy controller enabled

[out] Dev_Name_Char_Handle

Device Name Characteristics handle

[out] Appearance_Char_Handle

Appearance Characteristics handle

[out] Periph_Pref_Conn_Param_Char_Handle

Appearance Characteristics handle

Return values:

- *Value* indicating success or error code.

2.4.15 aci_gap_is_device_bonded

```
tBleStatus aci_gap_is_device_bonded    ( uint8_t Peer_Address_Type,
                                         uint8_t Peer_Address[6]
                                        )
```

The command finds whether the device, whose address is specified in the command, is bonded. If the device is using a resolvable private address and it has been bonded, then the command returns BLE_STATUS_SUCCESS.

Parameters

Peer_Address_Type

Address type. Values:

- 0x00: Public Device Address
- 0x01: Random Device Address

Peer_Address

Address used by the peer device while advertising.

Return values:

- *Value* indicating success or error code.

2.4.16 aci_gap_numeric_comparison_value_confirm_yesno

```
tBleStatus aci_gap_numeric_comparison_value_confirm_yesno    ( uint16_t Connection_Handle,
                                                               uint8_t  Confirm_Yes_No
                                                              )
```

This command allows the User to validate/confirm or not the Numeric Comparison value showed through the ACI_GAP_Numeric_Comparison_Value_Event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Confirm_Yes_No

Values:

- 0x00: NO. The Numeric Values showed on both local and peer device are different.
- 0x01: YES. The Numeric Values showed on both local and peer device are equal.

Return values:

- *Value* indicating success or error code.

2.4.17 aci_gap_pairing_resp

```
tBleStatus aci_gap_pairing_resp    ( uint16_t Connection_Handle,
                                      uint8_t  Accept
                                     )
```

This command shall be given in response to an aci_gap_paring_event_rp0, to allow or reject either the pairing request from the Central or the security request from the Peripheral.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Accept

Values:

- 0x00: REJECT
- 0x01: ACCEPT

Return values:

- *Value* indicating success or error code.

2.4.18 aci_gap_passkey_input

```
tBleStatus aci_gap_passkey_input    ( uint16_t Connection_Handle,
                                       uint8_t  Input_Type
                                      )
```

This command permits to signal to the Stack the input type detected during Passkey input.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Input_Type

Passkey input type detected. Values:

- 0x00: Passkey entry started
- 0x01: Passkey digit entered
- 0x02: Passkey digit erased
- 0x03: Passkey cleared
- 0x04: Passkey entry completed

Return values:

- *Value* indicating success or error code.

2.4.19 aci_gap_passkey_resp

```
tBleStatus aci_gap_passkey_resp    ( uint16_t Connection_Handle,
                                     uint32_t Passkey
                                    )
```

This command should be send by the host in response to aci_gap_passkey_req_event_rp0 event. The command parameter contains the pass key that is used during the pairing process.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Passkey

Pass key that is used during the pairing process. Must be a six-digit decimal number. Values:

- 0 ... 999999

Return values:

- *Value* indicating success or error code.

2.4.20 aci_gap_remove_advertising_set

```
tBleStatus aci_gap_remove_advertising_set ( uint8_t Advertising_Handle )
```

The GAP_Remove_Advertising_Set command is used to remove an advertising set from the Controller. If the advertising set corresponding to the Advertising_Handle parameter does not exist, then the Controller shall return the error code Unknown Advertising Identifier (0x42). If advertising on the advertising set is enabled, then the Controller shall return the error code Command Disallowed (0x0C).

Parameters

Advertising_Handle

It is used to identify an advertising set. Values:

- 0x00 ... 0xEF: Used to identify an advertising set

Return values:

- *Value* indicating success or error code.

2.4.21 aci_gap_remove_bonded_device

```
tBleStatus aci_gap_remove_bonded_device    ( uint8_t Peer_Identity_Address_Type,
                                             uint8_t Peer_Identity_Address[6]
                                            )
```

This command can be used to remove a specified device from the bonding table.

Attention: *The device removed from the Bonding Table preserves its connection and authentication, until explicit disconnection is requested by the user.*

Parameters

Peer_Identity_Address_Type

Identity address type. Values:

- 0x00: Public Identity Address
- 0x01: Random (static) Identity Address

Peer_Identity_Address

Public or Random (static) Identity address of the peer device.

Return values:

- *Value* indicating success or error code.

2.4.22 aci_gap_resolve_private_addr

```
tBleStatus aci_gap_resolve_private_addr    ( uint8_t Address[6],
                                             uint8_t Actual_Address[6]
                                            )
```

This command tries to resolve the address provided with the IRKs present in its database. If the address is resolved successfully with any one of the IRKs present in the database, it returns success and also the corresponding public/static random address stored with the IRK in the database.

Parameters

Address

Address to be resolved

`[out]` **Actual_Address**

The public or static random address of the peer device, distributed during the pairing phase.

Return values:

- *Value* indicating success or error code.

2.4.23 aci_gap_set_advertising_configuration

```
tBleStatus aci_gap_set_advertising_configuration ( uint8_t  Advertising_Handle,
                                                   uint8_t  Discoverable_Mode,
                                                   uint16_t Advertising_Event_Properties,
                                                   uint32_t Primary_Advertising_Interval_Min,
                                                   uint32_t Primary_Advertising_Interval_Max,
                                                   uint8_t  Primary_Advertising_Channel_Map,
                                                   uint8_t  Peer_Address_Type,
                                                   uint8_t  Peer_Address[6],
                                                   uint8_t  Advertising_Filter_Policy,
                                                   int8_t  Advertising_Tx_Power,
                                                   uint8_t  Primary_Advertising_PHY,
                                                   uint8_t  Secondary_Advertising_Max_Skip,
                                                   uint8_t  Secondary_Advertising_PHY,
                                                   uint8_t  Advertising_SID,
                                                   uint8_t  Scan_Request_Notification_Enable
                                                  )
```

This commands configures the advertising parameters for the legacy advertising or for a given extended advertising set. For general or limited discoverable mode or connectable advertising, Flags AD type must be present in advertising data. See also Bluetooth Core specifications, Vol. 4, part E, section 7.8.53 (LE Set Extended Advertising Parameters command).

Parameters

Advertising_Handle

Used to identify an advertising set. This parameter is only meaningful if Extended Advertising Feature is enabled. Values:

- 0x00 ... 0xEF

Discoverable_Mode

Specifies the discoverable mode of the device. Values:

- 0: Not Discoverable
- 1: Limited Discoverable
- 2: General Discoverable
- 3: Broadcast

Advertising_Event_Properties

The Advertising_Event_Properties parameter describes the type of advertising event that is being configured and its basic properties according to V5.1, Vol 2, Part E, section 7.8.53. Flags:

- 0x0001: Connectable
- 0x0002: Scannable
- 0x0004: Directed
- 0x0008: High Duty Cycle Directed Connectable
- 0x0010: Legacy
- 0x0020: Anonymous
- 0x0040: Include TX Power

Primary_Advertising_Interval_Min

Minimum advertising interval for undirected and low duty cycle directed advertising. Time = N * 0.625 msec. Values:

- 0x00000020 (20.000 ms) ... 0x00FFFFFF (10485759.375 ms)

Primary_Advertising_Interval_Max

Maximum advertising interval for undirected and low duty cycle directed advertising. Time = N * 0.625 msec. Values:

- 0x00000020 (20.000 ms) ... 0x00FFFFFF (10485759.375 ms)

Primary_Advertising_Channel_Map

It is a bit field that indicates the advertising channels that shall be used when transmitting advertising packets. Flags:

- 0x01: CH_37
- 0x02: CH_38
- 0x04: CH_39

Peer_Address_Type

The peer address type. Values:

- 0x00: Public
- 0x01: Random

Peer_Address

Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected.

Advertising_Filter_Policy

Advertising Filter Policy. If Directed advertising is selected, the Peer_Address_Type and Peer_Address shall be valid and the Advertising_Filter_Policy parameter shall be ignored. Values:

- 0x00: Scan and Connection requests from any
- 0x01: Connection requests from any, scan requests from Filter Accept List only
- 0x02: Scan requests from any, connection requests from Filter Accept List only
- 0x03: Scan and connection requests from Filter Accept List only

All other values reserved for future use.

Advertising_Tx_Power

The Advertising_Tx_Power parameter indicates the maximum power level at which the advertising packets are to be transmitted on the advertising channels. The Controller shall choose a power level lower than or equal to the one specified by the Host. (Units: dBm). This parameter is ignored if extended advertising is not enabled. Values:

- -127 ... 126
- 127: No preference

Primary_Advertising_PHY

The Primary_Advertising_PHY parameter indicates the PHY on which the advertising packets are transmitted on the primary advertising channel. If legacy advertising PDUs are being used, the Primary_Advertising_PHY shall indicate the LE 1M PHY. This parameter is ignored if extended advertising is not enabled. Values:

- 0x01: LE_1M_PHY
- 0x03: LE_CODED_PHY

Secondary_Advertising_Max_Skip

The Secondary_Advertising_Max_Skip parameter is the maximum number of advertising events that can be skipped before the AUX_ADV_IND can be sent. This parameter is ignored if extended advertising is not enabled. 0x00 AUX_ADV_IND shall be sent prior to the next advertising event 0x01-0xFF Maximum advertising events the Controller can skip before sending the AUX_ADV_IND packets on the secondary advertising channel.

Secondary_Advertising_PHY

The Secondary_Advertising_PHY parameter indicates the PHY on which the advertising packets are be transmitted on the secondary advertising channel. This parameter is ignored if extended advertising is not enabled. Values:

- 0x01: LE_1M_PHY
- 0x02: LE_2M_PHY
- 0x03: LE_CODED_PHY

Advertising_SID

The Advertising_SID parameter specifies the value to be transmitted in the Advertising SID subfield of the ADI field of the Extended Header of those advertising channel PDUs that have an ADI field. If the advertising set only uses PDUs that do not contain an ADI field, Advertising_SID is ignored. This parameter is ignored if extended advertising is not enabled. Values:

- 0x00 ... 0x0F

Scan_Request_Notification_Enable

The Scan_Request_Notification_Enable parameter indicates whether the Controller shall send notifications upon the receipt of a scan request PDU that is in response to an advertisement from the specified advertising set that contains its device address and is from a scanner that is allowed by the advertising filter policy. This parameter is ignored if extended advertising is not enabled. Values:

- 0x00: Scan request notifications disabled
- 0x01: Scan request notifications enabled

Return values:

- *Value* indicating success or error code.

2.4.24 aci_gap_set_advertising_data

```
tBleStatus aci_gap_set_advertising_data    ( uint8_t  Advertising_Handle,
                                              uint8_t  Operation,
                                              uint16_t Advertising_Data_Length,
                                              uint8_t  Advertising_Data[]
                                             )
```

The ACI_GAP_SET_ADVERTISING_DATA function is used to set the data in advertising PDUs. Data must be formatted as defined in Bluetooth Core spec Vol. 3 Part C, Section 11.

If the device is in Limited Discoverable Mode, Flags data type (0x06) in advertising data must have the flags set as described:

- The LE Limited Discoverable Mode flag set to one.
- The 'BR/EDR Not Supported' flag set to one.
- All other flags set to zero.

If the device is in General Discoverable Mode, Flags data type (0x06) in advertising data must have the flags set as described:

- The LE General Discoverable Mode flag set to one.
- The 'BR/EDR Not Supported' flag set to one.
- All other flags set to zero.

If the device is in one of the other modes, Flags data type (0x06) in advertising data must have the flags set as described:

- The 'BR/EDR Not Supported' flag set to one.
- All other flags set to zero.

In this case (none of the discoverable modes is used), Flags data type may be omitted in advertising data if a device is sending non connectable events, otherwise it must be present.

For non-legacy PDUs, the length of advertising data is limited to 245 octets in case of connectable advertising and cannot be present for scannable advertising. See also Bluetooth Core specifications, Vol. 4, part E, section 7.8.54 (LE Set Extended Advertising Data command).

Parameters

Advertising_Handle

Used to identify an advertising set. This parameter is only meaningful if Extended Advertising Feature is enabled. Values:

- 0x00 ... 0xEF

Operation

If set to Unchanged data, just update the Advertising DID. Values:

- 0x03: Complete data
- 0x04: Unchanged data

Advertising_Data_Length

Length of advertising data. For legacy PDUs which supports advertising data maximum value is 31 octets. Data must be formatted as defined in Bluetooth Core spec Vol. 3 Part C, Section 11.

Advertising_Data

Pointer to the buffer containing properly formatted advertising data (see Core v5.1 Vol 3, part C, chapter 11). Its content must not change, until an aci_hal_adv_scan_resp_data_update_event_rp0 is received, which informs the application that the buffer is no more used by the Bluetooth stack.

Return values:

- *Value* indicating success or error code.

2.4.25 aci_gap_set_advertising_enable

```
tBleStatus aci_gap_set_advertising_enable ( uint8_t Enable,
                                            uint8_t Number_of_Sets,
                                            Advertising_Set_Parameters_t
                                            Advertising_Set_Parameters[]
                                          )
```

This command is used to request the Controller to enable or disable one or more advertising sets using the advertising sets identified by the Advertising_Handle[i] parameter. The Controller manages the timing of advertisements in accordance with the advertising parameters given with aci_gap_set_advertising_configuration command.

Only the Enable parameter is used if extended advertising feature is disabled through modular configuration (CONTROLLER_EXT_ADV_SCAN_ENABLED=0), others are ignored. The command returns an error if adverting data are not set properly, according to the used discoverable mode: Flags AD type may be required (see aci_gap_set_advertising_data). An error is also returned if either the length of advertising data is greater than 245 octets and advertising type is connectable, or if no scan response data is set and advertising type is scannable. See also Bluetooth Core specifications, Vol. 4, part E, section 7.8.56 (LE Set Extended Advertising Enable command).

Parameters

Enable

Allows the enabling or disabling of one or more advertising sets using the advertising sets identified by the Advertising_Handle[i] parameter. Values:

- 0x00: Disable
- 0x01: Enable

Number_of_Sets

The Number_of_Sets parameter is the number of advertising sets contained in the parameter arrays. Ignored if extended advertising feature is disabled through modular configuration (CONTROLLER_EXT_ADV_SCAN_ENABLED=0). Values:

- 0x00: Disable all advertising sets.
- 0x01 ... 0x3F: Number of advertising sets to enable or disable.

Advertising_Set_Parameters

See Advertising_Set_Parameters_t.

Return values:

- *Value* indicating success or error code.

2.4.26 aci_gap_set_connection_configuration

```
tBleStatus aci_gap_set_connection_configuration    ( uint8_t  Initiating_PHY,
                                                     uint16_t Connection_Interval_Min,
                                                     uint16_t Connection_Interval_Max,
                                                     uint16_t Max_Latency,
                                                     uint16_t Supervision_Timeout,
                                                     uint16_t Min_CE_Length,
                                                     uint16_t Max_CE_Length
                                                    )
```

This function configures the connection parameters. To configure more than one PHY, this function must be called more times.

Parameters

Initiating_PHY

PHY that is going to be configured. Only one bit can be set. Flags:

- 0x01: LE_1M_PHY_BIT
- 0x02: LE_2M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

Connection_Interval_Min

Minimum value for the connection event interval. This shall be less than or equal to Connection_Interval_Max. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Connection_Interval_Max

Maximum value for the connection event interval. This shall be greater than or equal to Connection_Interval_Min. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Max_Latency

Maximum Peripheral latency for the connection in number of connection events. Values:

- 0x0000 ... 0x01F3

Supervision_Timeout

Supervision timeout for the LE Link. It shall be a multiple of 10 ms and larger than (1 + connPeripheralLatency) * connInterval * 2. Time = N * 10 msec. Values:

- 0x000A (100 ms) ... 0x0C80 (32000 ms)

Min_CE_Length

The minimum length of connection event recommended for this LE connection. Time = N * 0.625 msec.

Max_CE_Length

The maximum length of connection event recommended for this LE connection. Time = N * 0.625 msec.

Return values:

- *Value* indicating success or error code.

2.4.27 aci_gap_set_io_capability

```
tBleStatus aci_gap_set_io_capability ( uint8_t IO_Capability )
```

Set the IO capabilities of the device. This command cannot be sent during a pairing procedure.

Parameters

IO_Capability

IO capability of the device. Values:

- 0x00: IO_CAP_DISPLAY_ONLY
- 0x01: IO_CAP_DISPLAY_YES_NO
- 0x02: IO_CAP_KEYBOARD_ONLY
- 0x03: IO_CAP_NO_INPUT_NO_OUTPUT
- 0x04: IO_CAP_KEYBOARD_DISPLAY

Return values:

- *Value* indicating success or error code.

2.4.28 aci_gap_set_le_event_mask

```
tBleStatus aci_gap_set_le_event_mask ( uint8_t LE_Event_Mask[8] )
```

Set the IO capabilities of the device. This command cannot be sent during a pairing procedure.

Parameters

LE_Event_Mask

LE event mask. Default: 0x0000 0000 0000 001F. Flags:

- 0x0000 0000 0000 0000: No LE events specified
- 0x0000 0000 0000 0001: LE Connection Complete Event
- 0x0000 0000 0000 0002: LE Advertising Report Event
- 0x0000 0000 0000 0004: LE Connection Update Complete Event
- 0x0000 0000 0000 0008: LE Read Remote Used Features Complete Event
- 0x0000 0000 0000 0010: LE Long Term Key Request Event
- 0x0000 0000 0000 0020: LE Remote Connection Parameter Request Event
- 0x0000 0000 0000 0040: LE Data Length Change Event
- 0x0000 0000 0000 0080: LE Read Local P-256 Public Key Complete Event
- 0x0000 0000 0000 0100: LE Generate DHKey Complete Event
- 0x0000 0000 0000 0200: LE Enhanced Connection Complete Event
- 0x0000 0000 0000 0400: LE Directed Advertising Report Event
- 0x0000 0000 0000 0800: LE PHY Update Complete event
- 0x0000 0000 0000 1000: LE Extended Advertising Report event
- 0x0000 0000 0000 2000: LE Periodic Advertising Sync Established event
- 0x0000 0000 0000 4000: LE Periodic Advertising Report event
- 0x0000 0000 0000 8000: LE Periodic Advertising Sync Lost event
- 0x0000 0000 0001 0000: LE Scan Timeout event
- 0x0000 0000 0002 0000: LE Advertising Set Terminated event
- 0x0000 0000 0004 0000: LE Scan Request Received event
- 0x0000 0000 0008 0000: LE Channel Selection Algorithm event
- 0x0000 0000 0010 0000: LE Connectionless IQ Report event
- 0x0000 0000 0020 0000: LE Connection IQ Report event
- 0x0000 0000 0040 0000: LE CTE Request Failed event
- 0x0000 0000 0080 0000: LE Periodic Advertising Sync Transfer Received event
- 0x0000 0000 0100 0000: LE CIS Established event

- 0x0000 0000 0200 0000: LE CIS Request event
- 0x0000 0000 0400 0000: LE Create BIG Complete event
- 0x0000 0000 0800 0000: LE Terminate BIG Complete event
- 0x0000 0000 1000 0000: LE BIG Sync Established event
- 0x0000 0000 2000 0000: LE BIG Sync Lost event
- 0x0000 0000 4000 0000: LE Request Peer SCA Complete event
- 0x0000 0000 8000 0000: LE Path Loss Threshold event
- 0x0000 0001 0000 0000: LE Transmit Power Reporting event
- 0x0000 0002 0000 0000: LE BIGInfo Advertising Report event
- 0x0000 0004 0000 0000: LE Subrate Change event
- 0x0000 0008 0000 0000: LE Periodic Advertising Sync Established event [v2]
- 0x0000 0010 0000 0000: LE Periodic Advertising Report event [v2]
- 0x0000 0020 0000 0000: LE Periodic Advertising Sync Transfer Received event [v2]
- 0x0000 0040 0000 0000: LE Periodic Advertising Subevent Data Request event
- 0x0000 0080 0000 0000: LE Periodic Advertising Response Report event
- 0x0000 0100 0000 0000: LE Enhanced Connection Complete event [v2]

Return values:

- *Value* indicating success or error code.

2.4.29 aci_gap_set_oob_data

```
tBleStatus aci_gap_set_oob_data    ( uint8_t Device_Type,
                                     uint8_t Address_Type,
                                     uint8_t Address[6],
                                     uint8_t OOB_Data_Type,
                                     uint8_t OOB_Data_Len,
                                     uint8_t OOB_Data[16]
                                    )
```

This command can be used to input the Authentication data exchanged via OOB channel: either local authentication data to be sent to the peer device through the OOB channel or the peer device authentication data received through OOB channel. Moreover, it can also be used to generate authentication data for LE Secure Connections. Device_Type must be set to 0x00 (Local Device) to provide or generate local authentication data that are sent to the peer through OOB channel. In this case Address_Type and Address parameters are ignored. With Device_Type=0 and OOB_Data_Len=0x00, OOB_Data_Type is ignored and the command triggers an automatic generation of OOB Authentication data r and C (that can be read with aci_gap_get_oob_data()), used for Secure Connections, otherwise the OOB_Data carried by the command overwrites the current local authentication OOB Data.

To generate OOB authentication data, the stack requires the availability of the local Public Key, to be previously generated with hci_le_read_local_p256_public_key command. When peer authentication data are received through OOB channel for either Legacy Pairing or Secure Connections, aci_gap_set_oob_data() must be called with Device_Type set to 0x01 (Remote Device): the command sets the OOB data for the specified remote device (only one device at a time is supported). For Legacy pairing, the TK must be provided as both local and remote data.

Parameters

Device_Type

Values:

- 0x00: Local device. Sets the local OOB authentication data.
- 0x01: Remote device. Sets the OOB data for the specified remote device (only one device at a time is supported).

Address_Type

Identity address type of the remote device. Ignored if Device_Type is 0. Values:

- 0x00: Public Identity Address
- 0x01: Random (static) Identity Address

Address

Public or Random (static) address of the peer device. Ignored if Device_Type is 0.

OOB_Data_Type

OOB Data type. Values:

- 0x00: SM_TK. Legacy Pairing (LP) v.4.1 TK (Temporary Key)
- 0x01: SM_RANDOM_VALUE. Secure Connections (SC) v.4.2 Random value r used for generation of Confirm value
- 0x02: SM_CONFIRM_VALUE. Secure Connections (SC) v.4.2 Confirm value C generated through AES-CMAC-128 based cryptographic function: C=f4(PKx, PKx, r, 0)

OOB_Data_Len

Length of OOB Data carried by next data field. It may be set to 0x00 to trigger the automatic generation of local Random and Confirm values for LE Secure Connections pairing. Values:

- 0x00 ... 0x10

OOB_Data

OOB Data to be exported via OOB.

Return values:

- *Value* indicating success or error code.

2.4.30 aci_gap_set_scan_configuration

```
tBleStatus aci_gap_set_scan_configuration    ( uint8_t  Filter_Duplicates,
                                               uint8_t  Scanning_Filter_Policy,
                                               uint8_t  Scanning_PHY,
                                               uint8_t  Scan_Type,
                                               uint16_t Scan_Interval,
                                               uint16_t Scan_Window
                                              )
```

The ACI_GAP_SET_SCAN_CONFIGURATION function configures the scan parameters for a given PHY. To configure more than one PHY, this function must be called more times.

Parameters

Filter_Duplicates

The Filter_Duplicates parameter controls whether the Link Layer should filter out duplicate advertising reports (filtering duplicates enabled) to the Host or if the Link Layer should generate advertising reports for each packet received (filtering duplicates disabled). See [Vol 6] Part B, Section 4.4.3.5. Values:

- 0x00: Duplicate filtering disabled
- 0x01: Duplicate filtering enabled
- 0x02: Duplicate filtering enabled, reset for each scan period

Scanning_Filter_Policy

Values:

- 0x00: Accept all advertisement packets. Directed advertising packets which are not addressed for this device shall be ignored.
- 0x01: Ignore advertisement packets from devices not in the Filter Accept List Only. Directed advertising packets which are not addressed for this device shall be ignored
- 0x02: Accept all undirected advertisement packets (use resolving list). Directed advertisement packets where initiator address is a RPA and Directed advertisement packets addressed to this device shall be accepted.
- 0x03: Filter Accept List Only (use resolving list). Accept all undirected advertisement packets from devices that are in the Filter Accept List. Directed advertisement packets where initiator address is RPA and Directed advertisement packets addressed to this device shall be accepted.

Scanning_PHY

PHY that is going to be configured. Only one bit can be set. Flags:

- 0x01: LE_1M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

Scan_Type

The Scan_Type parameter specifies the type of scan to perform. Values:

- 0x00: Passive Scanning. No scan request PDUs shall be sent.
- 0x01: Active Scanning. Scan request PDUs may be sent.

Scan_Interval

Time interval from when the Controller started its last scan until it begins the subsequent scan on the primary advertising physical channel. Time = N * 0.625 ms. Values:

- 0x0004 (2.500 ms) ... 0xFFFF (40959.375 ms)

Scan_Window

Time interval from when the Controller started its last scan until it begins the subsequent scan on the primary advertising physical channel. Time = N * 0.625 msec. Values:

- 0x0004 (2.500 ms) ... 0xFFFF (40959.375 ms)

Return values:

- *Value* indicating success or error code.

2.4.31 aci_gap_set_scan_response_data

```
tBleStatus aci_gap_set_scan_response_data    ( uint8_t  Advertising_Handle,
                                               uint16_t Scan_Response_Data_Length,
                                               uint8_t  Scan_Response_Data[]
                                              )
```

The ACI_GAP_SET_SCAN_RESPONSE_DATA function configures the scan response data as requested by the application.

Parameters

Advertising_Handle

Used to identify an advertising set. This parameter is only meaningful if Extended Advertising Feature is enabled. Values:

- 0x00 ... 0xEF

Scan_Response_Data_Length

Length of scan response data. If the advertising set uses scannable legacy advertising PDUs maximum length is 31 octets.

Scan_Response_Data

Pointer to the buffer containing properly formatted scan response data (see Core v5.1 Vol 3, part C, chapter 11). Its content must not change, until an aci_hal_adv_scan_resp_data_update_event_rp0 is received, which informs the application that the buffer is no more used by the Bluetooth® stack.

Return values:

- *Value* indicating success or error code.

2.4.32 aci_gap_set_security

```
tBleStatus aci_gap_set_security    ( uint16_t Connection_Handle,
                                     uint8_t  Security_Level,
                                     uint8_t  Force_Pairing
                                    )
```

This command sets the security level for the given connection (LE security mode 1), by enabling encryption if needed. It enables encryption on the link if the peer is bonded with at least the specified security level. Otherwise, it starts pairing with the peer device. This command may be given either on a Central or on a Peripheral device. The Security_Level indicates the minimum Security Level to be achieved: 1 for no security (no authentication or encryption), 2 for unauthenticated pairing with encryption, 3 for authenticated pairing with encryption, 4 for authenticated LE Secure Connections pairing with encryption using a 128-bit strength encryption key.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Security_Level

Indicates the minimum Security Level to be achieved. Values:

- 0x01: GAP_SECURITY_LEVEL_1
- 0x02: GAP_SECURITY_LEVEL_2
- 0x03: GAP_SECURITY_LEVEL_3
- 0x04: GAP_SECURITY_LEVEL_4

Force_Pairing

Return values:

- *Value* indicating success or error code.

2.4.33 aci_gap_set_security_requirements

```
tBleStatus aci_gap_set_security_requirements    ( uint8_t Bonding_Mode,
                                                  uint8_t MITM_Mode,
                                                  uint8_t SC_Support,
                                                  uint8_t KeyPress_Notification_Support,
                                                  uint8_t Min_Encryption_Key_Size,
                                                  uint8_t Max_Encryption_Key_Size,
                                                  uint8_t Pairing_Response
                                                 )
```

This command shall be given to set security requirements at device level. MITM_Mode setting is used only when responding to an incoming Peripheral Security Request or Pairing Request received from peer device. It is recommended to force the use of Secure Connections for pairing by using Secure Connections Only Mode, since it is the only way to protect the pairing from passive eavesdropping.

The Pairing_Response parameter is used to control the way a pairing is accepted. 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 automatically accepted (no user interaction) except when the request comes from an already bonded device; in this case aci_gap_pairing_event_rp0 is notified and the application has to give a confirmation through aci_gap_pairing_resp to accept the request to rebond. If Pairing_Response is set to 2 (explicit pairing response) a pairing confirmation is always required since aci_gap_pairing_event_rp0 is always raised when a Pairing Request from a Central or a Peripheral Security Request form a Peripheral is received.

If the command is given during pairing, the command returns BLE_STATUS_NOT_ALLOWED. The command returns BLE_STATUS_INVALID_PARAMS if some of the parameters are out of admitted range. If KeyPress_Notification_Support is set to 1 but Secure Connection feature is not supported, BLE_ERROR_UNSUPPORTED_FEATURE is returned.

Parameters

Bonding_Mode

Bonding mode. Only if bonding is enabled (0x01), the bonding information is stored in flash memory. Values:

- 0x00: NO_BONDING
- 0x01: BONDING

MITM_Mode

MITM mode. Values:

- 0x00: MITM_PROTECTION_NOT_REQUIRED
- 0x01: MITM_PROTECTION_REQUIRED

SC_Support

LE Secure connections support. Secure Connections Only Mode (0x02) is the recommended value. Values:

- 0x00: GAP_SC_NOT_SUPPORTED. Secure Connections Pairing not supported.
- 0x01: GAP_SC_OPTIONAL. Secure Connections Pairing supported but optional.
- 0x02: GAP_SC_MANDATORY. Secure Connections Pairing supported and mandatory (SC Only Mode). This is the recommended value.

KeyPress_Notification_Support

Keypress notification support. Values:

- 0x00: GAP_KEYPRESS_NOT_SUPPORTED
- 0x01: GAP_KEYPRESS_SUPPORTED

Min_Encryption_Key_Size

Minimum encryption key size to be used during pairing. Values:

- 7 ... 16

Max_Encryption_Key_Size

Maximum encryption key size to be used during pairing. Values:

- 7 ... 16

Pairing_Response

This parameter controls how pairing confirmation is managed. Values:

- 0x00: GAP_PAIRING_RESP_NONE
- 0x01: GAP_PAIRING_RESP_FOR_BONDED_DEVICES
- 0x02: GAP_PAIRING_RESP_FOR_ALL

Return values:

- *Value* indicating success or error code.

2.4.34 aci_gap_start_connection_update

```
tBleStatus aci_gap_start_connection_update    ( uint16_t Connection_Handle,
                                                uint16_t Connection_Interval_Min,
                                                uint16_t Connection_Interval_Max,
                                                uint16_t Max_Latency,
                                                uint16_t Supervision_Timeout,
                                                uint16_t Min_CE_Length,
                                                uint16_t Max_CE_Length
                                               )
```

Start the connection update procedure (only when role is Central). A hci_le_connection_update is called. On completion of the procedure, an hci_le_connection_update_complete_event_rp0 event is returned to the upper layer.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Connection_Interval_Min

Minimum value for the connection event interval. This shall be less than or equal to Connection_Interval_Max. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Connection_Interval_Max

Maximum value for the connection event interval. This shall be greater than or equal to Connection_Interval_Min. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Max_Latency

Maximum Peripheral latency for the connection in number of connection events. Values:

- 0x0000 ... 0x01F3

Supervision_Timeout

Supervision timeout for the LE Link. It shall be a multiple of 10 ms and larger than (1 + connPeripheralLatency) * connInterval * 2. Time = N * 10 msec. Values:

- 0x000A (100 ms) ... 0x0C80 (32000 ms)

Min_CE_Length

The minimum length of connection event recommended for this LE connection. Time = N * 0.625 ms.

Max_CE_Length

The maximum length of connection event recommended for this LE connection. Time = N * 0.625 ms.

Return values:

- *Value* indicating success or error code.

2.4.35 aci_gap_start_procedure

```
tBleStatus aci_gap_start_procedure    ( uint8_t  Procedure_Code,
                                        uint8_t  PHYs,
                                        uint16_t Duration,
                                        uint16_t Period
                                       )
```

Starts a GAP procedure according to the procedure code.

Parameters

Procedure_Code

Procedure to be started. Values:

- 0x00: GAP_LIMITED_DISCOVERY_PROC
- 0x01: GAP_GENERAL_DISCOVERY_PROC
- 0x02: GAP_AUTO_CONNECTION_ESTABLISHMENT_PROC
- 0x03: GAP_GENERAL_CONNECTION_ESTABLISHMENT_PROC
- 0x04: GAP_SELECTIVE_CONNECTION_ESTABLISHMENT_PROC
- 0x05: GAP_OBSERVATION_PROC

PHYs

PHYs that are used for Scanning or Initiating. Flags:

- 0x01: LE_1M_PHY_BIT
- 0x04: LE_CODED_PHY_BIT

Duration

Ignored. Reserved for future use.

Period

Ignored. Reserved for future use.

Return values:

- *Value* indicating success or error code.

2.4.36 aci_gap_terminate

```
tBleStatus aci_gap_terminate    ( uint16_t Connection_Handle,
                                  uint8_t  Reason
                                 )
```

Command the controller to terminate the connection. A hci_disconnection_complete_event_rp0 is generated when the link is disconnected. After this event is received, the Bluetooth® stack may request to save GATT database information in non-volatile memory. So it is important not to reset or power off the system immediately after hci_disconnection_complete_event_rp0 is received. This operation is normally completed within less than a few milliseconds.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Reason

The reason for ending the connection. Values:

- 0x05: Authentication Failure
- 0x13: Remote User Terminated Connection
- 0x14: Remote Device Terminated Connection due to Low Resources
- 0x15: Remote Device Terminated Connection due to Power Off
- 0x1A: Unsupported Remote Feature
- 0x3B: Unacceptable Connection Parameters

Return values:

- *Value* indicating success or error code.

2.4.37 aci_gap_terminate_proc

```
tBleStatus aci_gap_terminate_proc ( uint8_t Procedure_Code )
```

Terminate the specified GAP procedure. An aci_gap_proc_complete_event_rp0 event is generated when the procedure has been completed, with the procedure code set to the corresponding procedure.

Parameters

Procedure_Code

Code identifying the procedure. Values:

- 0x00: GAP_LIMITED_DISCOVERY_PROC
- 0x01: GAP_GENERAL_DISCOVERY_PROC
- 0x02: GAP_AUTO_CONNECTION_ESTABLISHMENT_PROC
- 0x03: GAP_GENERAL_CONNECTION_ESTABLISHMENT_PROC
- 0x04: GAP_SELECTIVE_CONNECTION_ESTABLISHMENT_PROC
- 0x05: GAP_OBSERVATION_PROC
- 0x06: GAP_DIRECT_CONNECTION_ESTABLISHMENT_PROC
- 0x07: GAP_NAME_DISCOVERY_PROC

Return values:

- *Value* indicating success or error code.

2.5 GATT commands

This section describes the supported GATT commands.

Table 5. **GATT commands opcodes**

SoC command	Network coprocessor command	OpCode
aci_gatt_srv_profile_init	aci_gatt_srv_profile_init	0xFD01
aci_gatt_srv_add_service	aci_gatt_srv_add_service_nwk	0xFD02
aci_gatt_srv_include_service	aci_gatt_srv_include_service_nwk	0xFD03
aci_gatt_srv_add_char	aci_gatt_srv_add_char_nwk	0xFD04
aci_gatt_srv_add_char_desc	aci_gatt_srv_add_char_desc_nwk	0xFD05
-	aci_gatt_srv_write_handle_value_nwk	0xFD06
aci_gatt_srv_rm_char	aci_gatt_srv_rm_char_nwk	0xFD07
aci_gatt_srv_rm_service	aci_gatt_srv_rm_service_nwk	0xFD08
aci_gatt_srv_rm_include_service	aci_gatt_srv_rm_include_service_nwk	0xFD09
aci_gatt_set_event_mask	aci_gatt_set_event_mask	0xFD0A

SoC command	Network coprocessor command	OpCode
aci_gatt_clt_exchange_config	aci_gatt_clt_exchange_config	0xFD0B
aci_gatt_clt_prepare_write_req	aci_gatt_clt_prepare_write_req	0xFD10
aci_gatt_clt_execute_write_req	aci_gatt_clt_execute_write_req	0xFD11
aci_gatt_clt_disc_all_primary_services	aci_gatt_clt_disc_all_primary_services	0xFD12
aci_gatt_clt_disc_primary_service_by_uuid	aci_gatt_clt_disc_primary_service_by_uuid	0xFD13
aci_gatt_clt_find_included_services	aci_gatt_clt_find_included_services	0xFD14
aci_gatt_clt_disc_all_char_of_service	aci_gatt_clt_disc_all_char_of_service	0xFD15
aci_gatt_clt_disc_char_by_uuid	aci_gatt_clt_disc_char_by_uuid	0xFD16
aci_gatt_clt_disc_all_char_desc	aci_gatt_clt_disc_all_char_desc	0xFD17
aci_gatt_clt_read	aci_gatt_clt_read	0xFD18
aci_gatt_clt_read_using_char_uuid	aci_gatt_clt_read_using_char_uuid	0xFD19
aci_gatt_clt_read_long	aci_gatt_clt_read_long	0xFD1A
aci_gatt_clt_read_multiple_char_value	aci_gatt_clt_read_multiple_char_value	0xFD1B
aci_gatt_clt_write_nwk	aci_gatt_clt_write_nwk	0xFD1C
aci_gatt_clt_write_long	aci_gatt_clt_write_long_nwk	0xFD1D
aci_gatt_clt_write_char_reliable	aci_gatt_clt_write_char_reliable_nwk	0xFD1E
aci_gatt_clt_write_without_resp	aci_gatt_clt_write_without_resp	0xFD23
aci_gatt_clt_signed_write_without_resp	aci_gatt_clt_signed_write_without_resp	0xFD24
aci_gatt_clt_confirm_indication	aci_gatt_clt_confirm_indication	0xFD25
-	aci_gatt_srv_set_security_permission_nwk	0xFD28
aci_gatt_srv_read_handle_value	aci_gatt_srv_read_handle_value_nwk	0xFD2A
-	aci_gatt_srv_set_access_permission_nwk	0xFD2E
aci_gatt_srv_notify	aci_gatt_srv_notify	0xFD2F
-	aci_gatt_srv_exec_write_resp_nwk	0xFD31
-	aci_gatt_srv_authorize_resp_nwk	0xFD33
-	aci_gatt_srv_read_prepare_queue_nwk	0xFD35
aci_gatt_srv_read_multiple_instance_handle_value_nwk	aci_gatt_srv_read_multiple_instance_handle_value_nwk	0xFD37
aci_gatt_srv_multi_notify	aci_gatt_srv_multi_notify	0xFD38
aci_gatt_clt_read_multiple_var_len_char_value	aci_gatt_clt_read_multiple_var_len_char_value	0xFD39
aci_gatt_clt_add_subscription_security_level_nwk	aci_gatt_clt_add_subscription_security_level_nwk	0xFD3A
aci_gatt_srv_get_char_decl_handle	-	0xFD69
aci_gatt_srv_get_descriptor_handle	-	0xFD6A
aci_gatt_srv_get_include_service_handle	-	0xFD68
aci_gatt_srv_get_service_handle	-	0xFD67
aci_gatt_srv_resp	-	0xFD6C

2.5.1 aci_gatt_clt_confirm_indication

```
tBleStatus aci_gatt_clt_confirm_indication    ( uint16_t Connection_Handle,
                                                 uint16_t CID
                                                )
```

Allow application to confirm indication. This command has to be sent when the application receives the event aci_gatt_clt_indication_event_rp0.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Return values:

- *Value* indicating success or error code.

2.5.2 aci_gatt_clt_disc_all_char_desc

```
tBleStatus aci_gatt_clt_disc_all_char_desc    ( uint16_t Connection_Handle,
                                                uint16_t CID,
                                                uint16_t Char_Handle,
                                                uint16_t End_Handle
                                               )
```

Starts the procedure to discover all characteristic descriptors on the server. When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. Before procedure completion the response packets are given through aci_att_clt_find_info_resp_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Char_Handle

Handle of the characteristic value. Values:

- 0x0001 ... 0xFFFF

End_Handle

End handle of the characteristic. Values:

- 0x0001 ... 0xFFFF

Return values:

- *Value* indicating success or error code.

2.5.3 aci_gatt_clt_disc_all_char_of_service

```
tBleStatus aci_gatt_clt_disc_all_char_of_service    ( uint16_t Connection_Handle,
                                                      uint16_t CID,
                                                      uint16_t Start_Handle,
                                                      uint16_t End_Handle
                                                     )
```

Starts the procedure to discover all the characteristics of a given service. When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. Before procedure completion the response packets are given through aci_att_clt_read_by_type_resp_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Start_Handle

Start attribute handle of the service. Values:

- 0x0001 ... 0xFFFF

End_Handle

End handle of the characteristic. Values:

- 0x0001 ... 0xFFFF

Return values:

- *Value* indicating success or error code.

2.5.4 aci_gatt_clt_disc_all_primary_services

```
tBleStatus aci_gatt_clt_disc_all_primary_services    ( uint16_t Connection_Handle,
                                                        uint16_t CID
                                                      )
```

Starts the GATT client procedure to discover all primary services on the server. The responses of the procedure are given through the aci_att_clt_read_by_group_type_resp_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Return values:

- *Value* indicating success or error code.

2.5.5 aci_gatt_clt_disc_char_by_uuid

```
tBleStatus aci_gatt_clt_disc_char_by_uuid    ( uint16_t Connection_Handle,
                                                uint16_t CID,
                                                uint16_t Start_Handle,
                                                uint16_t End_Handle,
                                                uint8_t  UUID_Type,
                                                UUID_t * UUID
                                              )
```

Starts the procedure to discover all the characteristics specified by a UUID. When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. Before procedure completion the response packets are given through aci_gatt_clt_disc_read_char_by_uuid_resp_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Start_Handle

Start attribute handle of the service. Values:
- 0x0001 ... 0xFFFF

End_Handle

End handle of the characteristic. Values:
- 0x0001 ... 0xFFFF

UUID_Type

UUID type. Values:
- 0x01: 16-bit UUID
- 0x02: 128-bit UUID

UUID

See UUID_t.

Return values:
- *Value* indicating success or error code.

2.5.6 aci_gatt_clt_disc_primary_service_by_uuid

```
tBleStatus aci_gatt_clt_disc_primary_service_by_uuid    ( uint16_t Connection_Handle,
                                                          uint16_t CID,
                                                          uint8_t  UUID_Type,
                                                          UUID_t * UUID
                                                         )
```

Starts the procedure to discover all the characteristics of a given service. When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. Before procedure completion the response packets are given through aci_att_clt_read_by_type_resp_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:
- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

UUID_Type

UUID type. Values:
- 0x01: 16-bit UUID
- 0x02: 128-bit UUID

UUID

See UUID_t.

Return values:
- *Value* indicating success or error code.

2.5.7 aci_gatt_clt_exchange_config

```
tBleStatus aci_gatt_clt_exchange_config ( uint16_t Connection_Handle )
```

Performs an ATT MTU exchange procedure. When the ATT MTU exchange procedure is completed, a aci_att_exchange_mtu_resp_event_rp0 event is generated. A aci_gatt_clt_proc_complete_event_rp0 event is also generated to indicate the end of the procedure.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Return values:

- *Value* indicating success or error code.

2.5.8 aci_gatt_clt_execute_write_req

```
tBleStatus aci_gatt_clt_execute_write_req    ( uint16_t Connection_Handle,
                                               uint16_t CID,
                                               uint8_t  Execute
                                              )
```

Sends an Execute Write Request. The Execute Write Request is used to request the server to write or cancel the write of all the prepared values currently held in the prepare queue from this client. The result of the procedure is given through the aci_att_clt_exec_write_resp_event_rp0 event. The end of the procedure is indicated by a aci_gatt_clt_proc_complete_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Execute

Execute or cancel writes. Values:

- 0x00: Cancel all prepared writes
- 0x01: Immediately write all pending prepared values

Return values:

- *Value* indicating success or error code.

2.5.9 aci_gatt_clt_find_included_services

```
tBleStatus aci_gatt_clt_find_included_services    ( uint16_t Connection_Handle,
                                                    uint16_t CID,
                                                    uint16_t Start_Handle,
                                                    uint16_t End_Handle
                                                   )
```

Starts the procedure to find all included services. The responses of the procedure are given through the aci_att_clt_read_by_type_resp_event_rp0 event. The end of the procedure is indicated by a aci_gatt_clt_proc_complete_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Start_Handle

Start attribute handle of the service. Values:

- 0x0001 ... 0xFFFF

End_Handle

End handle of the service. Values:

- 0x0001 ... 0xFFFF

Return values:

- *Value* indicating success or error code.

2.5.10 aci_gatt_clt_prepare_write_req

```
tBleStatus aci_gatt_clt_prepare_write_req    ( uint16_t Connection_Handle,
                                               uint16_t CID,
                                  uint16_t Attr_Handle,
                                               uint16_t Val_Offset,
                                               uint16_t Attribute_Val_Length,
                                               uint8_t  Attribute_Val[]
                                             )
```

Sends a Prepare Write Request. The Prepare Write Request is used to request the server to prepare to write the value of an attribute. The responses of the procedure are given through the aci_att_clt_prepare_write_resp_event_rp0 event. The end of the procedure is indicated by a aci_gatt_clt_proc_complete_event_rp0.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Attr_Handle

Handle of the attribute to be written. Values:

- 0x0001 ... 0xFFFF

Val_Offset

The offset of the first octet to be written. Values:

- 0 ... 511

Attribute_Val_Length

Length of attribute value (maximum value is ATT_MTU - 5).

Attribute_Val

The value of the attribute to be written.

Return values:

- *Value* indicating success or error code.

2.5.11 aci_gatt_clt_read

```
tBleStatus aci_gatt_clt_read    ( uint16_t Connection_Handle,
                                  uint16_t CID,
                                  uint16_t Attr_Handle
                                )
```

Starts the procedure to read an attribute value. When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. Before procedure completion the response packet is given through aci_att_clt_read_resp_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Attr_Handle

Handle of the attribute to be read. Values:

- 0x0001 ... 0xFFFF

Return values:

- *Value* indicating success or error code.

2.5.12 aci_gatt_clt_read_long

```
tBleStatus aci_gatt_clt_read_long    ( uint16_t Connection_Handle,
                                       uint16_t CID,
                                       uint16_t Attr_Handle,
                                       uint16_t Val_Offset
                                      )
```

Starts the procedure to read a long attribute value. the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. Before procedure completion the response packets are given through aci_att_clt_read_blob_resp_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Attr_Handle

Handle of the attribute to be read. Values:

- 0x0001 ... 0xFFFF

Val_Offset

Offset from which the value needs to be read Values:

- 0 ... 511

Return values:

- *Value* indicating success or error code.

2.5.13 aci_gatt_clt_read_multiple_char_value

```
tBleStatus aci_gatt_clt_read_multiple_char_value    ( uint16_t Connection_Handle,
                                                      uint16_t CID,
                                                      uint8_t  Number_of_Handles,
                                                      uint16_t Handle[]
                                                     )
```

Starts a procedure to read multiple characteristic values from a server. This sub-procedure is used to read multiple Characteristic Values from a server when the client knows the Characteristic Value Handles. Only values that have a known fixed size can be read, with the exception of the last value that can have a variable length. When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. Before procedure completion the response packets are given through aci_att_clt_read_multiple_resp_event_rp0 event. The response only contains a set of Characteristic Values that is less than or equal to (ATT_MTU - 1) octets in length. If the Set Of Values is greater than (ATT_MTU - 1) octets in length, only the first (ATT_MTU - 1) octets are included in the response.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Number_of_Handles

The number of handles for which the value has to be read. From 2 to (ATT_MTU-1)/2. Values:

- 0x02 ... 0xFF

Handle

The handles for which the attribute value has to be read.

Return values:

- *Value* indicating success or error code.

2.5.14 aci_gatt_clt_read_multiple_var_len_char_value

```
tBleStatus aci_gatt_clt_read_multiple_var_len_char_value    ( uint16_t Connection_Handle,
                                                              uint16_t CID,
                                                              uint8_t  Number_of_Handles,
                                                              uint16_t Handle[]
                                                             )
```

This sub-procedure is used to read multiple Characteristic Values from a server when the client knows the Characteristic Value Handles. This procedure is useful when the attributes to read have a variable or unknown value length (otherwise aci_gatt_clt_read_multiple_char_value may be used). When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. Before procedure completion the response packets are given through aci_att_clt_read_multiple_var_len_resp_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Number_of_Handles

The number of handles for which the value has to be read. From 2 to (ATT_MTU-1)/2. Values:

- 0x02 ... 0xFF

Handle

The handles for which the attribute value has to be read.

Return values:

- *Value* indicating success or error code.

2.5.15 aci_gatt_clt_read_using_char_uuid

```
tBleStatus aci_gatt_clt_read_using_char_uuid    ( uint16_t Connection_Handle,
                                                  uint16_t CID,
                                                  uint16_t Start_Handle,
                                                  uint16_t End_Handle,
                                                  uint8_t  UUID_Type,
                                                  UUID_t * UUID
                                                 )
```

Starts the procedure to read all the characteristics specified by the UUID. When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. Before procedure completion the response packets are given through aci_gatt_clt_disc_read_char_by_uuid_resp_event_rp0 event.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Start_Handle

Start attribute handle of the service. Values:

- 0x0001 ... 0xFFFF

End_Handle

End handle of the characteristic. Values:

- 0x0001 ... 0xFFFF

UUID_Type

UUID type. Values:

- 0x01: 16-bit UUID
- 0x02: 128-bit UUID

UUID

See UUID_t.

Return values:

- *Value* indicating success or error code.

2.5.16 aci_gatt_clt_signed_write_without_resp

```
tBleStatus aci_gatt_clt_signed_write_without_resp    ( uint16_t Connection_Handle,
                                                       uint16_t Attr_Handle,
                                                       uint16_t Attribute_Val_Length,
                                                       uint8_t  Attribute_Val[]
                                                      )
```

Starts a signed write without response from the server. The procedure is used to write a characteristic value with an authentication signature without waiting for any response from the server. It cannot be used when the link is encrypted, and therefore it cannot be used on enhanced ATT bearers.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Attr_Handle

Handle of the attribute to be written. Values:

- 0x0001 ... 0xFFFF

Attribute_Val_Length

Length of the value to be written (up to ATT_MTU-13).

Attribute_Val

Value to be written.

Return values:

- *Value* indicating success or error code.

2.5.17 aci_gatt_clt_write

```
tBleStatus aci_gatt_clt_write    ( uint16_t Connection_Handle,
                                    uint16_t CID,
                                    uint16_t Attr_Handle,
                                    uint16_t Attribute_Val_Length,
                                    uint8_t  Attribute_Val[]
                                  )
```

Starts the procedure to write an attribute (characteristic value or descriptor). When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated.

Note: The buffer containing the value to be written must be kept valid until the aci_gatt_clt_proc_complete_event_rp0 is received.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Attr_Handle

Handle of the attribute to be written. Values:

- 0x0001 ... 0xFFFF

Attribute_Val_Length

Length of the value to be written.

Attribute_Val

Return values:

- *Value* indicating success or error code.

2.5.18 aci_gatt_clt_write_char_reliable

```
tBleStatus aci_gatt_clt_write_char_reliable    ( uint16_t Connection_Handle,
                                                  uint16_t CID,
                                                  uint8_t Num_Attrs,
                                                  ble_gatt_clt_write_ops_t * Write_Ops_p
                                                )
```

Starts the procedure to write a characteristic reliably (a check is made on the written values). When the procedure is completed, a aci_gatt_clt_proc_complete_event_rp0 event is generated. During the procedure, aci_att_clt_prepare_write_resp_event_rp0 and aci_att_clt_exec_write_resp_event_rp0 events are raised. Note: The memory pointed by Write_Ops_p parameter and the buffer containing the value to be written must be kept valid while the procedure is running. They can be released (or their content can be changed) when the aci_gatt_clt_proc_complete_event_rp0 is emitted indicating that the procedure is completed, or an error was received.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Num_Attrs

The number of attributes to write, that is, the number of elements in the list pointed by Write_Ops_p.

Write_Ops_p

The pointer to the list of structures that hold the write information.

Return values:

- *Value* indicating success or error code.

2.5.19 aci_gatt_clt_write_long

```
tBleStatus aci_gatt_clt_write_long    ( uint16_t Connection_Handle,
                                         uint16_t CID,
                                         ble_gatt_clt_write_ops_t * Write_Ops_p
                                        )
```

This procedure is used to write an Attribute Value to a Server when the Client knows the Attribute Handle but the length of the Value is longer than what can be sent in a single Write Request Attribute Protocol message. During the procedure, aci_att_clt_prepare_write_resp_event_rp0 and aci_att_clt_exec_write_resp_event_rp0 are raised. Note: The memory pointed by Write_Ops_p parameter and the buffer containing the value to be written must be kept valid while the procedure is running. They can be released (or their content can be changed) when the aci_gatt_clt_proc_complete_event_rp0 is emitted indicating that the procedure is completed, or an error was received.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Write_Ops_p

The pointer to the list of structures that hold the write information.

Return values:

- *Value* indicating success or error code.

2.5.20 aci_gatt_clt_write_without_resp

```
tBleStatus aci_gatt_clt_write_without_resp    ( uint16_t Connection_Handle,
                                                 uint16_t CID,
                                                 uint16_t Attr_Handle,
                                                 uint16_t Attribute_Val_Length,
                                                 uint8_t  Attribute_Val[]
                                                )
```

Starts the procedure to write a characteristic value without waiting for any response from the server. No events are generated after this command is executed. Writing attributes using this function is not considered reliable by the standard: packets may be discarded by the peer if too many write commands are received.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Attr_Handle

Handle of the attribute to be written. Values:

- 0x0001 ... 0xFFFF

Attribute_Val_Length

Length of the value to be written.

Attribute_Val

Value to be written.

Return values:

- *Value* indicating success or error code.

2.5.21 aci_gatt_set_event_mask

```
tBleStatus aci_gatt_set_event_mask ( uint32_t GATT_Evt_Mask )
```

Masks events from the GATT. The default configuration is all the events unmasked (enabled).

Parameters

GATT_Evt_Mask

GATT/ATT event mask. Values:
- 0x00000001: ACI_GATT_ATTRIBUTE_MODIFIED_EVENT
- 0x00000002: ACI_GATT_PROC_TIMEOUT_EVENT
- 0x00000004: ACI_ATT_EXCHANGE_MTU_RESP_EVENT
- 0x00000008: ACI_ATT_FIND_INFO_RESP_EVENT
- 0x00000010: ACI_ATT_FIND_BY_TYPE_VALUE_RESP_EVENT
- 0x00000020: ACI_ATT_READ_BY_TYPE_RESP_EVENT
- 0x00000040: ACI_ATT_READ_RESP_EVENT
- 0x00000080: ACI_ATT_READ_BLOB_RESP_EVENT
- 0x00000100: ACI_ATT_READ_MULTIPLE_RESP_EVENT
- 0x00000200: ACI_ATT_READ_BY_GROUP_TYPE_RESP_EVENT
- 0x00000800: ACI_ATT_PREPARE_WRITE_RESP_EVENT
- 0x00001000: ACI_ATT_EXEC_WRITE_RESP_EVENT
- 0x00002000: ACI_GATT_INDICATION_EVENT
- 0x00004000: ACI_GATT_NOTIFICATION_EVENT
- 0x00008000: ACI_GATT_ERROR_RESP_EVENT
- 0x00010000: ACI_GATT_PROC_COMPLETE_EVENT
- 0x00020000: ACI_GATT_DISC_READ_CHAR_BY_UUID_RESP_EVENT
- 0x00040000: ACI_GATT_TX_POOL_AVAILABLE_EVENT

Return values:
- *Value* indicating success or error code.

2.5.22 aci_gatt_srv_add_char

```
tBleStatus aci_gatt_srv_add_char    ( ble_gatt_chr_def_t * Char_p,
                                      uint16_t Service_Handle
                                     )
```

Adds a characteristic to a service.

Parameters

Char_p

The pointer to the Characteristic definition.

Service_Handle

Handle of the Service to which the characteristic is added. Values:
- 0x0001 ... 0xFFFF

Return values:
- *Value* indicating success or error code.

2.5.23 aci_gatt_srv_add_char_desc

```
tBleStatus aci_gatt_srv_add_char_desc    ( ble_gatt_descr_def_t * Descr_p,
                                           uint16_t Char_Handle
                                          )
```

Adds a characteristic descriptor to a characteristic.

Parameters

Descr_p

The pointer to the Descriptor definition.

Char_Handle

The Characteristic handle. Values:
- 0x0001 ... 0xFFFF

Return values:
- *Value* indicating success or error code.

2.5.24 aci_gatt_srv_add_service

```
tBleStatus aci_gatt_srv_add_service ( ble_gatt_srv_def_t * Service_p )
```

Adds a service to the GATT database. When a service is created, the host may reserve a range of handles for this service.

Parameters

Service_p

The pointer to the service definition.

Return values:
- *Value* indicating success or error code.

2.5.25 aci_gatt_srv_get_char_decl_handle

```
uint16_t aci_gatt_srv_get_char_decl_handle ( ble_gatt_chr_def_t * Char_p )
```

This function retrieves the Attribute Handle assigned to the Characteristic registered using the provided definition structure.

Parameters

Char_p

The Characteristic definition structure.

Return values:
- Attribute Handle of Service or BLE_ATT_INVALID_ATTR_HANDLE on error.

2.5.26 aci_gatt_srv_get_descriptor_handle

```
uint16_t aci_gatt_srv_get_descriptor_handle ( ble_gatt_descr_def_t * Descr_p )
```

This function retrieves the Attribute Handle assigned to the Characteristic Descriptor registered using the provided definition structure.

Parameters

Descr_p

The Characteristic Descriptor definition structure.

Return values:
- Attribute Handle of Service or BLE_ATT_INVALID_ATTR_HANDLE on error.

2.5.27 aci_gatt_srv_get_include_service_handle

```
uint16_t aci_gatt_srv_get_include_service_handle ( uint16_t Serv_Attr_Handle,
                                                    ble_gatt_srv_def_t * Included_Srv_p
                                                  )
```

This function retrieves the Attribute Handle assigned to the Include Service.

Parameters

Serv_Attr_Handle

The Handle of the including Service.

Included_Srv_p

The Included Service definition structure.

Return values:

- Attribute Handle of Service or BLE_ATT_INVALID_ATTR_HANDLE on error.

2.5.28 aci_gatt_srv_get_service_handle

```
uint16_t aci_gatt_srv_get_service_handle ( ble_gatt_srv_def_t * Serv_p )
```

This function retrieve the Attribute Handle assigned to the Service registered using the provided definition structure.

Parameters

Serv_p

The Service definition structure.

Return values:

- Attribute Handle of Service or BLE_ATT_INVALID_ATTR_HANDLE on error.

2.5.29 aci_gatt_srv_include_service

```
tBleStatus aci_gatt_srv_include_service    ( uint16_t Service_Handle,
                                             uint16_t  Included_Service_Handle
                                            )
```

Includes a service given by Included_Service_Handle to another service given by Service_Handle. Attribute server creates an Include definition attribute and returns the handle of this attribute.

Parameters

Service_Handle

The pointer to the Descriptor definition.

Included_Service_Handle

Attribute Handle of the Service which has to be included in service. Values:

- 0x0001 ... 0xFFFF

Return values:

- *Value* indicating success or error code.

2.5.30 aci_gatt_srv_multi_notify

```
tBleStatus aci_gatt_srv_multi_notify    ( uint16_t Connection_Handle,
                                          uint16_t CID,
                                          uint8_t  Flags,
                                          uint8_t  Num_Of_Attr,
                                          Gatt_Srv_Notify_Attr_t Gatt_Srv_Notify_Attr[]
                                         )
```

Notify multiple characteristic values to a client.

Parameters

Connection_Handle

Connection handle for which the attribute value is read.

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Flags

Reserved for future use. Its value must be set to 0. Values:

- 0x00

Num_Of_Attr

Gatt_Srv_Notify_Attr

See Gatt_Srv_Notify_Attr_t.

Return values:

- *Value* indicating success or error code.

2.5.31 aci_gatt_srv_notify

```
tBleStatus aci_gatt_srv_notify     ( uint16_t Connection_Handle,
                                     uint16_t CID,
                                     uint16_t Attr_Handle,
                                     uint8_t  Flags,
                                     uint16_t Val_Length,
                                     uint8_t  Val[]
                                    )
```

Send an indication or notification for the provided attribute handle. The Flags parameter indicates what kind of message is sent:

- 0x00: Send a notification
- 0x02: Send an indication.

Parameters

Connection_Handle

Connection handle to be used to identify the connection with the peer device. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Attr_Handle

Handle of the attribute to be notified. Values:

- 0x0001 ... 0xFFFF

Flags

Select the notification type. Values:

- 0x00: GATT_NOTIFICATION
- 0x02: GATT_INDICATION

Val_Length

Length of the Val field.

Val

Return values:

- *Value* indicating success or error code.

2.5.32 aci_gatt_srv_read_handle_value

```
tBleStatus aci_gatt_srv_read_handle_value    ( uint16_t Attr_Handle,
                                                uint16_t * Length,
                                                uint8_t  ** Value
                                               )
```

Reads the value of the attribute handle specified from the local GATT database. This command cannot be used to read attributes that have different values for each connection, e.g. the Client Characteristic Configuration Descriptors or Client Supported Features Characteristic (in this case, aci_gatt_srv_read_multiple_instance_handle_value needs to be used).

Parameters

Attr_Handle

Handle of the attribute to read Values:

- 0x0001 ... 0xFFFF

`[out]` **Length**

Length of the attribute value.

`[out]` **Value**

Pointer to the attribute value.

Return values:

- *Value* indicating success or error code.

2.5.33 aci_gatt_srv_read_multiple_instance_handle_value

```
tBleStatus aci_gatt_srv_read_multiple_instance_handle_value    ( uint16_t Connection_Handle,
                                                                  uint16_t Attr_Handle,
                                                                  uint16_t * Value_Length,
                                                                  uint8_t  ** Value
                                                                 )
```

Read the value for that kind of attributes that have different values for each connection, i.e. Client Characteristic Configuration Descriptors or Client Supported Features Characteristic.

Parameters

Connection_Handle

Connection handle for which the attribute value is read.

Attr_Handle

Handle of the attribute. Values:

- 0x0001 ... 0xFFFF

`[out]` **Value_Length**

Value length

`[out]` **Value**

Pointer to the buffer containing the value. Content may no more be valid after another call to this function or to BLE_STACK_Tick().

Return values:

- *Value* indicating success or error code.

2.5.34 aci_gatt_srv_resp

```
tBleStatus aci_gatt_srv_resp    ( uint16_t Connection_Handle,
                                  uint16_t CID,
                                  uint16_t Attr_Handle,
                                  uint8_t  Error_Code,
                                  uint16_t Val_Length,
                                  uint8_t  Val[]
                                )
```

Command to be given in response to aci_gatt_srv_read_event_rp0, aci_gatt_srv_write_event_rp0, aci_att_srv_prepare_write_req_event_rp0 or aci_att_srv_exec_write_req_event_rp0. It ends the ATT transaction initiated by the remote client.

Parameters

Connection_Handle

Connection handle to be used to identify the connection with the peer device. Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Attr_Handle

Attribute handle for which the response command is issued.

Error_Code

The reason why the request has generated an error response (ATT error codes). Values:

- 0x01: Invalid handle
- 0x02: Read not permitted
- 0x03: Write not permitted
- 0x04: Invalid PDU
- 0x05: Insufficient authentication
- 0x06: Request not supported
- 0x07: Invalid offset
- 0x08: Insufficient authorization
- 0x09: Prepare queue full
- 0x0A: Attribute not found
- 0x0B: Attribute not long
- 0x0C: Insufficient encryption key size
- 0x0D: Invalid attribute value length
- 0x0E: Unlikely error
- 0x0F: Insufficient encryption
- 0x10: Unsupported group type
- 0x11: Insufficient resources

Val_Length

Length of the Val field.

Val

Pointer to the value that must be returned in the response, in case this is a reply to an aci_gatt_srv_read_event_rp0. In other cases it is ignored.

Return values:

- *Value* indicating success or error code.

2.5.35 aci_gatt_srv_rm_char

```
tBleStatus aci_gatt_srv_rm_char ( uint16_t Char_Handle )
```

Deletes the specified characteristic from the service.

Parameters

Char_Handle

Handle of the characteristic which has to be deleted. Values:

- 0x0001 ... 0xFFFF

Return values:

- *Value* indicating success or error code.

2.5.36 aci_gatt_srv_rm_include_service

```
tBleStatus aci_gatt_srv_rm_include_service ( uint16_t Include_Handle )
```

Deletes the include definition from the service.

Parameters

Include_Handle

Handle of the included service which has to be deleted. Values:

- 0x0001 ... 0xFFFF

Return values:

- *Value* indicating success or error code.

2.5.37 aci_gatt_srv_rm_service

```
tBleStatus aci_gatt_srv_rm_service ( uint16_t Serv_Handle )
```

Deletes the specified service from the GATT server database.

Parameters

Serv_Handle

Handle of the service to be deleted. Values:

- 0x0001 ... 0xFFFF

Return values:

- *Value* indicating success or error code.

2.5.38 aci_gatt_srv_write_multiple_instance_handle_value

```
tBleStatus aci_gatt_srv_write_multiple_instance_handle_value    ( uint16_t Connection_Handle,
                                                                   uint16_t Attr_Handle,
                                                                   uint16_t * Value_Length,
                                                                   uint8_t  ** Value
                                                                  )
```

Updates an attribute value for that kind of attributes that have different values for each connection, that is, the Client Characteristic Configuration Descriptors.

Warning: *Use of this function can affect interoperability. Do not use the function unless you are aware of what you are doing.*

Parameters

Connection_Handle

Connection handle for which the attribute value is written.

Attr_Handle

Handle of the attribute. Values:

- 0x0001 ... 0xFFFF

Value_Length

Length of the attribute value in octets.

Value

Attribute value.

Return values:

- *Value* indicating success or error code.

2.5.39 aci_gatt_clt_add_subscription_security_level

```
tBleStatus aci_gatt_clt_add_subscription_security_level ( ble_gatt_clt_sec_level_st *
sec_level_p )
```

Set minimum security level to accept server-initiated packets (notification, indication, multiple notification). A default level can be specified using the 0xFFFF value for both the Conn_Handle and Char_Value_Handle values. If a notification is received when the requested minimum security level is not reached, it is discarded. If a multiple notification is received and at least one of its attributes doesn't reach requested minimum security level, the multiple notification is discarded. If an indication is received when the requested minimum security level is not reached, a confirmation is sent and the indication is discarded.

Parameters

sec_level_p

Pointer to the structure containing the parameters for the command: connection handle, attribute handle and minimum security level.

Return values:

- *Value* indicating success or error code.

2.5.40 aci_gatt_srv_write_handle_value_nwk

```
tBleStatus aci_gatt_srv_write_handle_value_nwk(uint16_t Attr_Handle,
                                               uint16_t Val_Offset,
                                               uint16_t Value_Length,
                                               uint8_t Value[])
```

Update an attribute value.

Parameters

Attr_Handle

Handle of the attribute. Values:

- 0x0001 ... 0xFFFF

Val_Offset

The offset from which the attribute value has to be updated. If this is set to 0 and the attribute value is of variable length, then the length of the attribute is set to the Char_Value_Length. If the Val_Offset is set to a value greater than 0, then the length of the attribute is set to the maximum length as specified for the attribute while adding the characteristic. Values:

- 0 ... 511

Value_Length

Length of the attribute value in octets

Value

Attribute value

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.5.41 aci_gatt_srv_set_security_permission_nwk

```
tBleStatus aci_gatt_srv_set_security_permission_nwk(uint16_t Attr_Handle,
                                                    uint8_t Security_Permissions)
```

This command sets the security permission for the attribute handle specified. Currently the setting of security permission is allowed only for client characteristic configuration descriptor.

Parameters

Attr_Handle

Handle of the attribute whose security permission has to be modified. Values:

- 0x0001 ... 0xFFFF

Security_Permissions

Security permission flags. Flags:

- 0x00
- 0x01: AUTHEN_READ (needs authentication to read)
- 0x04: ENCRY_READ (needs encryption to read)
- 0x08: AUTHEN_WRITE (needs authentication to write)
- 0x20: ENCRY_WRITE (needs encryption to write)

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.5.42 aci_gatt_srv_set_access_permission_nwk

```
tBleStatus aci_gatt_srv_set_access_permission_nwk(uint16_t Attr_Handle,
                                                  uint8_t Access_Permissions)
```

This command sets the access permission for the attribute handle specified.

Parameters

Attr_Handle

Handle of the attribute whose security permission has to be modified. Values:

- 0x0001 ... 0xFFFF

Access_Permissions

Access permission flags. Flags:

- 0x00
- 0x01: READ
- 0x02: WRITE
- 0x04: WRITE_NO_RESP
- 0x08: SIGNED_WRITE

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.5.43 aci_gatt_srv_exec_write_resp_nwk

```
tBleStatus aci_gatt_srv_exec_write_resp_nwk(uint16_t Conn_Handle,
                                            uint16_t CID,
                                            uint8_t Exec)
```

Response to an aci_att_srv_exec_write_req_event.

Parameters

Conn_Handle

Connection handle that identifies the connection.Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Exec

If 1, enables execution of queued writes. If 0, flushes all queued writes for the given connection handle. Values:

- 0x00: FLUSH
- 0x01: EXECUTE

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.5.44 aci_gatt_srv_authorize_resp_nwk

```
tBleStatus aci_gatt_srv_authorize_resp_nwk(uint16_t Conn_Handle,
                                           uint16_t CID,
                                           uint16_t Attr_Handle,
                                           uint8_t Operation_Type,
                                           uint8_t Error_Code,
                                           uint16_t Attr_Val_Offset,
                                           uint16_t Data_Length,
                                           uint8_t Data[])
```

This command should be sent when ACI_GATT_SRV_AUTHORIZE_NWK_EVENT is received.

Parameters

Conn_Handle

Connection handle to be used to identify the connection with the peer device.Values:

- 0x0000 ... 0x0EFF

CID

Channel Identifier of the ATT bearer. It must be set to 0x0004 for unenhanced ATT bearer.

Attr_Handle

Offset from which the value needs to be read or write. Values:

- 0 ... 511

Operation_Type

Values:

- 0x00: Read
- 0x10: Write Request
- 0x11: Write Command or Signed Write Command
- 0x12: Prepare Write Request

Error_Code

Set to 0 if operation is authorized, otherwise Error_Code is the ATT error code that is sent to the peer in response to the request.Values:

- 0: Authorize
- - 0 ... 255

Attr_Val_Offset

Offset from which the attribute needs to be read or written. For a read operation it is always 0.Values:

- 0 ... 511

param Data_Length

Length of Data field.

Data

The data that the client has requested to write.

Return values:

- *Value* indicating success or error code.

Note: *This command is available only on network coprocessor framework.*

2.5.45 aci_gatt_srv_read_prepare_queue_nwk

```
tBleStatus aci_gatt_srv_read_prepare_queue_nwk(uint16_t Conn_Handle,
                                               uint8_t Item_Index,
                                               uint16_t *Attr_Handle,
                                               uint16_t *Value_Offset,
                                               uint16_t *Value_Length,
                                               uint8_t Value[])
```

Read the content of the prepare write queue. This command should be used to atomically read all the queued write operations after a aci_att_srv_exec_write_req_event is received and before sending the aci_gatt_srv_exec_write_resp.

Parameters

Conn_Handle

Connection handle to be used to identify the connection with the peer device.Values:

- 0x0000 ... 0x0EFF

Item_Index

The index of the entry in the queue for the selected connection handle.

`[out]` Attr_Handle

The attribute handle of the returned entry.

`[out]` Value_Offset

The offset from which the peer is requesting to start writing.

`[out]` **Value_Length**

Length in octets of the Value parameter.

`[out]` **Value**

The value to be written.

Return values:

- *Value* indicating success or error code.

Note: This command is available only on network coprocessor framework.

2.6 L2CAP commands

This section describes the supported L2CAP commands.

Table 6. **L2CAP commands opcodes**

SoC command	Network coprocessor command	OpCode
aci_l2cap_connection_parameter_update_req	aci_l2cap_connection_parameter_update_req	0xFD81
aci_l2cap_connection_parameter_update_resp	aci_l2cap_connection_parameter_update_resp	0xFD82
aci_l2cap_cos_connection_req	aci_l2cap_cos_connection_req	0xFD83
aci_l2cap_cos_connection_resp	aci_l2cap_cos_connection_resp	0xFD84
aci_l2cap_cos_flow_control_credits_ind	aci_l2cap_cos_flow_control_credits_ind	0xFD85
aci_l2cap_cos_disconnect_req	aci_l2cap_cos_disconnect_req	0xFD86
aci_l2cap_cos_sdu_data_transmit	aci_l2cap_cos_sdu_data_transmit	0xFD87
aci_l2cap_cos_reconfigure_req	aci_l2cap_cos_reconfigure_req	0xFD8A
aci_l2cap_cos_reconfigure_resp	aci_l2cap_cos_reconfigure_resp	0xFD8B
aci_l2cap_cos_sdu_data_extract	-	0xFD92

2.6.1 aci_l2cap_connection_parameter_update_req

```
tBleStatus aci_l2cap_connection_parameter_update_req    ( uint16_t Connection_Handle,
                                                          uint16_t Connection_Interval_Min,
                                                          uint16_t Connection_Interval_Max,
                                                          uint16_t Peripheral_Latency,
                                                          uint16_t Timeout_Multiplier
                                                         )
```

Send an L2CAP connection parameter update request from the peripheral to the central. An aci_l2cap_connection_update_resp_event_rp0 event is raised when the central responds to the request (accepts or rejects).

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Connection_Interval_Min

Minimum value for the connection event interval. This shall be less than or equal to Connection_Interval_Max. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Connection_Interval_Max

Maximum value for the connection event interval. This shall be greater than or equal to Connection_Interval_Min. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Peripheral_Latency

Maximum Peripheral latency for the connection in number of connection events. Values:

- 0x0000 ... 0x01F3

Timeout_Multiplier

Defines connection timeout parameter in the following manner: Timeout Multiplier * 10ms. Values:

- 10 (100 ms) ... 3200 (32000 ms)

Return values:

- *Value* indicating success or error code.

2.6.2 aci_l2cap_connection_parameter_update_resp

```
tBleStatus aci_l2cap_connection_parameter_update_resp    ( uint16_t Connection_Handle,
                                                           uint16_t Connection_Interval_Min,
                                                           uint16_t Connection_Interval_Max,
                                                           uint16_t Peripheral_Latency,
                                                           uint16_t Timeout_Multiplier,
                                                           uint16_t Min_CE_Length,
                                                           uint16_t Max_CE_Length,
                                                           uint8_t Identifier,
                                                           uint8_t Accept
                                                          )
```

Accept or reject a connection update. This command should be sent in response to a aci_l2cap_connection_update_req_event_rp0 event from the controller. The accept parameter has to be set if the connection parameters given in the event are acceptable.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

Connection_Interval_Min

Minimum value for the connection event interval. This shall be less than or equal to Connection_Interval_Max. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Connection_Interval_Max

Maximum value for the connection event interval. This shall be greater than or equal to Connection_Interval_Min. Time = N * 1.25 msec. Values:

- 0x0006 (7.50 ms) ... 0x0C80 (4000.00 ms)

Peripheral_Latency

Maximum Peripheral latency for the connection in number of connection events. Values:

- 0x0000 ... 0x01F3

Timeout_Multiplier

Defines connection timeout parameter in the following manner: Timeout Multiplier * 10 ms. Values:

- 10 (100 ms) ... 3200 (32000 ms)

Min_CE_Length

The minimum length of connection event recommended for this LE connection. Time = N * 0.625 ms.

Max_CE_Length

The maximum length of connection event recommended for this LE connection. Time = N * 0.625 ms.

Identifier

Identifier received in ACI_L2CAP_Connection_Update_Req event.

Accept

Specify if connection update parameters are acceptable or not. Values:

- 0x00: Reject
- 0x01: Accept

Return values:

- *Value* indicating success or error code.

2.6.3 aci_l2cap_cos_connection_req

```
tBleStatus aci_l2cap_cos_connection_req    ( uint16_t Connection_Handle,
                                               uint16_t SPSM,
                                               uint16_t MTU,
                                               uint16_t MPS,
                                               uint8_t  Channel_Type,
                                               uint8_t  CID_Count
                                             )
```

Create and configure an L2CAP channel between two devices using either LE Credit Based Flow Control Mode or Enhanced Credit Based Flow Control Mode.

Parameters

Connection_Handle

Handle identifying the connection.

SPSM

Simplified Protocol/Service Multiplexer. Values:

- 0x0001 ... 0x00FF

MTU

The maximum SDU size (in octets) that the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_REQ can receive on this channel. Values:

- 23 ... 65535

MPS

The maximum PDU payload size (in octets) that the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_REQ is capable of receiving on this channel. Values:

- 23 ... 65533

Channel_Type

Type of channel: LE Credit Based Flow Control Mode or Enhanced Credit Based Flow Control Mode. Values:

- 0x00: L2CAP_CHANNEL_TYPE_LE_CFC
- 0x01: L2CAP_CHANNEL_TYPE_ECFC

CID_Count

Return values:

- *Value* indicating success or error code.

2.6.4 aci_l2cap_cos_connection_resp

```
tBleStatus aci_l2cap_cos_connection_resp    ( uint16_t Connection_Handle,
                                              uint8_t Identifier,
                                              uint16_t MTU,
                                              uint16_t MPS,
                                              uint16_t Result,
                                              uint8_t  CID_Count,
                                              uint16_t CID[]
                                            )
```

Command to be sent to respond to a request to open an L2CAP channel using LE Credit based Flow Control or Enhanced Credit Based Flow Control Mode. The request is notified through aci_l2cap_cos_connection_req_event_rp0.

Parameters

Connection_Handle

Handle identifying the connection.

Identifier

Identifier of the request.

MTU

The maximum SDU size (in octets) that the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_REQ can receive on this channel. Values:

- 23 ... 65535

MPS

The MPS field specifies the maximum PDU payload size (in octets) that the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_RSP is capable of receiving on this channel. Values:

- 23 ... 65533

Result

It indicates the outcome of the connection request. A result value of 0x0000 indicates success while a non-zero value indicates a fail. Code values starting from 0x000C can be used only for ECFC channel type. Values:

- 0x0000: L2CAP_CONN_SUCCESSFUL
- 0x0002: L2CAP_CONN_FAIL_SPSM_NOT_SUPPORTED
- 0x0004: L2CAP_CONN_FAIL_INSUFFICIENT_RESOURCES
- 0x0005: L2CAP_CONN_FAIL_INSUFFICIENT_AUTHENTICATION
- 0x0006: L2CAP_CONN_FAIL_INSUFFICIENT_AUTHORIZATION
- 0x0007: L2CAP_CONN_FAIL_KEY_SIZE_TOO_SHORT
- 0x0008: L2CAP_CONN_FAIL_INSUFFICIENT_ENCRYPTION
- 0x000B: L2CAP_CONN_FAIL_UNACCEPTABLE_PARAMETERS
- 0x000C: L2CAP_CONN_FAIL_INVALID_PARAMETERS
- 0x000D: L2CAP_CONN_FAIL_NO_INFO
- 0x000E: L2CAP_CONN_FAIL_AUTHENTICATION_PENDING
- 0x000F: L2CAP_CONN_FAIL_AUTHORIZATION_PENDING

CID_Count

CID

Return values:

- *Value* indicating success or error code.

2.6.5 aci_l2cap_cos_disconnect_req

```
tBleStatus aci_l2cap_cos_disconnect_req    ( uint16_t Connection_Handle,
                                             uint16_t CID
                                            )
```

Command to terminate an L2CAP channel.

Parameters

Connection_Handle

CID

Local endpoint of the channel to be disconnected.

Return values:

- *Value* indicating success or error code.

2.6.6 aci_l2cap_cos_reconfigure_req

```
tBleStatus aci_l2cap_cos_reconfigure_req    ( uint16_t Connection_Handle,
                                              uint16_t MTU,
                                              uint16_t MPS,
                                              uint8_t  CID_Count,
                                              uint16_t CID[]
                                             )
```

Command to send an L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet in order to request to change its receive MTU or MPS values compared to when the channels were created or last reconfigured.

Parameters

Connection_Handle

Identifier received in the aci_eatt_connection_event_rp0.

MTU

The maximum SDU size (in octets) that the L2CAP layer entity can receive on each of the Source CID channels (represented by Local_CID array parameter). This is equal to the maximum size of an ATT packet on the Enhanced ATT bearer. Values:

- 0x0040 ... 0xFFFF

MPS

The maximum PDU payload size (in octets) that the local L2CAP layer is capable of receiving on each of the Source CID channels (represented by Local_CID array parameter). Values:

- 0x0040 ... 0xFFFF

CID_Count

The number of potential Enhanced ATT bearers that are going to be opened. This is the number of L2CAP channels to be opened in Enhanced Credit Based Flow Control mode. Values:

- 0x01 ... 0x05

CID

List of CID values representing the channel endpoints on the local device. Each entry in the array shall be non-zero and represents a request for a channel. The value of each CID shall be from the dynamically allocated range for LE devices (0x0040-0x007F) and shall not be already allocated to a different channel on the device sending the request.

Return values:

- *Value* indicating success or error code.

2.6.7 aci_l2cap_cos_reconfigure_resp

```
tBleStatus aci_l2cap_cos_reconfigure_resp    ( uint16_t Connection_Handle,
                                              uint8_t Identifier,
                                              uint16_t Result,
                                             )
```

Command to send an L2CAP_CREDIT_BASED_RECONFIGURE_RSP packet in order to respond to an incoming L2CAP_CREDIT_BASED_RECONFIGURE_REQ. It has to be used upon the reception of an ACI_L2CAP_ECFC_RECONFIGURATION_EVENT.

Parameters

Connection_Handle

Identifier received in the aci_eatt_connection_event_rp0.

Identifier

Identifier received in the aci_eatt_connection_event_rp0.

Result

It indicates the outcome of the connection request. A result value of 0x0000 indicates success while a non-zero value indicates the connection request was refused. Values:

- 0x0000: L2CAP_RECONFIG_SUCCESSFUL
- 0x0001: L2CAP_MTU_REDUCTION_NOT_ALLOWED
- 0x0002: L2CAP_MPS_REDUCTION_NOT_ALLOWED
- 0x0003: L2CAP_INVALID_DESTINATION_CID
- 0x0004: L2CAP_UNACCEPTABLE_PARAMETERS

Return values:

- *Value* indicating success or error code.

2.6.8 aci_l2cap_cos_sdu_data_extract

```
tBleStatus aci_l2cap_cos_sdu_data_extract    ( uint16_t Connection_Handle,
                                               uint16_t CID,
                                               uint16_t SDU_Data_Buffer_Size,
                                               void * SDU_Data_Buffer,
                                               uint16_t * SDU_Length
                                             )
```

Function to be used to extract an SDU from receiving buffer.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

The local channel endpoint that identifies the L2CAP channel.

SDU_Data_Buffer_Size

Size of the buffer where SDU is copied.

SDU_Data_Buffer

Buffer where the extracted SDU is copied.

`[out]` **SDU_Length**

Length of the extracted SDU.

Return values:

- *Value* indicating success or error code.

2.6.9 aci_l2cap_cos_sdu_data_transmit

```
tBleStatus aci_l2cap_cos_sdu_data_transmit    ( uint16_t Connection_Handle,
                                                uint16_t CID,
                                                uint16_t SDU_Length
                                                uint8_t  SDU_Data[]
                                              )
```

Function to be called to send an SDU using an L2CAP channel in LE Credit Based Flow Control mode or Enhanced Credit Based Flow Control Mode.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:

- 0x0000 ... 0x0EFF

CID

The local channel endpoint that identifies the L2CAP channel.

SDU_Length

Length of the SDU to be transmitted.

SDU_Data

Data contained in the SDU to be transmitted. Data must be valid until the SDU is transmitted.

Return values:

- *Value* indicating success or error code.

2.6.10 aci_l2cap_cos_flow_control_credits_ind

```
tBleStatus aci_l2cap_cos_flow_control_credits_ind    ( uint16_t Connection_Handle,
                                                       uint16_t  CID,
                                                       uint16_t  RX_Credits,
                                                       uint8_t   CFC_Policy,
                                                       uint16_t * RX_Credit_Balance )
```

Command to be issued when the device is capable of receiving additional K-frames in Bluetooth® LE credit-based flow control mode.

Parameters

Connection_Handle

Connection handle that identifies the connection. Values:
- 0x0001 ... 0xFFFF

CID

The local channel endpoint that identifies the L2CAP channel.

RX_Credits

Additional number of K-frames that the local L2CAP layer entity can currently receive from the peer.

CFC_Policy

Policy to handle flow control. If the value is 0, flow control is handled by application and credits must be sent using aci_l2cap_send_flow_control_credits(). If the value is 1, flow control is handled automatically by the stack. Values:
- 0x00: L2CAP_CFC_MANUAL
- 0x01: L2CAP_CFC_AUTO

`[out]` **RX_Credit_Balance**

Current number of K-frames that the L2CAP layer entity of the peer can send.

Return values:
- *Value* indicating success or error code.

3 ACI/HCI event codes

3.1 HCI event codes

This section describes the supported HCI event codes.

3.1.1 Disconnection Complete

Event code: HCI_DISCONNECTION_COMPLETE_EVT_CODE

Event code value: 0x05

Event description: The Disconnection Complete event occurs when a connection is terminated. The status parameter indicates if the disconnection was successful or not. The reason parameter indicates the reason for the disconnection if the disconnection was successful. If the disconnection was not successful, the value of the reason parameter can be ignored by the Host. For example, this can be the case if the Host has issued the Disconnect command and there was a parameter error, or the command was not presently allowed, or a Connection_Handle that did not correspond to a connection was given.

3.1.2 Encryption Change

Event code: HCI_ENCRYPTION_CHANGE_EVT_CODE

Event code value: 0x08

Event description: The Encryption Change event is used to indicate that the change of the encryption mode has been completed. The Connection_Handle is a Connection_Handle for an ACL connection. The Encryption_Enabled event parameter specifies the new Encryption_Enabled parameter for the Connection_Handle specified by the Connection_Handle event parameter. This event occurs on both devices to notify the Hosts when Encryption has changed for the specified Connection_Handle between two devices.

Note: This event shall not be generated if encryption is paused or resumed; during a role switch, for example. The meaning of the Encryption_Enabled parameter depends on whether the Host has indicated support for Secure Connections in the Secure_Connections_Host_Support parameter. When Secure_Connections_Host_Support is 'disabled' or the Connection_Handle refers to an LE link, the Controller shall only use Encryption_Enabled values 0x00 (OFF) and 0x01 (ON). (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.7.8.)

3.1.3 Read Remote Version Information Complete

Event code: HCI_READ_REMOTE_VERSION_INFORMATION_COMPLETE_EVT_CODE

Event code value: 0x0C

Event description: The Read Remote Version Information Complete event is used to indicate the completion of the process obtaining the version information of the remote Controller specified by the Connection_Handle event parameter. The Connection_Handle is for an ACL connection. The Version event parameter defines the specification version of the LE Controller. The Manufacturer_Name event parameter indicates the manufacturer of the remote Controller. The Subversion event parameter is controlled by the manufacturer and is implementation dependent. The Subversion event parameter defines the various revisions that each version of the Bluetooth® hardware goes through as design processes change and errors are fixed. This allows the software to determine what Bluetooth® hardware is used and, if necessary, to work around various bugs in the hardware. When the Connection_Handle is associated with an LE-U logical link, the Version event parameter shall be the Link Layer VersNr parameter, the Manufacturer_Name event parameter shall be the CompId parameter, and the Subversion event parameter shall be the SubVersNr parameter. (See Bluetooth Specification v.4.1, Vol. 2, Part E, 7.7.12.)

3.1.4 Hardware Error

Event code: HCI_HARDWARE_ERROR_EVT_CODE

Event code value: 0x10

Event description: The Hardware Error event is used to indicate some implementation specific type of hardware failure for the controller. This event is used to notify the Host that a hardware failure has occurred in the Controller.

3.1.5 Number Of Completed Packets

Event code: HCI_NUMBER_OF_COMPLETED_PACKETS_EVT_CODE

Event code value: 0x13

Event description: The Number Of Completed Packets event is used by the Controller to indicate to the Host how many HCI Data Packets have been completed (transmitted or flushed) for each Connection_Handle since the previous Number Of Completed Packets event was sent to the Host. This means that the corresponding buffer space has been freed in the Controller. Based on this information, and the HC_Total_Num_ACL_Data_Packets and HC_Total_Num_Synchronous_- Data_Packets return parameter of the Read_Buffer_Size command, the Host can determine for which Connection_Handles the following HCI Data Packets should be sent to the Controller. The Number Of Completed Packets event must not be sent before the corresponding Connection Complete event. While the Controller has HCI data packets in its buffer, it must keep sending the Number Of Completed Packets event to the Host at least periodically, until it finally reports that all the pending ACL Data Packets have been transmitted or flushed.

3.1.6 Data Buffer Overflow

Event code: HCI_DATA_BUFFER_OVERFLOW_EVT_CODE

Event code value: 0x1A

Event description: The Data Buffer Overflow event is used to indicate that the Controller's data buffers have overflowed. This can occur if the Host sends more packets than allowed. The Link_Type parameter is used to indicate that the overflow was caused by ACL data.

3.1.7 Encryption Key Refresh Complete

Event code: HCI_ENCRYPTION_KEY_REFRESH_COMPLETE_EVT_CODE

Event code value: 0x30

Event description: The Encryption Key Refresh Complete event is used to indicate to the Host that the encryption key was refreshed on the given Connection_Handle any time encryption is paused and then resumed. If the Encryption Key Refresh Complete event was generated due to an encryption pause and resume operation embedded within a change connection link key procedure, the Encryption Key Refresh Complete event shall be sent prior to the Change Connection Link Key Complete event. If the Encryption Key Refresh Complete event was generated due to an encryption pause and resume operation embedded within a role switch procedure, the Encryption Key Refresh Complete event shall be sent prior to the Role Change event.

3.1.8 Authenticated Payload Timeout Expired

Event code: HCI_AUTHENTICATED_PAYLOAD_TIMEOUT_EXPIRED_EVT_CODE

Event code value: 0x57

Event description: The Authenticated Payload Timeout Expired event is used to indicate that a packet containing a valid MIC on the Connection_Handle was not received within the authenticatedPayloadTO.

Note: A Host may choose to disconnect the link when this occurs.

3.1.9 LE Meta

Event code: HCI_LE_META_EVT_CODE

Event code value: 0x3E

Event description: Format of Standard Bluetooth® LE Meta Events.

3.1.10 Vendor

Event code: HCI_VENDOR_EVT_CODE

Event code value: 0xFF

Event description: Format of proprietary events.

3.2 HCI LE meta subevent codes

This section describes the supported HCI LE meta subevent codes.

3.2.1 LE Connection Complete

Event code: HCI_LE_CONNECTION_COMPLETE_SUBEVT_CODE

Event code value: 0x01

Event description: The LE Connection Complete event indicates to both of the Hosts forming the connection that a new connection has been created. Upon the creation of the connection a Connection_Handle shall be assigned by the Controller, and passed to the Host in this event. If the connection establishment fails this event shall be provided to the Host that had issued the LE_Create_Connection command. This event indicates to the Host which issued a LE_Create_Connection command and received a Command Status event if the connection establishment failed or was successful. The Central_Clock_Accuracy parameter is only valid for a peripheral. On a central, this parameter shall be set to 0x00.

3.2.2 LE Advertising Report

Event code: HCI_LE_ADVERTISING_REPORT_SUBEVT_CODE

Event code value: 0x02

Event description: The LE Advertising Report event indicates that a Bluetooth® device or multiple Bluetooth® devices have responded to an active scan or received some information during a passive scan. The Controller may queue these advertising reports and send information from multiple devices in one LE Advertising Report event.

3.2.3 LE Connection Update Complete

Event code: HCI_LE_CONNECTION_UPDATE_COMPLETE_SUBEVT_CODE

Event code value: 0x03

Event description: The LE Connection Update Complete event is used to indicate that the Controller process to update the connection has completed. On a peripheral, if no connection parameters are updated, then this event shall not be issued. On a central, this event shall be issued if the Connection_Update command was sent.

3.2.4 LE Read Remote Features Complete

Event code: HCI_LE_READ_REMOTE_FEATURES_COMPLETE_SUBEVT_CODE

Event code value: 0x04

Event description: The LE Read Remote Features Complete event is used to indicate the completion of the process of the Controller obtaining the used features of the remote Bluetooth® device specified by the Connection_Handle event parameter.

3.2.5 LE Long Term Key Request

Event code: HCI_LE_LONG_TERM_KEY_REQUEST_SUBEVT_CODE

Event code value: 0x05

Event description: The LE Long Term Key Request event indicates that the central device is attempting to encrypt or re-encrypt the link and is requesting the Long Term Key from the Host. (See [Vol 6] Part B, Section 5.1.3.)

3.2.6 LE Data Length Change

Event code: HCI_LE_DATA_LENGTH_CHANGE_SUBEVT_CODE

Event code value: 0x07

Event description: The LE Data Length Change event notifies the Host of a change to either the maximum Payload length or the maximum transmission time of Data Channel PDUs in either direction. The values reported are the maximum values used on the connection following the change.

3.2.7 LE Read Local P-256 Public Key Complete

Event code: HCI_LE_READ_LOCAL_P256_PUBLIC_KEY_COMPLETE_SUBEVT_CODE

Event code value: 0x08

Event description: The LE Read Local P-256 Public Key Complete event is generated when local P-256 key generation is complete.

3.2.8 LE Generate DHKey Complete

Event code: HCI_LE_GENERATE_DHKEY_COMPLETE_SUBEVT_CODE

Event code value: 0x09

Event description: The LE Generate DHKey Complete event indicates that the LE Diffie-Hellman key generation has been completed by the Controller.

3.2.9 LE Enhanced Connection Complete

Event code: HCI_LE_ENHANCED_CONNECTION_COMPLETE_SUBEVT_CODE

Event code value: 0x0A

Event description: The LE Enhanced Connection Complete event indicates to both of the Hosts forming the connection that a new connection has been created. Upon the creation of the connection a Connection_Handle shall be assigned by the Controller, and passed to the Host in this event. If the connection establishment fails, this event shall be provided to the Host that had issued the LE_Create_Connection command. If this event is unmasked and LE Connection Complete event is unmasked, only the LE Enhanced Connection Complete event is sent when a new connection has been completed. This event indicates to the Host that issued a LE_Create_Connection command and received a Command Status event if the connection establishment failed or was successful. The Central_Clock_Accuracy parameter is only valid for a peripheral. On a central, this parameter shall be set to 0x00.

3.2.10 LE Directed Advertising Report

Event code: HCI_LE_DIRECTED_ADVERTISING_REPORT_SUBEVT_CODE

Event code value: 0x0B

Event description: The LE Directed Advertising Report event indicates that directed advertisements have been received where the advertiser is using a resolvable private address for the TargetA field of the advertising PDU which the Controller is unable to resolve and the Scanning_Filter_Policy is equal to 0x02 or 0x03. Direct_Address_Type and Direct_Address specify the address the directed advertisements are being directed to. Address_Type and Address specify the address of the advertiser sending the directed advertisements. The Controller may queue these advertising reports and send information from multiple advertisers in one HCI_LE_Directed_Advertising_Report event. This event shall only be generated if scanning was enabled using the HCI_LE_Set_Scan_Enable command. It only reports advertising events that used legacy advertising PDUs.

3.2.11 LE PHY Update Complete

Event code: HCI_LE_PHY_UPDATE_COMPLETE_SUBEVT_CODE

Event code value: 0x0C

Event description: The LE PHY Update Complete Event is used to indicate that the Controller has changed the transmitter PHY or receiver PHY in use. If the Controller changes the transmitter PHY, the receiver PHY, or both PHYs, this event shall be issued. If an LE_Set_PHY command was sent and the Controller determines that neither PHY changes as a result, it issues this event immediately.

3.2.12 LE Extended Advertising Report

Event code: HCI_LE_EXTENDED_ADVERTISING_REPORT_SUBEVT_CODE

Event code value: 0x0D

Event description: The LE Extended Advertising Report event indicates that one or more Bluetooth® devices have responded to an active scan or have broadcast advertisements that were received during a passive scan. The Controller may coalesce multiple advertising reports from the same or different advertisers into a single LE Extended Advertising Report event, provided all the parameters from all the advertising reports fit in a single HCI event. This event shall only be generated if scanning was enabled using the LE Set Extended Scan Enable command. It reports advertising events using either legacy or extended advertising PDUs. The Controller may split the data from a single advertisement (whether one PDU or several) into several reports. If so, each report except the last shall have an Event_Type with a data status field of "incomplete, more data to come", while the last shall have the value "complete"; the Address_Type, Address, Advertising_SID, Primary_PHY, and Secondary_PHY fields shall be the same in all the reports. When a scan response is received, bits 0-2 and 4 of the event type shall indicate the properties of the original advertising event. An Event_Type with a data status field of "incomplete, data truncated" indicates that the Controller attempted to receive an AUX_CHAIN_IND PDU but was not successful.

3.2.13 LE Periodic Advertising Sync Established

Event code: HCI_LE_PERIODIC_ADVERTISING_SYNC_ESTABLISHED_SUBEVT_CODE

Event code value: 0x0E

Event description: The LE Periodic Advertising Report event indicates that the Controller has received a Periodic Advertising packet. The Sync_Handle parameter indicates the identifier for the periodic advertisements specified by the Advertising SID subfield of the ADI field in the ADV_EXT_IND PDU. The Controller may split the data from a single periodic advertisement (whether one PDU or several) into several reports. If so, each report except the last shall have a Data_Status of "incomplete, more data to come", while the last shall have the value "complete". A Data_Status of "incomplete, data truncated" indicates that the Controller attempted to receive an AUX_CHAIN_IND PDU but was not successful. The Unused parameter shall be set to 0xFF by the Controller and ignored by the Host.

3.2.14 LE Periodic Advertising Report

Event code: HCI_LE_PERIODIC_ADVERTISING_REPORT_SUBEVT_CODE

Event code value: 0x0F

Event description: The LE Periodic Advertising Report event indicates that the Controller has received a Periodic Advertising packet. The Sync_Handle parameter indicates the identifier for the periodic advertisements specified by the Advertising SID subfield of the ADI field in the ADV_EXT_IND PDU. The Controller may split the data from a single periodic advertisement (whether one PDU or several) into several reports. If so, each report except the last shall have a Data_Status of "incomplete, more data to come", while the last shall have the value "complete". A Data_Status of "incomplete, data truncated" indicates that the Controller attempted to receive an AUX_CHAIN_IND PDU but was not successful. The Unused parameter shall be set to 0xFF by the Controller and ignored by the Host.

3.2.15 LE Periodic Advertising Sync Lost

Event code: HCI_LE_PERIODIC_ADVERTISING_SYNC_LOST_SUBEVT_CODE

Event code value: 0x10

Event description: The LE Periodic Advertising Sync Lost event indicates that the Controller has not received a Periodic Advertising packet identified by Sync_Handle within the timeout period.

3.2.16 LE Scan Timeout

Event code: HCI_LE_SCAN_TIMEOUT_SUBEVT_CODE

Event code value: 0x11

Event description:

3.2.17 LE Advertising Set Terminated

Event code: HCI_LE_ADVERTISING_SET_TERMINATED_SUBEVT_CODE

Event code value: 0x12

Event description: The LE Advertising Set Terminated event indicates that the Controller has terminated advertising in the advertising sets specified by the Advertising_Handle parameter. This event shall be generated every time connectable advertising in an advertising set results in a connection being created. This event shall only be generated if advertising was enabled using the LE Set Extended Advertising Enable command. The Connection_Handle parameter is only valid when advertising ends because a connection was created. If the Max_Extended_Advertising_Events parameter in the LE_Set_Extended_Advertising_Enable command was non-zero, the Num_Completed_Extended_Advertising_Events parameter shall be set to the number of completed extended advertising events the Controller had transmitted when either the duration elapsed or the maximum number of extended advertising events was reached; otherwise it shall be set to zero. If advertising has terminated as a result of the advertising duration elapsing, the Status parameter shall be set to the error code Advertising Timeout (0x3C). If advertising has terminated because the Max_Extended_Advertising_Events was reached, the Status parameter shall be set to the error code Limit Reached (0x43).

3.2.18 LE Scan Request Received

Event code: HCI_LE_SCAN_REQUEST_RECEIVED_SUBEVT_CODE

Event code value: 0x13

Event description: The LE Scan Request Received event indicates that a SCAN_REQ PDU or an AUX_SCAN_REQ PDU has been received by the advertiser. The request contains a device address from a scanner that is allowed by the advertising filter policy. The advertising set is identified by Advertising_Handle. This event shall only be generated if advertising was enabled using the LE Set Extended Advertising Enable command. The Scanner_Address_Type and Scanner_Address indicates the type of the address and the address of the scanner device.

3.2.19 LE Channel Selection Algorithm

Event code: HCI_LE_CHANNEL_SELECTION_ALGORITHM_SUBEVT_CODE

Event code value: 0x14

Event description: The LE Channel Selection Algorithm Event indicates which channel selection algorithm is used on a data channel connection (see [Vol 6] Part B, Section 4.5.8).

3.2.20 LE Connectionless IQ Report

Event code: HCI_LE_CONNECTIONLESS_IQ_REPORT_SUBEVT_CODE

Event code value: 0x15

Event description: The LE Connectionless IQ Report event is used by the Controller to report IQ information from the Constant Tone Extension of a received advertising packet forming part of the periodic advertising train identified by Sync_Handle and to report IQ information from the Constant Tone Extension of a received Test Mode packet (see Section 7.8.28). The index of the channel on which the packet was received, the RSSI of the packet (excluding the Constant Tone Extension), the ID of the antenna on which this was measured, the type of Constant Tone Extension, the value of paEventCounter, and the IQ samples of the Constant Tone Extension of the advertisement are reported in the corresponding parameters. For any given sample, either both or neither of I_Sample[i] and Q_Sample[i] shall equal 0x80. The Slot_Durations parameter specifies the sampling rate used by the Controller. The Packet_Status parameter indicates whether the received packet had a valid CRC and, if not, whether the Controller has determined the position and size of the Constant Tone Extension using the Length and CTETime fields. Note: A Controller is not required to generate this event for packets that have a bad CRC. The Constant Tone Extension format is defined in [Vol 6] Part B, Section 2.5.1. If the PDU contains AdvData, then the HCI_LE_Periodic_Advertising_Report event shall be generated before this event. The Controller is not required to generate this event for a Constant Tone Extension with a type that it does not support. This event is also used by the Controller to report that it has insufficient resources to report IQ samples for all received Constant Tone Extensions and has failed to sample at least once. In this case Packet_Status shall be set to 0xFF and Sample_Count to 0x00.

3.2.21 LE Connection IQ Report

Event code: HCI_LE_CONNECTION_IQ_REPORT_SUBEVT_CODE

Event code value: 0x16

Event description: The LE Connection IQ Report event is used by the Controller to report the IQ samples from the Constant Tone Extension of a received packet (see [Vol 6] Part B, Section 2.4.2.26). The Connection_Handle parameter identifies the connection that corresponds to the reported information. The receiver PHY, the index of the data channel, the RSSI value of the packet (excluding the Constant Tone Extension), the ID of the antenna on which this was measured, the type of Constant Tone Extension, the value of connEventCounter, and the IQ samples of the Constant Tone Extension of the received packet are reported in the corresponding parameters. For any given sample, either both or neither of I_Sample[i] and Q_Sample[i] shall equal 0x80. The Slot_Durations parameter specifies the sampling rate used by the Controller. The Packet_Status parameter indicates whether the received packet had a valid CRC and, if not, whether the Controller has determined the position and size of the Constant Tone Extension using the Length and CTETime fields. Note: A Controller is not required to generate this event for packets that have a bad CRC. This event is also used by the Controller to report that it has insufficient resources to report IQ samples for all received Constant Tone Extensions and has failed to sample at least once. In this case Packet_Status shall be set to 0xFF and Sample_Count to 0x00. The Constant Tone Extension format is defined in [Vol 6] Part B, Section 2.1.5.

3.2.22 LE CTE Request Failed

Event code: HCI_LE_CTE_REQUEST_FAILED_SUBEVT_CODE

Event code value: 0x17

Event description: The LE CTE Request Failed event is used by the Controller to report an issue following a request to a peer device to reply with a packet containing an LL_CTE_RSP PDU and a Constant Tone Extension. It shall be generated if the packet containing the LL_CTE_RSP PDU sent in response did not contain a Constant Tone Extension or if the peer rejected the request. It shall not be generated if the packet containing the LL_CTE_RSP PDU had a CRC error or if the procedure response timeout timer (see [Vol 6] Part B, Section 5.2) expired.

3.2.23 LE Periodic Advertising Sync Transfer Received

Event code: HCI_LE_PERIODIC_ADVERTISING_SYNC_TRANSFER_RECEIVED_SUBEVT_CODE

Event code value: 0x18

Event description: The LE Periodic Advertising Sync Transfer Received event is used by the Controller to report that it has received periodic advertising synchronization information from the device referred to by the Connection_Handle parameter and either successfully synchronized to the periodic advertising train or timed out while attempting to synchronize. The Status is zero if it is successfully synchronized and non-zero otherwise. The Service_Data value is provided by the Host of the device sending the information. The Sync_Handle identifies the periodic advertising in subsequent commands and events and shall be assigned by the Controller. The remaining parameters provide information about the periodic advertising (see Section 7.7.65.14). If Status is non-zero, all parameter values are valid except Sync_Handle, which the Host shall ignore.

Note: If the Controller is already synchronized to the periodic advertising train described in the received information, no event is generated.

3.2.24 LE CIS Established

Event code: HCI_LE_PERIODIC_ADVERTISING_SYNC_TRANSFER_RECEIVED_SUBEVT_CODE

Event code value: 0x19

Event description: The LE CIS Established event indicates that a CIS has been established, was considered lost before being established, or (on the Central) was rejected by the Peripheral. It is generated by the Controller in the Central and Peripheral. The Connection_Handle parameter shall be set to the value provided in the HCI_LE_Create_CIS command on the Central and in the HCI_LE_CIS_Request event on the Peripheral. The CIG_Sync_Delay parameter is the maximum time, in microseconds, for transmission of PDUs of all CISes in a CIG event (see [Vol 6] Part B, Section 4.5.14.1). The CIS_Sync_Delay parameter is the maximum time, in microseconds, for transmission of PDUs of the specified CIS in a CIG event (see [Vol 6] Part B, Section 4.5.14.1). The Transport_Latency_C_To_P and Transport_Latency_P_To_C parameters are the actual transport latencies, in microseconds, as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2. The PHY_C_To_P parameter indicates the PHY selected for packets from the Central to Peripheral. The PHY_P_To_C parameter indicates the PHY selected for packets from the Peripheral to Central. The NSE, BN_C_To_P, BN_P_To_C, FT_C_To_P, FT_P_To_C, Max_PDU_- C_To_P, Max_PDU_P_To_C, and ISO_Interval parameters are the corresponding parameters of the CIS (see [Vol 6] Part B, Section 4.5.13.1). If this event is generated on the Peripheral with a non-zero status, the Controller shall delete the Connection_Handle and any associated ISO data paths.

3.2.25 LE CIS Request

Event code: HCI_LE_CIS_REQUEST_SUBEVT_CODE

Event code value: 0x1A

Event description: The LE CIS Request event indicates that a Controller has received a request to establish a CIS. If the Controller receives such a request while the HCI_LE_CIS_Request event is masked away, it shall reject it. Otherwise the Controller shall assign a connection handle for the requested CIS and send the handle in the CIS_Connection_Handle parameter of the event. When the Host receives this event it shall respond with either an HCI_LE_Accept_CIS_Request command or an HCI_LE_Reject_CIS_Request command before the timer Connection_Accept_Timeout expires. If it does not, the Controller shall reject the request and generate an HCI_LE_CIS_Established event with the status Connection Accept Timeout Exceeded (0x10). The ACL_Connection_Handle is the connection handle of the ACL connection that is associated with the requested CIS. The CIG_ID parameter contains the identifier of the CIG that contains the requested CIS. This parameter is sent by the Central in the request to establish the CIS. The CIS_ID parameter contains the identifier of the requested CIS. This parameter is sent by the Central in the request to establish the CIS.

3.2.26 LE Create BIG Complete

Event code: HCI_LE_CREATE_BIG_COMPLETE_SUBEVT_CODE

Event code value: 0x1B

Event description: The LE Create BIG Complete event indicates that the HCI_LE_Create_BIG command has completed.

3.2.27 LE Terminate BIG Complete

Event code: HCI_LE_TERMINATE_BIG_COMPLETE_SUBEVT_CODE

Event code value: 0x1C

Event description: The LE Terminate BIG Complete event indicates that the transmission of all the BISes in the BIG are terminated.

3.2.28 LE BIG Sync Established

Event code: HCI_LE_BIG_SYNC_ESTABLISHED_SUBEVT_CODE

Event code value: 0x1D

Event description: The LE BIG Sync Established event indicates that the HCI_LE_BIG_Create_Sync command has completed.

3.2.29 LE BIG Sync Lost

Event code: HCI_LE_BIG_SYNC_LOST_SUBEVT_CODE

Event code value: 0x1E

Event description: The LE BIG Sync Lost event indicates that the Controller has not received any PDUs on a BIG within the timeout period BIG_Sync_Timeout or the BIG has been terminated by the remote device.

3.2.30 LE Request Peer SCA Complete

Event code: HCI_LE_REQUEST_PEER_SCA_COMPLETE_SUBEVT_CODE

Event code value: 0x1F

Event description: The LE Request Peer SCA Complete event indicates that the HCI_LE_Request_Peer_SCA command has been completed. The Peer_Clock_Accuracy parameter contains the sleep clock accuracy of the peer. The Connection_Handle is the connection handle of the ACL connection in which the HCI_LE_Request_Peer_SCA command is issued.

3.2.31 LE Path Loss Threshold

Event code: HCI_LE_PATH_LOSS_THRESHOLD_SUBEVT_CODE

Event code value: 0x20

Event description: The LE Path Loss Threshold event reports a path-loss threshold crossing on the ACL connection identified by the Connection_Handle parameter.

3.2.32 LE Transmit Power Reporting

Event code: HCI_LE_TRANSMIT_POWER_REPORTING_SUBEVT_CODE

Event code value: 0x21

Event description: The LE Transmit Power Reporting event reports the transmit power level on the ACL connection identified by the Connection_Handle parameter.

3.2.33 LE BIG Info Advertising Report

Event code: HCI_LE_BIGINFO_ADVERTISING_REPORT_SUBEVT_CODE

Event code value: 0x22

Event description: The LE BIG Info Advertising Report event indicates that the Controller has received an Advertising PDU that contained a BIG Info field.

3.2.34 LE Subrate Change

Event code: HCI_LE_SUBRATE_CHANGE_SUBEVT_CODE

Event code value: 0x23

Event description: The LE Subrate Change event is used to indicate that a Connection Subrate Update procedure has completed and some parameters of the specified connection have changed. This event shall be issued if the HCI_LE_Subrate_Request command was issued by the Host or the parameters are updated successfully following a request from the peer device. If no parameters are updated following a request from the peer device or the parameters were changed using the Connection Update procedure, then this event shall not be issued.

3.2.35 LE Periodic Advertising Sync Established

Event code: HCI_LE_PERIODIC_ADVERTISING_SYNC_ESTABLISHED_V2_SUBEVT_CODE

Event code value: 0x24

Event description: The LE Periodic Advertising Sync Established event indicates that the Controller has received the first periodic advertising packet from an advertiser after the HCI_LE_Periodic_Advertising_Create_Sync command has been sent to the Controller. The Sync_Handle parameter identifies the periodic advertising train in subsequent commands and events and shall be assigned by the Controller. The Advertising_SID parameter is set to the value of the Advertising SID subfield in the ADI field of the advertising PDU referring to the periodic advertising train. The Advertiser_Address_Type and Advertiser_Address parameters specify the address of the periodic advertiser. The Advertiser_PHY parameter specifies the PHY used for the periodic advertising. The Periodic_Advertising_Interval parameter specifies the interval between the periodic advertising events. The Advertiser_Clock_Accuracy parameter specifies the accuracy of the periodic advertiser's clock. If the periodic advertising has subevents or response slots, then the Num_- Subevents, Subevent_Interval, Response_Slot_Delay, and Response_Slot_- Spacing specify the parameters for these subevents, otherwise these values shall be set to 0x00.

3.2.36 LE Periodic Advertising Report

Event code: HCI_LE_PERIODIC_ADVERTISING_REPORT_V2_SUBEVT_CODE

Event code value: 0x25

Event description: The LE Periodic Advertising Report event indicates that the Controller has received a periodic advertisement or has failed to receive an AUX_SYNC_SUBEVENT_IND PDU. The Sync_Handle parameter identifies the periodic advertising train that the report relates to. The RSSI parameter contains the RSSI value, excluding any Constant Tone Extension. If the Controller supports the Connectionless CTE Receiver feature, RSSI shall not be set to 0x7F. When multiple advertising packets are used to complete a periodic advertising report (e.g., a packet containing an AUX_SYNC_IND PDU combined with one containing an AUX_CHAIN PDU), the RSSI event parameter shall be set based on the last packet received and the TX_Power event parameter shall be set based on the AUX_SYNC_IND PDU. However, the second or subsequent events for the same periodic advertisement may instead have a TX_Power value of 0x7F. The Controller may split the data from a single periodic advertisement (whether one PDU or several) into several reports. If so, each report except the last shall have a Data_Status of "incomplete, more data to come", while the last shall have the value "complete". No further reports shall be sent for a given periodic advertisement after one with a Data_Status other than "incomplete, more data to come". A Data_Status of "incomplete, data truncated" indicates that the Controller attempted to receive an AUX_CHAIN_IND PDU but was not successful or received it but was unable to store the data. The CTE_Type parameter indicates the type of Constant Tone Extension in the periodic advertising packets. The Periodic_Event_Counter parameter indicates the periodic advertising event counter (paEventCounter) of the event that the periodic advertising packet was received in. The Subevent parameter indicates the Periodic Advertising with Responses subevent that the periodic advertising packet was received in. If the Periodic Advertising does not have subevents, then Subevent shall be set to 0xFF. If the Controller receives an AUX_CHAIN_IND PDU with no AdvData, it should send the report (or the last report if it has split the data) immediately without waiting for any subsequent AUX_CHAIN_IND PDUs.

3.2.37 LE Periodic Advertising Sync Transfer Received

Event code: HCI_LE_PERIODIC_ADVERTISING_SYNC_TRANSFER_RECEIVED_V2_SUBEVT_CODE

Event code value: 0x26

Event description: The LE Periodic Advertising Sync Transfer Received event is used by the Controller to report that it has received periodic advertising synchronization information from the device referred to by the Connection_Handle parameter and either successfully synchronized to the periodic advertising train or timed out while attempting to synchronize. The Status is zero if it is successfully synchronized and non-zero otherwise. The Service_Data value is provided by the Host of the device sending the information. The Sync_Handle identifies the periodic advertising in subsequent commands and events and shall be assigned by the Controller. The remaining parameters provide information about the periodic advertising (see Section 7.7.65.14). If there are no subevents or response slots, then the Controller shall set the Num_Subevents parameter to zero and the Host shall ignore the Subevent_Interval, Response_Slot_Delay, and Response_Slot_- Spacing parameters. If Status is non-zero, all parameter values are valid except Sync_Handle, which the Host shall ignore.

Note: If the Controller is already synchronized to the periodic advertising train described in the received information, no event is generated.

3.2.38 LE Periodic Advertising Subevent Data Request

Event code: HCI_LE_PERIODIC_ADVERTISING_SUBEVENT_DATA_REQUEST_SUBEVT_CODE

Event code value: 0x27

Event description: The LE Periodic Advertising Subevent Data Request event is used to allow the Controller to indicate that it is ready to transmit one or more subevents and is requesting the advertising data for these subevents. The Subevent_Data_Count parameter shall be less than or equal to the number of subevents. The Subevent_Start parameter is the first subevent being requested and the Subevent_Data_Count parameter determines the subsequent subevents being requested. The subevent numbers wrap from one less than the number of subevents to zero. This event should be sent from the Controller when it has no data for upcoming subevents. The Controller should request data for as many subevents as it has memory to accept to minimize the number of events generated by the Controller.

3.2.39 LE Periodic Advertising Response Report

Event code: HCI_LE_PERIODIC_ADVERTISING_RESPONSE_REPORT_SUBEVT_CODE

Event code value: 0x28

Event description: The LE Periodic Advertising Response Report event indicates that one or more Bluetooth® devices have responded to a periodic advertising subevent during a PAwR train. The Controller may queue these advertising reports and send information from multiple devices in one HCI_LE_Periodic_Advertising_-Response_Report event. The Controller may fail to transmit the synchronization packet required to enable the response packets to be sent. If this happens, the Controller can report this to the Host using the Tx_Status parameter. The Controller may split the data from a single response into several reports. If so, each report except the last shall have a Data_Status of "incomplete, more data to come", while the last shall have the value "complete". No further reports shall be sent for a given periodic advertisement after one with a Data_Status other than "incomplete, more data to come".

3.2.40 LE Enhanced Connection Complete

Event code: HCI_LE_ENHANCED_CONNECTION_COMPLETE_V2_SUBEVT_CODE

Event code value: 0x29

Event description: The LE Enhanced Connection Complete event indicates to both of the Hosts forming the connection that a new connection has been created. Upon the creation of the connection a Connection_Handle shall be assigned by the Controller, and passed to the Host in this event. If the connection creation fails, this event shall be provided to the Host that had issued the HCI_LE_Create_- Connection or HCI_LE_Extended_Create_Connection command. If this event is unmasked and the HCI_LE_Connection_Complete event is unmasked, only the HCI_LE_Enhanced_Connection_Complete event is sent when a new connection has been created. This event indicates to the Host that issued an HCI_LE_Create_Connection or HCI_LE_Extended_Create_Connection command and received an HCI_Command_Status event if the connection creation failed or was successful. The Peer_Address, Peer_Resolvable_Private_Address, and Local_- Resolvable_Private_Address shall always reflect the most recent packet sent and received on air. The Central_Clock_Accuracy parameter is only valid for a Peripheral. On a Central, this parameter shall be set to 0x00. If the connection is established from periodic advertising with responses and Role is 0x00, then the Advertising_Handle parameter shall be set according to the periodic advertising train the connection was established from. If the connection is established from periodic advertising with responses and Role is 0x01, then the Sync_Handle parameter shall be set according to the periodic advertising train the connection was established from. In all other circumstances, Advertising_Handle and Sync_Handle shall be set to No Advertising_Handle and No Sync_Handle and shall be ignored by the Host.

3.3 GAP event codes

This section describes the supported GAP event codes.

3.3.1 Limited Discoverable

Event code: ACI_GAP_LIMITED_DISCOVERABLE_VSEVT_CODE

Event code value: 0x0400

Event description:

3.3.2 Pairing Complete

Event code: ACI_GAP_PAIRING_COMPLETE_VSEVT_CODE

Event code value: 0x0401

Event description: This event is generated when the pairing process has completed successfully or a pairing procedure timeout has occurred or the pairing has failed. This is to notify the application has been paired with a remote device so it can take further action, or to notify that a timeout has occurred so the upper layer can decide to disconnect the link.

3.3.3 Passkey Requested

Event code: ACI_GAP_PASSKEY_REQ_VSEVT_CODE

Event code value: 0x0402

Event description: This event is generated by the security manager to the application when a passkey is required for pairing. When this event is received, the application has to respond with the aci_gap_passkey_resp command.

3.3.4 Procedure Complete

Event code: ACI_GAP_PROC_COMPLETE_VSEVT_CODE

Event code value: 0x0407

Event description: This event is sent by the GAP to the upper layers when a procedure previously started has been terminated by the upper layer or has completed for any other reason.

3.3.5 Address Not Resolved

Event code: ACI_GAP_ADDR_NOT_RESOLVED_VSEVT_CODE

Event code value: 0x0408

Event description: This event is sent only by a privacy enabled Peripheral. The event is sent to the upper layers when the peripheral is unsuccessful in resolving the resolvable address of the peer device after connecting to it.

3.3.6 Numeric Comparison Value

Event code: ACI_GAP_NUMERIC_COMPARISON_VALUE_VSEVT_CODE

Event code value: 0x0409

Event description: This event is sent only during SC v.4.2 Pairing, when the Numeric Comparison Association model is selected, in order to show the generated Numeric Value and to ask for Confirmation by the User. When this event is received, the application has to respond with the aci_gap_numeric_comparison_value_confirm_yesno command.

3.3.7 Keypress Notification

Event code: ACI_GAP_KEYPRESS_NOTIFICATION_VSEVT_CODE

Event code value: 0x040A

Event description: This event is sent only during SC v.4.2 Pairing, when Keypress Notifications are supported, in order to show the input type signalled by the peer device, having keyboard-only I/O capabilities. When this event is received, no action is required from the User.

3.3.8 Pairing

Event code: ACI_GAP_PAIRING_VSEVT_CODE

Event code value: 0x040B

Event description: This event may be generated when there is a request to start a pairing process. The application shall respond with the aci_gap_pairing_resp command to accept or reject the incoming pairing procedure notified through this event. If the pairing starts with a non-bonded device, the Bonded parameter is set to 0. If the pairing process starts with an already bonded device, the event is raised with the Bonded parameter set to 1. 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 is given with Pairing_Response set to 1 (pairing confirmation only for bonded devices), the event is raised only if pairing is initiated with a bonded device.

3.4 GATT/ATT event codes

This section describes the supported GATT/ATT event codes.

3.4.1 Server Attribute Modified

Event code: ACI_GATT_SRV_ATTRIBUTE_MODIFIED_VSEVT_CODE

Event code value: 0x0C01

Event description: This event is generated to the application by the GATT server when a client modifies any attribute on the server, as consequence of one of the following GATT procedures:

- Write without response
- Signed write without response
- Write characteristic value
- Write long characteristic value
- Reliable write

3.4.2 Procedure Timeout

Event code: ACI_GATT_PROC_TIMEOUT_VSEVT_CODE

Event code value: 0x0C02

Event description: This event is generated by the client/server to the application on a GATT timeout (30 seconds). This is a critical event that must not happen during normal operating conditions. It is an indication of either a major disruption in the communication link or a mistake in the application which does not provide a reply to GATT procedures. After this event, the GATT channel is closed and no more GATT communication can be performed. The applications is exptected to issue an aci_gap_terminate to disconnect from the peer device.

3.4.3 Exchange MTU Response

Event code: ACI_ATT_EXCHANGE_MTU_RESP_VSEVT_CODE

Event code value: 0x0C03

Event description: This event is generated in response to an Exchange MTU request (local or from the peer), which can happen only on an unenhanced ATT bearer. (See aci_gatt_clt_exchange_config().)

3.4.4 Find Information Response

Event code: ACI_ATT_CLT_FIND_INFO_RESP_VSEVT_CODE

Event code value: 0x0C04

Event description: This event is generated in response to a Find Information Request during a discovery procedure for all the characteristic descriptors. (See aci_gatt_clt_disc_all_char_desc() and Find Information Response in the Bluetooth Core spec.)

3.4.5 CLT Find By Type Value Response

Event code: ACI_ATT_CLT_FIND_BY_TYPE_VALUE_RESP_VSEVT_CODE

Event code value: 0x0C05

Event description: This event is generated during a "discover service by UUID" procedure. (See aci_gatt_clt_disc_primary_service_by_uuid().)

3.4.6 CLT Read By Type Response

Event code: ACI_ATT_CLT_READ_BY_TYPE_RESP_VSEVT_CODE

Event code value: 0x0C06

Event description: This event is generated in response to a ATT_READ_BY_TYPE_REQ, during a "find included service" procedure or a "discover all characteristics" procedure. (See aci_gatt_clt_find_included_services() and aci_gatt_clt_disc_all_char_of_service().)

3.4.7 CLT Read Response

Event code: ACI_ATT_CLT_READ_RESP_VSEVT_CODE

Event code value: 0x0C07

Event description: This event is generated in response to a Read Request. (See aci_gatt_clt_read().)

3.4.8 CLT Read Blob Response

Event code: ACI_ATT_CLT_READ_BLOB_RESP_VSEVT_CODE

Event code value: 0x0C08

Event description: This event can be generated during a read long characteristic value procedure. (See aci_gatt_clt_read_long().)

3.4.9 CLT Read Multiple Response

Event code: ACI_ATT_CLT_READ_MULTIPLE_RESP_VSEVT_CODE

Event code value: 0x0C09

Event description: This event is generated in response to a Read Multiple Request. (See aci_gatt_clt_read_multiple_char_value().)

3.4.10 CLT Read By Group Type Response

Event code: ACI_ATT_CLT_READ_BY_GROUP_TYPE_RESP_VSEVT_CODE

Event code value: 0x0C0A

Event description: This event is generated in response to a Read By Group Type Request, during a "discover all primary services" procedure. (See aci_gatt_clt_disc_all_primary_services().)

3.4.11 CLT Prepare Write Response

Event code: ACI_ATT_CLT_PREPARE_WRITE_RESP_VSEVT_CODE

Event code value: 0x0C0C

Event description: This event is generated in response to an ATT_PREPARE_WRITE_REQ during a write long characteristic value procedure. (See aci_gatt_clt_write_long().)

3.4.12 CLT Execute Write Response

Event code: ACI_ATT_CLT_EXEC_WRITE_RESP_VSEVT_CODE

Event code value: 0x0C0D

Event description: This event is generated in response to an ATT Execute Write Request, during a write long characteristic value procedure. (See aci_gatt_clt_write_long().)

3.4.13 CLT Indication

Event code: ACI_GATT_CLT_INDICATION_VSEVT_CODE

Event code value: 0x0C0E

Event description: This event is generated when an indication is received from the server.

3.4.14 CLT Notification

Event code: ACI_GATT_CLT_NOTIFICATION_VSEVT_CODE

Event code value: 0x0C0F

Event description: This event is generated when a notification is received from the server.

3.4.15 CLT Procedure Complete

Event code: ACI_GATT_CLT_PROC_COMPLETE_VSEVT_CODE

Event code value: 0x0C10

Event description: This event is generated when a GATT client procedure completes either with error or successfully.

3.4.16 CLT Error Response

Event code: ACI_GATT_CLT_ERROR_RESP_VSEVT_CODE

Event code value: 0x0C11

Event description: This event is generated when an Error Response is received from the server. The error response can be given by the server at the end of one of the GATT discovery procedures. This does not mean that the procedure ended with an error, but this error event is part of the procedure itself.

3.4.17 CLT Discover Read Characteristic By UUID

Event code: ACI_GATT_CLT_DISC_READ_CHAR_BY_UUID_RESP_VSEVT_CODE

Event code value: 0x0C12

Event description: This event can be generated during a "Discover Characteristics By UUID" procedure or a "Read using Characteristic UUID" procedure. During a "Discover Characteristics By UUID" procedure, Attribute_Value is a characteristic declaration as defined in the Bluetooth Core spec (vol.3, Part G, ch. 3.3.1), that is, it is composed by:

- Characteristic Properties (1 octet)
- Characteristic Value Handle (2 octets)
- Characteristic UUID (2 or 16 octets)

During a "Read using Characteristic UUID" procedure, Attribute_Value is the value of the characteristic.

3.4.18 TX Pool Available

Event code: ACI_GATT_TX_POOL_AVAILABLE_VSEVT_CODE

Event code value: 0x0C16

Event description: Each time the Bluetooth® stack raises the error code BLE_STATUS_INSUFFICIENT_RESOURCES, aci_gatt_tx_pool_available_event() is generated as soon as the available buffer size is greater than maximum ATT MTU.

3.4.19 Server Confirmation

Event code: ACI_GATT_SRV_CONFIRMATION_VSEVT_CODE

Event code value: 0x0C17

Event description: This event is generated when the client has sent the confirmation to a previously sent indication.

3.4.20 Server Read

Event code: ACI_GATT_SRV_READ_VSEVT_CODE

Event code value: 0x0C19

Event description: This event is generated when the Bluetooth$^{®}$ LE stack needs an attribute value to be returned to the peer, as a result of a remote read operation (Read By Type Request, Read Request, Read Blob Request, Read Multiple Request). After this event is received, aci_gatt_srv_resp() must be used to send the response. This event is not generated if the read is requested on a characteristic or a descriptor that has an associated buffer handled by the stack (see ble_gatt_chr_def_t and ble_gatt_descr_def_t).

Note: This event is not supported on network coprocessor framework.

3.4.21 Server Write

Event code: ACI_GATT_SRV_WRITE_VSEVT_CODE

Event code value: 0x0C1A

Event description: This event is generated when the peer wants to write into a writable characteristic value or descriptor using a write request or command (Write Request, Write command, Signed Write command). If a response is needed, application must respond with an aci_gatt_srv_resp().

Note: This event is not supported on network coprocessor framework.

3.4.22 Server Prepare Write Request

Event code: ACI_ATT_SRV_PREPARE_WRITE_REQ_VSEVT_CODE

Event code value: 0x0C1B

Event description: This event is generated when a prepare write request is received. Application should queue this request and execute or discard it only when a aci_att_srv_exec_write_req_event is received.

Note: This event is not supported on network coprocessor framework.

3.4.23 Server Execute Write Request

Event code: ACI_ATT_SRV_EXEC_WRITE_REQ_VSEVT_CODE

Event code value: 0x0C1C

Event description: This event is generated when an execute write request is received from the peer. This happens when the client wants to write a long attribute (that is, an attribute with a size greater than ATT_MTU - 3) or more than one attribute in a single operation. The aci_gatt_srv_resp command must be sent to give a response to the peer.

3.4.24 Server Prepare Write Request Event code

Event code: ACI_GATT_SRV_AUTHORIZE_NWK_EVENT

Event code value: 0x0C1D

Event description: This event is generated if authorization is needed to access the attribute value. aci_gatt_srv_authorize_resp_nwk command must be sent in response to this event.

Note: This event is supported only on network coprocessor framework.

3.4.25 CLT Read Multiple Variable Length Response

Event code: ACI_ATT_CLT_READ_MULTIPLE_VAR_LEN_RESP_VSEVT_CODE

Event code value: 0x0C1E

Event description: This event is generated in response to a Read Multiple Variable Request. See aci_gatt_clt_read_multiple_var_len_char_value().

3.5 L2CAP event codes

This section describes the supported L2CAP event codes.

3.5.1 Connection Update Response

Event code: ACI_L2CAP_CONNECTION_UPDATE_RESP_VSEVT_CODE

Event code value: 0x0800

Event description: This event is generated when the central responds to the connection update request packet with a connection update response packet.

3.5.2 Procedure Timeout

Event code: ACI_L2CAP_PROC_TIMEOUT_VSEVT_CODE

Event code value: 0x0801

Event description: This event is generated when the central does not respond to the connection update request packet with a connection update response packet or a command reject packet within 30 seconds.

3.5.3 Connection Update Request

Event code: ACI_L2CAP_CONNECTION_UPDATE_REQ_VSEVT_CODE

Event code value: 0x0802

Event description: The event is given by the L2CAP layer when a connection update request is received from the peripheral. The upper layer which receives this event has to respond by sending a aci_l2cap_connection_parameter_update_resp command.

3.5.4 Disconnection Complete

Event code: ACI_L2CAP_COS_DISCONNECTION_COMPLETE_VSEVT_CODE

Event code value: 0x0804

Event description: Event raised when an L2CAP channel using LE Credit Based Flow Control mode is terminated.

3.5.5 Flow Control Credit

Event code: ACI_L2CAP_COS_FLOW_CONTROL_CREDIT_VSEVT_CODE

Event code value: 0x0805

Event description: Event raised when an L2CAP_FLOW_CONTROL_CREDIT_IND is received from the peer, which means that it is capable of receiving additional K-frames (for example, after it has processed one or more K-frames) in LE Credit Based Flow Control.

3.5.6 SDU Data TX

Event code: ACI_L2CAP_COS_SDU_DATA_TX_VSEVT_CODE

Event code value: 0x0806

Event description: Event raised when an SDU to be transmitted has been processed by the local L2CAP layer entity.

3.5.7 L2CAP SDU received

Event code: ACI_L2CAP_COS_SDU_DATA_RX_NWK_EVENT

Event code value: 0x0807

Event description: Event raised when an SDU has been received.

Note: *This event is available only on network coprocessor framework.*

3.5.8 Reconfiguration

Event code: ACI_L2CAP_COS_RECONFIGURATION_VSEVT_CODE

Event code value: 0x0809

Event description: Event raised when an SDU to be transmitted has been processed by the local L2CAP layer entity.

3.5.9 Command Reject

Event code: ACI_L2CAP_COMMAND_REJECT_VSEVT_CODE

Event code value: 0x080A

Event description: Event generated when receiving an L2CAP_CREDIT_BASED_RECONFIGURE_REQ or an L2CAP_CREDIT_BASED_RECONFIGURE_RSP, to reconfigure one or more (up to 5) L2CAP Enhanced Credit Based Flow Control channels.

3.5.10 SDU Data RX

Event code: ACI_L2CAP_COS_SDU_DATA_RX_VSEVT_CODE

Event code value: 0x080D

Event description: Event raised when an SDU has been received. Use aci_l2cap_extract_sdu_data() to extract SDU from buffer.

Note: This event is available on the system on chip framework.

3.5.11 Connection Request

Event code: ACI_L2CAP_COS_CONNECTION_REQ_VSEVT_CODE

Event code value: 0x080E

Event description: Event generated when a request is received from the peer to create one L2CAP Credit Based Flow Control channel or one or more (up to 5) L2CAP Enhanced Credit Based Flow Control channels.

3.5.12 Connection Response

Event code: ACI_L2CAP_COS_CONNECTION_RESP_VSEVT_CODE

Event code value: 0x080F

Event description: Event generated when a response is received from the peer to create one L2CAP Credit Based Flow Control channel or one or more (up to 5) L2CAP Enhanced Credit Based Flow Control channels.

3.6 HAL event codes

This section describes the supported HAL event codes.

3.6.1 Blue Initialization

Event code: ACI_BLUE_INITIALIZED_EVENT

Event code value: 0x0801

Event description: This event inform the application that the network coprocessor has been reset. If the reason code is a system crash, a following event ACI_BLUE_CRASH_INFO_EVENT provides more information regarding the system crash details.

Note: This event is not supported on network coprocessor framework.

3.6.2 Crash Information

Event code: ACI_BLUE_CRASH_INFO_EVENT

Event code value: 0x0803

Event description: This event is given to the application after the ACI_BLUE_INITIALIZED_EVENT when a system crash is detected. This events returns system crash information for debugging purposes. Information reported are useful to understand the root cause of the crash.

Note: This event is not supported on network coprocessor framework.

3.6.3 End Of Radio Activity

Event code: ACI_HAL_END_OF_RADIO_ACTIVITY_VSEVT_CODE

Event code value: 0x0004

Event description: This event is generated when the device completes a radio activity and provide information when a new radio activity is performed. Information provided includes type of radio activity and absolute time in system ticks when a new radio activity is schedule, if any. Application can use this information to schedule user activities synchronous to selected radio activities. A command aci_hal_set_radio_activity_mask is provided to enable radio activity events of user interests, by default no events are enabled. User should take into account that enabling radio events in application with intense radio activity could lead to a fairly high rate of events generated. Application use cases includes synchronizing notification with connection interval, switching antenna at the end of advertising or performing flash erase operation while radio is idle.

3.6.4 Firmware Error

Event code: ACI_HAL_FW_ERROR_VSEVT_CODE

Event code value: 0x0006

Event description: This event is generated to report firmware error information.

3.6.5 LE test end

Event code: ACI_HAL_LE_TEST_END_EVENT

Event code value: 0x0807

Event description: This event is generated when the amount of transmitted test packets specified with aci_hal_transmitter_test_packets() has been reached.

Note: This event is not supported on network coprocessor framework.

3.6.6 Advertising Scan Response Data Update

Event code: ACI_HAL_ADV_SCAN_RESP_DATA_UPDATE_VSEVT_CODE

Event code value: 0x0010

Event description: This event is raised when the advertising or scan response data pointer provided by application becomes active or inactive.

Note: This event is not supported on network coprocessor framework.

3.6.7 PAwR Data Free

Event code: ACI_HAL_PAWR_DATA_FREE_VSEVT_CODE

Event code value: 0x0011

Event description: This event is raised when the PAwR subevent data or the PAwR response data pointer provided by application is no more used by the stack and the associated memory can be freed.

Note: This event is not supported on network coprocessor framework.

4 Bluetooth® LE stack initialization and process framework

This section describes the Bluetooth® LE stack initialization and process framework.

4.1 BLE_STACK_Event

```
void BLE_STACK_Event ( hci_pckt * hci_pckt,
                       uint16_t  length
                     )
```

This function is called when an event is coming from the Bluetooth® LE stack.

Parameters

hci_pckt

The user event received from the Bluetooth® LE stack.

length

Length of the event.

4.2 BLE_STACK_Init

```
tBleStatus BLE_STACK_Init ( const BLE_STACK_InitTypeDef * BLE_STACK_InitStruct )
```

The Bluetooth® LE stack initialization routine.

Parameters

BLE_STACK_InitStruct (BLE_STACK_InitTypeDef Struct Reference)

Pointer to the const structure containing memory and low level hardware configuration data for the device

Returns

- Value indicating success or error code.

BLE_STACK_InitTypeDef Struct Reference

This structure contains memory and low level hardware configuration data for the device.

Parameters

uint16_t BLE_STACK_InitTypeDef::ATT_MTU

Maximum supported ATT_MTU size [23-1020].
Definition at line 675 of `ble_stack.h`.

uint8_t* BLE_STACK_InitTypeDef::BLEStartRamAddress

Start address of the RAM buffer required by the Bluetooth® stack. It must be 32-bit aligned. Use BLE_STACK_TOTAL_BUFFER_SIZE to calculate the correct size.
Definition at line 668 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::CTE_MaxNumAntennaIDs

Maximum number of Antenna IDs in the antenna pattern used in CTE connection oriented mode.
Definition at line 686 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::CTE_MaxNumIQSamples

Maximum number of IQ samples in the buffer used in CTE connection oriented mode.
Definition at line 687 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::ExtraLLProcedureContexts

Maximum number of simultaneous link layer procedures that can be managed, in addition to the minimum required by the stack. The minimum number guarantees one LL procedure initiated by the peer for each link, one LL procedure automatically initiated by the controller and one LL procedure initiated by the host.

Note: This parameter is available on Bluetooth LE stack v4.1 or later.

uint8_t BLE_STACK_InitTypeDef::FilterAcceptListSizeLog2

Twos logarithm of Filter Accept, Resolving, and Advertiser list size.

Definition at line 683 of `ble_stack.h`.

uint16_t BLE_STACK_InitTypeDef::isr0_fifo_size

Size of the internal FIFO used for critical controller events produced by the ISR (for instance, rx data packets).

Definition at line 694 of `ble_stack.h`.

uint16_t BLE_STACK_InitTypeDef::isr1_fifo_size

Size of the internal FIFO used for non-critical controller events produced by the ISR (for instance, advertising or IQ sampling reports).

Definition at line 695 of `ble_stack.h`.

uint16_t BLE_STACK_InitTypeDef::L2CAP_MPS

The maximum size of payload data in octets that the L2CAP layer entity is capable of accepting [0-1024].

Definition at line 684 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::L2CAP_NumChannels

Maximum number of channels in LE Credit Based Flow Control mode [0-255].

Definition at line 685 of `ble_stack.h`.

uint32_t BLE_STACK_InitTypeDef::MaxConnEventLength

Maximum duration of the connection event when the device is peripheral, in units of 625/256 us (~2.44 us).

Definition at line 676 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::MaxNumOfClientProcs

Maximum number of concurrent client's procedures. This value shall be less or equal to NumOfRadioTasks.

Definition at line 671 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::MaxPAwRSubeventDataCount

Maximum number of Periodic Advertising with Responses subevents that data can be requested for

Definition at line 680 of `ble_stack.h`.

uint16_t BLE_STACK_InitTypeDef::NumAttrRecords

Maximum number of attributes that can be stored in the GATT database.

Definition at line 670 of `ble_stack.h`.

uint16_t BLE_STACK_InitTypeDef::NumBlockCount

Number of allocated memory blocks.

Definition at line 674 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfAdvDataSet

Maximum number of advertising data sets, valid only when Advertising Extension Feature is enabled.

Definition at line 678 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfAuxScanSlots

Maximum number of slots for scanning on the secondary advertising channel, valid only when Advertising Extension Feature is enabled.

Definition at line 681 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfBrcBIG

Maximum number of ISO Broadcaster groups.

Definition at line 689 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfBrcBIS

Maximum number of ISO Broadcaster streams.

Definition at line 691 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfCIG

Maximum number of Connected Isochronous Groups.

Definition at line 692 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfCIS

Maximum number of Connected Isochronous Streams.

Definition at line 693 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfEATTChannels

Maximum number of simultaneous EATT active channels.

Definition at line 673 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfRadioTasks

Maximum number of simultaneous radio tasks. Radio controller supports up to 128 simultaneous radio tasks, but actual usable max value depends on the available device RAM (NUM_LINKS used in the calculation of BLE_STACK_TOTAL_BUFFER_SIZE).

Definition at line 672 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfSubeventsPAwR

Maximum number of Periodic Advertising with Responses subevents.

Definition at line 679 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfSyncBIG

Maximum number of ISO Synchronizer groups.

Definition at line 688 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfSyncBIS

Maximum number of ISO Synchronizer streams.

Definition at line 690 of `ble_stack.h`.

uint8_t BLE_STACK_InitTypeDef::NumOfSyncSlots

Maximum number of slots for synchronizing to a periodic advertising train, valid only when Periodic Advertising and Synchronizing Feature is enabled

Definition at line 682 of `ble_stack.h`.

uint16_t BLE_STACK_InitTypeDef::SleepClockAccuracy

Sleep clock accuracy (ppm value)

Definition at line 677 of `ble_stack.h`.

uint32_t BLE_STACK_InitTypeDef::TotalBufferSize

BLE_STACK_TOTAL_BUFFER_SIZE return value, used to check the MACRO correctness

Definition at line 669 of `ble_stack.h`.

uint16_t BLE_STACK_InitTypeDef::user_fifo_size

Size of the internal FIFO used for controller and host events produced outside the ISR

Definition at line 696 of `ble_stack.h`.

4.3 BLE_STACK_RadioHandler

```
void BLE_STACK_RadioHandler ( uint32_t BlueInterrupt )
```

Radio ISR routine.

This is the base function called for any radio ISR.

Parameters

BlueInterrupt

Value of the radio interrupt register.

4.4 BLE_STACK_ReadNextRadioActivity

```
uint8_t BLE_STACK_ReadNextRadioActivity ( uint32_t * NextStateSysTime )
```

This function provide information when a new radio activity is performed. Information provided includes type of radio activity and absolute time in system ticks when a new radio activity is schedule, if any.

Parameters

NextStateSysTime

32-bit absolute current time expressed in internal time units.

Return values:

- *Value* indicating the next state:
  - 0x00: Idle
  - 0x01: Advertising
  - 0x02: Connection event Peripheral
  - 0x03: Scanning
  - 0x04: Connection request
  - 0x05: Connection event Central
  - 0x06: TX test mode
  - 0x07: RX test mode

4.5 BLE_STACK_SleepCheck

```
uint8_t BLE_STACK_SleepCheck ( void )
```

Returns the Bluetooth® LE stack matching sleep mode.

Note: The API name and parameters are subject to change in future releases.

Returns: SLEEPMODE_RUNNING = 0, SLEEPMODE_NOTIMER = 3

4.6 BLE_STACK_Tick

```
void BLE_STACK_Tick ( void )
```

This function executes the processing of all Host Stack layers.

The Bluetooth® LE Stack Tick function has to be executed regularly to process incoming Link Layer packets and to process Host Layer procedures. All stack callbacks are called by this function.

If the Low-Speed Ring Oscillator is used instead of the LS Crystal oscillator, this function also performs the LS RO calibration, and must be called at least once at every system wake-up to keep the 500-ppm accuracy (500- ppm accuracy is mandatory if acting as a Central).

No Bluetooth® LE stack function must be called while the BLE_STACK_Tick is running. For example, if a Bluetooth® LE stack function is called inside an interrupt routine, that interrupt must be disabled during the execution of BLE_STACK_Tick(). Example (if a stack function may be called inside UART ISR):

```
NVIC_DisableIRQ(UART_IRQn);
BLE_STACK_Tick();
NVIC_EnableIRQ(UART_IRQn);
```

4.7 llc_conn_per_statistic

```
tBleStatus llc_conn_per_statistic ( uint16_t conn_handle,
                                    llc_conn_per_statistic_st * statistics_p
                                   )
```

LLC function to collect statistics per link:

- Statistics are stored in a buffer allocated by the application.
- Counters are reset every time the function is called.
- Counters are stopped when the function is called with a pointer to NULL.

Parameters

conn_handle

Connection handle that identifies the connection.

statistics_p

Pointer to the structure where statistics are noted.

Returns: BLE_ERROR_UNKNOWN_CONNECTION_ID in case of invalid conn_handle, BLE_STATUS_SUCCESS otherwise

4.8 llc_conn_per_statistic_by_channel

```
tBleStatus llc_conn_per_statistic_by_channel ( uint16_t conn_handle,
                                               llc_conn_per_statistic_by_channel_st * statist
ics_p
                                              )
```

LLC function to collect statistics per link and per channel:

- Statistics are stored in a buffer allocated by the application.
- Counters are reset every time the function is called.
- Counters are stopped when the function is called with a pointer to NULL.

Parameters

conn_handle

Connection handle that identifies the connection.

statistics_p

Pointer to the structure where statistics are noted.

Returns: BLE_ERROR_UNKNOWN_CONNECTION_ID in case of invalid conn_handle, BLE_STATUS_SUCCESS otherwise

5 Status error codes

Status error codes are used for the return status of all commands. Codes 0x00 to 0x3E are used for HCI commands (see Core Specification v5.2, Vol. 2, part D), all other codes are defined for ACI commands.

5.1 HCI status error codes

Table 7 lists the HCI status error codes.

Table 7. **HCI status error codes**

Error code	Value
BLE_STATUS_SUCCESS	0x00
BLE_ERROR_UNKNOWN_HCI_COMMAND	0x01
BLE_ERROR_UNKNOWN_CONNECTION_ID	0x02
BLE_ERROR_HARDWARE_FAILURE	0x03
BLE_ERROR_AUTHENTICATION_FAILURE	0x05
BLE_ERROR_KEY_MISSING	0x06
BLE_ERROR_MEMORY_CAPACITY_EXCEEDED	0x07
BLE_ERROR_CONNECTION_TIMEOUT	0x08
BLE_ERROR_CONN_LIMIT	0x09
BLE_ERROR_CONNECTION_ALREADY_EXISTS	0x0B
BLE_ERROR_COMMAND_DISALLOWED	0x0C
BLE_ERROR_CONN_REJECT_DUE_TO_LIMITED_RESOURCES	0x0D
BLE_ERROR_CONNECTION_ACCEPT_TIMEOUT_EXCEEDED	0x10
BLE_ERROR_UNSUPPORTED_FEATURE	0x11
BLE_ERROR_INVALID_HCI_CMD_PARAMS	0x12
BLE_ERROR_TERMINATED_REMOTE_USER	0x13
BLE_ERROR_TERMINATED_LOCAL_HOST	0x16
BLE_ERROR_UNSUPP_RMT_FEATURE	0x1A
BLE_ERROR_INVALID_LMP_LL_PARAMS	0x1E
BLE_ERROR_UNSPECIFIED	0x1F
BLE_ERROR_UNSUPP_LMP_LL_PARAM_VALUE	0x20
BLE_ERROR_LL_RESPONSE_TIMEOUT	0x22
BLE_ERROR_LL_PROCEDURE_COLLISION	0x23
BLE_ERROR_LMP_PDU_NOT_ALLOWED	0x24
BLE_ERROR_ENC_MODE_NOT_ACCEPTABLE	0x25
BLE_ERROR_INSTANT_PASSED	0x28
BLE_ERROR_DIFFERENT_TRANSACTION_COLLISION	0x2A
BLE_ERROR_CHANNEL_ASSESSMENT_NOT_SUPPORTED	0x2E
BLE_ERROR_PARAMETER_OUT_OF_RANGE	0x30
BLE_ERROR_HOST_BUSY_PAIRING	0x38
BLE_ERROR_CONTROLLER_BUSY	0x3A
BLE_ERROR_UNACCEPTABLE_CONNECTION_PARAMS	0x3B
BLE_ERROR_ADVERTISING_TIMEOUT	0x3C

Error code	Value
BLE_ERROR_CONNECTION_END_WITH_MIC_FAILURE	0x3D
BLE_ERROR_CONNECTION_FAILED_TO_ESTABLISH	0x3E
BLE_ERROR_COARSE_CLOCK_ADJ_REJECTED	0x40
BLE_ERROR_TYPE0_SUBMAP_NOT_DEFINED	0x41
BLE_ERROR_UNKNOWN_ADVERTISING_IDENTIFIER	0x42
BLE_ERROR_LIMIT_REACHED	0x43
BLE_ERROR_OPERATION_CANCELLED_BY_HOST	0x44
BLE_ERROR_PACKET_TOO_LONG	0x45
BLE_ERROR_TOO_LATE	0x46
BLE_ERROR_TOO_EARLY	0x47

5.2 ACI status error codes

Table 8 lists the ACI command status error codes.

Table 8. ACI status error codes

Error code	Value	Description
BLE_STATUS_UNKNOWN_CONNECTION_ID	BLE_ERROR_UNKNOWN_ CONNECTION_ID	The Connection Identifier does not exist. Temporary remapped to corresponding Controller Error.
BLE_STATUS_FAILED	0x81	The Host failed while performing the requested operation.
BLE_STATUS_INVALID_PARAMS	BLE_ERROR_INVALID_ HCI_CMD_PARAMS	Invalid parameters passed at Host layer.
BLE_STATUS_BUSY	0x83	The Host is already processing another request received in advance.
BLE_STATUS_PENDING	0x84	The operation requested cannot be completed immediately by the Host (usually because of lack of resources). The operation is generally put on hold by the caller and it's usually retried on later time.
BLE_STATUS_NOT_ALLOWED	BLE_ERROR_COMMAND_ DISALLOWED	The requested operation cannot be performed by the Host in the current status.
BLE_STATUS_ERROR	0x86	The requested operation violates the logic of the called layer/function or the format of the data to be processed during the operation.
BLE_STATUS_OUT_OF_MEMORY	0x87	The requested operation failed because of lack of memory. Out of memory shall be returned for situations where memory will never become available again.
BLE_STATUS_INSUFFICIENT_RESOURCES	0x88	The requested operation failed for a temporary lack of resources (for example, packet pool or timers), but it may be retried later when resources may become available (packets or timers may have been released by other consumers).
BLE_STATUS_NULL_PARAM	BLE_ERROR_INVALID_ HCI_CMD_PARAMS	A NULL pointer was passed as function parameter.
BLE_STATUS_WAITING_FOR_USER_ RESPONSE	0x90	The Host is temporarily waiting for user response, usually required through a (vendor-specific) event that notifies the required action/response that shall be typically provided through a (vendor-specific) command.

Error code	Value	Description
BLE_STATUS_SECURITY_REQUIREMENTS_ NOT_ACHIEVABLE	0x91	Indicate that the requested Security Level cannot be achieved based on available IO Capabilities and Authentication requirements.
BLE_STATUS_INVALID_CID	0xA0	An invalid L2CAP CID/channel has been selected to send data over.
BLE_STATUS_DEV_IN_BLACKLIST	0xB0	The remote device in the Blacklist and the pairing operation it requested cannot be performed.
BLE_STATUS_CSRK_NOT_FOUND	0xB1	CSRK not found during validation of an incoming signed packet.
BLE_STATUS_IRK_NOT_FOUND	0xB2	Currently not used.
BLE_STATUS_DEV_NOT_FOUND	0xB3	A search for a specific remote device was unsuccessful because no entry exists either into Security/GATT Database (flash-based) or in volatile database.
BLE_STATUS_SEC_DB_FULL	0xB4	The security database is full and no more records can be added.
BLE_STATUS_DEV_NOT_BONDED	0xB5	The remote device is not bonded, and no operations related to bonded devices may be performed (for example, writing GATT Client data).
BLE_INSUFFICIENT_ENC_KEYSIZE	0xB6	The encryption key size used for encrypting the link is insufficient.
BLE_STATUS_SEC_DB_BUSY	0xB7	The security database is temporarily inaccessible because the underlying physical NVM module is busy with other operations.
BLE_STATUS_SEC_PERMISSION_ERROR	0xC0	Notification/Indication can't be sent to the requested remote device because it doesn't satisfy the needed security permission.
BLE_STATUS_ADDRESS_NOT_RESOLVED	0xD0	The address of the device could not be resolved using the IRK stored.
BLE_STATUS_INVALID_SCAN_CONFIGURATION	0xD1	The configuration set by the aci_gap_set_scan_configuration command is not coherent with the GAP procedure that is requested to start.
BLE_STATUS_INVALID_CONNECT_ CONFIGURATION	0xD2	The configuration set by the aci_gap_set_connect_configuration command is not coherent with the GAP procedure that is requested to start.
BLE_STATUS_INVALID_ADV_CONFIGURATION	0xD3	The configuration set by the aci_gap_set_advertising_configuration command is not valid.
BLE_STATUS_INVALID_ADV_FLAGS	0xD4	The discoverability flags in the advertising data are not coherent with the discoverability mode set in the advertising configuration.
BLE_STATUS_NVM_READ_FAILED	0xF0	NVM read failure.
BLE_STATUS_NVM_WRITE_FAILED	0xF1	NVM write failure.
BLE_STATUS_NVM_ERASE_FAILED	0xF2	NVM erase failure.
BLE_STATUS_TIMEOUT	0xFF	Timeout error

6 Modular configuration options and supported APIs

This section lists the STM32WB0 Bluetooth® LE stack v4.x commands and events in terms of which modular configuration options are needed to support each of them.

The commands and events are listed in two separated tables referring to the values provided in Table 9 below.

Table 9. **Modular option values description**

Modular option	Description	Modular option define in file app_conf.h
PRI	Controller Privacy	CFG_BLE_CONTROLLER_PRIVACY_ENABLED
SEC	LE Secure Connections	CFG_BLE_SECURE_CONNECTIONS_ENABLED
SCN	Controller Scanning	CFG_BLE_CONTROLLER_SCAN_ENABLED
DLE	Controller Data Length Extension	CFG_BLE_CONTROLLER_DATA_LENGTH_EXTENSION_ENABLED
2MC	LE 2M/Coded PHY	CFG_BLE_CONTROLLER_2M_CODED_PHY_ENABLED
EAS	Extended Advertising and Scanning	CFG_BLE_CONTROLLER_EXT_ADV_SCAN_ENABLED
COS	L2CAP Connection Oriented Channels	CFG_BLE_L2CAP_COS_ENABLED
PAS	Periodic Advertising and Synchronization	CFG_BLE_CONTROLLER_PERIODIC_ADV_ENABLED
CTE	Constant Tone Extension	CFG_BLE_CONTROLLER_CTE_ENABLED
PCL	LE Power Control	CFG_BLE_CONTROLLER_POWER_CONTROL_ENABLED
CNS	ACL Connection Support	CFG_BLE_CONNECTION_ENABLED
CHC	LE Channel Classification	CFG_BLE_CONTROLLER_CHAN_CLASS_ENABLED
BIS	Broadcast Isochronous Streams	CFG_BLE_CONTROLLER_BIS_ENABLED
SUB	Connection Subrating	CFG_BLE_CONNECTION_SUBRATING_ENABLED
CIS	Connected Isochronous Streams	CFG_BLE_CONTROLLER_CIS_ENABLED
PWR	Periodic Advertising with Responses	CFG_BLE_CONTROLLER_PERIODIC_ADV_WR_ENABLED

Note: If the user wants to enable a specific Bluetooth® LE modular configuration option in their STM32_BLE application, they have to set the related define value in the `app_conf.h` file to 1; if not, they must set it to 0.

Note: The `app_conf.h` file is provided within the STM32CubeWB0 MCU Package, in the `Projects\{NUCLEO-WB09KE|NUCLEO-WB07CC|NUCLEO-WB05KZ}\Applications\BLE` folders.

6.1 Commands with modular options

Table 10. **Commands with modular options**

Command name	Required modular option(s)
aci_gap_clear_advertising_sets	EAS
aci_gap_clear_security_db	CNS
aci_gap_create_connection	SCN and CNS
aci_gap_create_periodic_advertising_connection	EAS and PAS and CNS and PWR
aci_gap_discover_name	SCN and CNS
aci_gap_get_bonded_devices	CNS
aci_gap_get_oob_data	CNS

Command name	Required modular option(s)
aci_gap_get_security_level	CNS
aci_gap_is_device_bonded	CNS
aci_gap_numeric_comparison_value_confirm_yesno	SEC and CNS
aci_gap_pairing_resp	CNS
aci_gap_passkey_input	SEC and CNS
aci_gap_passkey_resp	CNS
aci_gap_remove_advertising_set	EAS
aci_gap_remove_bonded_device	CNS
aci_gap_set_connection_configuration	SCN and CNS
aci_gap_set_io_capability	CNS
aci_gap_set_oob_data	CNS
aci_gap_set_scan_configuration	SCN
aci_gap_set_security	CNS
aci_gap_set_security_requirements	CNS
aci_gap_start_connection_update	CNS and (SCN or (EAS and PAS and CNS and PWR))
aci_gap_start_procedure	SCN
aci_gap_terminate	CNS
aci_gap_terminate_proc	SCN
aci_gatt_clt_add_subscription_security_level	CNS
aci_gatt_clt_confirm_indication	CNS
aci_gatt_clt_disc_all_char_desc	CNS
aci_gatt_clt_disc_all_char_of_service	CNS
aci_gatt_clt_disc_all_primary_services	CNS
aci_gatt_clt_disc_char_by_uuid	CNS
aci_gatt_clt_disc_primary_service_by_uuid	CNS
aci_gatt_clt_exchange_config	CNS
aci_gatt_clt_execute_write_req	CNS
aci_gatt_clt_find_included_services	CNS
aci_gatt_clt_prepare_write_req	CNS
aci_gatt_clt_read	CNS
aci_gatt_clt_read_long	CNS
aci_gatt_clt_read_multiple_char_value	CNS
aci_gatt_clt_read_multiple_var_len_char_value	CNS
aci_gatt_clt_read_using_char_uuid	CNS
aci_gatt_clt_signed_write_without_resp	CNS
aci_gatt_clt_write	CNS
aci_gatt_clt_write_char_reliable	CNS
aci_gatt_clt_write_long	CNS
aci_gatt_clt_write_without_resp	CNS
aci_gatt_set_event_mask	CNS
aci_gatt_srv_add_char	CNS

Command name	Required modular option(s)
aci_gatt_srv_add_char_desc	CNS
aci_gatt_srv_add_service	CNS
aci_gatt_srv_get_char_decl_handle	CNS
aci_gatt_srv_get_descriptor_handle	CNS
aci_gatt_srv_get_include_service_handle	CNS
aci_gatt_srv_get_service_handle	CNS
aci_gatt_srv_include_service	CNS
aci_gatt_srv_multi_notify	CNS
aci_gatt_srv_notify	CNS
aci_gatt_srv_read_handle_value	CNS
aci_gatt_srv_read_multiple_instance_handle_value	CNS
aci_gatt_srv_resp	CNS
aci_gatt_srv_rm_char	CNS
aci_gatt_srv_rm_include_service	CNS
aci_gatt_srv_rm_service	CNS
aci_gatt_srv_write_multiple_instance_handle_value	CNS
aci_hal_get_anchor_point	CNS
aci_hal_peripheral_latency_enable	CNS
aci_hal_set_le_power_control	PCL and CNS
aci_l2cap_connection_parameter_update_req	CNS
aci_l2cap_connection_parameter_update_resp	CNS and (SCN or (EAS and PAS and CNS and PWR))
aci_l2cap_cos_connection_req	COS and CNS
aci_l2cap_cos_connection_resp	COS and CNS
aci_l2cap_cos_disconnect_req	COS and CNS
aci_l2cap_cos_flow_control_credits_ind	COS and CNS
aci_l2cap_cos_reconfigure_req	COS and CNS
aci_l2cap_cos_reconfigure_resp	COS and CNS
aci_l2cap_cos_sdu_data_extract	COS and CNS
aci_l2cap_cos_sdu_data_transmit	COS and CNS
hci_disconnect	CNS
hci_le_accept_cis_request	CNS and CIS
hci_le_add_device_to_periodic_advertiser_list	SCN and EAS and PAS
hci_le_add_device_to_resolving_list	PRI
hci_le_big_create_sync	SCN and EAS and PAS and BIS
hci_le_big_terminate_sync	SCN and EAS and PAS and BIS
hci_le_clear_advertising_sets	EAS
hci_le_clear_periodic_advertiser_list	SCN and EAS and PAS
hci_le_clear_resolving_list	PRI
hci_le_connection_cte_request_enable	CTE and CNS
hci_le_connection_cte_response_enable	CTE and CNS
hci_le_connection_update	CNS and (SCN or (EAS and PAS and CNS and PWR))

Command name	Required modular option(s)
hci_le_create_big	EAS and PAS and BIS
hci_le_create_big_test	EAS and PAS and BIS
hci_le_create_cis	CNS and (CNS and CIS) and (SCN or (EAS and PAS and CNS and PWR))
hci_le_create_connection	SCN and CNS
hci_le_create_connection_cancel	CNS
hci_le_enable_encryption	CNS and (SCN or (EAS and PAS and CNS and PWR))
hci_le_enhanced_read_transmit_power_level	PCL and CNS
hci_le_extended_create_connection	SCN and EAS and CNS
hci_le_extended_create_connection_v2	EAS and PAS and CNS and PWR
hci_le_generate_dhkey	CNS
hci_le_iso_read_test_counters	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_iso_receive_test	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_iso_test_end	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_iso_transmit_test	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_long_term_key_request_negative_reply	CNS
hci_le_long_term_key_request_reply	CNS
hci_le_periodic_advertising_create_sync	SCN and EAS and PAS
hci_le_periodic_advertising_create_sync_cancel	SCN and EAS and PAS
hci_le_periodic_advertising_set_info_transfer	EAS and PAS and CNS
hci_le_periodic_advertising_sync_transfer	SCN and EAS and PAS and CNS
hci_le_periodic_advertising_terminate_sync	SCN and EAS and PAS
hci_le_read_antenna_information	CTE
hci_le_read_buffer_size_v2	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_read_channel_map	CNS
hci_le_read_iso_link_quality	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_read_iso_tx_sync	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_read_local_p256_public_key	CNS
hci_le_read_local_resolvable_address	PRI
hci_le_read_maximum_data_length	DLE and CNS
hci_le_read_number_of_supported_advertising_sets	EAS
hci_le_read_peer_resolvable_address	PRI
hci_le_read_periodic_advertiser_list_size	SCN and EAS and PAS
hci_le_read_phy	2MC and CNS
hci_le_read_remote_features	CNS
hci_le_read_remote_transmit_power_level	PCL and CNS
hci_le_read_resolving_list_size	PRI
hci_le_read_suggested_default_data_length	DLE and CNS
hci_le_receiver_test_v2	2MC
hci_le_receiver_test_v3	CTE
hci_le_reject_cis_request	CNS and CIS
hci_le_remove_advertising_set	EAS

Command name	Required modular option(s)
hci_le_remove_cig	CNS and (CNS and CIS) and (SCN or (EAS and PAS and CNS and PWR))
hci_le_remove_device_from_periodic_advertiser_list	SCN and EAS and PAS
hci_le_remove_device_from_resolving_list	PRI
hci_le_remove_iso_data_path	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_request_peer_sca	CNS and CIS
hci_le_set_address_resolution_enable	PRI
hci_le_set_advertising_set_random_address	EAS
hci_le_set_cig_parameters	CNS and (CNS and CIS) and (SCN or (EAS and PAS and CNS and PWR))
hci_le_set_cig_parameters_test	CNS and (CNS and CIS) and (SCN or (EAS and PAS and CNS and PWR))
hci_le_set_connection_cte_receive_parameters	CTE and CNS
hci_le_set_connection_cte_transmit_parameters	CTE and CNS
hci_le_set_connectionless_cte_transmit_enable	EAS and PAS and CTE
hci_le_set_connectionless_cte_transmit_parameters	EAS and PAS and CTE
hci_le_set_connectionless_iq_sampling_enable	EAS and PAS and CTE
hci_le_set_data_length	DLE and CNS
hci_le_set_data_related_address_changes	PRI
hci_le_set_default_periodic_advertising_sync_transfer_parameters	SCN and EAS and PAS and CNS
hci_le_set_default_phy	2MC
hci_le_set_default_subrate	CNS and SUB
hci_le_set_extended_advertising_enable	EAS
hci_le_set_extended_advertising_parameters	EAS
hci_le_set_extended_advertising_parameters_v2	EAS
hci_le_set_extended_scan_enable	SCN and EAS
hci_le_set_extended_scan_parameters	SCN and EAS
hci_le_set_host_channel_classification	(CNS and (SCN or CHC)) or EAS
hci_le_set_host_feature	EAS or CNS
hci_le_set_path_loss_reporting_enable	PCL and CNS
hci_le_set_path_loss_reporting_parameters	PCL and CNS
hci_le_set_periodic_advertising_enable	EAS and PAS
hci_le_set_periodic_advertising_parameters	EAS and PAS
hci_le_set_periodic_advertising_parameters_v2	EAS and PAS and CNS and PWR
hci_le_set_periodic_advertising_receive_enable	SCN and EAS and PAS
hci_le_set_periodic_advertising_sync_transfer_parameters	SCN and EAS and PAS and CNS
hci_le_set_periodic_sync_subevent	SCN and EAS and PAS and CNS and PWR
hci_le_set_phy	2MC and CNS
hci_le_set_privacy_mode	PRI
hci_le_set_resolvable_private_address_timeout	PRI
hci_le_set_scan_enable	SCN

Command name	Required modular option(s)
hci_le_set_scan_parameters	SCN
hci_le_set_transmit_power_reporting_enable	PCL and CNS
hci_le_setup_iso_data_path	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_subrate_request	CNS and SUB
hci_le_terminate_big	EAS and PAS and BIS
hci_le_transmitter_test_v2	2MC
hci_le_transmitter_test_v3	CTE
hci_le_transmitter_test_v4	CTE or PCL
hci_le_write_suggested_default_data_length	DLE and CNS
hci_read_afh_channel_assessment_mode	CNS and CHC
hci_read_authenticated_payload_timeout	CNS
hci_read_connection_accept_timeout	CNS and CIS
hci_read_remote_version_information	CNS
hci_read_rssi	CNS
hci_read_transmit_power_level	CNS
hci_tx_iso_data	(CNS and CIS) or (EAS and PAS and BIS)
hci_write_afh_channel_assessment_mode	CNS and CHC
hci_write_authenticated_payload_timeout	CNS
hci_write_connection_accept_timeout	CNS and CIS

6.2 Events with modular options

Table 11. **Events with modular options**

Event name	Required modular option(s)
aci_att_clt_exec_write_resp_event	CNS
aci_att_clt_find_by_type_value_resp_event	CNS
aci_att_clt_find_info_resp_event	CNS
aci_att_clt_prepare_write_resp_event	CNS
aci_att_clt_read_blob_resp_event	CNS
aci_att_clt_read_by_group_type_resp_event	CNS
aci_att_clt_read_by_type_resp_event	CNS
aci_att_clt_read_multiple_resp_event	CNS
aci_att_clt_read_multiple_var_len_resp_event	CNS
aci_att_clt_read_resp_event	CNS
aci_att_exchange_mtu_resp_event	CNS
aci_att_srv_exec_write_req_event	CNS
aci_att_srv_prepare_write_req_event	CNS
aci_gatt_srv_authorize_nwk_event	CNS
aci_gap_addr_not_resolved_event	-
aci_gap_keypress_notification_event	SEC and CNS
aci_gap_limited_discoverable_event	CNS

Event name	Required modular option(s)
aci_gap_numeric_comparison_value_event	SEC and CNS
aci_gap_pairing_complete_event	CNS
aci_gap_pairing_event	CNS
aci_gap_passkey_req_event	CNS
aci_gap_proc_complete_event	-
aci_gatt_clt_disc_read_char_by_uuid_resp_event	CNS
aci_gatt_clt_error_resp_event	CNS
aci_gatt_clt_indication_event	CNS
aci_gatt_clt_multi_notification_event	CNS
aci_gatt_clt_multi_notification_int_event	CNS
aci_gatt_clt_notification_event	CNS
aci_gatt_clt_proc_complete_event	CNS
aci_gatt_proc_timeout_event	CNS
aci_gatt_srv_attribute_modified_event	CNS
aci_gatt_srv_confirmation_event	CNS
aci_gatt_srv_read_event	CNS
aci_gatt_srv_write_event	CNS
aci_gatt_tx_pool_available_event	CNS
aci_hal_adv_scan_resp_data_update_event	-
aci_hal_fw_error_event	-
aci_hal_pawr_data_free_event	EAS and PAS and CNS and PWR
aci_l2cap_command_reject_event	CNS
aci_l2cap_connection_update_req_event	CNS
aci_l2cap_connection_update_resp_event	CNS
aci_l2cap_cos_connection_req_event	COS and CNS
aci_l2cap_cos_connection_resp_event	COS and CNS
aci_l2cap_cos_disconnection_complete_event	COS and CNS
aci_l2cap_cos_flow_control_credit_event	COS and CNS
aci_l2cap_cos_reconfiguration_event	COS and CNS
aci_l2cap_cos_sdu_data_rx_event	COS and CNS
aci_l2cap_cos_sdu_data_tx_event	COS and CNS
aci_l2cap_cos_sdu_data_rx_nwk_event	COS and CNS
aci_l2cap_proc_timeout_event	CNS
hci_acl_data_ind_event	CNS
hci_acl_data_tx_cmpl_event	CNS
hci_authenticated_payload_timeout_expired_event	CNS
hci_data_buffer_overflow_event	CNS
hci_disconnection_complete_event	CNS
hci_encryption_change_event	CNS
hci_encryption_key_refresh_complete_event	CNS
hci_le_advertising_report_event	SCN

Event name	Required modular option(s)
hci_le_advertising_set_terminated_event	EAS
hci_le_big_sync_established_event	SCN and EAS and PAS and BIS
hci_le_big_sync_lost_event	SCN and EAS and PAS and BIS
hci_le_biginfo_advertising_report_event	SCN and EAS and PAS and BIS
hci_le_channel_selection_algorithm_event	CNS
hci_le_cis_established_event	CNS and CIS
hci_le_cis_request_event	CNS and CIS
hci_le_connection_complete_event	CNS
hci_le_connection_iq_report_event	CTE and CNS
hci_le_connection_update_complete_event	CNS
hci_le_connectionless_iq_report_event	CTE
hci_le_create_big_complete_event	EAS and PAS and BIS
hci_le_cte_request_failed_event	CTE and CNS
hci_le_data_length_change_event	DLE or 2MC
hci_le_directed_advertising_report_event	PRI and SCN
hci_le_enhanced_connection_complete_event	CNS
hci_le_enhanced_connection_complete_v2_event	EAS and PAS and CNS and PWR
hci_le_extended_advertising_report_event	SCN and EAS
hci_le_generate_dhkey_complete_event	CNS
hci_le_long_term_key_request_event	CNS
hci_le_path_loss_threshold_event	PCL and CNS
hci_le_periodic_advertising_report_event	SCN and EAS and PAS
hci_le_periodic_advertising_report_v2_event	SCN and EAS and PAS and CNS and PWR
hci_le_periodic_advertising_response_report_event	EAS and PAS and CNS and PWR
hci_le_periodic_advertising_subevent_data_request_event	EAS and PAS and CNS and PWR
hci_le_periodic_advertising_sync_established_event	SCN and EAS and PAS
hci_le_periodic_advertising_sync_established_v2_event	SCN and EAS and PAS and CNS and PWR
hci_le_periodic_advertising_sync_lost_event	SCN and EAS and PAS
hci_le_periodic_advertising_sync_transfer_received_event	EAS and PAS and CNS
hci_le_periodic_advertising_sync_transfer_received_v2_event	EAS and PAS and CNS and PWR
hci_le_phy_update_complete_event	2MC and CNS
hci_le_read_local_p256_public_key_complete_event	CNS
hci_le_read_remote_features_complete_event	CNS
hci_le_request_peer_sca_complete_event	CNS and CIS
hci_le_rx_iso_data_event	(CNS and CIS) or (EAS and PAS and BIS)
hci_le_scan_request_received_event	EAS
hci_le_scan_timeout_event	SCN and EAS
hci_le_subrate_change_event	CNS and SUB
hci_le_terminate_big_complete_event	EAS and PAS and BIS
hci_le_transmit_power_reporting_event	PCL and CNS
hci_number_of_completed_packets_event	CNS or (EAS and PAS and BIS)

Event name	Required modular option(s)
hci_read_remote_version_information_complete_event	CNS
hci_rx_acl_data_event	CNS
iso_rx_bn_pdu_event	SCN and EAS and PAS and BIS
iso_terminate_event	EAS and PAS and BIS
iso_tx_bn_pdu_event	EAS and PAS and BIS

Revision history

Table 12. **Document revision history**

Date	Revision	Changes
19-Jun-2024	1	Initial release
09-Jul-2024	2	Updated Introduction
04-Nov-2024	3	Added: • Section 2.5.39: aci_gatt_clt_add_subscription_security_level • Section 2.6.10: aci_l2cap_cos_flow_control_credits_ind • Section 6: Modular configuration options and supported APIs Updated Introduction
28-Jan-2025	4	Moved the HCI ISO test commands from Section 2.1: HCI commands to Section 2.2: HCI test commands
08-Apr-2025	5	Updated: • Section Introduction • Section 2: ACI/HCI commands with commands opcodes for SoC and network coprocessor frameworks. • Section 3: ACI/HCI event codes
30-Jul-2025	6	Updated Section 2.4.36: aci_gap_terminate.
27-Oct-2025	7	Added Section 3.4.24: Server Prepare Write Request Event code. Updated Section 4.2: BLE_STACK_Init and Table 11. Events with modular options.

Contents

List of tables