BLETOOL Commands Manual

Version: BLETOOL_Commands_Manual_V0.2

Date: 2019-5-06

History

Version	Date	Description
0.1	2019-4-18	Initial
0.2	2019-5-06	Add connection, gatt operation commands and error codes

Table of Contents

1. Description

This document contains the full bletool commands and the user guide. It will help you operate the BLE module. Before using these commands, you have to install the “bletool” package according your model. Every command starts with string “bletool”, then followed operation name and its parameters e.g, bletool reset 0 .
Attention: all the numbers (including parameters and returned codes) in this manual are shows in hexadecimal.

2. Commands Reference

2.1 Module Status

2.1.0 help

Get all the commands and their simple instructions.

2.1.1 status

This command is used to get the BLE module status. It has no parameters. It will return “on” and the local Bluetooth address, or “off”.

2.1.2 on

This command is used to open the BLE module. It has no parameters or return. After this command, you can use command “status” to get current status.

2.1.3 off

This command is used to close the BLE module. It has no parameters or return. Once used, all the BLE functions will be shut down.

2.1.4 reset

This command is used to reset the BLE module. Note that all the connections will be cut, connection information will be cleared.

Parameters

Type	Name	Description
uint8	Boot mode	0: Normal reset 1: Boot to UART DFU mode 2: Boot to OTA DFU mode

2.2 Generic Access Profile (le_gap)

2.2.1 discovery_type

This command can be used to set the scan type of the specified PHYs. If the device is currently scanning for

advertising devices on the PHYs, new parameters will take effect when the scanning is restarted.

Parameters

Type	Name	Description
uint8	phys	The PHYs for which the parameters are set. • 1: LE 1M PHY • 4: LE Coded PHY • 5: LE 1M PHY and LE Coded PHY
uint8	scan_type	Scan type indicated by a flag. Values: • 0: Passive scanning • 1: Active scanning In passive scanning mode the device only listens to advertising packets and will not transmit any packet In active scanning mode the device will send out a scan request packet upon receiving advertising packet from a remote device and then it will listen to the scan response packet from remote device. Default value: 0

2.2.2 discovery_time

This command can be used to set the timing parameters of the specified PHYs. If the device is currently scanning for advertising devices the PHYs, new parameters will take effect when the scanning is restarted.

Parameters

Type	Name	Description
uint8	phys	The PHYs for which the parameters are set. • 1: LE 1M PHY • 4: LE Coded PHY • 5: LE 1M PHY and LE Coded PHY
uint16	scan_interval	Scan interval. This is defined as the time interval from when the device started its last scan until it begins the subsequent scan,that is how often to scan • Time = Value x 0.625 ms • Range: 0x0004 to 0xFFFF • Time Range: 2.5 ms to 40.96 s Default value: 10 ms There is a variable delay when switching channels at the end of each scanning interval which is included in the scanning interval time itself. During this switch time no advertising packets will be received by the device. The switch time variation is dependent on use case, for example in case of scanning while keeping active connections the channel switch time might be longer than when only scanning without any active connections. Increasing the scanning interval will reduce the amount of time in which the device cannot

		receive advertising packets as it will switch channels less often. After every scan interval the scanner will change the frequency it operates at. It will cycle through all the three advertising channels in a round robin fashion. According to the specification all three channels must be used by a scanner.
uint16	scan_window	Scan window. The duration of the scan. scan_window shall be less than or equal to scan_interval • Time = Value x 0.625 ms • Range: 0x0004 to 0xFFFF • Time Range: 2.5 ms to 40.96 s Default value: 10 ms Note that packet reception is aborted if it has been started before scan window ends.

2.2.3 discovery

This command can be used to start the GAP discovery procedure to scan for advertising devices on the specified scanning PHY, that is to perform a device discovery. "Invalid Parameter" error will be returned if the scanning PHY value is invalid or the device does not support the PHY. Note that this command will not end until get a signal “CRTL+C” or “kill”.

Parameters

Type	Name	Description
uint8	mode	Bluetooth discovery Mode.Value: • 0: le_gap_discover_limited Discover only limited discoverable devices • 1: le_gap_discover_generic Discover limited and generic discoverable devices • 2: le_gap_discover_observation Discover all devices
uint8	scanning_phy	The scanning PHY. Value: • 1: LE 1M PHY • 4**:** LE Coded PHY
uint8	print_level	The level for print. • **0: original** Just print all the content of a received adv event in hexadecimal. The data order like that: • int8 rssi; • uint8 packet_type; • 6 uint8 address; (little endian) • uint8 address_type; • uint8 bonding; • uint8array data; • **1: simple** Analyzed the address , packet type, address type, rssi, bonding; And then print the original advertising data.

		• **2: detailed** Besides mode 1, analyzed the advertising data according the Bluetooth Core.

2.2.4 end_procedure

This command can be used to end a current GAP procedure. It has no parameters.

2.2.5 adv_data

This command can be used to set user defined data in advertising packets, scan response packets or periodic advertising packets. If advertising mode is currently enabled the new advertising data will be used immediately. Advertising mode can be enabled using command start_adv. Periodic advertising mode can be enabled using command periodic_adv. Note that to use these user defined data, you have to set the discover to le_gap_user_data in command start_adv.

The maximum data length is 31 bytes for legacy advertising and 191 bytes for extended advertising.

The invalid parameter error will be returned in following situations:

- The data length is more than 31 bytes but the advertiser can only advertise using legacy advertising PDUs;
- The data length is more than 191 bytes when the advertiser can advertise using extended advertising PDUs;
- Set the data of advertising data packet when the advertiser is advertising in scannable mode using extended advertising PDUs;
- Set the data of scan response data packet when the advertiser is advertising in connectable mode using extended advertising PDUs.

Note that the user defined data may be overwritten by the system when the advertising is later enabled in other discoverable mode than user_data.

Parameters

Type	Name	Description
uint8	scan_rsp	This value selects if the data is intended for advertising packets, scan response packets, periodic advertising packets or advertising packet in OTA. Values: • 0: Advertising packets • 1: Scan response packets • 2: OTA advertising packets • 4: OTA scan response packets • 8: Periodic advertising packets
uint8array	adv_data	Data to be set. Maximum data length: • 31 bytes for legacy advertising; • 191 bytes for extended advertising

2.2.6 start_adv

This command can be used to start the advertising of the given advertising set with specified discoverable and connectable modes.

The number of concurrent connectable advertisings is also limited by MAX_CONNECTIONS configuration. For example, only one connectable advertising can be enabled if the device has (MAX_CONNECTIONS - 1) connections

when this command is called. The limitation does not apply to non-connectable advertising.

The default advertising configuration in the stack is set to using legacy advertising PDUs on LE 1M PHY. The stack will automatically select extended advertising PDUs if either of the followings has occurred under the default configuration:

1. The connectable mode is set to le_gap_connectable_non_scannable.
2. The primary advertising PHY has been set to LE Coded PHY by command le_gap_set_advertise_phy.

If currently set parameters can't be used then an error will be returned. Specifically, this command fails with "Connection Limit Exceeded" error if it may cause the number of connections exceeding the configured MAX_CONNECTIONS value. It fails with "Invalid Parameter" error if one of the following cases occur:

1. Non-resolvable random address is used but the connectable mode is le_gap_connectable_scannable or le_gap_connectable_non_scannable.
2. The connectable mode is le_gap_connectable_non_scannable, but using legacy advertising PDUs has been explicitly enabled with command adv_config.
3. The primary advertising PHY is LE Coded PHY but using legacy advertising PDUs has been explicitly enabled with command adv_config.
4. The connectable mode is le_gap_connectable_scannable but using extended advertising PDUs has been explicitly enabled or the primary advertising PHY has been set to LE Coded PHY.

If advertising will be enabled in user_data mode, command adv-data should be used to set advertising and scan response data before issuing this command. When the advertising is enabled in other modes than user_data, the advertising and scan response data is generated by the stack using the following procedure:

1. Add a Flags field to advertising data.
2. Add a TX power level field to advertising data if TX power service exists in the local GATT database.
3. Add a Slave Connection Interval Range field to advertising data if the GAP peripheral preferred connection parameters characteristic exists in the local GATT database.
4. Add a list of 16-bit Service UUIDs to advertising data if there are one or more 16-bit service UUIDs to advertise. The list is complete if all advertised 16-bit UUIDs are in advertising data; otherwise the list is incomplete.
5. Add a list of 128-bit service UUIDs to advertising data if there are one or more 128-bit service UUIDs to advertise and there is still free space for this field. The list is complete if all advertised 128-bit UUIDs are in advertising data; otherwise the list is incomplete. Note that an advertising data packet can contain at most one 128-bit service UUID.
6. Try to add the full local name to advertising data if device is not in privacy mode. In case the full local name does not fit into the remaining free space, the advertised name is a shortened version by cutting off the end if the free space has at least 6 bytes; Otherwise, the local name is added to scan response data.

Parameters

Type	Name	Description
uint8	discover	This value define the available Discoverable Modes, which dictate how the device is visible to other devices. Values: • 0: le_gap_non_discoverable Not discoverable • 1: le_gap_limited_discoverable Discoverable using both limited and general discovery procedures • 2: le_gap_general_discoverable Discoverable using general discovery procedure

		• 3: le_gap_broadcast Device is not discoverable in either limited or generic discovery procedure, but may be discovered by using the Observation procedure • 4: le_gap_user_data Send advertising and/or scan response data defined by the user using adv_data. The limited/general discoverable flags are defined by the user.
uint8	connect	This value define the available connectable modes. Values: • 0: le_gap_non_connectable Non-connectable non-scannable. • 1:le_gap_directed_connectable Directed connectable (RESERVED, DO NOT USE) • 2: le_gap_connectable_scannable Undirected connectable scannable. This mode can only be used in legacy advertising PDUs. • 3: le_gap_scannable_non_connectable Undirected scannable (Non-connectable but responds to scan requests) • 4: le_gap_connectable_non_scannable Undirected connectable non-scannable. This mode can only be used in extended advertising PDUs.

2.2.7 adv_config

This command can be used to configure the type of advertising event and other advertising properties of the given advertising set. The command clear_adv_config can be used to clear the configurations set by this command. This setting will take effect on the next advertising enabling.

Parameters

Type	Name	Description
uint32	configurations	Advertising configuration flags to enable. This value can be a bitmask of multiple flags. Flags: • 1 (Bit 0): Use legacy advertising PDUs. • 2 (Bit 1): Omit advertiser's address from all PDUs (anonymous advertising). This flag is effective only in extended advertising. • 4 (Bit 2): Use le_gap_non_resolvable address type. Advertising must be in non-connectable mode if this configuration is enabled. • 8 (Bit 3): Include TX power in advertising packets. This flag is effective only in extended advertising. Default value: 1

2.2.8 clear_adv_config

This command can be used to disable advertising configurations on the given advertising set. The command

adv_config can be used to set configurations. This setting will take effect on the next advertising enabling.

Parameters

Type	Name	Description
uint32	configurations	Advertising configuration flags to disable. This value can be a bitmask of multiple flags. See adv_config for possible flags.

2.2.9 adv_channel_map

This command can be used to set the primary advertising channel map of the given advertising set. This setting will take effect on the next advertising enabling.

Parameters

Type	Name	Description
uint8	channel_map	Advertisement channel map which determines which of the three channels will be used for advertising. This value is given as a bitmask. Values: • 1: Advertise on CH37 • 2: Advertise on CH38 • 3: Advertise on CH37 and CH38 • 4: Advertise on CH39 • 5: Advertise on CH37 and CH39 • 6: Advertise on CH38 and CH39 • 7: Advertise on all channels Recommended value: 7 Default value: 7

2.2.10 adv_phy

This command can be used to set the advertising PHYs of the given advertising set. This setting will take effect on the next advertising enabling. "Invalid Parameter" error will be returned if a PHY value is invalid or the device does not support a given PHY.

Parameters

Type	Name	Description
uint8	primary_phy	The PHY on which the advertising packets are transmitted on the primary advertising channel. If legacy advertising PDUs are being used, the PHY must be LE 1M. Values: • 1: Advertising PHY is LE 1M • 4: Advertising PHY is LE Coded Default: 1
uint8	secondary_phy	The PHY on which the advertising packets are transmitted on the secondary advertising channel.

		Values: • 1: Advertising PHY is LE 1M • 2: Advertising PHY is LE 2M • 4: Advertising PHY is LE Coded Default: 1

2.2.11 adv_timing

This command can be used to set the advertising timing parameters of the given advertising set. This setting will take effect on the next advertising enabling.

Parameters

Type	Name	Description
uint32	interval_min	Minimum advertising interval. Value in units of 0.625 ms • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s Default value: 100 ms
uint32	interval_max	Maximum advertising interval. Value in units of 0.625 ms • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s • Note: interval_max should be bigger than interval_min Default value: 200 ms
uint16	duration	The advertising duration for this advertising set. Value 0 indicates no advertising duration limit and the advertising continues until it is disabled. A non-zero value sets the duration in units of 10 ms. The duration begins at the start of the first advertising event of this advertising set. • Range: 0x0001 to 0xFFFF • Time range: 10 ms to 655.35 s Default value: 0
uint8	maxevents	If non-zero, indicates the maximum number of advertising events to send before stopping advertiser. Value 0 indicates no maximum number limit. Default value: 0

2.2.12 adv_tx_power

This command can be used to limit the maximum advertising TX power on the given advertising set. If the value goes over the global value that has been set using set_system_txpower command, the global value will be the maximum limit. The maximum TX power of legacy advertising is further constrained to not go over +10 dBm. Extended advertising TX power can be +10 dBm and over if Adaptive Frequency Hopping has been enabled. This setting will take effect on the next advertising enabling.
By default, maximum advertising TX power is limited by the global value.

Parameters

Type	Name	Description
int16	power	TX power in 0.1dBm steps, for example the value of 10 is 1dBm and 55 is 5.5dBm

2.2.13 adv_req_report

This command can be used to enable or disable the scan request notification of the given advertising set. This setting will take effect on the next advertising enabling.

Parameters

Type	Name	Description
uint8	report_scan_req	If non-zero, enables scan request notification, and scan requests will be reported as events. Default value: 0

2.2.14 stop_adv

This command can be used to stop the advertising. This command has no parameter.

2.2.15 whitelist

This command is used to enable whitelisting. To add devices to the whitelist either bond with the device or add it manually with command add_whitelist.

Parameters

Type	Name	Description
uint8	enable	1 enable, 0 disable whitelisting.

2.2.16 connect

This command can be used to connect an advertising device with the specified initiating PHY. The Bluetooth stack will enter a state where it continuously scans for the connectable advertising packets from the remote device which matches the Bluetooth address given as a parameter. Scan parameters set in discovery_time are used in this operation. Upon receiving the advertising packet, the module will send a connection request packet to the target device to initiate a Bluetooth connection. To cancel an ongoing connection process use the disconnect command with the handle received in the response from this command.

A connection is opened in no-security mode. If the GATT client needs to read or write the attributes on GATT server requiring encryption or authentication, it must first encrypt the connection using an appropriate authentication method.

If a connection cannot be established at all for some reason (for example, the remote device has gone out of range, has entered into deep sleep, or is not advertising), the stack will try to connect forever. In this case the application will not get any event related to the connection request. To recover from this situation, application can implement a timeout and call disconnect to cancel the connection request.

This command fails with "Connection Limit Exceeded" error if the number of connections attempted to be opened

exceeds the configured MAX_CONNECTIONS value.

This command fails with "Invalid Parameter" error if the initiating PHY value is invalid or the device does not support the PHY.

Parameters

Type	Name	Description
uint8	initiating_phy	The initiating PHY. Value: • 1: LE 1M PHY • 4: LE Coded PHY
uint8	address_type	Address type of the device to connect to.
string	address	Address of the device to connect to.(e.g. ea:87:7d:28:59:f8)

2.2.17 connection_para

This command can be used to set the default Bluetooth connection parameters. The configured values are valid for all subsequent connections that will be established. For changing the parameters of an already established connection use the command inconnection_para.

Parameters

Type	Name	Description
uint16	min_interval	Minimum value for the connection event interval. This must be set be less than or equal to max_interval. • Time = Value x 1.25 ms • Range: 0x0006 to 0x0c80 • Time Range: 7.5 ms to 4 s Default value: 20 ms
uint16	max_interval	Maximum value for the connection event interval. This must be set greater than or equal to min_interval. • Time = Value x 1.25 ms • Range: 0x0006 to 0x0c80 • Time Range: 7.5 ms to 4 s Default value: 50 ms
uint16	latency	Slave latency. This parameter defines how many connection intervals the slave can skip if it has no data to send • Range: 0x0000 to 0x01f4 Default value: 0
uint16	timeout	Supervision timeout. The supervision timeout defines for how long the connection is maintained despite the devices being unable to communicate at the currently configured connection intervals. • Range: 0x000a to 0x0c80 • Time = Value x 10 ms • Time Range: 100 ms to 32 s • The value in milliseconds must be larger than (1 + latency) *

		max_interval * 2, where max_interval is given in milliseconds It is recommended that the supervision timeout is set at a value which allows communication attempts over at least a few connection intervals. Default value: 1000 ms

2.2.18 channel_classification

This command can be used to specify a channel classification for data channels. This classification persists until overwritten with a subsequent command or until the system is reset.

Parameters

Type	Name	Description
uint8array	channel_map	This parameter is 5 bytes and 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. • 0: Channel n is bad. • 1: Channel n is unknown. The most significant bits are reserved and shall be set to 0 for future use. At least two channels shall be marked as unknown.

2.2.19 extended_scan_response

This command can be used to enable and disable the extended scan response event.

Parameters

Type	Name	Description
uint8array	channel_map	Values: • **0:** Disable extended scan response event • **1:** Enable extended scan response event

2.2.20 privacy_mode

This command can be used to enable or disable privacy feature on all GAP roles. The new privacy mode will take effect for advertising on the next advertising enabling, for scanning on the next scan enabling, and for initiating on the next open connection command. When privacy is enabled and the device is advertising or scanning, the stack will maintain a periodic timer with the specified time interval as timeout value. At each timeout the stack will generate a new private resolvable address and use it in advertising data packets and scanning requests.
By default, privacy feature is disabled.

Parameters

Type	Name	Description
uint8	privacy	Values: • 0: Disable extended scan response event

		• 1: Enable extended scan response event
uint8	interval	The minimum time interval between private address change. This parameter is ignored if this command is issued for disabling privacy mode. Values: • 0: Use default interval, 15 minutes • others: The time interval in minutes

2.2.21 periodic_adv

This command can be used to start periodic advertising on the given advertising set. The stack will enable the advertising set automatically if the set has not been enabled and the set can advertise using extended advertising PDUs.

"Invalid Parameter" error will be returned if the application has configured to use legacy advertising PDUs or anonymous advertising, or advertising set has been enabled using legacy advertising PDUs.

Parameters

Type	Name	Description
uint16	interval_min	Minimum periodic advertising interval. Value in units of 1.25 ms • Range: 0x06 to 0xFFFF • Time range: 7.5 ms to 81.92 s Default value: 100 ms
uint16	interval_max	Maximum periodic advertising interval. Value in units of 1.25 ms • Range: 0x06 to 0xFFFF • Time range: 7.5 ms to 81.92 s • Note: interval_max should be bigger than interval_min Default value: 200 ms
uint32	falgs	Periodic advertising configurations. Bitmask of followings: • Bit 0: Include TX power in advertising PDU

2.2.22 stop_periodic_adv

This command can be used to stop the periodic advertising. It has no parameter.

This command does not affect the enable state of the advertising set.

2.3 DTM testing commands(test)

2.3.1 dtm_tx

This command can be used to start a transmitter test. The test is meant to be used against a separate Bluetooth tester device.

In the transmitter test, the device sends packets continuously with a fixed interval. The type and length of each packet is set by **packet_type** and **length** parameters. Parameter **phy** specifies which PHY is used to transmit the packets. All devices support at least the 1M PHY. There is also a special packet type, **test_pkt_carrier**, which can be

used to transmit continuous unmodulated carrier. The **length** field is ignored in this mode.

The test may be stopped using the test_dtm_end command.

Parameters

Type	Name	Description
uint8	packet_type	Test packet types supported by the stack. Values: • 0: test_pkt_prbs9 PRBS9 packet payload • 1: test_pkt_11110000 11110000 packet payload • 2: test_pkt_10101010 10101010 packet payload • 4: test_pkt_11111111 11111111 packet payload • 5: test_pkt_00000000 00000000 packet payload • 6: test_pkt_00001111 00001111 packet payload • 7: test_pkt_01010101 01010101 packet payload • 253: test_pkt_pn9 PN9 continuously modulated output • 254: test_pkt_carrier Unmodulated carrier
uint8	length	Packet length in bytes Range: 0-255
uint8	channel	Bluetooth channel Range: 0-39
uint8	phy	Test PHY types. Values: • 1: test_phy_1m 1M PHY • 1: test_phy_2m 2M PHY • 1: test_phy_125k 125k Coded PHY • 1: test_phy_500k 500k Coded PHY

2.3.2 dtm_rx

This command can be used to start a receiver test. The test is meant to be used against a separate Bluetooth tester device. Parameter **phy** specifies which PHY is used to receive the packets. All devices support at least the 1M PHY. The test may be stopped using the dtm_end command. This will return the number of packets received during the test.

Parameters

Type	Name	Description
uint8	channel	Bluetooth channel Range: 0-39
uint8	phy	Test PHY types. Values: • 1: test_phy_1m 1M PHY • 1: test_phy_2m 2M PHY • 1: test_phy_125k 125k Coded PHY • 1: test_phy_500k 500k Coded PHY

2.3.3 dtm_end

This command can be used to end a transmitter or a receiver test. This will return the number of packets transmitted or received during the test. It has no parameters.

2.4 Connection management(le_connection)

2.4.1 disconnect

This command can be used to close a Bluetooth connection or cancel an ongoing connection establishment process. The parameter is a connection handle which is reported when connection established.

Parameters

Type	Name	Description
uint8	connection	Handle of the connection to be closed

2.4.2 slave_latency

This command temporarily enables or disables slave latency. Used only when Bluetooth device is in slave role.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint8	disable	0 enable, 1 disable slave latency

2.4.3 get_rssi

This command can be used to get the latest RSSI value of a Bluetooth connection.

Parameters

Type	Name	Description
uint8	connection	Connection handle

2.4.4 inconnection_para

This command can be used to request a change in the connection parameters of a Bluetooth connection.

Parameters

Type	Name	Description
uint16	min_interval	Minimum value for the connection event interval. This must be set be less than or equal to max_interval. • Time = Value x 1.25 ms • Range: 0x0006 to 0x0c80 • Time Range: 7.5 ms to 4 s
uint16	max_interval	Maximum value for the connection event interval. This must be set greater than or equal to min_interval. • Time = Value x 1.25 ms

		• Range: 0x0006 to 0x0c80 • Time Range: 7.5 ms to 4 s
uint16	latency	Slave latency. This parameter defines how many connection intervals the slave can skip if it has no data to send • Range: 0x0000 to 0x01f4 Use 0x0000 for default value
uint16	timeout	Supervision timeout. The supervision timeout defines for how long the connection is maintained despite the devices being unable to communicate at the currently configured connection intervals. • Range: 0x000a to 0x0c80 • Time = Value x 10 ms • Time Range: 100 ms to 32 s • The value in milliseconds must be larger than (1 + latency) * max_interval * 2, where max_interval is given in milliseconds It is recommended that the supervision timeout is set at a value which allows communication attempts over at least a few connection intervals.

2.4.5 connection_phy

This command can be used to set preferred PHYs for connection. Preferred PHYs are connection specific. Other than preferred PHY can also be set if remote device does not accept any of the preferred PHYs.
NOTE: 2 Mbit and Coded PHYs are not supported by all devices.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint8	phy	Preferred PHYs for connection. This parameter is bitfield and multiple PHYs can be preferred by setting multiple bits. • 0x01: 1 Mbit PHY • 0x02: 2 Mbit PHY • 0x04: 125 kbit Coded PHY (S=8) • 0x08: 500 kbit Coded PHY (S=2)

2.5 System (system)

2.5.1 get_address

This command can be used to read the **local** Bluetooth public address used by the device. This command has no parameter.

2.5.2 get_count

This command is to get packet and error counters. It will return four numbers, they are **tx_packets**, **rx_packets**, **crc_errors**, **failures**.

- tx_packets: Number of successfully transmitted packets
- rx_packets: Number of successfully received packets
- crc_errors: Number of received packets with CRC errors
- failures: Number of radio failures like aborted tx/rx packets, scheduling failures, etc

Parameters

Type	Name	Description
uint8	reset	Reset counters if parameter value is nonzero

2.5.3 get_random_data

This command can be used to get random data up to 16 bytes. It will return a random bytes array whose length is that you pass in.

Parameters

Type	Name	Description
uint8	length	Length of random data. Maximum length is 16 bytes.

2.5.4 halt

This command forces radio to idle state and allows device to sleep. Advertising, scanning, connections and software timers are halted by this commands. Halted operations are resumed by calling this command with parameter 0. Connections stay alive if system is resumed before connection supervision timeout.
This command should only be used for a short time period (a few seconds at maximum). It halts Bluetooth activity, but all the tasks and operations are still existing inside stack with their own concepts of time. Halting the system for a long time period may have negative consequences on stack's internal states.
NOTE: Software timer is also halted. Hardware interrupts are the only way to wake up from energy mode 2 when system is halted.

Parameters

Type	Name	Description
uint8	halt	Values: • 1: halt • 0: resume

2.5.5 hello

This command does not trigger any event but the response to the command is used to verify that communication between the host and the device is working. It has no parameter.

2.5.6 link_config

Send configuration data to linklayer. This command is used to fine tune low level Bluetooth operation.

Parameters

Type	Name	Description
uint8	key	Key to configure: • 1:HALT, same as system_halt command, value-0 Stop Radio 1- Start Radio • 2:PRIORITY_RANGE, Sets RAIL priority_mapping offset field of linklayer Priority configuration structure to the first byte of value field. • 3:SCAN_CHANNELS, Sets channels to scan on. First byte of value is channel map. 0x1 = Channel 37, 0x2 = Channel 38, 0x4 = Channel 39 • 4:SET_FLAGS, Set Link Layer configuration flags. value is little endian 32bit integer. Flag Values: • 0x00000001 - Disable Feature Exchange when slave • 0x00000002 - Disable Feature Exchange when master • 5:CLR_FLAGS, value is flags to clear. Flags are same as in SET_FLAGS command. • 7:SET_AFH_INTERVAL, Set afh_scan_interval field of Link Layer priority configuration structure. • 0:
uint8array string	data	Configuration data. Length and contents of data field depend on the key value used.

2.5.7 set_name

This command can be used to set the device name. Currently it is possible to set the name which will be used during the OTA update. The name will be stored in persistent storage. If the OTA device name is also set in gecko configuration, the name stored in persistent storage is overwritten with the name in gecko configuration during device boot.

Parameters

Type	Name	Description
uint8	type	Device name to set. Values: • 0: OTA device name
uint8array string	name	Device name

2.5.8 set_address

This command can be used to set the device's Bluetooth identity address. The address can be a public device address or static random address. A valid address set with this command overrides the default Bluetooth public address that was programmed at production, and it will be effective in the next system reboot. The stack treats 00:00:00:00:00:00 and ff:ff:ff:ff:ff:ff as invalid addresses. Thus passing one of them into this command will cause the stack to use the default address in next system reboot.

Parameters

Type	Name	Description
uint8	type	Address type • 0: Public device address • 1: Static random address
string	address	Bluetooth identity address in little endian format. E.g. ea:87:7d:28:59:f8

2.5.9 set_system_txpower

This command can be used to set the global maximum TX power for Bluetooth. The returned value in the response is the selected maximum output power level after applying RF path compensation. If the GATT server contains a Tx Power service, the Tx Power Level attribute of the service will be updated accordingly.
The selected power level may be less than the specified value if the device does not meet the power requirements. For Bluetooth connections the maximum TX power will be limited to 10 dBm if Adaptive Frequency Hopping (AFH) is not enabled.
By default, the global maximum TX power value is 8 dBm.
NOTE: This command should not be used while advertising, scanning or during connection.

Parameters

Type	Name	Description
int16	power	TX power in 0.1dBm steps, for example the value of 10 is 1dBm and 55 is 5.5dBm

2.6 Generic Attribute Profile (gatt)

2.6.1 discovery_char

This command can be used to discover all characteristics of the defined GATT service from a remote GATT database. This command will print handle, properties and UUID for every discovered characteristic.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint32	service	GATT service handle

2.6.2 primary_service

This command can be used to discover all the primary services of a remote GATT database. This command return the handle and UUID for every discovered primary service.

Parameters

Type	Name	Description
uint8	connection	Connection handle

2.6.3 char_by_uuid

This command can be used to discover all the characteristics of the specified GATT service in a remote GATT database having the specified UUID. This command return the characteristic handle, properties and UUID for every discovered characteristic having the specified UUID.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint32	service	GATT service handle This value is normally received from the gatt_service event.
uint8array string	uuid	Characteristic UUID

2.6.4 discovery_desc

This command can be used to discover all the descriptors of the specified remote GATT characteristics in a remote GATT database. This command just return the descriptor handle and uuid but no value for every discovered descriptor. You can use command read_desc_value to get the descriptor value.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	characteristic	GATT characteristic handle

2.6.5 primary_service_by_uuid

This command can be used to discover primary services with the specified UUID in a remote GATT database. This command will return service handle and UUID for every discovered primary service.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint8array string	uuid	Service UUID

2.6.6 execute_char_write

This command can be used to commit or cancel previously queued writes to a long characteristic of a remote GATT server. Writes are sent to queue with pre_char_write command. Content, offset and length of queued values are validated by this procedure.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint8	flags	0: Cancel all queued writes 1: Commit all queued writes

2.6.7 pre_char_write

This command can be used to add a characteristic value to the write queue of a remote GATT server. This command can be used in cases where very long attributes need to be written, or a set of values needs to be written atomically. At most ATT_MTU - 5 amount of data can be sent once. Writes are executed or cancelled with the execute_char_write command. Whether the writes succeeded or not are indicated in the return of the execute_char_write command.
In all cases where the amount of data to transfer fits into the BGAPI payload the command write_char_value is recommended for writing long values since it transparently performs the pre_char_write and execute_char_write commands.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	characteristic	GATT characteristic handle
uint16	offset	Offset of the characteristic value
uint8array string	value	Value to write into the specified characteristic of the remote GATT database

2.6.8 pre_char_reliable_write

This command can be used to add a characteristic value to the write queue of a remote GATT server and verify if the value was correctly received by the server. Specifically, error code 0x0194 (data_corrupted) will be returned if the value received from the GATT server's response failed to pass the reliable write verification. At most ATT_MTU - 5 amount of data can be sent once. Writes are executed or cancelled with the execute_char_write command. Whether the writes succeeded or not are indicated in the response of the execute_char_write command.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	characteristic	GATT characteristic handle
uint16	offset	Offset of the characteristic value
uint8array string	value	Value to write into the specified characteristic of the remote GATT database

2.6.9 find_included_service

This command can be used to find out if a service of a remote GATT database includes one or more other services. This command will return service handle and UUID for each included service.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint32	service	GATT service handle

2.6.10 read_char_value

This command can be used to read the value of a characteristic from a remote GATT database.
Note that the GATT client does not verify if the requested atrribute is a characteristic value. Thus before calling this command the application should make sure the attribute handle is for a characteristic value in some means, for example, by performing characteristic discovery.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	characteristic	GATT characteristic handle

2.6.11 read_char_value_by_uuid

This command can be used to read the characteristic value of a service from a remote GATT database by giving the UUID of the characteristic and the handle of the service containing this characteristic.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	service	GATT service handle
uint8array string	uuid	Characteristic UUID

2.6.12 read_char_value_by_offset

This command can be used to read a partial characteristic value with specified offset and maximum length from a remote GATT database. It is equivalent to read_char_value if both the offset and maximum length parameters are 0.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	characteristic	GATT characteristic handle
uint16	offset	Offset of the characteristic value
uint16	maxlen	Maximum bytes to read. If this parameter is 0 all characteristic value starting at given offset will be read.

2.6.13 read_desc_value

This command can be used to read the descriptor value of a characteristic in a remote GATT database. It will return handle, offset and value of the specified descriptor.

Parameters

Type	Name	Description

uint8	connection	Connection handle
uint16	descriptor	GATT characteristic descriptor handle

2.6.14 read_mutiple_char_value

This command can be used to read the values of multiple characteristics from a remote GATT database at once. All the characteristic found in the given list will be returned.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint8array	characteristic_list	Little endian encoded uint16 list of characteristics to be read. List all the characteristic handle you want, like: bletool read_mutiple_char_value 1 0001 0002 0003 It will return the value data together.

2.6.15 send_char_confirm

This command must be used to send a characteristic confirmation to a remote GATT server after receiving an indication. Confirmation needs to be sent within 30 seconds, otherwise the GATT transactions between the client and the server are discontinued.

Parameters

Type	Name	Description
uint8	connection	Connection handle

2.6.16 char_notification

This command can be used to enable or disable the notifications and indications being sent from a remote GATT server. This procedure discovers a characteristic client configuration descriptor and writes the related configuration flags to a remote GATT database.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	characteristic	GATT characteristic handle
uint8	flags	Characteristic client configuration flags 0: Disable notifications and indications 1: Notification 2: Indication

2.6.17 set_max_mtu

This command can be used to set the maximum size of ATT Message Transfer Units (MTU). If the given value is too large according to the maximum BGAPI payload size, the system will select the maximal possible value as the

maximum ATT_MTU. If maximum ATT_MTU is larger than 23, MTU is exchanged automatically after a Bluetooth connection has been established.

Parameters

Type	Name	Description
uint8	max_mtu	Maximum size of Message Transfer Units(MTU) allowed Range: 23 to 250 (In Hexadecimal 17 to fa) Default: 247 (In Hexadecimal f7)

2.6.18 write_char_value

This command can be used to write the value of a characteristic in a remote GATT database. If the given value does not fit in one ATT PDU, "write long" GATT procedure is used automatically.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	characteristic	GATT characteristic handle
uint8array string	value	Characteristic value

2.6.19 write_char_value_without_rsp

This command can be used to write the value of a characteristic in a remote GATT server. This command does not generate any event. All failures on the server are ignored silently. For example, if an error is generated in the remote GATT server and the given value is not written into database no error message will be reported to the local GATT client. Note that this command cannot be used to write long values. At most ATT_MTU - 3 amount of data can be sent once.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	characteristic	GATT characteristic handle
uint8array string	value	Characteristic value

2.6.20 write_desc_value

This command can be used to write the value of a characteristic descriptor in a remote GATT database. If the given value does not fit in one ATT PDU, "write long" GATT procedure is used automatically.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	descriptor	GATT characteristic descriptor handle

uint8array string	value	Descriptor value

2.7 Generic Attribute Profile Server (gatt_server)

2.7.1 server_read_attribute_value

This command can be used to read the value of an attribute from a local GATT database. Only (maximum BGAPI payload size - 3) amount of data can be read once. The application can continue reading with increased offset value if it receives (maximum BGAPI payload size - 3) amount of data.

Parameters

Type	Name	Description
uint8	connection	Connection handle
uint16	attribute	Attribute handle
uint16	offset	Value offset

3. Error Codes

3.1 Errors related to hardware

Code	Description
0x0501	Flash reserved for PS store is full
0x0502	PS key not found
0x0503	Acknowledge for i2c was not received.
0x0504	I2C read or write timed out.

3.2 Errors related to BGAPI protocol

Code	Description
0x0101	Invalid GATT connection handle.
0x0102	Waiting response from GATT server to previous proce-dure.
0x0103	GATT connection is closed due procedure timeout.
0x0180	Command contained invalid parameter
0x0181	Device is in wrong state to receive command
0x0182	Device has run out of memory
0x0183	Feature is not implemented

0x0184	Command was not recognized
0x0185	A command or procedure failed or a link lost due totimeout
0x0186	Connection handle passed is to command is not a validhandle
0x0187	Command would cause either underflow or overflow er-ror
0x0188	User attribute was accessed through API which is notsupported
0x0189	No valid license key found
0x018a	Command maximum length exceeded
0x018b	Bonding procedure can't be started because device hasno space left for bond.
0x018c	Unspecified error
0x018d	Hardware failure
0x018e	Command not accepted, because internal buffers arefull
0x018f	Command or Procedure failed due to disconnection
0x0190	Too many Simultaneous Requests
0x0191	Feature is not supported in this firmware build
0x0192	The bonding does not exist.
0x0193	Error using crypto functions
0x0194	Data was corrupted.
0x0195	Data received does not form a complete command
0x0196	Feature or subsystem not initialized
0x0197	Invalid periodic advertising sync handle

3.3 Errors from Security Manager Protocol

Code	Description
0x0301	The user input of passkey failed, for example, the user cancelled the operation
0x0302	Out of Band data is not available for authentication
0x0303	The pairing procedure cannot be performed as authentication requirements cannot be met due to IO capabilities of one or both devices
0x0304	The confirm value does not match the calculated compare value
0x0305	Pairing is not supported by the device
0x0306	The resultant encryption key size is insufficient for the security requirements of this device
0x0307	The SMP command received is not supported on this device
0x0308	Pairing failed due to an unspecified reason
0x0309	Pairing or authentication procedure is disallowed be-cause too little time has elapsed since last pairing re-quest or security request
0x030a	The Invalid Parameters error code indicates: the command length is invalid or a parameter is outside of the specified range.
0x030b	Indicates to the remote device that the DH Key Check value received doesn't match the one

	calculated by the local device.
0x030c	Indicates that the confirm values in the numeric comparison protocol do not match.
0x030d	Indicates that the pairing over the LE transport failed due to a Pairing Request sent over the BR/EDR trans-port in process.
0x030e	Indicates that the BR/EDR Link Key generated on the BR/EDR transport cannot be used to derive and distribute keys for the LE transport.

3.4 Bluetooth errors

Code	Description
0x0202	Connection does not exist, or connection open request was cancelled.
0x0205	Pairing or authentication failed due to incorrect results in the pairing or authentication procedure. This could be due to an incorrect PIN or Link Key
0x0206	Pairing failed because of missing PIN, or authentication failed because of missing Key
0x0207	Controller is out of memory.
0x0208	Link supervision timeout has expired.
0x0209	Controller is at limit of connections it can support.
0x020a	The Synchronous Connection Limit to a Device Exceeded error code indicates that the Controller has reached the limit to the number of synchronous connections that can be achieved to a device.
0x020b	The ACL Connection Already Exists error code indicates that an attempt to create a new ACL Connection to a device when there is already a connection to this device.
0x020c	Command requested cannot be executed because the Controller is in a state where it cannot process this command at this time.
0x020d	The Connection Rejected Due To Limited Resources error code indicates that an incoming connection was rejected due to limited resources.
0x020e	The Connection Rejected Due To Security Reasons error code indicates that a connection was rejected due to security requirements not being fulfilled, like authentication or pairing.
0x020f	The Connection was rejected because this device does not accept the BD_ADDR. This may be because the device will only accept connections from specific BD_ADDRs.
0x0210	The Connection Accept Timeout has been exceeded for this connection attempt.
0x0211	A feature or parameter value in the HCI command is not supported.
0x0212	Command contained invalid parameters.
0x0213	User on the remote device terminated the connection.
0x0214	The remote device terminated the connection because of low resources
0x0215	Remote Device Terminated Connection due to Power Off
0x0216	Local device terminated the connection.
0x0217	The Controller is disallowing an authentication or pairing procedure because too little time has elapsed since the last authentication or pairing attempt failed.
0x0218	The device does not allow pairing. This can be for ex-ample, when a device only allows pairing

	during a certain time window after some user input allows pairing
0x021a	The remote device does not support the feature associated with the issued command.
0x021f	No other error code specified is appropriate to use.
0x0222	Connection terminated due to link-layer procedure time-out.
0x0223	LL procedure has collided with the same transaction or procedure that is already in progress.
0x0225	The requested encryption mode is not acceptable at this time.
0x0226	Link key cannot be changed because a fixed unit key is being used.
0x0228	LMP PDU or LL PDU that includes an instant cannot be performed because the instant when this would have occurred has passed.
0x0229	It was not possible to pair as a unit key was requested and it is not supported.
0x022a	LMP transaction was started that collides with an ongoing transaction.
0x022e	The Controller cannot perform channel assessment be-cause it is not supported.
0x022f	The HCI command or LMP PDU sent is only possible on an encrypted link.
0x0230	A parameter value requested is outside the mandatory range of parameters for the given HCI command or LMPPDU.
0x0237	The IO capabilities request or response was rejected because the sending Host does not support Secure Simple Pairing even though the receiving Link Manager does.
0x0238	The Host is busy with another pairing operation and un-able to support the requested pairing. The receiving de-vice should retry pairing again later.
0x0239	The Controller could not calculate an appropriate value for the Channel selection operation.
0x023a	Operation was rejected because the controller is busy and unable to process the request.
0x023b	Remote device terminated the connection because of an unacceptable connection interval.
0x023c	Advertising for a fixed duration completed or, for directed advertising, that advertising completed without a connection being created.
0x023d	Connection was terminated because the Message Integrity Check (MIC) failed on a received packet.
0x023e	LL initiated a connection but the connection has failed to e established. Controller did not receive any packets rom remote end.
0x023f	The MAC of the 802.11 AMP was requested to connect o a peer, but the connection failed.
0x0240	The master, at this time, is unable to make a coarse adjustment to the piconet clock, using the supplied parameters. Instead the master will attempt to move the clock using clock dragging.
0x0242	A command was sent from the Host that should identify an Advertising or Sync handle, but the Advertising or Sync handle does not exist.
0x0243	Number of operations requested has been reached and has indicated the completion of the activity (e.g., advertising or scanning).
0x0244	A request to the Controller issued by the Host and still pending was successfully canceled.
0x0245	An attempt was made to send or receive a packet that exceeds the maximum allowed packet length.

3.5 Application errors

Code	Description
0x0a01	File open failed.
0x0a02	XML parsing failed.
0x0a03	Device connection failed.
0x0a04	Device communication failed.
0x0a05	Device authentication failed.
0x0a06	Device has incorrect GATT database.
0x0a07	Device disconnected due to procedure collision.
0x0a08	Device disconnected due to failure to establish or rees-tablish a secure session.
0x0a09	Encrypion/decryption operation failed.
0x0a0a	Maximum allowed retries exceeded.
0x0a0b	Data parsing failed.
0x0a0c	Pairing established by the application layer protocol hasbeen removed.
0x0a0d	Inactive timeout.
0x0a0e	Mismatched or insufficient security level

3.6 Errors from Attribute Protocol

Code	Description
0x0401	The attribute handle given was not valid on this server
0x0402	The attribute cannot be read
0x0403	The attribute cannot be written
0x0404	The attribute PDU was invalid
0x0405	The attribute requires authentication before it can be read or written.
0x0406	Attribute Server does not support the request received from the client.
0x0407	Offset specified was past the end of the attribute
0x0408	The attribute requires authorization before it can be read or written.
0x0409	Too many prepare writes have been queued
0x040a	No attribute found within the given attribute handle range.
0x040b	The attribute cannot be read or written using the Read Blob Request
0x040c	The Encryption Key Size used for encrypting this link is insufficient.
0x040d	The attribute value length is invalid for the operation
0x040e	The attribute request that was requested has encountered an error that was unlikely, and therefore could not be completed as requested.
0x040f	The attribute requires encryption before it can be read or written.
0x0410	The attribute type is not a supported grouping attribute as defined by a higher layer specification.

0x0411	Insufficient Resources to complete the request
0x0412	The server requests the client to rediscover the data-base.
0x0413	The attribute parameter value was not allowed.
0x0480	When this is returned in a BGAPI response, the application tried to read or write the value of a user attribute from the GATT database.

3.7 Filesystem errors

Code	Description
0x0901	file_not_found

3.8 Errors from Logical Link Control and Adaptation Protocol

Code	Description
0x0d01	Returned when remote disconnects the connection-oriented channel by sending disconnection request.
0x0d02	Returned when local host disconnect the connection-oriented channel by sending disconnection request.
0x0d03	Returned when local host did not find a connection-oriented channel with given destination CID.
0x0d04	Returned when connection-oriented channel disconnected due to LE connection is dropped.
0x0d05	Returned when connection-oriented channel disconnected due to remote end send data even without credit.
0x0d06	Returned when connection-oriented channel disconnected due to remote end send flow control credits exceed65535.
0x0d07	Returned when connection-oriented channel has run out of flow control credit and local application still trying to send data.
0x0d08	Returned when connection-oriented channel has not received connection response message within maximum timeout.
0x0d09	Returned when local host received a connection-oriented channel connection response with an invalid destination CID.
0x0d0a	Returned when local host application tries to send a command which is not suitable for L2CAP channel's cur-rent state.

3.9 Security errors

Code	Description
0x0b01	Device firmware signature verification failed.
0x0b02	File signature verification failed.
0x0b03	Device firmware checksum is not valid.

‘